One Solution For Multimedia Subscription Using Blockchain

Igor Srdić, Đorđe Glišić, Marija Jovanović

***Abstract*—With the expansion of blockchain technologies in different areas from health care to voting systems and emerging demand on video content delivery platforms, it is interesting to investigate possibilities to combine those two into a new system that will help digital content availability. We present and discuss relevant work in the industry. Paper proposes one solution for subscription rights management using blockchain technologies. Proof of concept is done using the Ethereum blockchain ecosystem for a video-on-demand service. A similar approach could be used in other fields of DTV services, like cable TV subscription services.**

***Index Terms*—subscription management, blockchain, smart contracts, Ethereum, DTV content subscription, Android TV, solidity, web3j.**

I. Introduction

Blockchain is a growing list of records linked to the use of the cryptocurrency method. After the sudden expansion of Bitcoin [1], it became clear that the potential of blockchain is a lot greater than its use for money. This technology's decentralized system is fertile ground for many ideas and provides a new approach and opportunities [2]. There are ideas like a voting system for elections, where vote theft and rigging of results could not occur, and many other ideas, each of which aims to improve the current system, which has its virtues and many flaws. The system's main problem with the central authority, with a central database, is that such systems have one critical point - the system's bottleneck, which, if endangered, puts the whole system in danger.

Ethereum is a platform that works on top of blockchain technology which has revolutionized this field. It is a platform that allows its users to develop their decentralized applications [3]. It also resolves one additional issue in such networks: they cannot function without many users - nodes - who work on them.

Buying and renting multimedia content is the default functionality on all TVs, most often as Video On Demand (VOD), where users temporarily purchase rights to view a movie or series. With increased online shopping abilities, new options come to mind. What if a user can sell its rights to another user? Does it always have to be between the content provider and the end-user? In the real world, there are second-hand shops for used goods. Why don't such digital shops exist in the entertainment industry? Blockchain technology may be the right step towards that. It is reasonable to think that users would rather sell digital valuables than offer them free to the public. Some movements are done in that direction with the appearance of NFT (non-fungible token).

Some work has been done toward decentralized video streaming platforms using blockchain technologies in work by Tan et al. [4]. Focus that work was around content creators, advertisers, and consumers and the availability of the content in networks based on blockchain. More discussion on the usability of blockchain smart contracts in the entertainment industry could be found in the work of Pons [5].

Working consortium grouped around Sony, Samsung, and Google work on creating a next-generation video entertainment blockchain. A published white paper [6] presents the new architecture of the distributed content delivery network called Theta Mainnet 4.0. They have anticipated Theta Metachain and Theta Edge Network, as illustrated in Fig. 1. The first network is distributed blockchain of blockchains, where the "metachain" name comes from, resulting in support for an unlimited number of blockchains. The second network is responsible for interaction with users. Using publicly available Theta Video API, users can upload content and request access to content. The network has endpoints for protected content storage, encoding new content uploaded, and delivery to the user with an appropriate NFT-based digital rights management (DRM) system. It is expected to be available by the end of 2022.

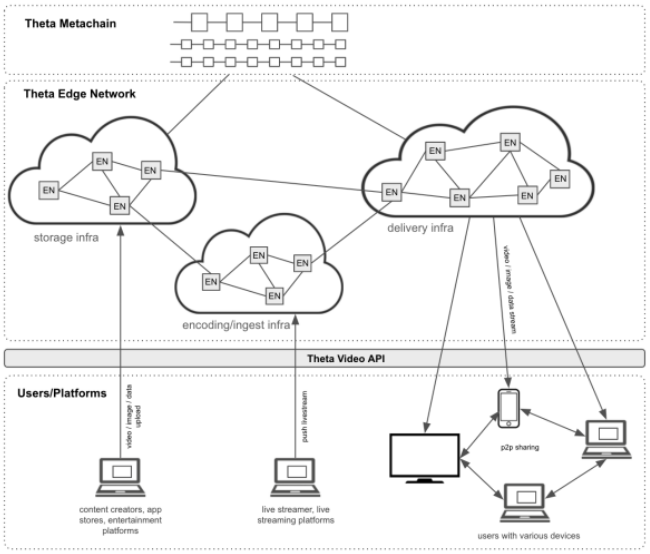


Fig. 1. The architecture of Theta Mainnet 4.0 with Theta Metachain and Theta Edge Network

Igor Srdić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Igor.Srdic@rt-tk.com)

Đorđe Glišić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Djordje.Glisic@rt-tk.com)

Marija Jovanović – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Marija Jovanovic@rt-tk.com

 ISBN 978-86-7466-930-3

In this paper, we are investigating the possibilities of using blockchain technology in the world of digital television. In particular, we tried to use the smart contracts concept from the Ethereum ecosystem to allow users to buy video content from their homes [7]. Once a transaction is initiated, it takes more than 10 seconds for the change to be permanently stored in the blockchain, which is the waiting time for the return values of these functions. Detailed workflow can be seen in Fig. 2. This implies that Ethereum applications are not suitable for all systems that need faster data exchange and cannot correctly use current blockchain technology.

A smart contract as a programming code cannot be changed once it enters the blockchain. The mistakes and oversights are only solvable by creating new smart contracts and redirecting the applications to new contracts. The main problem is that errors are discovered only when they happen, and in such a system, the results of errors are harmful to the user.

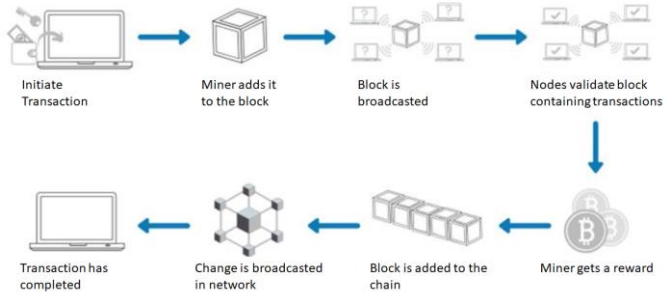


Fig. 2. Workflow for a transaction in the Ethereum network

In this work, smart contracts will be used for exchanging digital currencies for video-on-demand (VOD) services. We will introduce a digital wallet, an application used for storing digital funds (for example, cryptocurrency). On the side of the DTV device, a DTV application will be altered to support additional payment methods for its video-on-demand service. Those are the building blocks necessary to complete the task of creating a subscription management system with blockchain technology.

The paper consists of three parts. The second section will present functional requirements. The third section will explain all technologies and implementation of the system and the TV application adopted to be the client for the blockchain payment method. The fourth and last part contains an overview of the solution itself, shortcomings, and problems that have arisen during implementation and testing, and finally, we will discuss possible improvements.

II. Requirements

Purchase of content should be made on the chain itself. It should be implemented as an Ethereum smart contract that will provide functionalities such as purchase and storage purchase history. Purchased history has two parts, valid and expired. A valid subscription is the one that is still active and allows the owner to consume it (Fig. 3), in our case, to watch a movie or TV show. An expired subscription is essential for validating existing payments and resolving any problems. Additional requirements involve checking content access rights for specific users, content validation by the provider, etc. There is a set of functions related to administration, which can be performed exclusively by the provider. It must be ensured that the user cannot call them.

Fig. 3. The user interface for the Video on Demand service

In addition to the account user created in the TV application, the user also received his Ethereum account and can deposit money at physical ATMs or online. That order is permanently linked to a TV application account, and every content purchase is made through that account.

In this research, we did not want to use any currency assets. Instead, we selected the Ethereum test network Rinkeby. In this network, the currency has no real value, so we could deposit any amount of money into the account for testing purposes. For that reason, work does not cover depositing currencies. That does not reduce the applicability of the proposed solution, as only the target network has to be changed to make it usable in the real world.

Executing the method from smart contracts may take some time, but it is necessary to ensure that the application runs smoothly while waiting for the answers to those calls. An example screen for purchasing content is shown in Fig. 4. In particular, Android has a built-in capability to interrupt application execution that does not work for a while by sending ANR - Application Not Responding error.

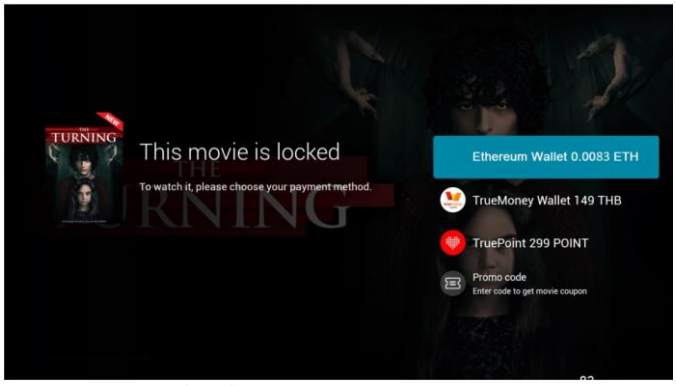


Fig. 4. The user interface for Video on Demand service when content is locked (needs payment)

After the purchase, the user should be able to view the content for 30 days before the first review or two days from

 ISBN 978-86-7466-930-3

when the content was first viewed. After the period expires, the content should be locked again until the next purchase. Given that money can be sent by calling smart contract functions, it is necessary to ensure that, in case of any error when calling these functions, the payments are aborted, and user funds remain untouched in the account.

III. IMPLEMENTATION

To implement our system, we had to implement two main components. The first one is smart to contact that will be positioned in the Ethereum test network. We used Solidity high-level object-oriented programming language especially designed for Ethereum networks. We selected Remix development environment (IDE), written in JavaScript and running under any web browser as a development environment.

On the client-side, we had an Android TV application [7] running on a set-top box (STB) device [8][9]. The application was written in Java. We have selected the web3j library for integration into the existing application to establish a connection with Ethereum smart contract API [10].

For storing the user's digital currency, we selected a digital wallet Metamask, which was used to create the Ethereum account. Its purpose is to save access to the digital currency details in the Ethereum network.

To simulate the Ethereum network, we created a network of miners. For that purpose, we used an Infura cluster made of full nodes. The cluster can execute contracts and alter the chain.

Part of the smart contract is present in Fig. 5. There are three fields of integer type. The *myEtherValue* field is an auxiliary used to convert some numerical data into currency Ether. At the same time, *shortDuration* and *longDuration* represent the lengths of the period in which the buyer is allowed to watch content. The *ctAddress* field is of the address type, the type of data it represents in Solidity public addresses of users in the Ethereum network. This is the address of the crypto-telecom (provider) which will have administrator privileges in the system.

In addition, there are two structures, *Content* and *Deal*. Structure *Content* represents the content and contains fields for name, price, and one auxiliary field. Structure *Deal* is proof of purchase, the contract concluded between provider and customer. It has the identifier of purchased content, the time of the purchase, the time when the user first looked at the content, and the field that indicates whether the content is still viewed. The content folder, which maps the content identifiers to data of the *Content* type, contains all available content of the provider. Maps *validDeals* and *archivedDeals* map users (addresses) into concluded contracts and are in the valid and expired contracts, respectively.

The Web3j library receives the output of the Solidity compiler [11][12] consisting of binary and ABI file and generates a Java surrounding class smart contract, in this case, class *CryptoTelecom*. Some of the methods of this class:

- *load* - loading the contract instance with the given address, web3j object, user credentials, and gas variables.

- *order* – Wrapper method of the smart contract order function. The return value is *RemoteFunctionCall <TransactionReceipt>*, which represents identification of transaction. Once the function is called, it will return its value after the function is executed, the state of the contract is changed, and that state is saved in the new transaction as part of a new block in the blockchain.

```
contract CryptoTelekom {

    uint256 private myEtherValue;
    uint256 private shortDuration;
    uint256 private longDuration;

    address payable private ctAddress;

    struct Content {
        string name;
        uint256 price;
        bool isValid;
    }

    mapping(string => Content) content;

    struct Deal {
        string contentId;
        uint256 startedTime;
        uint256 startedWatchingTime;
        bool startedWatching;
    }

    mapping(address => Deal[]) validDeals;
    mapping(address => Deal[]) archivedDeals;

    modifier onlyCT() {
        assert(msg.sender == ctAddress);
        _;
    }

    event Order(uint256 ret);
    event CheckDeal(bool ret);
    event ValidDeal(string contentId,
                    uint256 startedTime,
                    uint256 startedWatchingTime,
                    bool startedWatching,
                    bool isRetValid);
```

Fig. 5. The interface of the contract class in Solidity.

- *getOrderEvents* - Read events from the order method in a given transaction.

- *getNumValidDeals* - Wrapper method for function *getNumValidDeals()* from smart contract. Example of a return method that does not change the state of the contract. The return value is immediately available because only data is read.

Details about how we used Metamask, Rinkeby network, and Infura cluster can be found in [13]. Also, the authors omitted implementation details about the Android Java application as reference code can be found in [9]. Work does not lose clarity or applicability without those details, as that information can be found both in given references and online resources.

 ISBN 978-86-7466-930-3

IV. Conclusion

The novelty that this approach brings, and at the same time one of its advantages, is that proof of the contract is no longer stored in the provider's database - all are stored on the Ethereum chain. This is interesting for the provider for two reasons. The first is that maintenance of the database by the provider is completely avoided, and the second is that no one can write data to a chain without the consensus of the entire network. Therefore, no one can falsely present themselves as having the right to specific content. Furthermore, users who purchase content are confident that their content rights will remain untouched because data ends on a chain that cannot be changed.

The virtues of this approach are numerous. However, there are also drawbacks. Developers must pay close attention when developing smart contracts because consequences in a system that operates with money can be substantial [14]. Also, it is important to optimize the program code as much as possible. The gas is charged to the initiator of the transaction. Calls to the smart contract function are calculated for every assembler instruction executed. Therefore, additional effort should be made to make the code as optimal as possible to lower gas prices. Also, as transaction initiators are charged for gas, and transactions are initiated by the users who buy the content, the provider has no operating cost for the network. Another approach is to subtract the price gas from the total price of the content that the user buys, which would result in the situation where the provider pays for network service and the user for the content.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review, 2008.

[2] M. Å. Hugoson, “Centralized versus decentralized information systems.” IFIP Conference on History of Nordic Computing. Springer, Berlin, Heidelberg, 2007. Vitalik Buterin, “Ethereum Whitepaper”, https://ethereum.org/whitepaper/, 2013.

[3] Y. Tan, S. Kadhe, K. Ramchandran, “Proof-of-Stream: A Robust Incentivization Protocol for Blockchain-based Hybrid Video on Demand Systems”, UCB/EECS-2021-42, 2021

[4] J. Pons, “Blockchains and smart contracts in the culture and entertainment business”, Réalités industrielles, 2017

[5] N. Szabo, “The Idea of Smart Contracts”, https://nakamotoinstitute.org/the-idea-ofsmart-contracts/, 1997.

[6] Theta Labs, “Theta Mainnet 4.0 - Introducing Theta Metachain to Power Web3 Businesses”, accessed May 2022, https://assets.thetatoken.org/theta-mainnet-4-whitepaper.pdf

[7] I. Pan, N. Lukić, “Design and architecture of software systems: Android-based systems”, FTN Publishing, Novi Sad, 2015.

[8] K. Yaghmour, “Embedded Android”, O ‘Reilly Media, 2013.

[9] N. Schapeler, “A example to Ethereum Development On Android using Web3j and Infura”, accessed May 2022, https://medium.datadriveninvestor.com/an-introduction-to-ethereum-development-on-android-using-web3j-and-infura-763940719997

[10] "Ethereum documentation", accessed May 2022, https://ethdocs.org/en/latest/

[11] "GitHub", Solidity, accessed May 2022, https://github.com/ethereum/solidity

[12] “Solidity documentation”, accessed May 2022, https://solidity.readthedocs.io/en/v0.6.9/.

[13] I. Srdic, BSc thesis “Realizacija multimedijalne pretplate korišćenjem blokčejna”, ETF, 2020

[14] K. O'Hara, "Smart Contracts - Dumb Idea," in IEEE Internet Computing, vol. 21, no. 2, pp. 97-101, Mar.-Apr. 2017, doi: 10.1109/MIC.2017.48.

 ISBN 978-86-7466-930-3

Model-Driven Approach to Blockchain-Enabled MLOps

Nenad Petrović

***Abstract*—In recent years, machine learning has reached quite sophisticated level of usability within applications across various domains – ranging from booking reservations and media content delivery to business and healthcare. However, the deployment of machine learning models, together with parameter tuning and periodic training, which are necessary to maintain satisfiable performance, represent time consuming processes, requiring various types of skills - both DevOps and data analysis-related. In this paper, we leverage model-driven approach in synergy with code generation with aim to automatize the so-called MLOps activities, relying on ZenML framework for pipeline automation and Kubernetes for containerized task orchestration. On top of that, we leverage Blockchain for infrastructure provisioning. Our goal is to reduce the cognitive load of infrastructure and services management within systems relying on machine learning. The framework is evaluated in scenarios using PyTorch-based deep learning predictive models. According to the results, the proposed approach reduces both the time and skill required for successful MLOps activities.**

***Index Terms*—DevOps; Kubernetes; MLOps; PyTorch; Blockchain.**

I. Introduction

Continuous integration and delivery have become standard in software engineering workflow within the last decade. The goal of so-called DevOps practice is to align the deployment of software artifacts with business goals which are enabled by them, so the customer's organization can benefit from them as quickly as possible. However, operations related to underlying infrastructure management are becoming more and more complex, due to heterogeneity of services, devices and increasing performance demands. Therefore, due to fact that machine learning (ML) services are recognized as crucial enablers of novel usage scenarios across various domains, a distinct subfield with focus on them has emerged, known as MLOps [1-4]. It is an extension of now well-established DevOps paradigm with aspects specific to service delivery in machine learning, such as continuous model training for prediction performance improvement, rapid deployment and parameter tunning towards automated generation of complex ML task pipelines [1-4].

In this paper, the focus is on reducing the complexity of MLOps-related activities and service delivery relying on model-driven approach [5]. Moreover, the business-related aspects of infrastructure resource provisioning and usage charging using blockchain by the provider are also considered. The main contributions of this paper are the following: 1) MLOps metamodel – defining the structure of user-created model instances representing machine learning pipelines with several distinct steps together with aspects related to its deployment 2) code generator – leverages the model for automated code generator covering several aspects: pipeline script, predictive model, infrastructure management 3) blockchain-based transaction model making use of smart contracts for renting high-performance computing resources aiming accelerated machine learning.

In our previous works, metamodel-based approach in synergy with ontologies was leveraged for automated container-based service deployment in Fog Computing [6]. On the other side, a similar method was adopted in [7] for generation of predictive models starting from high-level predictive problem descriptions aiming state-of-art mobile network infrastructure planning and management.

Nenad Petrović is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: nenad.petrovic@elfak.ni.ac.rs), (https://orcid.org/0000-0003-2264-7369)

II. Background

A. ZenML

ZenML [8] is open-source, high-level Python framework for machine learning pipeline automation. It is available as Python library in form of function decorators and specific classes inside scripts, while it imposes pre-defined code structure. The overview of main ZenML-related concepts and terminology is given in Table I.

TABLE I
ZenML Concepts Overview

Concept	Description
Repository	A special type of directory, declared used `zenml init` command. Each ZenML action must take place within a repository, which is created
Step	Single stage within ML flow, representing a node of in ML flow computation graph. Implementation-wise, they represent Python functions with typed parameters in signature for both arguments and return value. Decorator used is `@step`, while step result caching can be enabled using `enable_cache=True` parameter
Pipeline	A sequence of steps. It connects all the steps, their inputs and outputs. Decorator used is `@pipeline`. Moreover, it is possible to set the list of external libs/dependencies from .txt file can be done using `requirements_file` attribute of the decorator. It is run by invoking `pipeline.run()` method inside Python script. Optionally, a scheduler object can be assigned for periodic execution of certain steps enabling scenarios such as continuous model training.

 ISBN 978-86-7466-930-3

Stack	Represents environment and configuration of MLOps platform infrastructure. It consists of: artifact store, metadata store, container registry, orchestrator and custom step operators. Creating a new stack with desired parameters is done using `stack register`, while it is run using `stack up` command., which has to be done before running any pipeline.
Artifact store	Persistent storage of step results.
Materializer	Defines how data is passed between steps. Serialization and deserialization are used while storing/retrieving results.
Metadata store	Keeps the data related to pipeline, step and experiment configuration and references for tracking of inputs/outputs within artifact store created within ML pipeline.
Orchestrator	Component for scheduling and running pipeline steps
Container registry	Stores Docker images required for running the steps
Custom step operator	User-defined environments for running ML flow tasks within Docker containers
Integration	Enable usage of various third-party tools enriching ML development like pytorch, tensorflow and sklearn. Apart from that, it also includes orchestrator-enabled stacks, such as local-kubeflow (Kubernetes-based) and airflow.

Depiction of ZenML architecture showing how the previously mentioned concepts are related is given in Fig. 1.

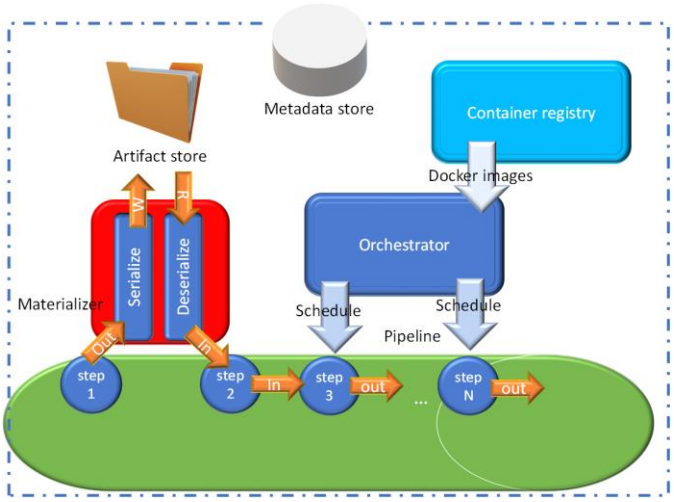


Fig. 1. ZenML concepts and their relations.

Additionally, Fig. 2 shows the programming workflow using ZenML. First, we initialize a repository inside the desired directory where we place our Python script. After that, in Python code we define the typical steps of ML flow: 1) importer – downloading and loading dataset 2) trainer – passing through dataset and updating model weights for new predictions 3) evaluator – estimates how good the prediction performance is, according to the given metric (accuracy for classification; mean relative error - MRE for regression). Moreover, we connect the steps in a sequence as shown within the pipeline object and finally run the pipeline object instance.

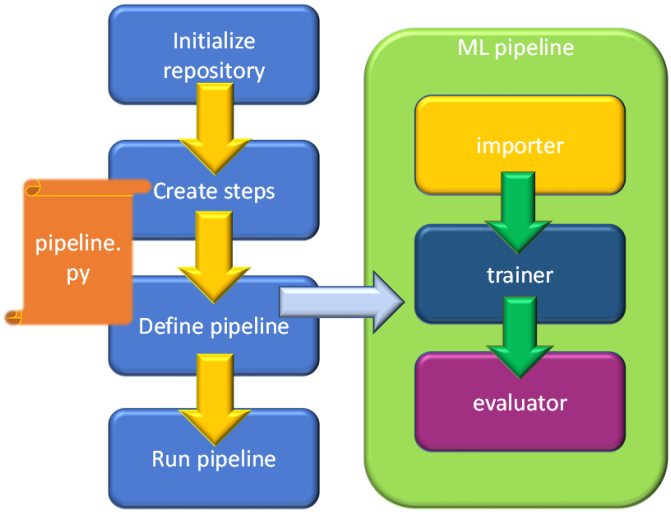


Fig. 2. ML pipeline creation using ZenML in Python.

B. *Kubernetes*

Kubernetes [9] represents an open-source platform whose goal is to enable deployment and management of containerized services run on multi-server clusters. Furthermore, it provides convenient access to useful features, such as scalability, fault-tolerance and declarative configuration. Table III gives an overview of key concepts within Kubernetes-based architectures.

TABLE III
KUBERNETES CONCEPTS OVERVIEW

Concept	Description
Control plane	Cluster management component, responsible for global decisions, and scheduling
Node	Worker machines within the cluster running containerized apps
Pod	Smallest deployable unit, which consists of one or more app containers. These containers share storage, network specification and config. Optionally, might include data volumes for persistent storage
Service	A logical set of pods
Kubectl	Comamnd-Line Interface (CLI) tool for running commands against Kubernetes cluster, such as: -Deployment (`kubectl apply deployment.json` and `kubectl create deployment dep_name --image=docker_image` -Scaling up/down (`kubectl scale --replicas=num resource_name`) -Retrieval of node, pod and service info (`kubectl get pods, nodes, services`)

Despite the fact that Kubernetes provides automatic scheduling capabilities, there might be situations where deployment of pod has to be done on a specific node. In that case, we leverage node labels. The corresponding command for labelling a node has the following form: `kubectl label nodes <node_name> label_name=label_value`. After that, when we want to create a pod using YAML configuration file, it would be necessary to make use of `nodeSelector` property and set it as `label_name:label_value`.

 ISBN 978-86-7466-930-3

Kubernetes-based architecture is depicted in Fig. 3. In this paper, we make use of containerized custom step operators in ZenML run as Kubernetes pods, which are actually containers running PyTorch code. Moreover, node labels are leveraged in order to have low-level scheduling control and determine where each of these steps will be executed, enabling additional flexibility.

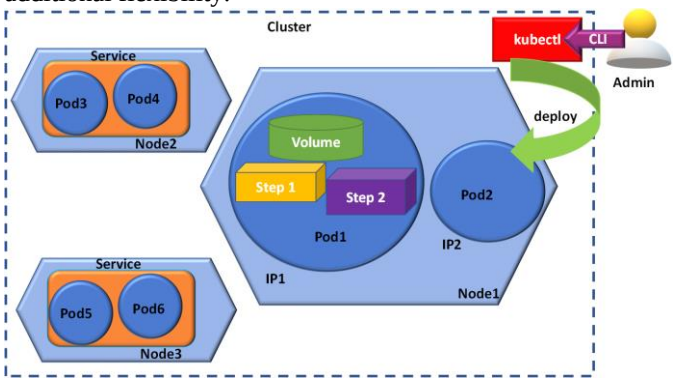


Fig. 3. Kubernetes-based containerized app architecture.

III. IMPLEMENTATION

A. Workflow Overview

Model-driven workflow of the proposed framework for automated ML-related task deployment is illustrated in Fig. 4. In the first step, user creates a deployment diagram model instance which describes the ML pipeline and underlying execution infrastructure details. After that, the model is parsed, so code generator constructs Python script containing the ML pipeline code relying on ZenML and PyTorch in synergy with numpy. Moreover, corresponding Kubernetes orchestrator commands are generated in order to ensure that distinct pipeline steps are executed on the desired cluster nodes. Additionally, the allocated resources are charged to the customer by parametrizing smart contracts for blockchain-based transactions, while the price might vary due to presence of deep learning accelerator cards on some of the nodes. Finally, the generated machine learning Python script is executed on the allocated computing nodes.

Fig. 4. Blockchain-enabled model-driven MLOps workflow.

B. MLOps Metamodel

When it comes to adoption of model-driven engineering, we make use of metamodel which defines the structure of user-created deployment diagram (shown in Fig. 5). For implementation, Ecore [10] within Eclipse Modelling Framework (EMF) [11] in Java is used, which automatically generates all the auxiliary classes for model manipulation, together with convenient GUI-enabled editor.

The highest-level concept is *Pipeline*, which consists of one or more machine learning tasks, referred to as *Step*. A Pipeline can be executed periodically for purpose of continuous training, which is defined by *repeatTime* property. Moreover, each of the Steps can be one of the following type with specific, distinct properties: *Importer*, *Trainer* or *Evaluator*. Importer represents ML task which downloads the corresponding dataset and opens the downloaded file. In this context, it is necessary to set URL corresponding to the location where dataset is stored online, denoted as *onlineData*. Otherwise, if dataset is local and already present on disk, another parameter is used – *localData*. When it comes to trainer step, it is possible to set its learning rate, number of batches, select the target implementation technology, but we make use of PyTorch in this paper. Each trainer can use pre-created model, given by *modelPath* or it is necessary to define a custom neural network. For custom neural network, its architecture is described using *Layer* element, while each of them has type (such as Convolutional – in image classification or standard Fully Connected in Multi-Layer Perceptron), number of processing units (neurons) and activation function (such as ReLU, softmax, sigmoid). Finally, the performance metric used within Evaluator step depends on the type of machine learning task, and we cover two possibilities relevant to supervised learning as *predictionType* property of *Pipeline* – classification (*Accuracy*) and regression (*Mean Relative Error*).

On the other side, the aspects of distinct Step deployment are covered by the metamodel as well. For each pipeline part, there is an attribute *targetLabel*, describing which worker node within Kubernetes cluster would be preferred for execution of pod created within custom step operator. Additionally, infrastructure executing the pipeline is represented as *Cluster* that consists of *Nodes*. For each *Node*, the following properties are customizable, such as label, location, IP address, accelerator (whether it has dedicated hardware for deep learning attached) and unit price (depending on the node performance).

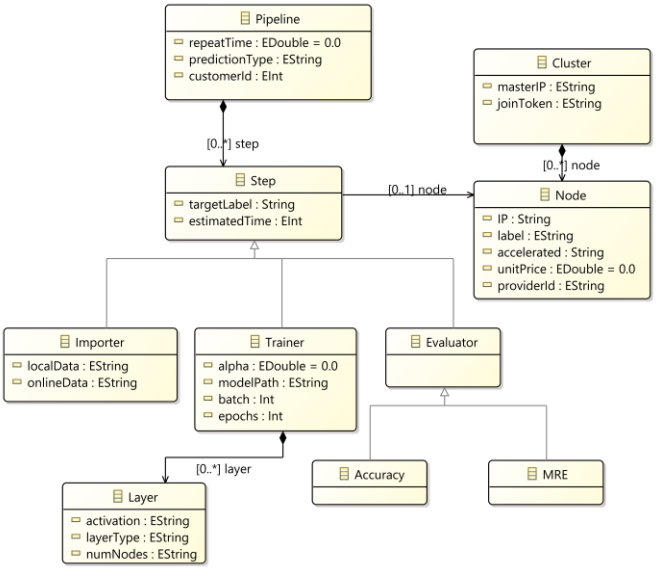


Fig. 5. UML class diagram of MLOps metamodel.

 ISBN 978-86-7466-930-3

C. Code Generation

The user-drawn deployment model is first parsed and then traversed relying on Ecore-generated classes. Pipeline element and iterated for each of the contained steps. When it comes to each step, it is necessary to get the target label and insert it into nodeSelector section within Kubernetes deployment YAML file. After that, depending on the step type, corresponding template is used. For importer steps, it is only necessary to change the path where pre-created model is located. When it comes to trainer, it contains model training loop iterating for given number of epochs through batch number of dataset samples and desired learning rate (α) value. For evaluator step, the corresponding method is chosen according to the type of prediction problem. Additionally, the leasing of resources is charged to the customer by valorizing blockchain smart contract with unit price of the selected node. Finally, the previously created Kubernetes YAML describing step deployment is applied, so the pod with Docker container running ML task inside is spawned on the selected node. Pseudocode of the code generation procedure is given in Table IV.

TABLE IV
CODE GENERATION PSEUDOCODE

Input: MLOps deployment model
Output: Python script, Kubernetes commands, Smart contract
Steps:
1. deployment.elements:=parse(model);
2. Retrieve pipeline from deployment.elements;
3. For each step in pipeline
4. Create nodeSelector for step.targetLabel;
5. If(step is Importer)
6. Generate importer loading dataset from step.localData or onlineData;
7. If(step is Trainer)
8. Load model from step.modelPath;
9. Generate TrainerCode(step.epochs, step.batch, step.alpha);
10. If(step is Evaluator)
11. If pipeline.predictionType is regression
12. Use Mean Relative Error;
13. Else
14. Use Accuracy;
15. Get node.unitPrice;
16. Calculate total leasing price as step.estimatedTime*step.node.unitPrice
17. Genrate smart contract between pipeline.customerId and step.node.providerId for total price;
18. Apply Kubernetes deployment for pod with step.id;
19. End for each
20. End

D. ZenML pipeline Relying on PyTorch Deep Learning Models

Deep learning refers to approach in artificial intelligence making use of neural networks with one or many hidden layers between the inputs and outputs. PyTorch [12] is library for Python which covers the required set of capabilities for deep learning: tensor manipulation and high-level object-oriented representation of both models and datasets. In Fig. 6, an excerpt of typical ZenML pipeline relying on PyTorch models is given. This kind of Python script actually represents one of the outputs of code generator. When it comes to neural network models in PyTorch, their capabilities are encapsulated within *Module* class which has to be inherited by any custom model. In this class, within the constructor we define neural network architecture (layers, nodes and activation functions), while *forward* connects the layers defining how data passes through neural network. In given example, we use MNIST dataset [13] of handwritten digits for purpose of classification.

```
class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        self.flat_network = nn.Sequential(
            nn.Flatten(),
            nn.Linear(784, 311),
            nn.ReLU(),
            nn.Linear(311,10)
        )
        # fully connected layer, output 10 classes
        self.out = nn.Linear(10, 10)

    def forward(self, x):
        x = torch.unsqueeze(x, dim=0)
        x = self.flat_network(x)
        x = self.out(x)
        output = self.out(x)
        return output

def get_data_loader_from_np(X: np.ndarray, y: np.ndarray) -> DataLoader:
    tensor_x = torch.Tensor(X)  # transform to torch tensor
    tensor_y = torch.Tensor(y).type(torch.LongTensor)

    torch_dataset = TensorDataset(tensor_x, tensor_y)
    torch_dataloader = DataLoader(torch_dataset)
    return torch_dataloader

@step(custom_step_operator="trainer1", enable_cache=False)
def torch_trainer(
    X_train: np.ndarray,
    y_train: np.ndarray,
) -> nn.Module:

    train_loader = get_data_loader_from_np(x_train, y_train)

    model = Net().to(DEVICE)
    optimizer = optim.Adadelta(model.parameters(), lr=0.001)

    scheduler = StepLR(optimizer, step_size=1, gamma=0.01)
    for epoch in range(1, num_epochs):
        model.train()
        for batch_idx, (data, target) in enumerate(train_loader):
            data, target = data.to(DEVICE), target.to(DEVICE)
            optimizer.zero_grad()
            output = model(data)
            loss = F.nll_loss(output, target)
            loss.backward()
            optimizer.step()
        scheduler.step()

    return model

@step(custom_step_operator="evaluator1", enable_cache=False)
def classification_evaluator(
    X_test: np.ndarray,
    y_test: np.ndarray,
    model: nn.Module,
) -> float:
    model.eval()
    test_loader = get_data_loader_from_np(x_test, y_test)
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(DEVICE), target.to(DEVICE)
            output = model(data)
            test_loss += F.nll_loss(
                output, target, reduction="sum"
            ).item()
            pred = output.argmax(
                dim=1, keepdim=True
            )
            correct += pred.eq(target.view_as(pred)).sum().item()

    return correct / len(test_loader.dataset)

@step(custom_step_operator="importer1", enable_cache=False)
def my_importer() -> Output(
    x_train=np.ndarray, y_train=np.ndarray, x_test=np.ndarray, y_test=np.ndarray
):
    (X_train, y_train), (
        X_test,
        y_test,
    ) = load_data(dataset_path)
    return x_train, y_train, x_test, y_test

@pipeline(required_integrations=[PYTORCH])
def my_pipeline(
    importer,
    trainer,
    evaluator,
):
    x_train, y_train, x_test, y_test = importer()
    model = trainer(x_train=x_train, y_train=y_train)
    evaluator(x_test=x_test, y_test=y_test, model=model)

continous_train = Schedule(
     start_time = datetime.now(),
     end_time = datetime.now() + timedelta(minutes = 5),
     interval_second = 60
)

if __name__ == "__main__":
    torch_pipeline = my_pipeline(
        importer=my_importer(),
        trainer=torch_trainer(),
        evaluator=classification_evaluator(),
    )

    torch_pipeline.run(schedule = continous_train)
```

Fig. 6. ZenML PyTorch training script for MNIST dataset.

 ISBN 978-86-7466-930-3

E. Resource Leasing Relying on Solidity Smart Contract

Blockchain enables decentralized approach to immutable and irreversible transactions relying on approval by huge network of computer nodes, making it secure and reliable. On the other side, smart contracts define actions executed within protocol for realization of blockchain-based transaction. In this paper, we make se of Ethereum blockchain in synergy with Solidity smart contracts [14]. Solidity code of the underlying transaction mechanism for resource leasing in context of ML task execution is given in Fig. 7. As it can be seen, the information stored as part of transaction consists of *customerId*, *providerId* and identifier of node which will execute some ML task which represents a step within pipeline. First, the total price is calculated by multiplying *unitPrice* and *estimatedTime* required for step execution. After that, the transaction itself is performed by transferring the previously calculated total amount of tmokens from customer's to provider's account.

```
contract LeasingInfrastructure {
   address public providerId;
   uint32 public nodeId;
   uint32 public stepId;
   mapping (address => uint) public balances;

   event Sent(address customerId, address providerId, uint total);

   function leaseNode(address received, uint unitPrice, uint estimateTime) public {
      total = uintPrice*estimatedTime;
       require(total <= balances[msg.sender], "Not enough tokens");
      balances[msg.customerId] -= total;
      balances[providerId] += total;
      emit Sent(msg.customerId, providerId, total);
   }
}
```

Fig. 7. Solidity smart contract for ML task resource leasing.

IV. Experiments and Evaluation

For evaluation of the proposed framework, three publicly available image classification datasets were used. The first two tackle image classification problem: 1) yoga pose determination (our previous work presented in [15]) - 5 poses in dataset of 1551 images 2) MNIST [13] – 70 000 images of handwritten digits 0-9. On the other side, a regression problem of service demand prediction in telco networks from [7] was considered as the third case. In all of the experiments, test was 20% of the overall dataset with no overlapping samples from training set. The presented experiments were run on MacBook Pro (16-inch, 2019) laptop, equipped with 2.3GHz 8-core Intel Core i9 CPU, 16GB of DDR4 memory, 1TB SSD and Intel UHD Graphics 630 with 1.5GB VRAM. On the other side, Kubernetes cluster consisted of two more Ubuntu machines equipped with Intel i5 CPU, 8GB DDR4 RAM and 4GB GPU.

The results of the experiments are given in Table IV. Several aspects were considered: code generation time, model training time, speed-up compared to manual pipeline creation including model creation (moderately experienced machine learning engineer) and achieved prediction performance (MRE for regression, accuracy for classification).

TABLE IV
Experiment results

Case	Code generation [s]	Model training [s]	Speed-up [times]	Performance [%]	Manual pipe [s]
Yoga pose [15]	0.911	317	45	Accuracy 73%	104
MNIST [13]	0.87	124	36	Accuracy 96%	91
Telco [7]	0.93	27	21	MRE 9%	88

As it can be seen, in all the cases, the achieved speed-up was more than 20 times compared to traditional approach involving manual Python code writing from scratch. However, the speed-up is more significant is case of more complex models based on convolutional neural networks with huge number of layers – it was yoga pose determination. In our case, the only manual operation is pipeline deployment model creation using GUI tool, which took about 1.5 minutes in our experiments. All the models show almost identical performance to traditional counterparts, as expected. When it comes to code generation, execution time does not exceed 1 second in the presented case studies. Finally, the overhead of model training compared to execution without MLOps framework is around 15% when run on single machine and k3d [16] local Kubernetes cluster, but can be compensated by smart scheduling techniques, especially for larger datasets.

V. Conclusion and Future Work

According to the achieved experimental results, the proposed model-driven approach to MLOps leveraging automated code generation further speeds up the development of machine learning services, required administration operations and their delivery to the customers. Moreover, it also accelerates resource leasing protocols adopting blockchain-based smart contracts for transactions and their automated generation. Finally, the adoption of intuitive model-driven tools opens new horizons of machine learning service adoption and management even by persons without expertise in this area.

However, there are several possible research directions in future. First, we would work on integration of model-driven resource allocation mechanisms relying on multi-objective optimization approach [17] for energy and cost-efficient ML pipeline task scheduling. Moreover, the incorporation more sophisticated federated learning mechanisms and neural network layer splitting strategies across multiple cluster nodes aiming time-critical scenarios would be considered as well.

Acknowledgment

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

 ISBN 978-86-7466-930-3

REFERENCES

[1] G. Symeonidis, E. Nerantzis, A. Kazakis and G. A. Papakostas, "MLOps - Definitions, Tools and Challenges," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), pp. 453-460, 2022. https://doi.org/10.1109/CCWC54503.2022.9720902

[2] S. Moreschini, F. Lomio, D. Hästbacka, D. Taibi, "MLOps for evolvable AI intensive software systems", IEEE International Conference on Software Analysis, Evolution and Reengineering 2022, pp. 1-2, 2022.

[3] D. Kreuzberger, N. Kühl, S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture", preprint, 2022.

[4] D. A. Tamburri, "Sustainable MLOps: Trends and Challenges", 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 17-23, 2020. https://doi.org/10.1109/SYNASC51798.2020.00015

[5] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice, 2nd Edition, Morgan & Claypool Publishers, 2017.

[6] N. Petrovic, M. Tosic, "SMADA-Fog: Semantic model driven approach to deployment and adaptivity in Fog Computing", Simulation Modelling Practice and Theory, 102033, pp. 1-25, 2019. https://doi.org/10.1016/j.simpat.2019.102033

[7] D. Krstić, N. Petrović, I. Al-Azzoni, "Model-Driven Approach to Fading-Aware Wireless Network Planning Leveraging Multiobjective Optimization and Deep Learning", Mathematical Problems in Engineering, vol. 2022, 4140522, Special Issue: Mathematical Modelling of Data Transmission in Next Generation Wireless Systems, 2022, pp. 1-23, 2022. https://doi.org/10.1155/2022/4140522

[8] ZenML [online]. Available on: https://zenml.io/, last accessed: 08/05/2022.

[9] Kuberentes [online]. Available on: https://kubernetes.io/, last accessed: 08/05/2022.

[10] Eclipse Modeling Framework [online]. Available on: https://www.eclipse.org/modeling/emf/, last accessed: 08/05/2022.

[11] Ecore [online]. Available on: https://wiki.eclipse.org/Ecore, last accessed: 08/05/2022.

[12] E. Stevens, L. Antiga, T. Viehmann, *Deep Learning with PyTorch*, Manning Publications, 2020

[13] The MNIST database of handwritten digits [online]. Available on: http://yann.lecun.com/exdb/mnist/ , last accessed: 08/05/2022.

[14] Solidity [online]. Available on: https://docs.soliditylang.org/en/v0.8.13/, last accessed: 08/05/2022.

[15] M. Radenković, V. Nejković, N. Petrović, "Adopting AR and Deep Learning for Gamified Fitness Mobile Apps: Yoga Trainer Case Study", AIIT 2021 International conference on Applied Internet and Information Technologies, pp. 167-171, 2021.

[16] K3d [online]. Available on: https://k3d.io/v5.4.1/ , last accessed: 08/05/2022.

[17] I. Al-Azzoni, J. Blank, N. Petrović, "A Model-Driven Approach for Solving the Software Component Allocation Problem", Algorithms 2021; 14(12):354, pp. 1-19, 2021. https://doi.org/10.3390/a14120354

 ISBN 978-86-7466-930-3

Controllability of the multi-agent system modeled by the chain graphs with repeated degree

Milica Anđelić
Department of Mathematics,
Kuwait University,
Safat 13060, Kuwait
milica.andelic@ku.edu.kw

Edin Dolićanin
Department of Technical Sciences
State University of Novi Pazar
Novi Pazar, Serbia
edin@np.ac.rs

Zoran Stanić
Faculty of Mathematics,
University of Belgrade,
Serbia
zstanic@matf.bg.ac.rs

***Abstract*—We consider the controllability of multi-agent dynamical systems modeled by a special class of bipartite graphs, called chain graphs. Our particular attention is focused on chain graphs that have one repeated degree. We derive properties of eigenvectors of graphs under consideration as well as some of their Laplacian spectra. On the basis of the obtained theoretical results, we determine the minimum number of leading agents that make the system in question controllable and locate them in the corresponding graph.**

***Index Terms*—Chain graph, Laplacian spectrum, Eigenvectors, Controllable dynamical system**

I. Introduction

Let $G = (V(G), E(G))$ be a simple graph (without loops or multiple edges) of *order* $n = |V(G)|$. By $A(G)$ we denote its $(0,1)$-*adjacency matrix*. If $D(G)$ is the diagonal matrix of vertex degrees, then $L(G) = D(G) - A(G)$ stands for the *Laplacian matrix* of G. The *Laplacian eigenvalues* of G are the eigenvalues of $L(G)$ and they form $\sigma(G)$, the *Laplacian spectrum* of G.

We consider a multi-agent system with n linear agents $\{1, 2, \ldots, n\}$ modeled by a graph G. If x_i denotes the state of the agent i, its dynamics is described by the single integrator

$$\dot{\mathbf{x}}(t) = -\sum_{j \in N(i)} (x_i(t) - x_j(t)),$$

where $N(i)$ denotes the set of neighbours of i. The compact dynamics can be written as $\dot{\mathbf{x}}(t) = -L(G)\mathbf{x}(t)$, where $\mathbf{x}$ is the vector of the agents' states and $L(G)$ is the graph Laplacian.

Following [6] by ℓ and f we denote affiliations with leaders and followers. A *follower graph* G_f of G is the subgraph induced by the set of followers. Consequently, the graph Laplacian $L(G)$ of G may be written as

$$L(G) = \begin{pmatrix} \mathcal{L}_f(G) & l_{f\ell}(G) \\ l_{f\ell}^{\mathsf{T}}(G) & \mathcal{L}_\ell(G) \end{pmatrix}. \tag{I.1}$$

The control system we consider is the leader-follower system

$$\begin{pmatrix} \dot{\mathbf{x}}_f(t) \\ \dot{\mathbf{u}}(t) \end{pmatrix} = -\begin{pmatrix} \mathcal{L}_f(G) & l_{f\ell}(G) \\ l_{f\ell}^{\mathsf{T}}(G) & \mathcal{L}_\ell(G) \end{pmatrix} \begin{pmatrix} \mathbf{x}_f(t) \\ \mathbf{u}(t) \end{pmatrix},$$

where followers evolve through the Laplacian-based dynamics

$$\dot{\mathbf{x}}_f(t) = -\mathcal{L}_f(G)\mathbf{x}_f(t) - l_{f\ell}(G)\mathbf{u}(t), \tag{I.2}$$

and $\mathbf{u}$ denotes the external control signal ran by the leaders' states.

The system modeled by (I.2) is said to be *controllable* if it can be driven from any initial state to any desired final state in a finite time. In the study of the controllability of multi-agent systems, the main problem is to determine the locations of leaders under which the controllability can be realized. The multi-agent system (I.2) is said to be k-*leaders controllable* if there exist minimum number of k leaders to make (I.2) controllable. In particular, if $k = 1$, the system (I.2) is called *single leader controllable*.

We recall a useful argument for further analysis of controllability of multi-agent systems.

Lemma I.1. ([5]) *The system* (I.2) *is controllable if and only if there is no eigenvector for $L(G)$ taking 0 on all entries corresponding to leaders, i.e. if and only if $L(G)$ and $\mathcal{L}_f(G)$ do not share any common eigenvalues.*

Multi-agent systems arise in many areas of science and engineering (see for example [1], [5], [7], [8], [10], [12]). In this paper we focus on controllability of chain graphs, in particular to chain graphs with one repeated degree. Chain graphs are $2K_2, C_3, C_5$ graphs, which implies that they are also bipartite graphs. We determine the minimum number of leaders needed to make the corresponding system (I.2) modeled by such a graph controllable and provide the locations of leaders in the graph.

The paper is organized as follows. In Section II we give some preliminary results on the structure of chain graphs and on their spectrum. In Section III we present several results concerning Laplacian spectrum and eigenvectors of chain graphs with one repeated degree. In Section IV we consider the controllability of systems (I.2) modeled by a corresponding chain graph. In Section V we present several concluding remarks.

 ISBN 978-86-7466-930-3

II. PRELIMINARIES

The Laplacian matrix $L(G)$ of any graph G is symmetric and positive semidefinite. Moreover, 0 is an eigenvalue of G afforded by the all-1 vector $\mathbf{j}$. Therefore, we may assume that the eigenvalues of G (in fact, the roots of the characteristic polynomial $\phi(L(G), x) = \det(xI - L(G))$) are indexed in non-increasing order and given as follows:

$$\mu_1(G) \geq \mu_2(G) \geq \cdots \geq \mu_n(G) = 0\,.$$

We denote by $\sigma(G)$ the spectrum of G, i.e. the multiset of its eigenvalues.

The vertex set of a chain graph G consists of two colour classes that are partitioned into h non-empty cells $\bigcup_{i=1}^{h} U_i$ and $\bigcup_{i=1}^{h} V_i$, respectively. All vertices in U_s are joined to all vertices in $\bigcup_{k=1}^{h+1-s} V_k$, for $1 \leq s \leq h$. Therefore, all vertices in U_i (resp. V_j) are *co-neighbours*, i.e. they share the same set of neighbours. If $m_s = |U_s|$ and $n_s = |V_s|$, for $1 \leq s \leq h$, then G is denoted by

$$\mathrm{DNG}(m_1, m_2, \ldots, m_h; n_1, n_2, \ldots, n_h)\,.$$

A chain graph is sketched in Figure II.1.

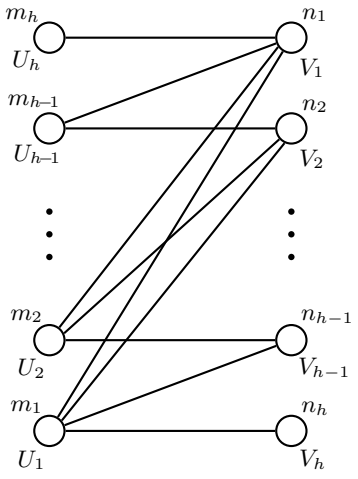


Fig. II.1. The chain graph $G = \mathrm{DNG}(m_1, m_2, \ldots, m_h; n_1, n_2, \ldots, n_h)$.

If we follow the vertex ordering from the partition $\left(\bigcup_{i=1}^{h} U_i\right) \cup \left(\bigcup_{i=1}^{h} V_i\right)$, then the *quotient matrix* $Q(G)$ of a chain graph G has the form

$$\left(\begin{array}{cccc|cccc} d_1 & & & & -n_1 & \cdots & -n_{h-1} & -n_h \\ & d_2 & & & -n_1 & \cdots & -n_{h-1} & \\ & & \ddots & & \vdots & \iddots & & \\ & & & d_h & -n_1 & & & \\ \hline -m_1 & \cdots & -m_{h-1} & -m_h & d_1^* & & & \\ -m_1 & \cdots & -m_{h-1} & & & d_2^* & & \\ \vdots & \iddots & & & & & \ddots & \\ -m_1 & & & & & & & d_h^* \end{array}\right). \tag{II.1}$$

The corresponding diagonal blocks we shortly denote by D_1, D_2, while off-diagonal ones we denote by B_1, B_2.

It is well-known that every eigenvalue of $Q(G)$ is an eigenvalue of G. For more results on spectral properties of chain graphs the reader is referred to [4], [9], [11].

III. LAPLACIAN SPECTRUM OF $\mathrm{DNG}(k, 1, \ldots, 1; 1, \ldots, 1)$

In this section we investigate spectral properties of chain graphs with one repeated degree. These graphs are of the form $\mathrm{DNG}(\underbrace{k, 1, \ldots, 1}_{h}; \underbrace{1, \ldots, 1}_{h})$. Since G has only one repeated degree, then $k > h$.

Theorem III.1. *Let* $\mathrm{DNG}(\underbrace{k, 1, \ldots, 1}_{h}; \underbrace{1, \ldots, 1}_{h})$, $k > h$. *Then*

$$\sigma(G) = \{0, h^{k-1}, \kappa_1, \kappa_2, \ldots, \kappa_{2h-1}\},$$

where

$$\begin{cases} \kappa_i \in (i-1, i), & i \in \{1, \ldots, h-1\} \\ \kappa_{h+i} \in (k+i-1, k+i), & i \in \{1, \ldots, h-1\} \\ \kappa_{2h} \geq k+h, \end{cases}$$

Proof. Taking into account that $d_i = h + 1 - i$, $1 \leq i \leq h$ and $d_j^* = k + h - j$, $1 \leq j \leq h$ and employing [13, Theorem 3.5], we get that the characteristic polynomial $\phi(L(G), x)$ of $L(G)$ is given by

$$x(x-h)^{k-1} \prod_{i=1}^{k+h-1} (x-i) \left(\frac{1}{p_1} + x \sum_{j=2}^{h} \frac{1}{(x - d_{h+2-j})p_j} + \frac{1}{x - d_1} \right).$$

Since $x(x-h)^{k-1}$ is a factor of $\phi(L(G), x)$, the remaining eigenvalues are the roots of the polynomial

$$p(x) = \prod_{i=1}^{k+h-1} (x-i) \left(\frac{1}{p_1} + x \sum_{j=2}^{h} \frac{1}{(x - d_{h+2-j})p_j} + \frac{1}{x - d_1} \right).$$

Then we have:

- $p(0) = (-1)^{k+h}(2h + k - 1)\frac{(k+h-2)!}{h}$;
- $p(1) = (-1)^{k+h+1}(k + h - 3)(k + h - 3)!$;
- $p(\ell) = (-1)^{k+h+\ell} 2\ell(h+1)(\ell-1)!(k + h - 2\ell - 2)! \frac{(k+h-\ell-1)!}{(k+h-2\ell)!}$, for $2 \leq \ell \leq h-1$;
- $p(k) = (-1)^{h+1} \frac{k!}{(k-h+1)(k-h)}(h-1)!$
- $p(k+\ell) = (-1)^{h-\ell+1} 2(\ell+1) \frac{(k+\ell)!}{(k+2\ell-h+1)(k+2\ell-h)}(h - \ell - 1)!$, for $1 \leq \ell \leq h-1$;
- $p(k+h) = -(k+h-2) \cdot (h-1)! < 0$.

From the obtained values, we conclude that $p(0), p(1), \ldots, p(h-1)$ alternate in sign. Therefore, for any $i \in \{1, 2, \ldots, h-1\}$, we have $p(t) = 0$, for some $t \in (i-1, i)$. Similar argument holds for $p(k), p(k+1), \ldots, p(k+h-1)$, and consequently, for every $i \in \{k+1, \ldots, k+h-1\}$ we have $p(t) = 0$ for some $t \in (i-1, i)$. Also, since p is a monic polynomial and $p(k+h) < 0$, it follows that $p(t) = 0$ holds for some $t > k + h$. □

 ISBN 978-86-7466-930-3

We illustrate the results of Theorem III.1 on the following example.

Example III.2. Let $G = \mathrm{DNG}(6,1,1,1,1;1,1,1,1,1)$. Then

$$\sigma(G) = \{13.03, 9.64, , 08.6, 7.58, 6.58, 3.82, 2.86, 1.92, 0.96\} \cup \{5^5, 0\}.$$

Next we observe the structure of eigenvectors of $L(G)$ corresponding to non-integer eigenvalues.

Theorem III.3. *Let $G = \mathrm{DNG}(k,1,\ldots,1;1,1,\ldots,1)$ with $k > h$, μ a non-integer eigenvalue of G and $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\intercal$ an associated eigenvector. Then $x_i \neq 0$ for any $1 \leq i \leq k$.*

Proof. We recall first, (see, for example, [13]) a relation between the eigenvectors of $Q(G)$ and those of G for the same eigenvalue. A vector $\mathbf{v} = (y_1, y_2, \ldots, y_h, z_1, z_2, \ldots, z_h)^\intercal$ is an eigenvector of $Q(G)$ for μ, if and only if the corresponding eigenvector of G for the same eigenvalue has the form

$$\mathbf{x} = (\underbrace{y_1, y_1, \ldots, y_1}_{k}, y_2, \ldots, y_h, z_1, \ldots, z_h)^\intercal.$$

Assume on the contrary that $\mathbf{x}$ is an eigenvector for the non-integer eigenvalue μ of $L(G)$ such that $x_i = 0, 1 \leq i \leq k$. By [13, Lemma 3.4], μ is also an eigenvalue of $Q(G)$. So there exists a non-zero vector $(\mathbf{y} \quad \mathbf{z})^\intercal \in \mathbb{R}^{2h}$ such that $Q(G)(\mathbf{y} \quad \mathbf{z})^\intercal = \mu(\mathbf{y} \quad \mathbf{z})^\intercal$ with $y_1 = 0$. Then the eigenvalue equation

$$\begin{pmatrix} D_1 & -B_1 \\ -B_2 & D_2 \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \mu \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix}$$

can be rewritten as

$$D_1\mathbf{y} - B_1\mathbf{z} = \mu\mathbf{y}$$
$$-B_2\mathbf{y} + D_2\mathbf{z} = \mu\mathbf{z}.$$

The matrices B_1, B_2 have full rank, and therefore are invertible. Next, from

$$\mathbf{z} = B_1^{-1}(D_1 - \mu I_h)\mathbf{y}$$
$$\mathbf{y} = B_2^{-1}(D_2 - \mu I_h)\mathbf{z},$$

we conclude that

$$\mathbf{y} = B_2^{-1}(D_2 - \mu I_h)B_1^{-1}(D_1 - \mu I_h)\mathbf{y},$$

i.e. $\mathbf{y}$ is an eigenvector of

$$P = B_2^{-1}(D_2 - \mu I_h)B_1^{-1}(D_1 - \mu I_h)$$

for the eigenvalue 1. The latter product is the product of two anti-bidiagonal matrices $B_2^{-1}(D_2 - \mu I_h)$ that is

$$\begin{pmatrix} & & & & (k-\mu)/k \\ & & & & -(k-\mu) \\ & & \iddots & \iddots & \\ & & & & \\ & & (k+h-2-\mu) & & \\ (k+h-1-\mu) & & -(k+h-2-\mu) & & \end{pmatrix}$$

and $B_1^{-1}(D_1 - \mu I_h)$

$$\begin{pmatrix} & & & & (1-\mu) \\ & & & (2-\mu) & -(1-\mu) \\ & & \iddots & \iddots & \\ & & & & \\ & & (h-1-\mu) & & \\ (h-\mu) & & -(h-1-\mu) & & \end{pmatrix},$$

and hence it is a tridiagonal matrix with

$$p_{1,1} = \frac{(h-\mu)(k-\mu)}{k}$$
$$p_{\ell,\ell} = (h+1-\ell-\mu)\Big(\frac{k+\ell-1-\mu}{m_{h+1-\ell}} + (k+\ell-2-\mu)\Big),$$

$2 \leq \ell \leq h$,

$$p_{\ell,\ell-1} = -(h-\ell+2-\mu)(k+\ell-2-\mu), \quad 2 \leq \ell \leq h,$$
$$p_{\ell,\ell+1} = -\frac{(h-\ell-\mu)(k+\ell-1-\mu)}{m_\ell}, \quad 1 \leq \ell \leq h-1,$$

taking into account that $m_1 = k$ and $m_i = 1$, $i \geq 2$. From $\mu \notin \mathbb{Z}$, we have $p_{\ell,\ell-1}, p_{\ell,\ell+1} \neq 0$.

If $y_1 = 0$, then from the first equation in $P\mathbf{y} = \mathbf{y}$ we obtain $y_2 = 0$ ($p_{1,2} \neq 0$). Next, in the similar way, the second equation gives $y_3 = 0$, and so on, until we obtain $y_h = 0$, i.e. $\mathbf{y} = \mathbf{z} = \mathbf{0}$.

Therefore, we obtain that $\mathbf{x} = \mathbf{0}$, which is a contradiction. This completes the proof. □

IV. Controllability of systems modeled by $\mathrm{DNG}(k,1,\ldots,1;1,\ldots,1)$

Previously obtained results, in this section will be employed to determine the number of leading agents in (I.2), where the system is modeled by a chain graph $\mathrm{DNG}(k,1,\ldots,1;1,\ldots,1)$ for $k > h$.

Theorem IV.1. *Let G be a chain graph $\mathrm{DNG}(k,1,\ldots,1;1,\ldots,1)$ with $k > h$. Then the system (I.2) modeled by G is controllable with $k-1$ co-neighbour vertices in the role of leaders.*

 ISBN 978-86-7466-930-3

Proof. The eigenvectors corresponding to the eigenvalue h of the multiplicity $k-1$ are of the form

$$\begin{aligned}
\mathbf{v}_1 &= (\underbrace{1,-1,0,0,\ldots,0}_{k},0,\ldots,0)\\
\mathbf{v}_2 &= (\underbrace{1,0,-1,0,\ldots,0}_{k},0,\ldots,0)\\
&\vdots\\
\mathbf{v}_{k-1} &= (\underbrace{1,0,\ldots,0,0,-1}_{k},0,\ldots,0).
\end{aligned}$$

We first conclude that vertices $\{2,\ldots,k\}$ should be selected as leaders. Moreover, any vector corresponding to $\mu = h$ is of the form $(t_1,\ldots,t_k,0,\ldots,0)^{\intercal}$. Any $k-1$ of these t_i's cannot be zeros simultaneously. For any $\mathbf{x}_1,\ldots,\mathbf{x}_l$, $l < k-1$ there exists $\mathbf{x}_s$, such that $\mathbf{x}_s\mathbf{x}_i = 0$ for each $i, 1 \leq i \leq l$.

The remaining eigenvalues by Theorem III.1 are non-integer and therefore their eigenvectors, by Theorem III.3 satisfy $x_i \neq 0, 1 \leq i \leq k$. Now the statement follows by Lemma I.1. □

Example IV.2. For $G = \mathrm{DNG}(6,1,1,1,1;1,1,1,1,1)$ the system (I.2) is 5 leader controllable. The leaders $\ell_1,\ldots,\ell_5 \in U_1$ are 5 of 6 vertices with repeated degrees, that are joined to the followers $v_1,\cdots,v_5$ as illustrated in Figure IV.1.

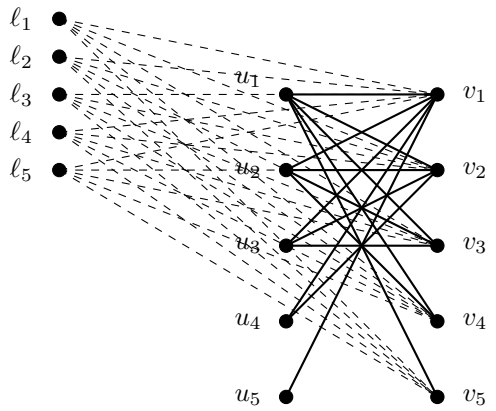


Fig. IV.1. A 5-leader controllable system modelled by $\mathrm{DNG}(6,1,1,1,1;1,1,1,1,1)$.

V. Conclusion

In this paper we have covered the controllability of multi-agent systems that are modelled by special class of bipartite graphs: chain graphs. We have proved that if a chain graph has only one repeated degree with multiplicity k, then the system requires at least $k-1$ controllers in order to be controllable. In this way we positively addressed the questions raised in [7], where the authors asked if there is a family of graphs other than threshold graphs with one multiple degree of multiplicity m for whose controllability at least $m-1$ controllers are needed. Consequently, we expanded the known classes of the controllable multi-agent systems. Taking into account that many engineering systems are modelled by graphs, the obtained results are of particular importance in creating new controllable systems, since the known structures are limited (they mainly include paths, grids, cycles and circulant networks). Another advantageous aspect is a possibility to generate graphs with some desirable properties. One of them is algebraic connectivity, i.e. the second smallest Laplacian eigenvalue. It is a useful tool to measure the robustness and synchronizations of the graphs. For the chain graphs that we considered the algebraic connectivity is always in $(0,1)$ and it approaching to 1 as the size of the graph increases. This brings another benefit, since in general graphs the algebraic connectivity usually decreases if the order of a graph is increased.

Acknowledgment

Research of M.A and Z.S. is supported by the Science Fund of the Republic of Serbia; grant number 7749676: Spectrally Constrained Signed Graphs with Applications in Coding Theory and Control Theory – SCSG-ctct.

References

[1] T. Kailath, Linear systems, Prentice-Hall, Englewood Cliffs, 1980.
[2] N.V.R. Mahadev, U.N. Peled, it Threshold Graphs and Related Topics, North-Holland, New York, 1995.
[3] K. Ogata, Modern Control Engineering, Prentice-Hall, Upper Saddle River, 2002.
[4] F.K. Bell, D. Cvetković, P. Rowlinson, S.K. Simić, Graphs for which the least eigenvalue is minimal, II, Linear Algebra Appl., 429 (2008), 2168–2179.
[5] Z. Ji, Z. Wang, H. Lin, Z. Wang, Interconnection topologies for multi-agent coordination under leader-follower framework, Automatica, (2009), 45(12), 2857–2863.
[6] A. Rahmani, M. Ji, M. Mesbahi, M. Egerstedt, Controllability of multi-agent systems from a graph theoretic perspective, SIAM J. Control Optim., 48 (2009), 162–186.
[7] S-P Hsu, Controllability of the multi-agent system modelled by the threshold graph with one repeated degree, Systems Control Lett., 97 (2016), 149–156.
[8] X. Liu, Z. Ji, Controllability of multiagent systems based on path and cycle graphs, Int. J. Robust Nonlinear Control, 28 (2016), 296–309.
[9] M. Anđelić, S.K. Simić, D. Živković, E. Dolićanin, Fast algorithms for computing the characteristic polynomial of threshold and chain graphs, Appl. Math. Comput. 332 (2018), 329-337.
[10] M. Anđelić, M. Brunetti, Z. Stanić, Laplacian controllability for graphs obtained by some standard products, Graphs Combin., 36 (2020), 1593–1602.
[11] K.Ch. Das, A. Alazemi, M. Anđelić, On energy and Laplacian energy of chain graphs, Discrete Appl. Math., 284 (2020), 391–400.
[12] A. Farrugia, T. Koledin, Z. Stanić, Controllability of NEPSes of graphs, Linear Multilinear Algebra, https://doi.org/10.1080/03081087.2020.1778622
[13] A. Alazemi, M. Anđelić, K.Ch. Das, C.M. da Fonseca, Chain graph sequences and Laplacian spectra of chain graphs, Linear Multilinear Algebra, https://doi.org/10.1080/03081087.2022.2036672

Secret Keys Distillation using Speech Signals and Discussion over Public Authenticated Channel

Jelica Radomirović, Milan Milosavljević, and Aleksandra Krstić

***Abstract*— This paper discusses a system for generating and distributing secret cryptographic keys based on the principle of common randomness and discussion over an authenticated public channel. The use of speech as a source of common randomness is one possibility and to our knowledge the first for this type of system. We consider different reconciliation algorithms and compare them with experiments. Experimental results show that it is possible to generate information-theoretical secret keys with rates between 3 and 5%. This result proves the practical feasibility of absolutely secret autonomous cipher systems with speech control.**

***Index Terms*—secret key; distillation; symmetric cryptography; speech signal; reconciliation; privacy amplification; secret key rate;**

I. Introduction

The information-theoretical approach to the analysis and synthesis of cipher systems came into focus with the availability of quantum computers in the near future. The classical result of this approach states that the entropy of secret keys in a cryptographic system must be no less than the entropy of plaintext [1]. As is well known, systems designed in this way are resistant to the unlimited computing resources of the adversary, and thus to cryptanalysis based on quantum computers [2].

From the point of view of generating and distributing high-quality secret keys, special attention is drawn to the fundamental results of Alswede and Csiszar [3], Maurer [4], and Csiszar and Narayan [5]. The basic idea of information-theoretical approach in these results is to identify and use mutually correlated signals available to legitimate parties.

Depending on the location of the source of common randomness, there are two approaches, [3]:

(i) Secret key extraction from sources independent of communication channels (source model),

(ii) Secret key extraction from existing communication channels (channel model).

The difference between these two models is how the parties observe the initial sequence. While in the source model, random source is controlled by nature, in the channel model, one of the parties governs the input of a noisy channel (independent of the main channel) while others observe the output.

In this paper, we will analyze the possibility of extracting cryptographic keys from a speech signal, applying an approach based on the source model.

In Section 2, the basic blocks of the proposed secret key generation system will be presented, in two variants: (i) when the input is a speech signal and (ii) when the input is a residual speech signal, filtered by an adaptive linear predictive model [6].

In Section 3, the information and statistical characteristics of this source will be analyzed and the key parameters of the sequential procedure for extracting secret keys will be identified, separately for each of the phases: Advantage Distillation (AD), Information Reconciliation (IR) and Privacy Amplification. -PA).

In Section 4 we present the results of the experiment of obtaining secret keys for all pairs (Alice, Bob) of legitimate participants for 5 speakers, one of which was chosen as an eavesdropper (Eve).

The Conclusion discusses the upper limits of the rate of generating secret keys and the possibility of further improving the performance of the proposed system.

II. Discrete Memoryless Source

As illustrated in Figure 1, a source model for secret-key agreement represents a situation in which three parties, Alice, Bob, and Eve, observe the realizations of a DMS - Discrete Memoryless Source (XYZ, P_{XYZ}) with three components.

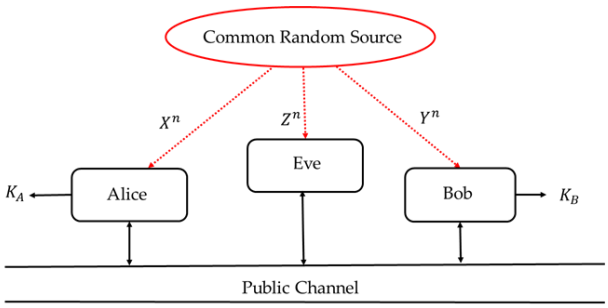


Fig. 1. Secret-key Agreement by Public Discussion from Common Randomness [4].

Jelica Radomirović is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia and with Vlatacom Institute of High Technologies, 5 Bulevar Milutina Milankovića, 11070 Belgrade, Serbia (e-mail: jelica.radomirovic@vlatacom.com).

Milan Milosavljević is with Singidunum University, 32 Danijelova, 11000 Belgrade, Serbia (e-mail: mmilosavljevic@singidunum.ac.rs).

Aleksandra Krstić is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: amarjanovic@etf.bg.ac.rs).

 ISBN 978-86-7466-930-3

The DMS is assumed to be outside the control of all parties, but its statistics are known. By convention, component X is observed by Alice, component Y by Bob, and component Z by Eve. Alice and Bob's objective is to process their observations and agree on a key K about which Eve should have no information.

Alice and Bob can exchange messages over a noiseless, two-way, public and authenticated channel. That is, all messages are overheard by Eve and the existence of the public channel does not provide Alice and Bob with an explicit advantage over Eve. The rules by which Alice and Bob compute the messages they exchange over the public channel and agree on a key define a four-stage key distillation strategy, [4]:

1. Randomness sharing. Alice, Bob, and Eve observe *n* realizations of a DMS (XYZ, P_{XYZ}).
2. Advantage distillation. If needed, Alice and Bob exchange messages over the public channel to process their observations and to "distill" observations for which they have an advantage over Eve.
3. Information reconciliation. Alice and Bob exchange messages over the public channel to process their observations and agree on a common bit sequence.
4. Privacy amplification. Alice and Bob publicly agree on a deterministic function they apply to their common sequence to generate a secret key.

The largest achievable key rate is defined as the key capacity and is given by

$$C_K = max\{I(X;Y), I(X:Y|Z)\}, \quad (1)$$

where I(X;Y) denotes mutual information between X and Y, while I(X:Y|Z) denotes the same quantity conditioned by Z. In the special case, when Eva is totally independent of Alice and Bob, or equivalently, when Z is independent of X and Y, maximal key capacity is equal to

$$C_{K\,max} = I(X;Y). \quad (2)$$

In this work, we use speech the signals of participants as DMS of the proposed system, see Fig.2.

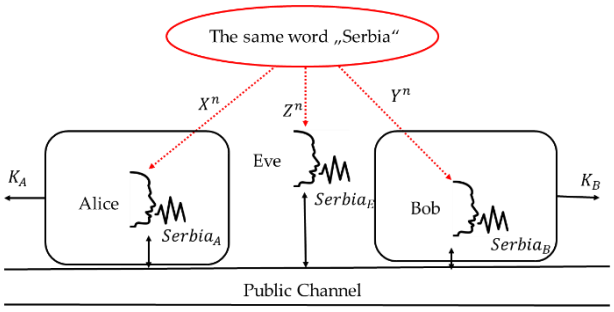


Fig. 2. Secret-key Agreement by Public Discussion based on the speech signals obtained by pronouncing the word "Serbia".

III. System Architecture

As already mentioned, we will analyze two DMS, the first one corresponding to the original speech signal, and the second one corresponding to the residual signal. Residual DMS is obtained after inverse filtering by an adaptive linear autoregressive model, estimated every 10 ms of input speech, see Fig.3.

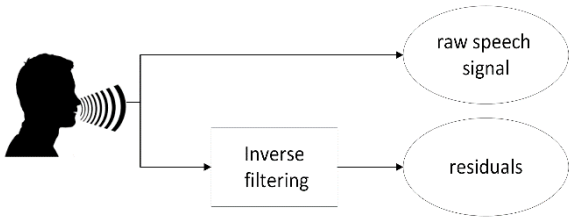


Fig. 3. Two different DMS based on the same input speech signal

The general architecture of the system is given in Fig.4. Speech input (or residual) is transformed into binary DMS by a non-uniform quantization, based on estimating the probability density function of input samples.

Advantage Distillation (AD) blocks are used to eliminate the advantage that the eavesdropper may have over legitimate parties. In that case, Eve knows more information about Alice's initial bit string, than Bob does. We will use the bit pair advantage distillation/degeneration protocol [7]. Algorithms are based on the exchange of parity information of 2-bit blocks and the elimination of one bit of each block to ensure security. After this step, the number of different bits of the sequence is reduced. The number of non-matching bits of the eavesdropper sequence decreases but much slower than for the legitimate parties. The protocol runs for several rounds until the sequences differ only in a few bits.

The Information Reconciliation (IR) is intended to correct the remaining erroneous bits in the Alice and Bob sequences. The most popular IR protocols are Winnow [7] and Cascade [8] protocols. They are based on the exchange of block parity information until the error of a certain block is detected. For each block parity query, some bits are deleted to ensure security. In the end, we get the same sequence on legitimate sides that represents the secret key. Even though we tried to ensure privacy by deleting potentially compromised bits, the eavesdropper has still gained some information about the secret key. To make the secret key absolutely secure, we proceed to the next step of our system, privacy amplification (PA).

In Privacy Amplification (PA) block sequences are transformed such that m bits are discarded due to the eavesdropper's knowledge. One of the possible transformations is a hash function $g: \{0,1\}^n \rightarrow \{0,1\}^r$ where n is the length of a sequence before PA and r is the length after PA, i.e., the length of the final secret key. It is common to use so-called universal hash functions, such as random *r x n* matrices, over GF(2), [9].

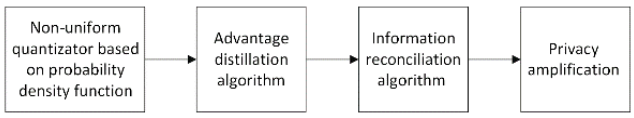


Fig. 4. Secret key agreement algorithm [3]

 ISBN 978-86-7466-930-3

IV. Results

Raw speech signal and its residuals, obtained from inverse filtration of AR model of 10th order, were used for experimental proof of the proposed system, Fig 5. Four participants recorded the word 'Srbija' for two seconds. Beginning. as well as the end of the word, were determined so the initial signal is reduced to 0.6 second length. The recordings were sampled at 44.1 kHz.

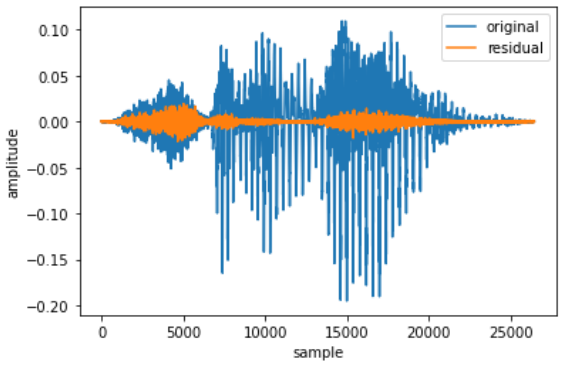


Fig. 5. Comparison of source signals

Normalized Hamming distance is an appropriate metric for measuring the difference between two binary sequences. Depending on the number of bits we use to quantize continuous signal we get more or less similar sequences. That directly affects how much information circulates over a public channel and how long is the final secret key length. The key rate is an indicator of how much of the sequence at the beginning is useful

$$\text{key rate} = \frac{\text{final length of secret key}}{\text{length of input sequence}} * 100\ [\%].$$

In Fig 6. the key rate in function of normalized Hamming distance is presented. In order to determine the optimal value for the number of quantization bits, we try five different values 6,8,10,12, and 14.

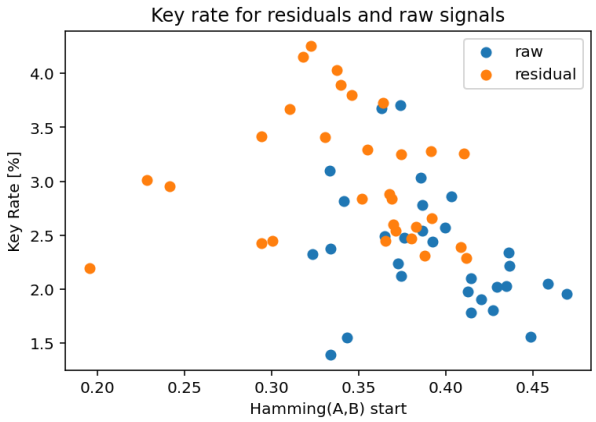


Fig. 6. Comparison of key rate for source signals. A and B denote legitimate parties

From Fig.6 can be seen that residuals are closer to each other than raw signals because of the corresponding higher key rate.

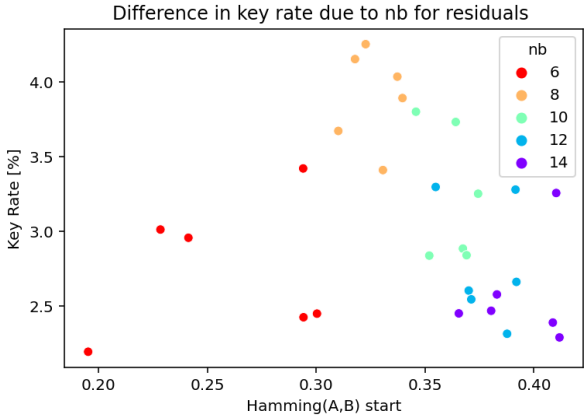


Fig. 7. How the number of quantization bits affects distance at the beginning

As shown in Fig 7., the highest rate is achieved for normalized Hamming distance between 0.3 and 0.35, which corresponds to 8 quantization bits.

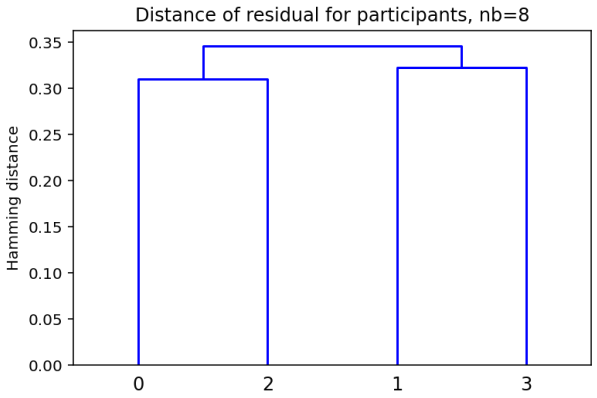


Fig. 8. Dendrogram represented distance of all sequences

In Fig 8. we use a dendrogram to show how close to each other are the participant sequences. The dendrogram was obtained as a result of hierarchical cluster analysis by the Ward method [10]. To compare Cascade and Winnow algorithms we conduct 4 experiments, 2 for each, that is one for residual signals and one for speech signals. Results are represented in table 1.

TABLE I
Experimental results

		Final key size	Key rate [%]
Winnow	Residual	10116.83 ± 788.93	4.78 ± 0.37
	Raw	8670.30 ± 1033.87	4.10 ± 0.49
Cascade	Residual	8432.24 ± 624.18	3.98 ± 0.29
	Raw	6833.72 ± 1433.26	3.22 ± 0.68

 ISBN 978-86-7466-930-3

Based on the results we conclude that the Winnow algorithm is a better choice for reconciliation because it gives an almost 1% higher key rate. If we compare two DMS, the one corresponding to the residual signal gives longer secret keys due to a smaller normalized Hamming distance at the beginning, for the same pair of sequences. Final hamming for all sequences and all experiments toward the eavesdropper after PA is ~0.5. In other words, all the information that leaked through the public channel will not reveal anything to the eavesdropper about the distilled secret key.

V. Conclusion

In this work we proposed a speech based secret key agreement system with message transmission over a public channel. The proposed system can distill secret keys from speech signals, with the key rate of up to 5%, and with negligible information leakage to an eavesdropper. This opens up the possibility of practical realization of absolutely secret cipher systems controlled by voice. Such systems can be used both in the security services for critical information and communication infrastructure of the government, as well as in commercial applications.

Future work will include generalization in terms of the largest achievable key rate and a testing of the proposed system on more participants as well as more different spoken words.

VI. Acknowledgment

The authors would like to thank the Vlatacom Institute of High Technologies, where the research was done as part of the project Prj_164.

References

[1] Shannon C.E., “Communication theory of secrecy systems”. *BSTJ*, vol. 28, no. *4*, pp. *656–715*. October 1949.

[2] Wolf S., “Unconditional Security in Cryptography”, in *Lectures on Data Security: Modern Cryptology in Theory and Practice*, *Lecture Notes in Computer Science*, Berlin, 1999, vol. 1561, pp. *217–250*.

[3] Ahlswede R., Csiszar I., “Common randomness in information theory and cryptography, Part I: Secret sharing”, *IEEE Transaction on Information Theory*, vol. 39, pp. *1121–1132*, 1993.

[4] Maurer U., “Secret Key Agreement by Public Discussion from Common Information”, *IEEE Transaction on Information Theory*, vol. 39, no. *3*, May, 1993.

[5] Csiszar I., Narayan P., “Secrecy capacities for multiple terminals”, *IEEE Transaction on Information Theory*, vol. 50, pp. *3047–3061*, 2004.

[6] Kovačević B., Milosavljević M., Veinović M., “Robust Digital Processing of Speech Signals“, Springer, 2017.

[7] Wang Q., Wang X., Lv Q., Ye X., Luo Y., You L., “Analysis of the information theoretically secret key agreement by public discussion”, *Security and Communication Networks*, vol. 8, January, 2015.

[8] Reis A., “Quantum Key Distribution Post Processing - A Study on the Information Reconciliation Cascade Protocol”. Master’s Thesis, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal, 2019.

[9] Bennett C.H., Brassard G., Crepeau C., Maurer U. “Generalized privacy amplification”. *IEEE Transaction on Information Theory*, vol. 41, pp. 1915–1923, 1995.

[10] sklearn.cluster.Ward – scikit-learn 0.15-git documentation. https://scikit-learn.org/0.15/modules/generated/sklearn.cluster.Ward.html

 ISBN 978-86-7466-930-3

One solution for voice commands on Android based STB

Jovana Simić, Đorđe Glišić, Nikola Vranić, Marija Jovanović

***Abstract*—With the introduction of voice recognition and its support by various research groups, it became possible to add voice commands to different devices in use. We are seeing them on PCs, phones, and tablets. This paper presents a solution to order and control set-top box devices and TVs based on Android OS. It is a cloud-based solution supported by Google API for voice recognition.**

***Index Terms*—set-top box, android, voice commands, voice recognition, TV voice commands, Actions on Google, Android, Google Assistant, Dialogflow, Firebase.**

I. Introduction

Virtual assistance is becoming more popular with artificial intelligence and machine learning advances. Particularly with voice recognition available on PC and even mobile devices, with capabilities to support more languages, not just English. Popular operating systems come with support for virtual assistance and voice recognition as one service. There is Cortana on Windows operating system. In iOS, there is Siri. For Android devices and Google products, there is Google Assistant. Additionally, Samsung has Bixby, and Amazon has Alexa.

All mentioned virtual assistance is based on cloud architecture that employs the internet. Some solutions are standalone and work reasonably well in a narrow field, but cloud-based solutions give better results for general-purpose tasks.

This paper will investigate the possibilities of adding virtual assistance to set-top box devices to speed operations with DTV sets and improve user experience. We have selected an Android-based device with a remote controller that supports voice recording [1]. Early work on the subject was done with [1] and [2]. More similar work was done on narrow areas for content playback on set-top box devices by Petrovic et al. [3] and Visekruna et al. [4], and Lazic et al. [5].

II. Section Title (e.g., The Method)

Google Assistant is Google's virtual assistant available on mobile phones, smart home devices, TVs, cars, etc. It has artificial intelligence and enables two-way communication in voice communication and text messages with the user. It can browse the Internet, set alarms and events, access device settings, display user account information, etc. It also supports a lot of functionality for smart homes. For example, the user can control lighting, temperature, TV, and other items in the house. To prevent the service from being constantly active, the recognition of the keywords "OK, Google" and "Hey, Google" was introduced, after which the service became active. After activating Google Assistant, the voice command recording begins, and it collects an audio message which is further processed. The resulting response is reproduced depending on the device on which Google Assistant is used, e.g., in the case of Google Home devices, the answer is in the form of audio sound, while when using a smartphone or TV, a text record is also obtained [6]. If you want to run Google Assistant on an Android TV, you need a microphone to record your voice, integrated into the remote control [6].

Actions on Google (AoG) is a platform that allows you to expand your personal Google Assistant by adding your services called Actions. To enable device management, Action has been implemented. Suppose you want to access the Action described in this paper. In that case, you need to tell Google Assistant: "Talk to True TV" or "Ask True TV", after which he asks the AoG platform to run the application, AoG sends a request to the web service and receives a response forwards to Google Assistant, which displays the answer to the user. Furthermore, Google Assistant forwards the user input directly to the Action, and the Action responds directly to the Assistant.

When creating an Action, it is also possible to use the Dialogflow platform to simplify the understanding of user input. The Dialogflow agent translates user input during a conversation into structured data that applications and services can understand. As the call center, they are trained to handle expected conversation scenarios. The platform itself provides the ability to reply with static responses. If you want to respond with dynamic answers and additional logic, it is possible to implement a web service. In this case, Dialogflow is a proxy between Actions on Google and the web service, as shown in the figure. Instead of sending the request directly to the web service, AoG sends it to Dialogflow. Also, the web service sends a response to Dialogflow, which forwards it to AoG.

The answer must be returned to the user within 10 seconds. Otherwise, the request will expire. Also, the response must be less than or equal to 64 kilobytes. Dialogflow processes the user's input, and it sends an HTTP POST request to the web

Jovana Simić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Jovana.Simic@rt-tk.com)
Đorđe Glišić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Djordje.Glisic@rt-tk.com)
Marija Jovanović – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Marija Jovanović@rt-tk.com)
Nikola Vranić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Nikola.Vranic@rt-tk.com)

 ISBN 978-86-7466-930-3

service, which is set as a function on the Firebase platform. The Cloud Functions for Firebase service automatically starts the function when an HTTP request arrives.

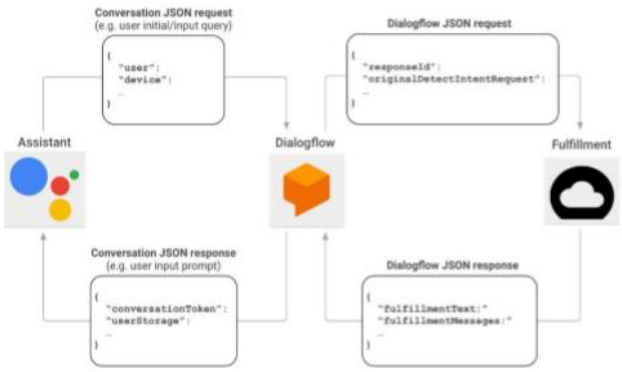


Fig. 1. The diagram shows the path that requests go from Assistant through the Dialogflow till the Fulfillment and response traveling back to the Assistant.

The Actions on Google, Dialogflow, and Firebase platforms are well integrated, as seen in Fig. 2. The figure shows a simple interaction between the end-user, the assistant, Dialogflow, and the web service built in the cloud, like Functions for Firebase and the Firebase Real-time Database.

The user enters his request as a voice command and starts the assistant himself. The assistant converts the voice command into text and starts the Dialogflow searching for the initial action (Welcome intent). Dialogflow uses machine learning to map text to intent and to isolate recognized entities (such as TV channel, TV channel number, time, etc.). Dialogflow triggers a web service, implemented as Cloud Functions for Firebase, sending it a JSON request that contains all the necessary information from the text. The web service processes the JSON request and implements logic (checks the Firebase Real-time Database) to respond to the assistant (or Dialogflow). Device Assistant converts the answer to speech and display (for devices that do not have a screen).

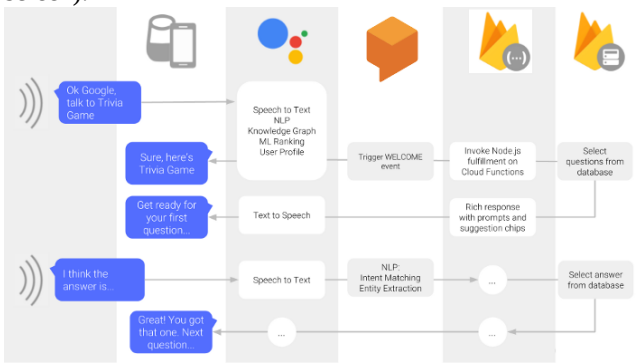


Fig. 2. Voice command workflow from Actions on Google, Dialogflow to Firebase fulfillment web service and back.

III. Implementation

This section describes one way to train and integrate Google Assistant. The Dialogflow agent needs to be trained first. The intent has been created for each supported command, in which it is necessary to enter a large number of phrases that the agent will understand, as shown in Fig. 3.

The pairing of the Action user and the STB device user was solved using a Google account, precisely his email address. To be able to manage the STB during the first launch of the Action, it is first necessary to link the account with the Action. The AoG platform provides the Google Sign-In option, the easiest way to connect and create accounts with the Action. The Action may request access to the user's Google profile, including the user's name, email address, and profile picture. Of course, you need to ask the user if he agrees to access his Google profile.

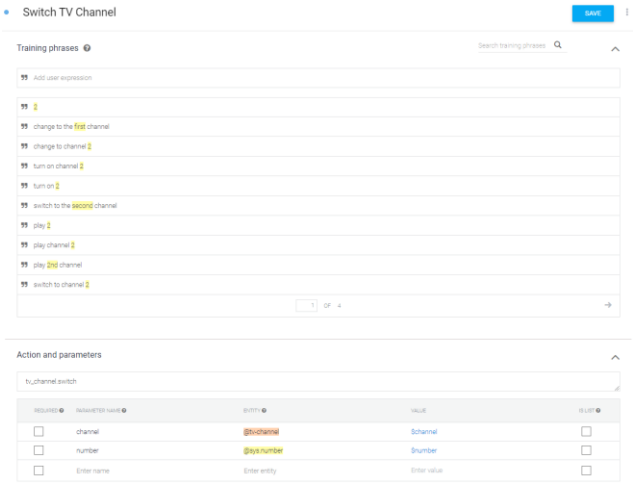


Fig. 3. Training phrases for specific action in Dialog flow and the required parameters defined.

After recognizing the command by the Dialogflow platform, it is necessary to send a signal to the TV application, which should execute the given command. Therefore, for each intent in the agent, an intent handler has been created in the web service. The operator first checks whether the user has linked his account with the Action. If this is not done, the user is first asked to link his account, and the user receives a list of suggested cards with commands for linking the account. Otherwise, the triggered command is entered in the Firebase Realtime Database database in a node whose email field value equals the user's email address accessing the Action. If such a node does not exist, the user with that email address is not logged in to any TV application. If there is, wait a while for the TV application to process the command and return the answer to be displayed to the user.

As already mentioned, the pairing of users of the Action and users of STB devices was done through a Google account. Therefore, the *AccountHandler* class is implemented in the TV application, which detects adding and removing accounts from the application. The class diagram of the *AccountHandler* class is shown in Fig. 4. Each account should have its node in the Firebase Realtime Database. Each account is associated with an object of the type *VoiceHandler* given in Fig. 5, which is used to detect database changes and execute commands.

 ISBN 978-86-7466-930-3

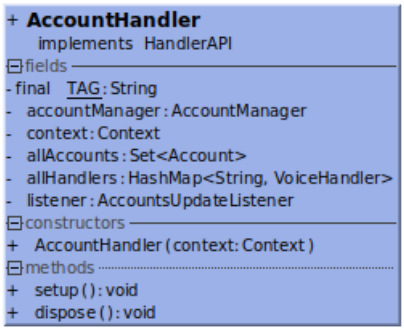


Fig. 4. Account handle class diagram.

When a new account is added to the TV application, it is necessary to add a new node to the database for that account and set the application to listen for changes on that node, for which the *VoiceCommand* class is used. In the class constructor, a reference to the root node of the database is retrieved, and in the *subscribe()* method, a listener is registered to the node in the database for which changes need to be detected.

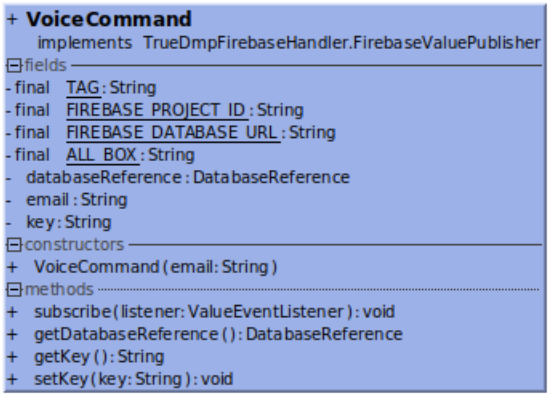


Fig. 5. Voice command class diagram.

IV. Results

A vital result of the presented work is that the voice commands apply to set-top-box devices with acceptable latency. Additionally, the phrase database could be upgraded without changing the actual code on any STB device.

Testing was performed via the Google Assistant application on a mobile device. The response time of the Action from the user's input to the return of the response to the user was tested. The tests were performed under normal conditions, under certain noises, using commands in English.

The table gives a comparative overview of the response time for commands implemented in this system. Response time averages 1 to 2 seconds, depending on the command. As shown in Table 1, the most time-consuming commands are for displaying information about events. These timings are comparable with the times it takes for users to do those actions using RCU. Key benefits could be seen in more complex scenarios, like adding services to favorite lists or scheduling PVR recordings based on the TV show's name for a specific period. Those cloud-based services will outperform manual user interaction with the graphical user interface.

TABLE I
VOICE COMMANDS AND RESPONSE TIME

Command for set-top box	Response Time[s]
Change to the next channel	~1,174
Change to the previous channel	~1,172
Change to a channel with a specific name	~1,195
Change to a channel with a specific number	~1,222
Complete tone reduction	~1,156
Restore volume before complete mute	~1,190
Tone amplification	~1.075
Mute the tone	~1,199
Increase the tone to maximum	~1,036
Check if the STB is in sleep mode	~1,353
View information about the current event on the current channel	~1,535
View information about the current event on a specific channel	~1,599
View information about the next event on the current channel	~1,469
View information about the next event on a specific channel	~1,445
Setting reminders	~1,169

V. Conclusion

In this paper, we have implemented a proof of concept for the voice commands on STB. For more implementation details refer to works [5], [6], and [7]. As with any cloud-based service, it relies on the internet and its stability. Further work shall be done to benchmark latency depending on the internet connection speed, location, and type of connection (WiFi, cable, mobile). It makes that operators supporting this feature offer stable enough internet.

If a set of basic commands is appropriately selected to cover all actions that the user can execute, then a cloud solution can make a chain of commands based on the user's voice request. Dialog flow cloud platform could process arbitrary complex requests that could be transformed and fulfilled as a series of actions. This hides a real benefit, as requests like "make a list of my top five sports channels and record NBA finals on internal disk" could be executed.

Additionally, requests like "Find me a movie with Tom Hanks for Saturday evening" contains more than just a request for STB. They hold user preferences so that operators can train recommendation systems. It opens up a vast field of opportunities as well as privacy concerns.

It does not end with voice commands for STB. It can be a command supported by a third-party application for food delivery or a calendar application.

 ISBN 978-86-7466-930-3

REFERENCES

[1] A. B. Garayalde, MSc thesis, “Speech Control & Media Sharing for Media Centers”, ENST Bretagne France, 2007

[2] KH Lin, CH Lin, KH Chung, KS Lin , “A Compressive Sensing-based Speech Signal Processing System for Wearable Computing Device in IPTV Environment”, ICMT 2013

[3] D. Petrović, M. Zeković and N. Vranić, "One solution for extension of the system for recording multimedia content on Android based devices" 2017 25th Telecommunication Forum (TELFOR), Belgrade, 2017, pp. 1-4.

[4] U. Višekruna and M. Savić, "Integration of Google Assistant in Android Application for Voice Control of Media Playback," 2018 26th Telecommunications Forum (TELFOR), Belgrade, 2018, pp. 1-4.

[5] A. Lazić, M. Z. Bjelica, D. Nad and B. M. Todorović, "Google Assistant Integration in TV Application for Android OS," 2018 26th Telecommunications Forum (TELFOR), Belgrade, 2018, pp. 420-425.

[6] E. Nan: „Upravljanje pametnom kućom uz pomoć Google asistenta”, University of Novi Sad, Faculty of Technical Sciences, Novi Sad, 2017.

[7] J. Simic, “Realizacija glasovnih komandi za uredjaj baziran na operativnom sistemu Android”, UB ETF, Belgrade, Serbia, 2022

 ISBN 978-86-7466-930-3

One solution for simulating conditional access in DTV Software on PC platform

Milan Petrović, Đorđe Glišić, Uroš Jokić and Marija Jovanović

***Abstract*— Digital television (DTV) software runs on various hardware platforms, from low-cost low-performance devices to high-end devices that could compare with modern smartphones and PC configurations. The development quality depends on the tools available for the target platform. A new approach was taken to improve development by moving to the PC platform to avoid this dependency. The benefits are apparent, but it comes with some constraints. Typical examples are components available for target platforms but not PC platforms for security and legal reasons. One such component is the conditional access system (CAS) and digital rights management (DRM) components. This paper will present one solution to simulate conditional access (CA) in software without vendor CA libraries and support in hardware. The aim is to get the ability to test and verify various parts of DTV software that depend on CA functionalities.**

***Index Terms*— digital television, simulation, conditional access, DTV stack test environment.**

I. Introduction

A device that can reproduce digital television needs to comply with some DTV standards (DVB, ATSC, ISDB, etc.). Often it needs to support some content protection mechanism (encryption, signing, etc.). Additionally, the device needs to have a certain number of standard features and a few unique features dictated by the operator that will be available to the user.

In developing DTV software, specific components are delivered from third parties, like a software development kit (SDK) for the target platform or CA libraries for content protection. Content protection certification is an essential step in the development life-cycle, and DTV software is adopted according to the specification documents and APIs delivered. Upon development completion, the application is verified using several test suites that prove it behaves in the required way. This process repeats for every new target platform.

The DTV software development is tightly coupled with the target platform. Depending on the platform and its supporting packages, it may be impractical to develop a more complex project using them as a development platform. Instead, one way to overcome those difficulties is to develop on more suitable platforms. That platform should support at least logging mechanisms, the ability to re-write persistent memory, access to hardware debuggers, and good enough software packages to use those features. In practice, this is not the case, and almost always, given components are missing, and software packages are always behind the state-of-the-art counterpart packages available for PC. Selecting a more applicable platform instead of the target one for development is not applicable if the target CA library has different requirements (hardware or software) compared to its counterpart on a development platform.

For DTV software to be as robust as possible, there was a need to implement support for different CAS vendors. They shared core concepts for content access rights, content protection, operator box management, operator messaging to users, and other customized product and feature protections.

The CAS vendors' APIs significantly differ, although concepts are very similar. The differences between versions from the same vendor may not be compatible. Older libraries tend to have fewer restrictions, while newer versions have more demands and APIs to support, as new scrambling algorithms are added, and more security protocols are employed. It is necessary to have a level of abstraction in DTV middleware to adopt those changes and differences.

As a result, the first DTV simulator was developed on a PC platform [1]. It aimed to support the development of a graphical user interface. It became clear it could be used for implementing DTV middleware features as well. Those features were related to the DTV standard. To support it, the middleware test environment (MTE) [2] was created to test and verify different parts of the software on a PC platform using white box testing [3] [4]. This approach could not cover the code developed for a CA subsystem and the application code that was connected and dependent on that CA subsystem.

This paper aims to discuss paths that could be taken to overcome those obstacles. It gives one solution that is implemented and tested to prove the concepts. We could not find any relevant work on this topic. Closes to the work are discussions on testing approaches made in [5], [6], and [7].

Section two details the challenge and introduces the DTV system's architecture. Section three provides more information on the implementation and final solution. Section four explains verification and test results. Section five concludes the work.

Milan Petrović – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Milan.Petrovic@rt-tk.com)

Đorđe Glišić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Djordje.Glisic@rt-tk.com)

Marija Jovanović – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Marija Jovanovic@rt-tk.com

Uroš Jokić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Uros.Jokic@rt-tk.com.

 ISBN 978-86-7466-930-3

II. PROBLEM STATEMENT

The CA vendor dictates two primary CA integration approaches depending on the target platform and selected operating system. If the target platform runs an operating system (OS) that does not support processes, only threads (tasks), the architecture looks as in Fig. 1. Here DTV software consists of OS, software development kit (SDK, drivers), hardware abstraction layer (HAL), middleware, and application layer. The application depends on middleware, and middleware depends on the abstraction layer (HAL) API that abstracts OS and SDK APIs [8].

The middleware and application layers contain all the business logic, whereas the remaining layers, like HAL, are porting layers designed to be very thin. Application is oriented toward user interface and feature logic, whereas middleware is oriented toward controlling hardware, supporting DTV standards, and interacting with CA subsystems.

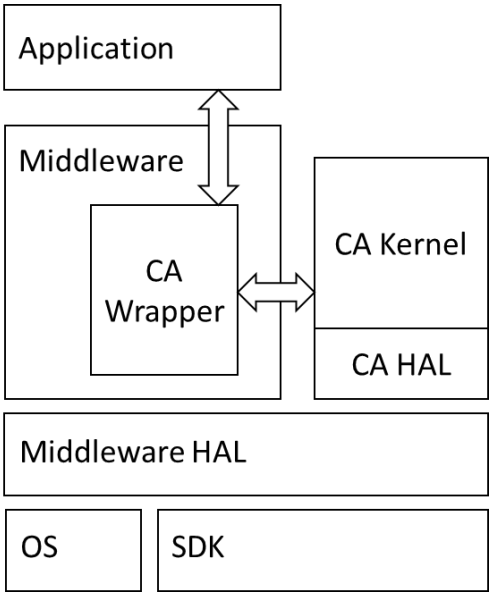


Fig. 1. Typical architecture of DTV software in case of OS where processes are not supported.

The module depicted as CA Wrapper is the actual module seen by the DTV middleware. It interacts with the middleware and application-level modules. It behaves like a proxy between the DTV stack and the existing vendor-specific CA subsystem. The conditional access subsystem consists of the kernel part where the logic is implemented and the hardware abstraction part (CA HAL) used as a glue layer between the CA kernel and underlying OS and SDK APIs. In this architecture, middleware HAL acts like a resource manager and has the information about allocated resources and tasks running in the system. This allows better resource management compared to the second approach.

The second approach is required with the OS supporting processes, like Linux and Android. As depicted in Fig. 2, the CA kernel and CA HAL depend directly on the underlying OS and SDK. They are running in a separate process. If the DTV stack is unstable or crashes, it does not affect the CA kernel. This approach ensures that the rest of the system never compromises CA. Still, the CA wrapper serves as a proxy between the CA kernel and DTV stack. It is up to the SDK vendor to ensure that multiple clients can access the same hardware peripherals. If not provided, some features like PVR may need to be carefully designed to ensure that components do not overlap in responsibilities.

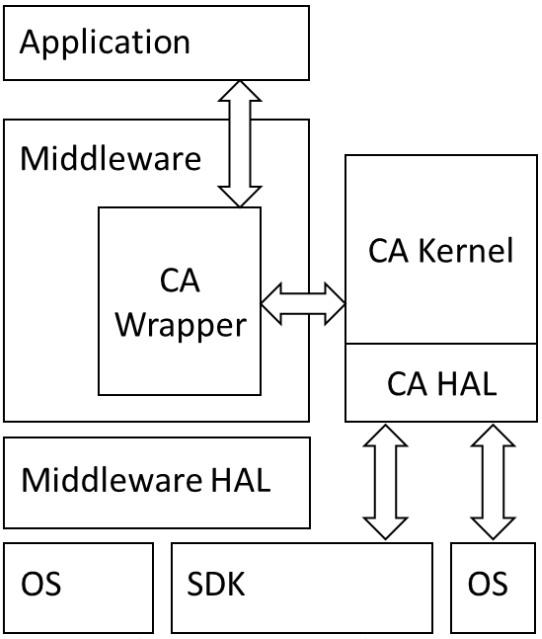


Fig. 2. Typical DTV software architecture in platforms with OS supporting processes.

We need to add CA subsystem support on a PC platform to test CA-related features. There are two possible paths:

1. Implement CA wrapper replacement module
2. Implement CA kernel replacement module (supporting CAS API)

The first solution gives us the ability to have a general CAS subsystem, irrespective of the actual CAS vendor. However, it puts aside CA wrapper code that interacts with the existing CA subsystem. Changes in the requirements of the CAS do not directly affect this solution.

The second approach is to develop the CA kernel module and the CA HAL module. It will preserve the CA wrapper module and allow it to be appropriately tested. However, this approach is considerably more time-consuming and has open questions related to all behaviors implemented in CA kernel API.

Our aim is not to implement content protection as software or hardware encryption. That is transparent to the middleware. Middleware only knows that content is protected and that the CA subsystem must start. CA subsystem is entirely responsible for the content decryption.

Encrypted content is never used in testing on PC because it has a complicated decryption procedure requiring specialized hardware protected by patents and legal documents. Only unencrypted content is used. This type of content can be generated using open-source tools like TS-duck [11] and video content available.

 ISBN 978-86-7466-930-3

III. IMPLEMENTATION

We have decided to take a hybrid approach given the above pros and cons. We implemented CA wrapper API on the DTV stack side as it already exists, allowing the remaining parts of the system to be unaware of the difference. CA kernel is partially shifted to the Middleware Test Environment (MTE). It is a framework for testing the DTV stack on PC.

The DTV software is running as a standalone executable. It has a middleware hardware abstraction layer (HAL) adjusted for the PC platform. Hardware devices are simulated in HAL using SDL [9] and FFmpeg [10] open-source libraries. The test environment is written in Python and communicates with the PC simulator using interprocess communication, particularly sockets. The test environment supported remote control, logging, and execution of automated tests. It can fully control the PC simulator, user input, and DTV stream input. Automated tests are supported by different APIs that are implemented in MTE. More about it can be found in [1] and [2] papers.

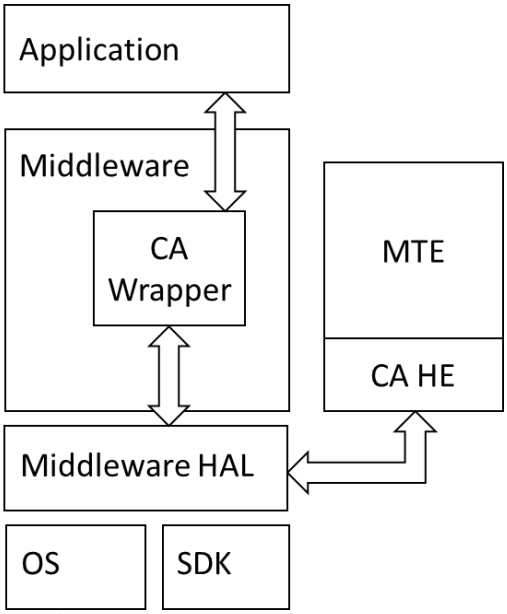


Fig. 3. DTV software architecture with middleware test environment (MTE) supporting conditional access head-end (CA HE)

As given in Fig. 3. the middleware test environment communicates with the PC simulator through the HAL layer that implements interprocess communication. A module CA wrapper uses send/receive routines from HAL. This is to mimic actual data flow, where CA information comes from a demultiplexer connected to the data stream. It will parse received commands and act accordingly. One typical example is the zapping procedure, where service is changed from one to another. In that case, middleware notifies the CA wrapper who needs to check access rights for that service in the database, sharing the data about the service being connected to and additional information about tracks to be descrambled. The module checks access rights in the database and responds to middleware. In our work, descrambling is not implemented, as it does not add any test value since all the descrambling is done in hardware, and none of that logic is done in the DTV stack.

Module CA wrapper is responsible for maintaining the CA kernel database. It exchanges data with the remaining parts of the DTV system. The middleware test environment can get the CA kernel database and modify it by sending appropriate commands. It communicates with a PC simulator using conditional access head-end (CA HE).

Following features (commands) we implemented in the CA HE subsystem and CA wrapper:

1. Device activation in a network
    a. Smart card
    b. Virtual smart card
2. Product access rights
    a. Checking rights
    b. Adding rights
    c. Removing rights
3. Service access rights
    a. Checking rights
    b. Adding rights
    c. Removing rights
4. Content protection
    a. Covered fingerprint
    b. Periodic fingerprint
    c. Permanent fingerprint
5. Mails
6. Changing service bouquet
7. Forced software update
8. CA notification messages
    a. Periodic messages
    b. Permanent messages
    c. User acknowledges messages

In Fig. 4, a window containing the setup for generating CA messages is presented. This CA HE submodule of MTE supports creating test cases. The test case is a message with all the parameters for that particular command. This way, the QA tester or developer does not have to enter test commands each time manually. Instead, he can select test cases saved as an XML file.

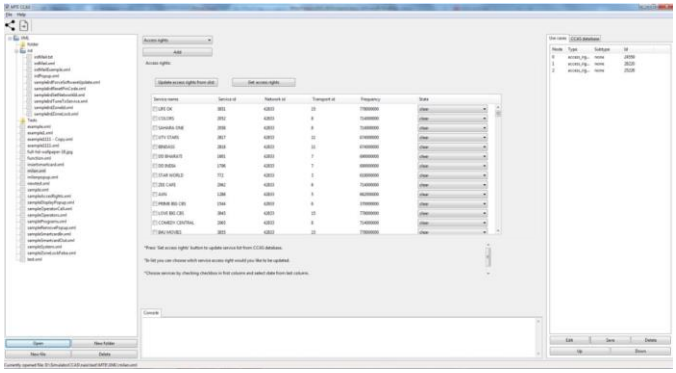


Fig. 4. CA HE main window consists of three parts, the user can re-use existing test cases or make new CA commands and send them in bulk.

CA HE main window consists of three parts, the left part reserved for displaying a tree of saved test cases, the middle

 ISBN 978-86-7466-930-3

part consists of a panel for generating CA commands, and the right part for listing generated commands ready for sending.

The middle panel for adjusting access rights for particular services is depicted in Fig. 5. There can be a list of services available on the box and the access right for that service.

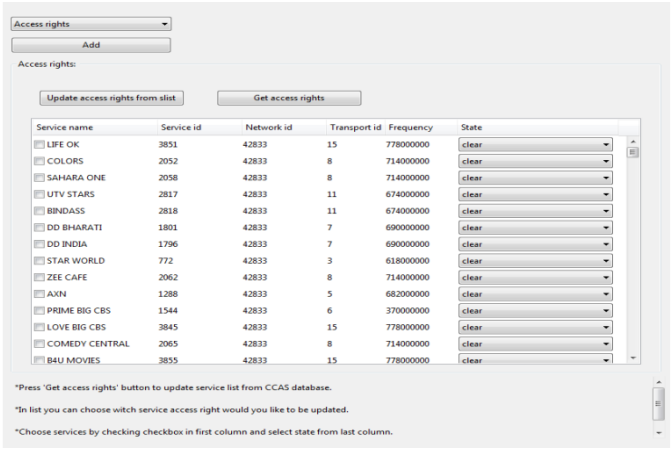


Fig. 5. The middle panel of the CA HE main window, where users select options to modify the service database's access rights.

The service access rights database is saved on the simulator side at runtime. The database can be exchanged between the simulator and MTE upon request sent from MTE. Once they are synced, MTE can send commands related to services access rights to the simulator. When the CA wrapper module receives a command, it processes the message, updates the database and saves it in an XML document. The CA state simulated in the CA wrapper can be restored from the XML file upon simulator restart. With this approach, the simulator is a standalone application, and MTE can communicate with it, but there is no dependency on MTE.

IV. Verification and Results

To test prepared CA subsystems, we have created a set suite that covers all supported types of messages that could be sent to the CA or received from the CA module by the DTV middleware [4]. We have observed that the code is executing correctly and that middleware behaves in the same manner as it is expected in the production environment.

In the case of sending chains of commands, we have observed new failure cases that were not covered by the DTV middleware and application. Those cases involve low probability cases like at the same time receiving a fingerprint message and a CA message. Those cases uncovered several combinations that could not be adequately tested on the development side, the operator's production live network or the lab network. They are not simple to prepare as a test case in those environments.

Scenarios that combine user interaction, CA signaling, and DTV signaling can reveal hidden bugs. Those bugs could be reported as software malfunction in a production. Yet those issues are impossible to reproduce manually unless the exact preconditions are known, which is rarely the case. Troubled combinations may be of low probability, but in networks with many end users, the chances that the failure will be seen and reported are very high. Still, the ability to troubleshoot it efficiently is very poor.

V. Conclusion

This paper focused on expanding capabilities for testing DTV software on a PC platform. It allowed more complex test cases to be executed that would be very hard or impossible to replicate in a network with real hardware. A further way of improving the solution is making a CA API on the MTE side. That could allow the creation of automated tests for testing application behavior as a response to CA events and user interaction.

Work could be extended toward implementing specific CA vendors' API allowing the whole DTV stack to be tested for required functionalities. It will increase the coverage of testable code to almost 100%. But gains versus cost ratio for doing this may not prove as an appropriate step. Another improvement can be made towards implementing some descrambling capabilities.

References

[1] A. Šuka, Đ. Glišić, M. Jovanović, "One solution of DTV simulator for PC platform", TELFOR, 2019
[2] M. Petrović, Đ. Glišić, M. Jovanović, "One solution for testing embedded DTV software on the PC platform", ETRAN 2022,
[3] S. Nidhra1, J. Dondeti, "BLACK BOX AND WHITE BOX TESTING TECHNIQUES – A LITERATURE REVIEW", IJESA, Vol.2, No.2, June 2012
[4] I. Jovanovic, "Software Testing Methods and Techniques", IPSI TIR, 2009
[5] T. Tarkan, "User-driven Automatic Test-case Generation for DTV/STB Reliable Functional Verification"; IEEE Transaction on Consumer Electronics, vol.58, no.2, pp. 587-595, ISBN: ISSN:0098-3063, 2012
[6] Cabot Communications, "Automated testing of digital television devices", accessed 2022, http://www.cabot.co.uk/solutions/robotester-white-paper/at_download/CB.pdf
[7] M. Kovacevic, B. Kovacevic, D. Stefanovic, V. Pekovic "System for automatic testing of Android based digital TV receivers ", INDEL 2014, Banja Luka
[8] G. Miljkovic, "DTV Linux Device Abstraction for Embedded Systems", ISCE, ISBN:978-1-4244-6673-3, 2010
[9] Simple DirectMedia Layer, https://www.libsdl.org/, accessed May 2022
[10] FFMPEG library, https://ffmpeg.org/, accessed May 2022
[11] TSduck, https://tsduck.io/, accessed May 2022

 ISBN 978-86-7466-930-3

One solution for testing embedded DTV software on the PC platform

Branka Ševa, Đorđe Glišić, Uroš Jokić and Marija Jovanović

***Abstract*—In an embedded device industry, applicable software is developed for a particular platform and device. Reusability, functional correctness, and quality control of the software are of great importance. The digital television industry is no different. Moreover, it requires compliance with device safety, security, and functionality standards. Compliance testing is often done with near-end products, as most functionalities require that all components be put together. Secondly, most development is done using target platforms that often lack tools and add significant delays in development. This paper gives one solution for testing the embedded DTV software on PC. The authors give a road map for developing testing environment to safeguard the product's quality. It allows early-stage testing by the development team and helping the QA team test the end product.**

***Index Terms*— automated testing, DTV, middleware test environment, python, OpenCV, tesseract.**

I. Introduction

In embedded devices, hardware capabilities vary in many areas. Available RAM, platform instruction set, supported peripherals, hardware accelerators, and dedicated specialized hardware blocks. On the other side, depending on the product or manufacturer, there are support variations, incomplete documentation, and very little support for the supporting development software packages.

On the other side, there is a problem with integrating third-party components. They may or may not come with the test suite or test application. In the case of open-source software, source code is available, but it was written for specific operating systems (OS), sometimes depending on unique OS features.

A common component for all devices is DTV middleware software. It grows with new requirements, new standards, etc. Testing is always pushed to the end product, verified against predefined sets of tests. The reason behind it is that many features depend on all components being put together, and it is tough to test partially completed software [1].

Additionally, suppose such a DTV stack is inherited from another source without a test suite. In that case, it is always commercially unjustifiable to spend engineering time preparing a test suite that will verify the DTV stack. Instead, it is pushed to develop the end product and confirm its functional compliance [2].

Commercially available solutions are focused on testing the end product. Depending on the solution, it may offer hardware compliance testing or functional testing. Some tools like Intent+ [3][4] offer automated and manual testing. Automated testing is accomplished using dedicated test suite applications. Suitest [5] offers visual preparation of tests. Other solutions provide general-purpose languages like Stb-tester [6]. Others provide APIs like black-box-testing (BBT) API from Intent+. They mainly focus on automating the remote controller, capturing the screen, recording audio, and processing it using a test suite.

As a result of described practices, software products' quality may be at a reasonable level, but the quality of the code may be poor. Reuse of already developed code is very inconvenient across projects. Feature development may slow down as maintaining code becomes more and more expensive. Products may suffer from bugs that have low repeatability rates and high severity. In such cases, black-box testing [7] is not suitable. It is necessary to implement white box testing [8] procedures.

Section two details a problem and describes the system's architecture. Section three explains the proposed solution and provides implementation details. Section four discusses the results. In section five, we conclude our work.

II. Problem statement

In the development stage, verifying a new feature is time-consuming. Platforms with limited hardware capabilities offer unique tools to write software to devices. It may need from 30 seconds up to 5 minutes to run the software. Often those platforms do not support hardware debuggers.

A typical application in DTV consists of the following components:

1. Application layer (APP)
2. Middleware layer (MW)
3. Hardware abstraction layer (HAL)
4. Platform-specific SDK (SDK)
5. Operating system (OS)

Platform-specific SDK is a set of libraries and APIs that provide access to platform hardware components and allows control over them. This layer and the OS layer are closed for the development team. Also, those layers are highly platform-specific, so they cannot be ported to other platforms without

Branka Ševa – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Branka.Seva@rt-tk.com)
Đorđe Glišić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Djordje.Glisic@rt-tk.com)
Marija Jovanović – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Marija Jovanovic@rt-tk.com
Uroš Jokić – RT-RK Institute for Computer Based Systems, Novi Sad, Srbija, (e-mail: Uros.Jokic@rt-tk.com.

 ISBN 978-86-7466-930-3

considerable effort.

The hardware abstraction layer (HAL) provides a defined API [9] that exposes all necessary functions for upper layers (middleware and application) and abstracts platform devices and operating systems. It is implemented again with every new platform. It is common to have abstraction layers for every portable software and a test suite that verifies that the layer is ported correctly.

The middleware layer provides support for the DTV standard and is responsible for all functionalities in the DTV application. It consists of modules controlling hardware service change, acquiring information from DTV signal tables, maintaining program database, service lists, event information database, user interface engine, etc. Those modules are often interdependent. It is not simple to decouple one from the rest of the system and check their correctness using white box testing (e.g., unitary testing).

An essential component of the middleware layer is the conditional access system (CAS) or digital right management (DRM) system. It provides access to protected content. It is also a closed component that comes with the pre-defined test suite.

The application layer covers the graphical user interface and specific logic for the user interface. It is connected to the middleware layer and highly depends on it. Black-box testing mainly verifies this layer.

Architecturally higher-level components depend only on lower layer components. Key components that are developed are the application layer and middleware layer. Hardware abstraction layer API stays the same across different target platforms. We want to create a system that will test those two main components.

The goal is to prepare a software test environment that can support:

1. Functional tests as end-user
2. Scenario tests as end-user and operator
3. Monitoring and testing internal state
4. Code coverage

Functional tests cover black-box testing, where implemented features are verified [1]. Examples are video presence, audio presence, switching service, changing volume, displaying graphics, and event information presented. Besides core DTV tests, additional tests unique to the application have to be supported, like the position of some element on the screen, at the right time, for the correct period, etc.

Scenario tests verify DTV software in more complex cases. Those use-cases involve changing information in DTV tables signaling, new commands from the CAS/DRM system, or new data from other custom protocols that affect the device's state. Tests shell cover application responsiveness to the user interaction and user interface changes based on the system's internal state.

The monitoring system needs to monitor the execution and report critical situations. It should consist of a logging mechanism and software/hardware debuggers to automate the testing of internal states by inspecting calls to specific modules, APIs, and execution paths.

Code coverage gives insight into the test suite coverage of the existing code. If test coverage is low, it may mean that the test suite has to be expanded to cover some exceptional cases or that some source code is unnecessary occupying space (dead code). This work did not cover code coverage testing. Due to the complexity of this feature, implementation details are not covered in this paper.

The test environment defined would be capable of inspecting every module for its dependencies and behavior. Afterward, proper refactoring will allow white box testing (unitary testing, scenario testing).

III. Implementation

We decided to create a test environment to run and test DTV software on a PC. The reason behind it is to use current and future state-of-the-art tools. The first step was to port DTV software to the PC platform. It was done by porting the HAL layer. More details about it can be found in [10]. It supports working with actual transport stream data and makes DTV middleware fully operational. Compared to the commercial product, the only difference is that it does not support targeted CAS, as it is proprietary, and its libraries are only delivered for specific target platforms. Work is done to overcome this, using simulated CAS. Due to the complexity of this feature, implementation details are the subject of another paper and are not given here.

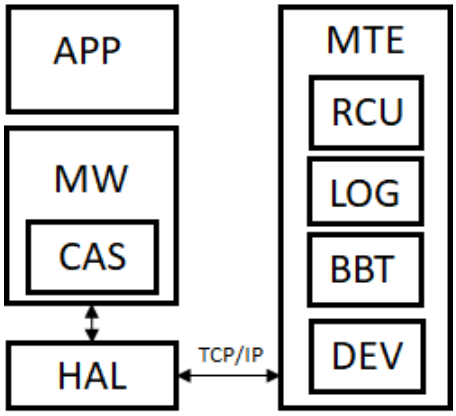


Fig. 1. Key components of DTV software running as part of the PC simulator are on the left side. On the right side are components of the MTE.

We decided to run separate processes for the test environment and DTV simulator. The DTV application runs stand-alone as it would be on the actual device. It allows us to have more options for the middleware test environment (MTE).

Communication with the PC simulator is done using TCP/IP. The communication protocol is designed to be minimalistic. The aim was not to disrupt the dynamics of the DTV middleware execution compared to its expected dynamics on the device. The protocol covers commands from MTE to PC simulator and data from PC simulator to MTE. Commands consisted of remote controller (RCU) events and requests for device state (screen capture, audio status, and

 ISBN 978-86-7466-930-3

similar).

We have decided to implement a test environment in Python language. We saw that this language is widely used in automation testing. Two STB automation test suites [4][5] already support Python scripting. It has extensive library support for user interface, computer vision, text recognition, communication protocols, etc. It is cross-platform, so we could design a tool to run on different platforms. It supports documenting code and a capable development environment (IDE).

We have selected the following frameworks to implement MTE:

1. wxWidgets - UI library (platform-independent, supports all major operating systems)
2. openCV - cross-platform library for computer vision, used for image manipulation and comparison
3. Tesseract - OCR engine for text recognition and extraction

The application was developed to support four different APIs:

1. Remote control API
2. Logger API
3. Black-box testing API
4. Development API

Remote control API covers control over RCU and sends commands to the PC simulator the same way a user would do using a remote control unit (RCU). To send commands, a TCP/IP protocol is used. On the side of the simulator, an existing module for receiving RCU input is adapted to receive TCP/IP commands from MTE.

Logger API is responsible for collecting log information from remote PC simulators using TCP/IP protocol. The existing logging module was improved to send log information over TCP/IP and the serial console on the simulator side. It supports filtering and searching for logging information.

Black-box testing API is a set of predefined APIs implemented on top of RCU API and an additional acquiring protocol for collecting screen output. It is aimed to be used for writing test cases. We selected to support the commercial black-box testing (BBT) API as part of Intent+. It was available to compare with the framework against an existing set of automated tests. Other solutions like Stb-tester API [5] are similar in API and exposed functionality.

Development API is created to support debugger integration in the MTE framework. It is implemented to support GNU GDB compatible debuggers. The framework can run the debugger and start the application or run the debugger and connect to the remote debugger server running the application (Fig. 2). This API makes it possible to start debugging software and send commands like setting breakpoints and watchpoints, printing values, etc. In this scenario, MTE spawns two processes, one for the GDB server that starts the PC Simulator and the second one for GDB used to control the remote PC simulator.

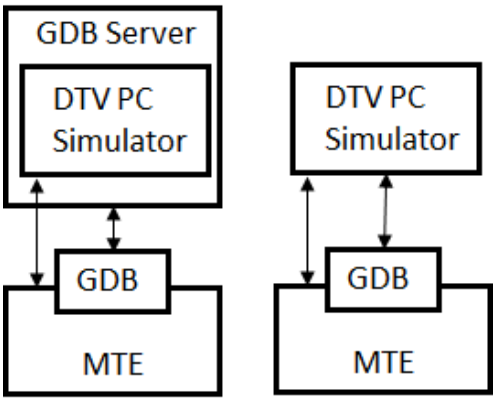


Fig. 2. Possible setups for running PC simulator using GDB debugging software with MTE.

Application consists of four parts similar to the APIs given above:

1. RCU controller
2. Stream controller
3. Logger
4. Test suite controller

Using an RCU controller, the user or developer can control the PC simulator using commands in the window that resemble the real RCU, as shown in Fig. 3.

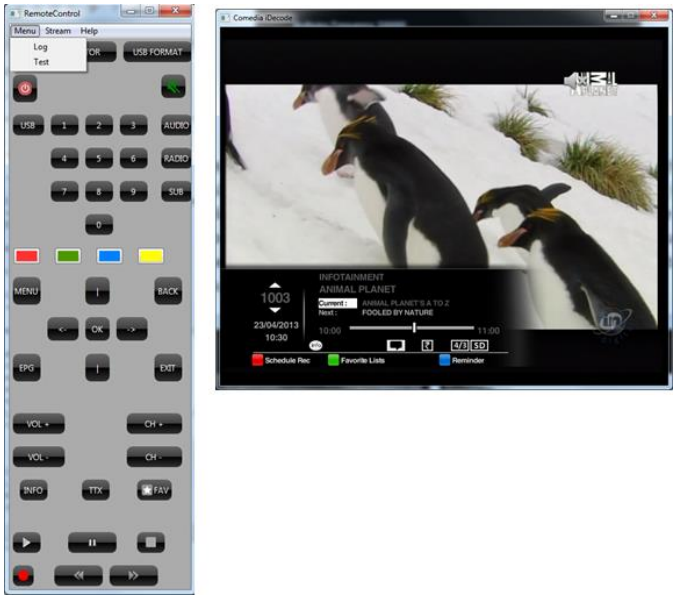


Fig. 3. Key components of DTV software running as part of the PC simulator are on the right side. On the left side are elements of the MTE.

The stream controller window is responsible for adjusting input DTV streams for the PC simulator. It allows setting stream files and broadcasting parameters.

Logger windows give information about logging data and allow users to filter and search for specific data in the log. The search pattern is highlighted in the log. In the filter window, only lines matching patterns are presented (Fig. 4).

 ISBN 978-86-7466-930-3

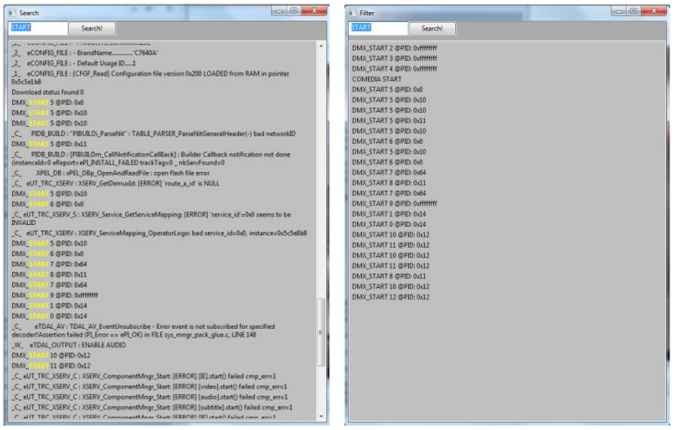


Fig. 4. Search and filter windows for logged information.

IV. Verification and Results

As a result of the following implementation, a test suite was created for commercial products using only BBT API (black-box testing). Test suites are grouped by the features they are testing. A list of all test groups and the number of tests are given in Table 1.

TABLE I
List of test suites prepared and results

Info channel	5 test cases	PASSED
EPG	6 test cases	PASSED
Genres	2 test cases	PASSED
Menus	10 test cases	PASSED
PVR	8 test cases	PASSED
Reminders	3 test cases	PASSED
Favorite lists	6 test cases	PASSED
Service lists	5 test cases	PASSED
Volume	6 test cases	PASSED
Service lists	5 test cases	PASSED
Zapping	5 test cases	PASSED

In the case of automated black-box testing, some graphical test cases may be challenging to create and prove reliable. User interface graphics blended with background video make it more difficult for AI-based engines to recognize certain visual elements' fonts, text, and shapes. Also, the comparison rate with expected images (shapes) may drop due to the background video. Our solution can compare video and graphical layers separately, resulting in higher recognition rates than blended image recognition using tools like OpenCV and tesseract. As a result, our MTE showed fewer errors than hardware running as part of the Intent+ solution.

Verification time was about 15 minutes, compared to manual testing, which will take 1-2h depending on tester skills. This allows developers to save considerable time when developing new features. Compared to automated hardware testing, execution time is around the same. It will enable continuous integration (CI) systems like Jenkins to repeat testing on selected changes.

V. Conclusion

With the proposed solution DTV application could be tested in the development phase by research and development teams or by dedicated QA teams. Automated tests written for MTE are usable for BBT devices in hardware testing, as they are written using the same API.

The essential contribution of this work is automated testing using software debuggers, where developers can inspect certain parts of the system multiple times and summarize information in reports. This type of testing can mimic unitary testing and complex scenario testing having internal systems state exposed for examination and reporting. It allows tightly coupled modules to be slowly refactored and isolated to introduce unitary testing and low-level verification.

Additionally, any other DTV software capable of porting to the PC platform could be tested using this MTE framework. It has to implement necessary features for that middleware and additional requirements to support communication protocol with MTE.

Further work could be done toward implementing support for CAS/DRM simulator or emulation. Also, it would be of great benefit to change DTV signaling from within the MTE application, as now it relies on signaling transported in DTV streams captured from live DTV networks. Another path for improvements is to add systems for code coverage and memory leak checks like Valgrind that could check applications in specific test scenarios as part of the automatic test.

References

[1] T. Tarkan, “User-driven Automatic Test-case Generation for DTV/STB Reliable Functional Verification”; IEEE Transaction on Consumer Electronics, vol.58, no.2, pp. 587-595, ISBN: ISSN:0098-3063, 2012
[2] Cabot Communications, “Automated testing of digital television devices“, accessed 2022, http://www.cabot.co.uk/solutions/robotester-white-paper/at_download/CB.pdf
[3] M. Kovacevic, B. Kovacevic, D. Stefanovic, V. Pekovic “System for automatic testing of Android based digital TV receivers “, INDEL 2014, Banja Luka
[4] Intent+, https://www.rt-rk.com/services/testing-centre, accessed May 2022
[5] STB Tester, https://stb-tester.com/, accessed May 2022
[6] Test suite, https://suite.st/, accessed May 2022
[7] S. Nidhra1, J. Dondeti, “BLACK BOX AND WHITE BOX TESTING TECHNIQUES – A LITERATURE REVIEW”, IJESA, Vol.2, No.2, June 2012
[8] I. Jovanovic, “Software Testing Methods and Techniques”, IPSI TIR, 2009
[9] G. Miljkovic, “DTV Linux Device Abstraction for Embedded Systems”, ISCE, ISBN:978-1-4244-6673-3, 2010
[10] A. Šuka, Đ. Glišić, M. Jovanović, “One solution of DTV simulator for PC platform“, TELFOR, 2019

Comparison of type-2 hypervisor performance on the example of VirtualBox, VMware Workstation player and MS Hyper-V

Borislav Đorđević, *Member IEEE*, Iva Jovičić, Nenad Kraljević and Valentina Timčenko, *Member IEEE*

***Abstract* – This paper presents a comparison of the performances of type-2 hypervisors, on the example of desktop virtualization applications, which include VirtualBox, VMware Workstation Player, and MS Hyper-V. The qualities of all three tested hypervisors, from many aspects of performance, were tested through the performance of the files system. Tests were performed under the same conditions and the same testing methods, using the Filebench program. CentOS 7 was used as the guest operating system. The hypervisor's performances were compared taking into consideration the tests performed for the system with one, two, and three virtual machines in operation. Hypotheses about expected behavior were set, and then they were validated through the obtained results using the Filebench program.**

***Index Terms* – VirtualBox; VMware Workstation; MS Hyper-V; CentOS; hypervisor; virtual machines.**

I. Introduction

Virtualization as a concept is increasingly used and conquers new spaces. It has become a part of everyday life for the simple reason that information technology are all around us. As these technologies are increasingly present in modern life and are constantly advancing, virtualization has taken its place in this development. The main advantages obtained by applying virtualization can be seen in the reduction of costs of IT equipment, electricity and storage space for this equipment. The concept itself provides high security and resistance to failures, and makes administration easier. The choice of virtualization methods and techniques depends on the specific situation and the needs of the end user. This is due to the fact that some virtualization techniques achieve greater flexibility in operation, while others achieve better performance or security. The most commonly used virtualization techniques are: virtualization of hardware, software, data, memory, storage space, virtualization of network infrastructure and virtualization of desktop computers. [1]

Borislav Đorđević – Institute Mihailo Pupin, Volgina 15 Belgrade, Serbia, (borislav.djordjevic@pupin.rs)
Iva Jovičić – VISER, School of Electrical and Computer Engineering of Applied Studies, Belgrade, Serbia, (iva.jovicic@yahoo.com)
Nenad Kraljević – VISER, School of Electrical and Computer Engineering of Applied Studies, Belgrade, Serbia, (nenadk@gs.viser.edu.rs)
Valentina Timčenko - Institute Mihailo Pupin, Volgina 15 Belgrade, Serbia, (valentina.timcenko@pupin.rs)

Virtualization involves the encapsulation and abstraction of computer components, so these components can be used in a way that suits a particular application. Hardware virtualization involves usage of hypervisor, a software layer, which is an intermediary between the hardware and the guest operating system in a virtual machine. This is a simulated environment that could have characteristics equal to the physical environment. The hypervisor can be native, or type-1, which runs directly on the hardware, or hosted, type-2, which runs on the operating system. Examples of this type of hypervisor are VirtualBox 6.1 and VMware Workstation 16 player which have been tested for the purpose of this paper. MS Hyper-V is a bare metal, type-1 hypervisor. However, when activated as a roll, in this case in Windows 10 Pro, it behaves as a type-2 hypervisor. This version of MS Hyper-V was used in the testing process for this paper. [2]

Other than above described classification, hardware virtualization also depends on whether full, partial, or paravirtualization is selected. Full virtualization (Figure 1), topic of this research, represents a simulation of complete hardware, so guest operating systems can be installed and ran without problems. The guest operating system is separated from the physical layer of the host, the hypervisor layer. The advantage of this method of virtualization is increased security and scalability, as well as system flexibility. This solution is the easiest to use, but the performance is slightly lower. [3,4]

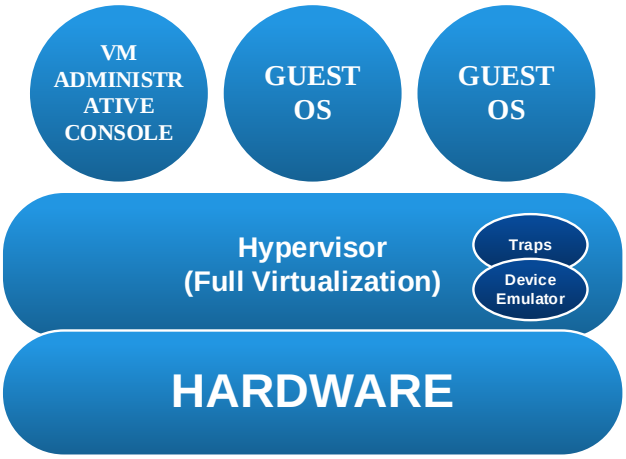


Figure 1. Full virtualization

 ISBN 978-86-7466-930-3

II. Related work, objective and motivation

The focus of this research are the performance characteristics of hypervisors as one of the basic factors in achieving service quality. The problem itself, which can be found in the literature as well as by using these systems in practice, can be viewed from several angles. There are many papers and discussions that use different methodology and approach to evaluating the performance of virtual platforms. The most common is a comparative performance analysis of VMware, Xen, MS Hyper-V and other hypervisors by using various benchmark tools such as HD Tune Pro, ATTO, Filebench, Bonnie ++, etc. Virtualization is a great solution in both desktop and server versions. The needs for virtual platforms for personal use are growing. The main contribution is the mathematical modeling of the file system performance in hypervisor-based virtualization. The modeling of complex virtual environment includes many factors, and modeling expects there is no single winner hypervisor. Similar mathematical model, we used in this paper and most our references [3], [5], [10], [11]. Our model is open for enhancing. We think we are different from related work by our methodology. Its essence is a mathematical model, apply it on a particular case study, and then provide the interpretation of practical results as a validation of model. Using by large number of case studies, we recommend the creation of Knowledge Data Base (KDB) related to the file system performance in virtual environment. Case study in this paper include the performance comparison of three hypervisors in the desktop version, namely VirtualBox, Vmware Workstation player and Hyper-V, in fair-play conditions. This implies identical hardware, the same virtual machines, and an identical version of the guest operating system, which in this case was CentOS 7, an operating system from the Linux distribution family. As VirtualBox and Vmware Workstation player use full virtualization, and MS Hyper-V and paravirtualization as well, the effects of full virtualization for three different hypervisors were examined, using the Filebench benchmark program with four different workloads. Hypotheses about expected behavior are set, followed by a mathematical model for workloads and a hypervisor environment. Performance was measured and the obtained results were interpreted on the basis of models and hypotheses. This paper has some similarities with reference at the serial number five in our literature. The results have the similarities and differences, because the hardware and many other factors are quite different, but we think that both papers are interesting and useful cases of study.

III. VirtualBox, VMware Workstation player and MS Hyper-V

Oracle's VirtualBox is a very powerful program for virtualizing 32-bit and 64-bit operating systems, on computers with Intel or AMD processors. VirtualBox is the only professional solution that is available for free as open source software under the terms of *"GNU"* version 2. This software runs on Windows, Linux, Mac and Solaris operating systems. The technical requirements for running this software are:

- 32-bit or 64-bit operating system with AMD or Intel processor,
- 512MB or more RAM (depending on the number and type of operating systems being virtualized. The RAM memory space allocated to the virtual machine environment can go up to half capacity, as the software itself will not allow more than half of the base system's RAM.
- available hard disk space for the virtual machine environment (recommended size is a minimum of 8GB).

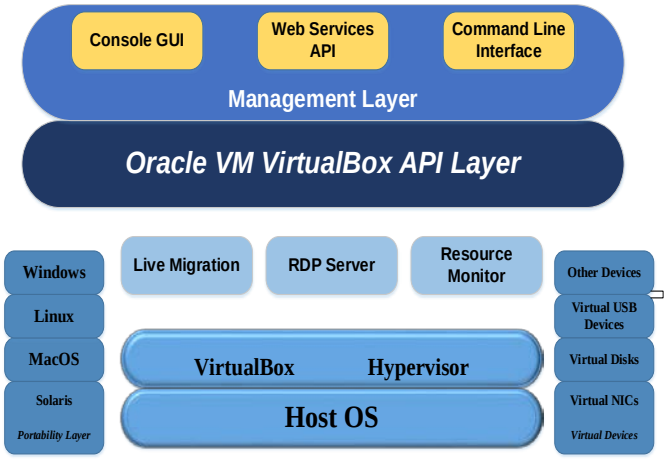


Figure 2. VirtualBox architecture

VirtualBox allows quick and easy data sharing between virtual machines (Figure 2). As VirtualBox is open source, or rather it's software, it tends to solve problems quickly and can be upgraded with new features. Since version 6.1, which is used in this paper, the ability to export and import virtual machines to Oracle Cloud has been added. Software virtualization has been deprecated since this version, and VirtualBox 6.1 uses only hardware-assisted virtualization. [5,6]

VMware® is considered one of the largest manufacturers of software virtualization. The solutions of this company occupy over 70% of the market share in this area, primarily due to the quality of products and the availability of technical support. VMware® has been acquired since 2004 and became part of the EMC Corporation. VMware Workstation 16 player is a software package that runs on standard x86-based hardware with 64-bit Intel and AMD processors and on 64-bit Linux and Windows operating systems. [7,8]

VMware Workstation 16 player (Figure 3) can run existing virtual machines and create their own virtual machines. It uses the same virtualization core as VMware Workstation Pro, a similar multi-featured, non-free program.

 ISBN 978-86-7466-930-3

VMware Workstation 16 player is available for personal non-commercial use (free), for distribution or other use by written agreement. The technical requirements for running this software are:

- 64-bit operating system from the Windows or Linux family with AMD or Intel processor,
- 2GB or more RAM (recommended 4GB or more),
- 1.3GHz or higher core speed,
- available hard disk space.

Figure 3. VMware Workstation player architecture

Microsoft is one of the leading companies in information technology. At the server level, virtualization has become the standard, but interest in virtualization has also emerged among users for personal use. Microsoft occupies about 15% of the market with MS Hyper-V virtual platform. With the release of Windows 8 in 2012, MS Hyper-V became an integral part of its Enterprise, Education and Pro editions. MS Hyper-V is a type-1 hypervisor-based system for x86-64 operating system architectures. It is activated in the Windows operating system as a roll, just like any other service in the Microsoft family. There are some MS Hyper-V features that work differently in Windows OS and Windows server. The memory management model is different for MS Hyper-V, where MS Hyper-V manages memory on the server assuming only virtual machines run on the server, and in the Windows operating system, it is managed with the expectation that most client machines run software on the host in addition to virtual machines. [9-11]

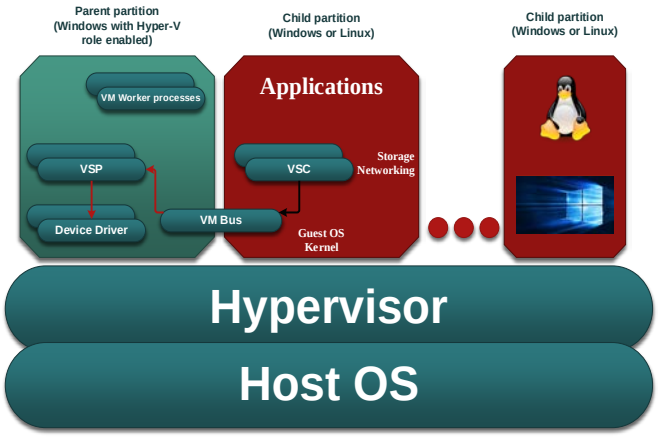


Figure 4. MS Hyper-V architecture

MS Hyper-V (Figure 4) supports virtual machine isolation and uses partitions in which guest operating systems will run.

IV. Hypotheses about expected behavior

As the Type-2 hypervisor running under the guest operating system was used, we can point out that each workload generates typical random and sequential data read times, as well as random and sequential data write times. Each workload is defined by the access time for file systems. Workload represents the total time to complete all operations, the time required to complete all operations related to directories, metadata, file blocks, free lists, house-keeping and journaling operations in the file system. There are five components in a virtual environment that have an impact on workload time (Tw – Time Workload):

$$Tw = f\,(Bn, gFS, VHw\text{-}pr, Hp\text{-}pr, hFS) \qquad (1)$$

The first and second components, Bn (Benchmark) and gFS (guest OS file system) are exactly the same for VMware Workstation player, Oracle VirtualBox and MS Hyper-V. The analysis focuses on the interaction between the reference values and the guest operating system. Because the test environment relies on the use of an identical benchmark, identical virtual machines, and ext4 as the guest file system, these components are expected to have an identical effect on Tw. Processing time for full hardware virtualization is the third component of VHw-pr (virtual hardware processing). Each hypervisor uses its own solution for full hardware virtualization, so the performance will be different too. The fourth component, Hp-pr (hypervisor processing), represents the time it takes for the hypervisor to receive requests from the virtual hardware and forward them to the host drivers. In particular, guest FS requests (guestOS-FS) are forwarded to host FS (hostOS-FS). All of these hypervisors, VMware Workstation player, Oracle VirtualBox and MS Hyper-V, generate different hypervisor processing times. The fifth component, hFS (host OS file system), represents the

 ISBN 978-86-7466-930-3

processing time of the host OS file system. All hypervisors have MS NTFS as hostOS-FS, so this component is expected to have similar processing times for all hypervisors. The dominant influence of the third and fourth components of formula (1) is expected as the tests are focused on the performance of natively virtualized guests (complete hardware virtualization). [12]

V. Test configuration and benchmark application

The prerequisite for quality and adequate testing is the application of one hardware configuration, the same operating system, then the selection of a quality benchmark program and the same measurement methodology for all testing procedures. Testing was done on a personal computer whose characteristics can be seen in Table I, while the characteristics of the disk are given in Table II. CentOS 7 from the Linux distribution family is installed as a guest operating system.

TABLE I - Test environment/PC

Components	Characteristics
Processor	Intel Core i5-4590S 3GHz
Memory	8GB DDR3
Cache	6MB L3
Hard drive	Seagate Barracuda 7200.12
Operating system	Windows 10 Pro, 64-bitni

VirtualBox 6.1 virtualization platforms, VMware Workstation 16 and MS Hyper-V, a version for Windows 10 Pro, are installed or activated on the hard drive, where the tests were done. The hard drive was also used to install virtual machines.

TABLE II - Test environment/Hard Disk

	Seagate Barracuda 7200.12
Model Number	Seagate ST3500418AS
Capacity	500GB
Interface	Serial-ATA/300
External Transfer Rate	3.0Gb/s
Max Sustained	300MB/s
Cash	16MB
Average Latency	4.17ms
Spindle Speed	7200rpm
Average Seek Time Read	8.5ms
Average Seek Time Write	8.5ms

All tests were performed using the benchmark program Filebench 1.4.9.-1. This program is designed to measure storage space and performance of file system. Filebench is capable of generating several types of workloads, it can simulate environments when using certain services such as mail server, fileserver, web server, etc. [13, 14]

VI. Testing and results

This paper shows a comparison of the performance of virtual platforms for personal use with their capabilities. Disk performance and data flow were measured. To make testing meaningful, all virtual machines are created exactly the same and have the same characteristics (Table III).

TABLE III - Virtual machine parameters

Components	Characteristics
Virtual processor	1
Memory	2GB
Virtual hard drive	60GB
Operating system	CentOS 7

During the testing, modified base code files were used, such as *varmail.f*, *fileserver.f*, *webserver.f* and *randomfileaccess.f*, which test the web, mail and file server. The appearance of the set parameters of the benchmark program can be seen in Table IV. To achieve the most realistic results, each test lasted 120 seconds. Each test was repeated ten times, and the obtained results were expressed as the average value of these tests.

TABLE IV – Base code parameters of **F* file

	Varmail	Web server	File server	Random file access
nthtreads	16	100	50	5
nfiles	1000	1000	10000	10000
meandir widht	1000000	20	20	20
meanfile size	16k	16k	128k	random

Tests were conducted by first installing VirtualBox and then creating a virtual machine in this program. It was tested, and then this procedure was repeated when two and afterwards three virtual machines were created. The testing system was exactly the same when the virtual machines in the VMware Workstation 16 player application were tested. Of course, before testing on this platform, previous virtual machines and VirtualBox applications were uninstalled. At the end of the testing, the MS Hyper-V roll was activated and tests were performed, as in the previous two cases. In that way, fair-play conditions were created for all three virtual platforms. The results of the Fileserver workload test can be seen in Table V and Figure 5.

TABLE V - Fileserver benchmark results

 ISBN 978-86-7466-930-3

Fileserver	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)
VMware	74,58	64,33	56,41
VirtualBox	42,03	35,16	29,76
MS Hyper-V	33,34	28,12	21,53

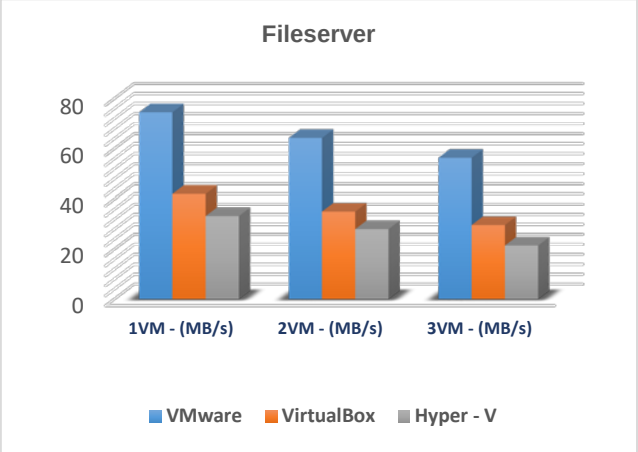


Figure 5. Fileserver test results

The results of testing other workloads are shown in Tables VI, VII and VIII, as well as graphical representations in Figures 6, 7 and 8.

For the *"Fileserver"* workload, we note that VMware is by far the best, while VirtualBox is better than Hyper-V. In a complex workload such as Fileserver with sequential and random write components, the FS cache effect on guest and hostOS is significant, so VMware wins convincingly primarily because of the 3rd component of formula (1) and the best cooperation with FS caching.

TABLE VI - VARMAIL BENCHMARK RESULTS

Varmail	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)
VMware	44,58	42,62	39,87
VirtualBox	18,46	17,83	16,62
MS Hyper-V	14,06	12,77	10,11

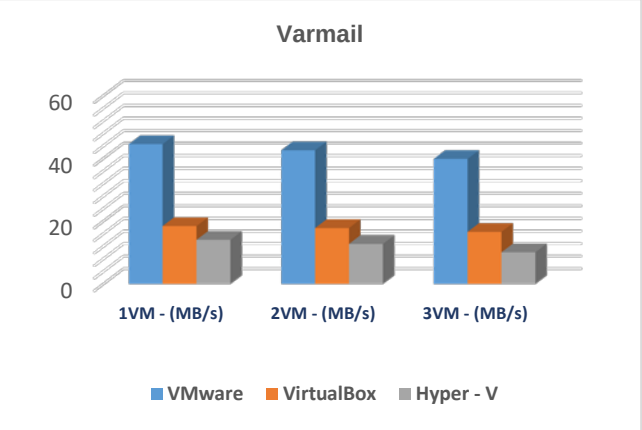


Figure 6. Varmail test results

For the *"Varmail"* workload, we note that VMware is by far the best, while VirtualBox is again slightly better than MS Hyper-V. In the Varmail workload, in addition to random reading, there are also synchronous components of random write, the impact of FS caching is small, so VMware and then VirtualBox obtain wins, primarily because of the 3rd and 4th components of formula (1).

TABLE VII - WEBSERVER BENCHMARK RESULTS

Webserver	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)
VMware	84,68	81,04	77,68
VirtualBox	47,73	42,26	37,86
MS Hyper-V	80,92	76,61	71,92

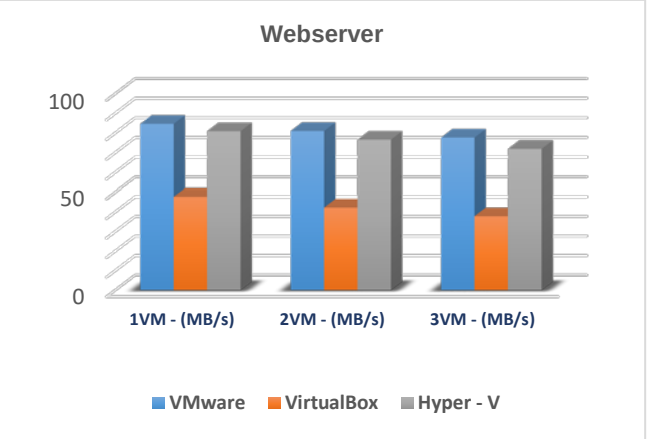


Figure 7. Webserver test results

For the *"Webserver"* workload, we note that VMware is slightly better than MS Hyper-V, and VirtualBox is significantly weaker. In the Webserver workload, which has both random read components and very few random write components, there is less influence of FS caching, so VMware and MS Hyper-V did better than VirtualBox, and win primarily because of the 3rd and 4th and the components of formula (1) and the cache effect in random reading, the dominant workload in Webserver environment.

 ISBN 978-86-7466-930-3

TABLE VIII – RANDOMFILEACCESS BENCHMARK RESULTS

Randomfile access	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)
VMware	5141,56	5034,02	5008,76
VirtualBox	2595,11	2546,56	2517,36
MS Hyper-V	4888,85	4816,64	4790,44

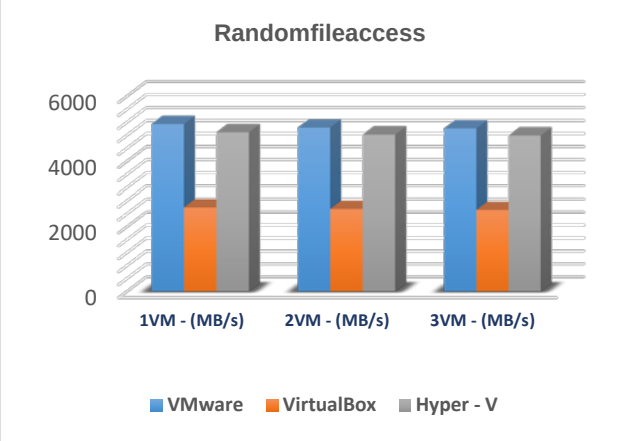


Figure 8. Randomfileaccess test results

For the *"Randomfileaccess"* workload, we can see that Hyper-V is slightly weaker than VMware, and VirtualBox is significantly weaker again. In the Randomfileaccess workload, which has both random read components and a lot of asynchronous random write components, there is a solid impact of FS caching primarily for random writing, so VMware and MS Hyper-V fared better than VirtualBox, primarily because 3rd and 4th components of formula (1) and solid cache effect in random entry.

VII. CONCLUSION

When it comes to virtualization, it should be noted that it brings major changes in the information technology and computer industry, primarily in reducing investment in infrastructure and saving electricity consumption. In area of personal computers, virtualization has made great strides by bringing a large number of software for this purpose. Some of them are also presented in this paper. The advantage of using these applications is the ease of installation and usage, and the fact that the platforms tested in this paper are completely free. For this case study, VMware is the absolute winner in all workloads. We believe that the differences is made by the 3rd and 4th components of formula (1), as well as by the powerful hypervisor usage of the FS cache effect. For workloads with many sequential features and weak cache effect (Fileserver and Mailserver) VirtualBox is better than MS Hyper-V, and for workloads with random dominance and solid cache effect (Webserver and Randomfileaccess) MS Hyper-V is better than VirtualBox. Future work on this topic will focus on testing other virtual platforms in the field of desktop computers, as well as the use and testing of various operating systems that are applied in practice.

ACKNOWLEDGEMENT

The work presented in this paper has partially been funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

LITERATURE:

[1] Matthew Portnoy: "Virtualization Essentials", 2016.
[2] Hypervisor Type-2. Online: https://www.ibm.com/developerworks /cloud/library/cl-hypervisor compare/, 2021.
[3] B.Đorđević, V.Timčenko, N.Kraljević, N.Davidović, "*File system performance in full hardware virtualization with ESXi and Xen hypervisors*", 18th International Symposium INFOTEH-JAHORINA, 2019.
[4] Full Virtualization. Online: https://www.sciencedirect.com/topics /computer-science/full-virtualization, 2021.
[5] Dejana T. Vojnak, Borislav S. Đorđević, Valentina V. Timčenko, Svetlana M. Štrbac: "*Performance Comparison of the type-2 hypervisor VirtualBox and VMware Workstation player*", 27th Telecommunication Forum, 2019.
[6] VirtualBox, Online: https://www.virtualbox.org/, 2021.
[7] Perfomance Evaluation of VMware and VirtualBox Online: https://pdfs.semanticscholar.org, pdf-2021.
[8] VMware, Online: https://www.vmware.com/product/workstation-player.html, 2021.
[9] Microsoft, Online: https://www.microsoft.com/, 2021.
[10] Jadran Torbić, Ivan Stanković, Borislav S. Đorđević, Valentina Timčenko: "*Hyper-V and ESXi hypervisors comparison in Windows Server 12 virtual environment*", 17th International Symposium INFOTEH-JAHORINA, 2018.
[11] B. Đorđević, V. Timčenko, N. Kraljević, N. Maček: "*File System Performance Comparison in Full Hardware Virtualization with ESXi, KVM, Hyper-V and Xen Hypervisors*", Advances in Electrical and Computer Engineering, vol.21, iss.1, 2021.
[12] M. Polenov, V. Guzik, V. Lukyanov: "*Hypervisors comparison and their performance*", Computer Science Online Conference, 2018.
[13] Filebench, Online: https://github.com/filebench/, 2021.
[14] Christopher Stracheyje, "*Time Sharing in Large Fast Computers*". Online:https://archive.org/details/large/fast/computers/page/n5/mode /2up, 2021.

 ISBN 978-86-7466-930-3

Comparison of file system performance in full virtualization with MS Hyper-V and KVM hypervisors

Borislav Đorđević, *Member IEEE*, Miloš Piljić, Nenad Kraljević and Valentina Timčenko, *Member IEEE*

***Abstract* - This paper presents a comparison of the performance of native hypervisors on the example of MS Hyper-V and QEMU/KVM virtual platforms. Their quality was examined through aspects of file system performance. Filebench program was used for testing procedure, which is an application that guarantees equality and independence from the impact of hardware environment. CentOS 7, an operating system from the Linux distribution family, was used as the guest operating system. The tests were performed for one, two and finally three virtual machines that are running simultaneously. The results were further validated based on the defined hypotheses related to the expected behavior of the hypervisors.**

***Index Terms* - MS Hyper-V; QEMU/KVM; CentOS; virtual machines; performance.**

I. Introduction

In the area of information technology, virtualization is a way of creating a virtual version of computer resources. Virtualization is a simulation of the hardware or software that other software, such as various operating system, is running. Virtualization is initially applied by IBM in 1960s as a method for the logical division of mainframe computer system resources between different applications. The need to manage the "one server-one application" model has been eliminated, opening the possibility of running multiple operating systems on the same hardware platform. The advantages and savings that are obtained by using such a system are more than obvious: hardware, CPU, memory resources, administration staff. All this is a plus for virtualization in the reliability segment. The virtualization solutions allow easiness in adding new servers, as well as in data migration from one server to another. This is an additional advantage of this technology in the field of scalability.

Borislav Đorđević - Mihailo Pupin Institute, Volgina 15 Belgrade, Serbia, (borislav.djordjevic@pupin.rs)
Miloš Piljić - VISER, School of Electrical and Computer Engineering of Applied Studies, Belgrade, Serbia, (milosrin4520@gs.viser.edu.rs)
Nenad Kraljević - VISER, School of Electrical and Computer Engineering of Applied Studies, Belgrade, Serbia, (nenadk@gs.viser.edu.rs)
Valentina Timčenko - Institute Mihailo Pupin, Volgina 15 Belgrade, Serbia, (valentina.timcenko@pupin.rs)

Hardware virtualization, which is the topic of this paper, is the most popular and widespread type of virtualization [1]. The software that controls virtualization is called a Virtual Machine Monitor (VMM). According to the most common form of use in a professional IT environment, the process of creating and managing virtual machines is also called server virtualization. There are two categories of hypervisor: type-1 (native) and type-2 (hosted). In this paper, type-1 hypervisors were tested for the case of MS Hyper-V and QEMU/KVM virtual platforms (Figure 1) [2].

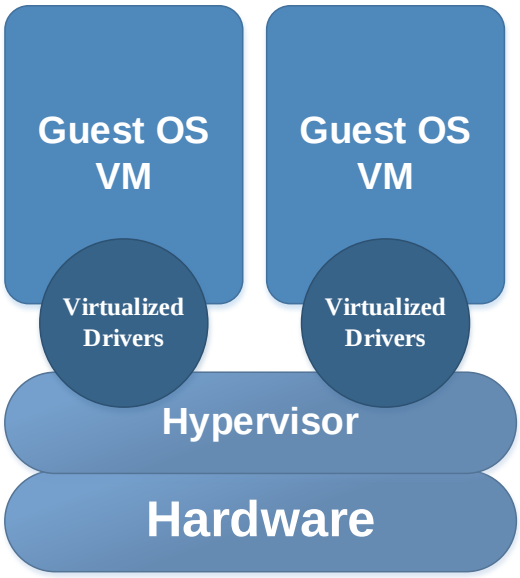


Figure 1. Native hypervisors

II. Research work, goal and motivation

The literature related to this field is mostly focused on comparative analysis of hypervisor performance, using by different test methodology and benchmark tools. For this purpose, some proven benchmark tools are usually used, which is one of the cornerstones for obtaining quality level results. We recommend the Filebench, as open source solution, because it is a versatile, powerful, multithread and it simulates the real application workloads. We recommend the Fio tool, similar benchmark as Filebench, and some synthetic benchmarks such are Bonnie++, Postmark etc. The main contribution of this paper is the mathematical modeling

 ISBN 978-86-7466-930-3

of hypervisor-based virtualization in the context of the file system performance and applying the model on a performance case study for the interpretation of benchmark results. Because the complex virtual environment includes large number of factors, model expects there is no single winner hypervisor and depends on the case study i.e. the workload characteristics. In relation to competition, we are forcing a mathematical model and a number of case studies based on model with practical performance tests. The server variant of the virtualization stands for a great solution, primarily due to the introduction of the infrastructure costs and hardware reduction, followed by the easier administration. Still, there is a lot of room and opened questions for the improvement in this area. This paper contribution is the validation and comparison of two hypervisors, namely MS Hyper-V and QEMU/KVM, for which we have tested the quality and performances in identical conditions. Both hypervisors use full virtualization, while MS Hyper-V is also suitable for the use of the paravirtualization. As the guest operating system we have used CentOS 7, popular distribution from the Linux OS family, while for testing needs we have applied Filebench benchmark program with 4 different workloads. After defining the hypotheses, a mathematical model was set up, and validated by the obtained results [3], [4].

III. MS HYPER-V AND QEMU/KVM

MS Hyper-V is an efficient hypervisor, developed by Microsoft, which enables virtualization of operating systems in a server environment (Figure 2). With the release of Windows Server 2008 R2 version, Microsoft has included a Hyper-V virtualization solution in the operating system itself. MS Hyper-V is a role that allows administrators to create multiple virtual machine, and supports isolation of partitions in which guest operating systems will run [5], [6].

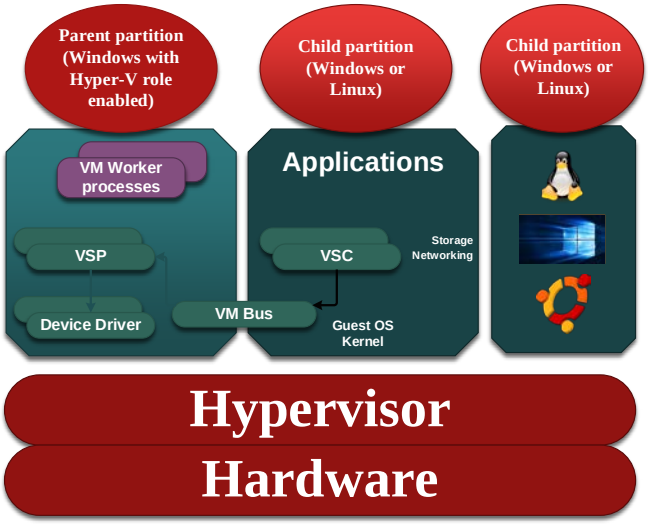


Figure 2. MS Hyper-V architecture

When it comes to virtualization under the Linux operating system, KVM (Kernel-based Virtual Machine) is almost indispensable technology. It is originally created as the Red Hat sponsored project. KVM is implemented in the form of a kernel module and is an integral part of the Linux kernel from version 2.6.20. For the KVM it cannot be said that it is a type-1 or type-2 hypervisor. On the one hand, KVM extends the Linux kernel and adds virtualization capabilities to it, allowing Linux itself to be treated as a native hypervisor (Figure 3). On the other hand, Linux is a standalone OS on which KVM functionality relies orthogonally, so it can be said that KVM runs above the main OS (hosted hypervisor), using already implemented system functions in the absence of its own (QEMU) [7-9].

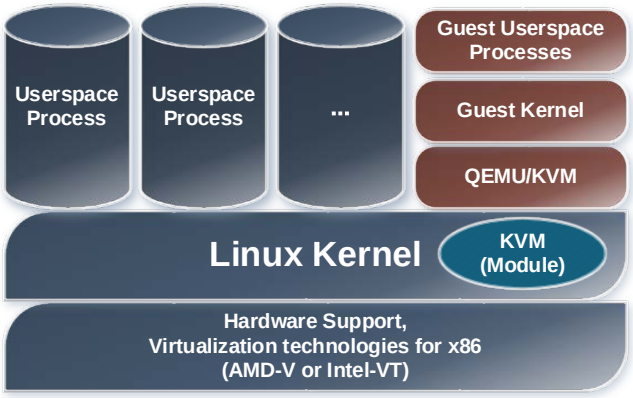


Figure 3. QEMU/KVM architecture

IV. HYPOTHESES ABOUT EXPECTED BEHAVIOR

Both hypervisors are native, they work directly on the hardware and are realized in the microkernel architecture. The total processing time for each load *Tw* (Time workload) can be calculated as follows (eq. 1):

$$T_W = T_{RW} + T_{SW} + T_{RR} + T_{SR} \quad (1)$$

where T_{RW} and T_{SW} represent random and sequential data entry times, while T_{RR} and T_{SR} represent random and sequential read times. For each of these workloads, there is an expected access time for a file system that includes five components (eq. 2):

$$T_{WL} = T_{FB} + T_{FL} + T_J + T_{HK} + T_{DIR} + T_{META} \quad (2)$$

where T_{WL} represents the total time for the implementation of all operations for a defined workload, and the elements from equation (2) represent the time required for the implementation all operations related to file blocks, file lists, journaling, house-keeping, metadata and directory in the file system. There are 5 components that have an impact on the workload time T_W (eq. 3):

$$T_W = f\,(\text{Bn, gOS-FS, Hp-proc, VH-proc, hOS-FS}) \quad (3)$$

 ISBN 978-86-7466-930-3

The first and second components Bn (Benchmark) and gOS- FS (guest file system) are identical for KVM and Hyper-V. Since an identical benchmark and the same virtual machines with their ext4 guest file system are used in testing, we can assume that these components will have the same impact on the third component, Hp-proc (hypervisor processing) which represents a typical delay of hypervisor (KVM-delay, Hyper-V-delay). This represents the time that takes the hypervisor to receive a request from the virtual hardware and forward it to the host drivers. The fourth component, VH-proc (virtual hardware processing) for KVM is QEMU full virtualization, and for Hyper-V MS full virtualization. Although these are full hardware emulations, both hypervisors have their own solutions that will certainly differ in performance. The fifth component is hOS-FS (host file system). KVM uses ext4 and the Hyper-V NTFS file system, and this component is expected to cause different processing time for hypervisors. Since the tests are focused on the performance of native virtualized guests, the dominant influence of the third, fourth and fifth components of formula (3) is expected.

V. Test configuration and benchmark application

In order for testing to be adequate and high quality, it is necessary to use the same hardware configuration, the same guest operating system, choose a quality benchmark test program and the same performance measurement methodology. The tests were performed on an IBM server, whose characteristics can be seen in Table I, and the characteristics of the hard disk on which the tests were performed can be seen in Table II. CentOS 7 was used as a guest OS [10].

TABLE I - Server/Test Environment

	IBM 7945J2G - System x3650 M3
Processor	Intel® Xeon E5620 2.4GHz
Memory	32GB DDR3
Cache	12MB L3
Hard Disk	8 x Kingston 240GB SSD Now V300 SATA 3 2.5 (SV300S37A / 240G)
Network	2 x 1Gb / s

Virtual Platforms MS Hyper-V and QEMU/KVM are installed on hard drives converted into RAID 10, size 960GB (4x240GB SSD), while the other (RAID10/960GB) served as a repository on which virtual machines were created.

TABLE II - Hard Disk/Test Environment

	Kingston 240GB SSD Now V300
Model Number	SV300S37A/240GB
Model Name	SSD Now V300
Capacity	240GB
Interface	SATA 3.0 (6Gb/s)
Connectivity Technology	SATA
Hard Disk Form Factor	2.5 Inches
Read / Write Speed	450MB/s
Cache Size	240GB

All tests were done using Filebench, a benchmark program version 1.4.9.1-3. This program is designed to measure the performance of file systems and storage space and is capable of generating a large number of workloads. In this paper, 4 different workloads are used simulating environments when using services: web, mail and file server [11].

VI. Testing and results

This paper presents a comparison of the performance of virtual platforms for server use. Disk performance and data throughput were tested. In order to make testing meaningful, all virtual machines were created with identical characteristics (Table III).

TABLE III - Virtual Machine Parameters

Components	**Characteristics**
Virtual Processor	1
Memory	8GB
Virtual Hard Disk	200GB

For mail, file and web server test needs, we have modified the base code files for analyzed workloads: *webserver.f, varmail.f, fileserver.f* and *randomfileaccess.f*. First, Hyper-V was tested, which was activated as a role on Windows Server 2016, by creating one virtual machine that was tested. The same procedure is repeated for testing the environment with two and three virtual machines. Each test lasted 120 seconds and was repeated ten times. The final result represents the average value of the obtained test results. Before testing the KVM virtual platform (using CentOS 7 with the KVM option checked), the Windows server with its virtual machines was uninstalled in order to clean the environment. An identical installation and testing procedure were then conducted with the KVM virtual platform. In this way, fair-play conditions were acquired for

 ISBN 978-86-7466-930-3

both virtual platforms. The results of Varmail workload testing can be seen in Table IV and Figure 3.

TABLE IV - VARMAIL BENCHMARK RESULTS

Varmail	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)	Native (MB/s)
MS Hyper- V	25.21	19.82	13.11	
QEMU/KVM	13.04	12.79	12.22	
Native OS				68.77

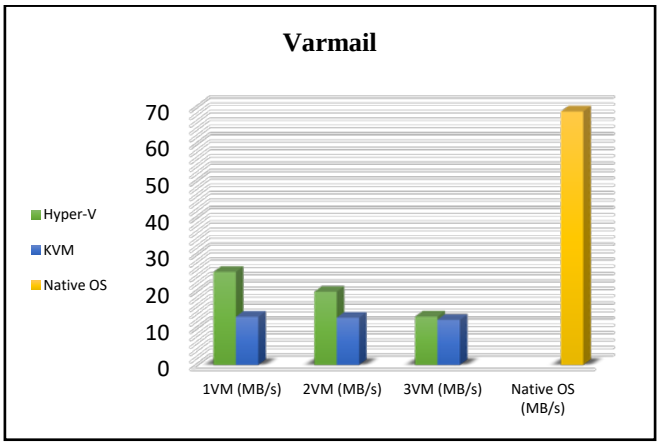


Figure 3. Varmail test results

For the "*Varmail*" workload, we notice that Hyper-V is solidly better than KVM. In this workload, besides the random read components these are synchronous random write components too for which the impact of the FS caching is very small. In this case, Hyper-V is better, primarily due to the fifth component of formula (3), where NTFS for this workload performed better in FS pair (ext4 on NTFS compared to ext4 on ext4).

The results of testing other workloads can be seen in the graphs (Figures 4.5 and 6), as well as in Tables V, VI and VII.

TABLE V - FILESERVER BENCHMARK RESULTS

Fileserver	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)	Native (MB/s)
MS Hyper-V	146.04	83.75	47.43	
QEMU/KVM	155.44	138.84	115.46	
Native OS				555.63

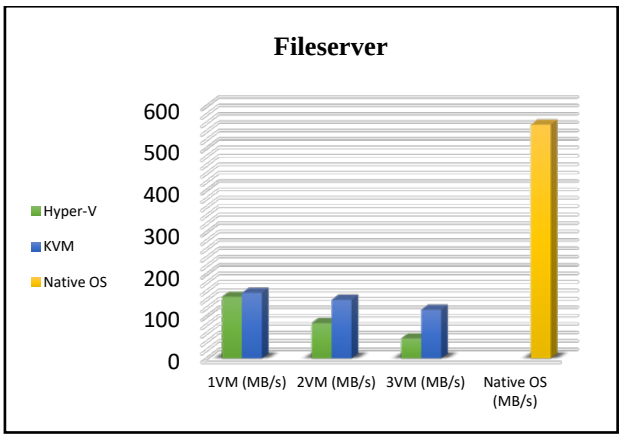


Figure 4. Fileserver test results

For the "*Fileserver*" workload, we notice that KVM is better than Hyper-V. In a complex workload such as Fileserver in which there are random and sequential write components, the FS cache effect on the guest and host OS is significant, so KVM wins primarily because of the third and fourth components of formula (3). We believe that KVM has better virtual hardware processing and less hypervisor latency.

TABLE VI - WEBSERVER BENCHMARK RESULTS

Webserver	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)	Native (MB/s)
MS Hyper-V	53.94	47.73	43.28	
QEMU/KVM	39.62	37.99	36.44	
Native OS				115.26

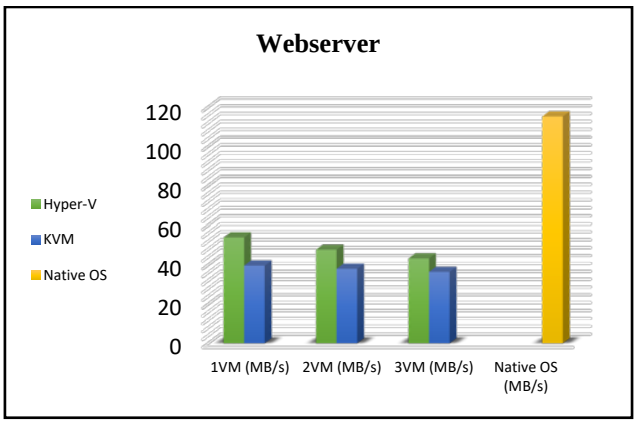


Figure 5. Webserver test results

For the "*Webserver*" workload, we can see that the Hyper-V is again solidly better than the KVM. In the Webserver workload, which has random read components and very few random write components, there is less influence of FS caching, so Hyper-V manages better, primarily due to the fifth component of formula (3), or FS pair (ext4 on NTFS in relative to ext4 to ext4) and the combined effect of FS caching.

 ISBN 978-86-7466-930-3

TABLE VII - RANDOMFILEACCESS BENCHMARK RESULTS

Random fileaccess	1VM (MB/s)	2VM (MB/s)	3VM (MB/s)	Native (MB/s)
MS Hyper-V	3153.46	2588.48	2056.96	
QEMU/ KVM	2121, 15	2007.26	1890.35	
Native OS				13780.52

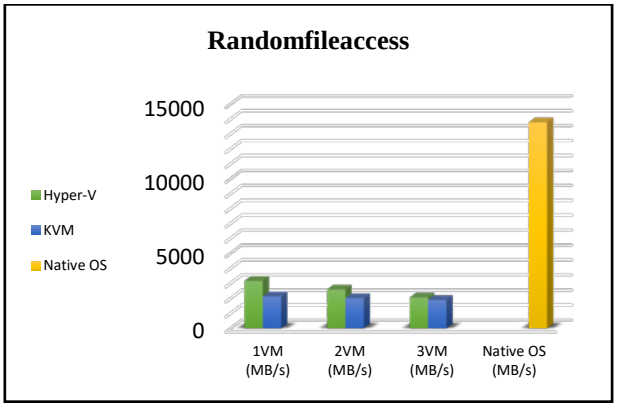


Figure 6. Randomfileaccess test results

For the "*Randomfileaccess*" workload, we again notice that Hyper-V is solidly better than KVM. In this workload, which has a lot of asynchronous random write components as well as random read components, there is a solid impact of FS caching, especially for random write, and for that reason Hyper-V performed better than KVM. This is primarily the effect of the fifth component of formula (3), NTFS, i.e. FS pair (ext4 on NTFS versus ext4 on ext4) and solid cache effect in random write.

VII. CONCLUSION

Virtualization has already proven itself in the field of information technology and has found an adequate place. In addition to all the benefits that this technology brings, it is necessary to emphasize that it's large share in the preservation of the human environment, and we can emphasize that it can be successfully used in the domain of green technologies. For the research presented in this paper, Hyper-V outperformed KVM in 3 out of 4 workloads, while in the most complex workload (*Fileserver*), KVM was dominant. For this kind of hardware and experiment, the crucial role in the differences in performance was brought by the difference in the file system of the host OS, the difference in the FS pair (ext4 on NTFS vs. ext4 on ext4). There are also differences in virtual hardware processing and hypervisor processing, which have proven to be the most complex workload (*Fileserver*). Future work in this area may focus on testing different types of servers, as well as other commonly used virtual platforms.

ACKNOWLEDGMENT

The work presented in this paper has been partially funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

LITERATURE:

[1] Christopher Strachey, "*Time Sharing in Large Fast Computers*". Online: https://archive.org/details/large-fast-computers/page/n5/mode/2up, 2021.

[2] Virtualization. Online: https://www.redhat.com/en/topics /virtualization /what-is-virtualization, 2021.

[3] Full Virtualization. Online: https://www.sciencedirect.com/topics/computer-science/full-virtualization, 2021.

[4] Matthew Portnoy: "*Virtualization Essentials*", 2016.

[5] Jadran Torbić, Ivan Stanković, Borislav S. Đorđević, Valentina Timčenko: "*Hyper-V and ESXi hypervisors comparison in Windows Server 12 virtual environment*", 17th International Symposium INFOTEH-JAHORINA, 2018.

[6] Microsoft, Online: https://www.microsoft.com/, 2021.

[7] B. Đorđević, V. Timčenko, N. Kraljević, N. Maček: "*File System Performance Comparison in Full Hardware Virtualization with ESXi, KVM, Hyper-V and Xen Hypervisors*", Advances in Electrical and Computer Engineering, vol.21, iss.1, 2021.

[8] Linux KVM, Online: https://www.linux-kvm.org/, 2021.

[9] Qemu process emulator, https://www.qemu-project.org/, 2021.

[10] M. Polenov, V. Guzik, V. Lukyanov: "*Hypervisors comparison and their performance*", Computer Science Online Conference, 2018.

[11] Filebench, Online: https://github.com/filebench/, 2021.

 ISBN 978-86-7466-930-3

A Review of Wazuh Tool Capabilities for Detecting Attacks Based on Log Analysis

Stefan Stanković, Slavko Gajin, and Ranko Petrović, *Member, IEEE*

***Abstract*— During the difficult times of the Covid pandemics and the transfer of work from the office to the home, security has never been more challenging. Because the development of information technology is expanding day by day, there is increasing amount of network traffic. Within that traffic, a potential attacker can often cover up his evil intentions. To detect attacks on host computer and prevent it from further malicious activities, Host Intrusion Detection Systems are often used. One of these systems is Wazuh and thanks to its powerful features it has been adopted by many companies. This paper provides an overview of the possibilities of Wazuh tools with a special emphasis on well-known attack detection on web servers.**

***Index Terms*—Wazuh, web-server, network, security, monitoring, attack.**

I. Introduction

The network monitoring system is used in the internal or external network in order to best identify risk components and prevent system crashes. The task of these systems is to find a weak point, submit a report and, if possible, solve the problem. Given the growing challenges in cyber security and the increasing amount of data generated globally, network and security administrators face a growing challenge. Most engineers use the Host Intrusion Detection System to detect and identify attacks and the Intrusion Prevention System to prevent them.

One of the main components of any application and an almost inevitable part of any data center is a web server - a hardware-software component that houses websites that serve end-users. Web servers are mostly exposed to the Internet and thus exposed to a large volume of potential attacks coming either from real attackers or from automated bots. Identifying attacks on web servers is a basic task of any administrator who maintains them because if protection is breached, the application may be inaccessible to a large number of users or permanently destroyed [1]. There are many network security monitoring solutions used worldwide [2]. One of the tools that helps identify and detect attacks on web servers is Wazuh. This paper will present the main features of the Wazuh tool installed in the University of Belgrade Computer Center.

Wazuh is a tool used around the world for various purposes. Paper [3] describes how the Wazuh tool can be used to test solutions that detect attacks within their constructed honeypot. On the other hand, Wazuh can integrate with machine learning as described in [4]. The advantage of this work is that it can serve network and security engineers very well in network and host security monitoring.

This work demonstrates Wazuh tools when collecting data exclusively from web servers. The results obtained through this paper give administrators an insight into what needs to be changed within their configurations in order to bring their servers and the entire infrastructure to the highest security level.

The paper is written in 4 major sections. After the introduction, Section 2 describes the basic functionalities of Wazuh tools and experimental setup, followed by Section 3, where statistic data are shown. Section 4 presents the results related to known attacks such as SSH (Secure Shell) brute force. The last section presents the main conclusions with ideas for future work.

II. Wazuh Overview and Experiment Setup

Wazuh is a free and open-source platform for threat detection and security monitoring according to predefined security rules. It can be used to monitor endpoints such as desktops, laptops, servers, or network devices such as firewalls and routers, and to aggregate and analyze data in real-time. Wazuh provides the following capabilities [5]:

- Security analytics - collection, aggregation, indexing and processing of security data, helping organizations detect intrusions, threats and behavioral anomalies.
- Intrusion Detection - Wazuh agents scan the monitored systems looking for malware, rootkits and suspicious anomalies. They can detect hidden files and processes.
- Log Data Analysis - Wazuh agents read operating system and application logs, and securely forward them to a manager for rule-based analysis.
- File Integrity Monitoring - Wazuh monitors the file system, identifying changes in content, permissions, ownership and attributes of files that need attention.
- Vulnerability Detector - Wazuh agents pull software inventory data and send this information to the server, where it is correlated with periodically updated CVEs (Common

Stefan Stanković is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia and with Vlatacom Institute oh High Technologies, 5 Milutina Milankovica, 11070 Belgrade, Serbia (email: stefan.stankovic@vlatacom.com).

Slavko Gajin is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (email: slavko.gajin@rcub.bg.ac.rs).

Ranko Petrović is with the Vlatacom Institute oh High Technologies, 5 Milutina Milankovica, 11070 Belgrade, Serbia (email: ranko.petrovic@vlatacom.com).

 ISBN 978-86-7466-930-3

Vulnerabilities and Exposures) databases, in order to identify well-known vulnerable software.

- Configuration Assessment - Wazuh monitors system and application configuration settings to ensure they are compliant with security policies and standards. Agents perform periodic scans to detect applications that are known to be vulnerable, unpatched, or insecurely configured.

A. *Wazuh components*

The Wazuh solution is based on the following 3 components [6]:

- Wazuh agent - Installed on endpoints such as laptops, desktops, servers or virtual machines, it provides prevention, detection and response capabilities. It supports Windows, Linux, MacOS, HP-UX, Solaris and AIX platforms.
- Wazuh server - It analyses data received from the agents, processing it through decoders and rules, and using threat intelligence to look for well-known indicators of compromise.
- Elastic Stack -Elastic Stack is a unified suite of open-source projects for log management, including Elasticsearch, Kibana, Filebeat, and others. The projects that are especially relevant to the Wazuh solution are: Filebeat, Elastic Search, Kibana. Filebeat is A lightweight forwarder used to transfer logs across a network, usually to Elasticsearch. It is used on the Wazuh server to transfer events and alerts to Elasticsearch. Elastic search is A highly scalable, full-text search and analytics engine. A flexible and intuitive web interface for mining, analyzing, and visualizing data. It runs on top of the indexed content in an Elasticsearch cluster. Wazuh web user interface has been fully embedded in Kibana, in the form of a plugin. Wazuh architecture.

B. *Wazuh architecture*

The Wazuh architecture is based on agents, running on the monitored endpoints, that forward security data to a central manager. Moreover, agentless devices (such as firewalls, switches, routers, access points, etc.) are supported and can actively submit log data via Syslog. The manager decodes and analyzes the incoming information, and passes the results along to an Elasticsearch for indexing and storage.

C. *Experiment setup*

The results described in this paper were collected from several web servers with CentOS operating system version of 7.9, located in the University of Belgrade Computer Centre. Wazuh agents are installed on them to send Wazuh manager data. Wazuh manager has been installed on a virtual machine with Ubuntu operating system. Alternative to the implementation on virtual machine, dockers can be used, according to [7]. Some Wazuh manager functionalities are not included by default, such as Vulnerability Detector.

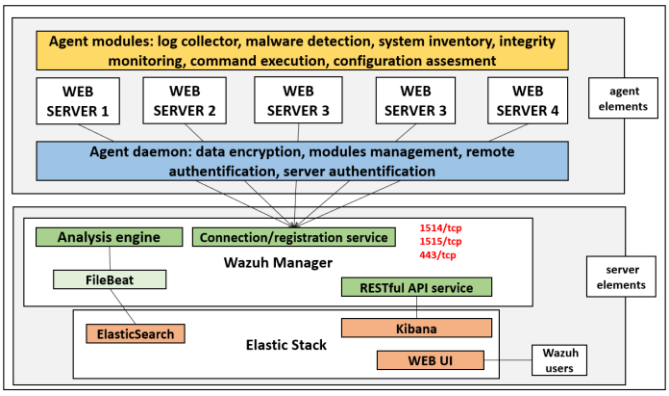


Fig. 1. Wazuh architecture on experiment setup

III. WEB SERVER ATTACK DETECTION OVERVIEW

In this section the results within each element of Wazuh Managers will be presented in brief outlines and special emphasis will be placed on the analysis of well-known attacks. Within the main dashboard, there are 4 basic sections with options in which you can monitor data in real-time.

A. *Security Information Management*

Within this module there are 2 units in which statistics on security events and integrity monitoring are located.

1) *Security events*

In this section, it is possible to search for all security events recorded within the Wazuh system. The operation of the system is based on agents that send data (logs) to the server where they are processed. There is a whole set of rules defined to identify threats. The results are processed and when a rule is met then it is recorded within the dashboard. By default, the rules are divided into 12 levels based on defined standards. Wazuh provides the option to write custom rules according to user needs. Figure 2 shows the sorted list of security alerts. We see that the 'Web server 400 error code' is the most prevalent error. In each unit within the Wazuh manager, it is possible to display the results in a given time range. Within each section, there is an option to generate reports and for better visibility top 10 alerts from each report will be displayed. Figure 2 presents the top 10 alerts from 31-Mar-2022 to 27-Apr-2022.

Description	Level	Count
Web server 400 error code.	5	603474
Auditd: SELinux permission check	3	473912
sshd: Attempt to login using a non-existent user	5	285016
PAM: User login failed.	5	247417
sshd: authentication failed.	5	215293
unix_chkpwd: Password check failed.	5	117091
syslog: User missed the password more than one time	10	50267
Multiple web server 400 error codes from same source ip.	10	30149
sshd: Reverse lookup error (bad ISP or attack).	5	19515
sshd: insecure connection attempt (scan).	6	15187

Fig. 2. Top 10 security alerts

2) *Integrity Monitoring*

In this module, it is possible to monitor the statistics of changes over system files on the host with the installed agent. These changes include modifying files, deleting files, and

 ISBN 978-86-7466-930-3

adding new files. Changes are detected based on the change in the checksum of each file.

Within the Events tab, it is possible to follow each change in detail, where you can find out the details of when a file was changed, which user did the action and what are the file permissions. This type of monitoring can be very useful, especially for sensitive systems. The performed actions and the list of files that are most often modified are shown in Figure 3.

Path	Action	Count
/etc/httpd/sites-available/h2020.rcub.bg.ac.rs.conf	modified	28
/etc/httpd/sites-available/lira.f.bg.ac.rs.conf	modified	28
/etc/httpd/conf/httpd.conf	modified	27
/etc/httpd/sites-available/amres.ac.rs.conf	modified	27
/etc/httpd/sites-available/amres.rs.conf	modified	27
/etc/httpd/sites-available/arhiva.fpu.bg.ac.rs.conf	modified	27
/etc/httpd/sites-available/arts.bg.ac.rs.conf	modified	27
/etc/httpd/sites-available/bpd.amres.ac.rs.conf	modified	27
/etc/httpd/sites-available/bsaae.bg.ac.rs.conf	modified	27
/etc/httpd/sites-available/careers.ac.rs.conf	modified	27
/etc/httpd/sites-available/cbp.rcub.bg.ac.rs.conf	modified	27

Fig. 3. The list of top modified files detected by Wazuh

B. *Auditing and Policy Monitoring*

This module offers 3 sections in which statistics and details about the system configuration and how much that configuration deviates from global standards.

1) *Policy monitoring*

This chapter shows the data obtained from the log analysis of policy monitoring. System configuration verification, such as kernel and security configuration files, is performed based on predefined rules. Wazuh uses 3 components to perform this task: Root check, OpenSCAP (Security Content Automation Protocol) and CIS-CAT. If some process is hidden from the virtual process file system (procfs), that file is marked as an alert.

Rule description	Control	Count
Possible kernel level rootkit	Anomaly detected in file '/etc/letsencrypt/.certbot.lock'.	1
Possible kernel level rootkit	Anomaly detected in file '/tmp/#sql-temptable-68c6-2faed-b5e7.MAD'.	1
Possible kernel level rootkit	Anomaly detected in file '/tmp/#sql-temptable-68c6-2faed-b5e7.MAI'.	1
Possible kernel level rootkit	Anomaly detected in file '/tmp/#sql-temptable-68c6-6e16f-f03c.MAD'.	1
Possible kernel level rootkit	Anomaly detected in file '/tmp/#sql-temptable-68c6-6e16f-f03c.MAI'.	1
Possible kernel level rootkit	Process '4859' hidden from /proc.	1
Possible kernel level rootkit	Process '8798' hidden from /proc.	1

Fig. 4. The list of detected anomalies

2) *System auditing*

This section presents data based on which user behavior can be monitored, command execution monitored and possibly an alert raised if sensitive files are accessed. User behavior is monitored by a powerful auditing facility called *auditd* which provides a detailed accounting of actions and changes in a system. Thanks to the Wazuh agent who sends *auditd* logs to the manager, administrators can have insight into users' behavior. The statistics from this section are shown in Figure 5.

Event	Command	Count
Auditd: SELinux permission check	/usr/sbin/php	192484
Auditd: SELinux permission check	/usr/sbin/php	119432
Auditd: SELinux permission check	/opt/remi/php74/root/usr/sbin/php	68985
Auditd: SELinux permission check	/usr/sbin/httpd	27612
Auditd: SELinux permission check	/usr/sbin/httpd	24822
Auditd: SELinux permission check	/usr/sbin/php	13641
Auditd: SELinux permission check	/usr/sbin/httpd	10602
Auditd: SELinux permission check	/opt/remi/php80/root/usr/sbin/php	4945
Auditd: SELinux permission check	/usr/sbin/postdrop	3292
Auditd: SELinux permission check	/usr/sbin/postdrop	2396
Auditd: SELinux permission check	/usr/libexec/postfix/smtp	2001

Fig. 5. SElinux permission checklist

3) *Security Configuration Assessment*

Within this unit, before displaying the data, it is necessary to select the agent where the configuration check will be performed. When the check is performed, the statistics and scores of that host are displayed. Verification is performed based on CIS (Center for Internet Security) benchmark recommendations for particular operating system distribution. The check consists of executing a set of commands whose result is binary: pass or fail. A detailed report is obtained when exported in CSV format where the columns show, among other things, commands, results, references and recommendations. This statistic gives a very good insight into the system configuration and draws the administrator's attention to important configuration elements.

C. *Threat Detection and Response*

In this unit there are 3 entities in which it is possible to gain insight into data related to threat detection. These entities are vulnerabilities, virus Total and MITRE ATT&CK, a globally accessible database of adversary tactics and techniques based on real-world observations. Evaluation of Wazuh tool with persistence tactic of MITRE ATT&CK is nicely described in [8]. As the targeted system is not related to antivirus software, no data has been collected.

1) *Vulnerabilities*

This module is not included in the main configuration file by default and needs to be enabled. It performs vulnerability searches according to the latest indexes that are updated in real-time with Canonical, RedHat and National Vulnerability databases. The detector on the manager inspects the list of installed applications periodically sent by the agents. Based on this list, search and verification processes are performed within the local database with the latest CVE elements (Common Vulnerabilities and Exposures). Alerts are generated when a CVE affects a package installed on one of the monitored servers. That package is then marked as vulnerable. There are 2 types of scanning: full and partial. A full scan is done the first time the Vulnerability Detector is activated and then each packet is scanned individually. Partial scanning is done when new packages are installed.

 ISBN 978-86-7466-930-3

Severity	Title	Published	CVE	Count
High	CVE-2021-32749 affects fail2ban	1626393600000	CVE-2021-32749	593
Low	CVE-2015-3243 affects rsyslog	1500940800000	CVE-2015-3243	498
Low	CVE-2019-15165 affects libpcap	1570060800000	CVE-2019-15165	498
Low	CVE-2019-1551 affects openssl	1575590400000	CVE-2019-1551	498
Low	CVE-2019-1563 affects openssl	1568073600000	CVE-2019-1563	498
Low	CVE-2020-1968 affects openssl	1599609600000	CVE-2020-1968	498
Low	CVE-2021-3601 affects openssl	1623715200000	CVE-2021-3601	498
Low	CVE-2021-3601 affects openssl-libs	1623715200000	CVE-2021-3601	498
Low	CVE-2021-3659 affects kernel	1617736920000	CVE-2021-3659	498
Low	CVE-2021-3659 affects kernel-tools	1617736920000	CVE-2021-3659	498

Fig. 6. The list of matched CVEs detected by Wazuh

2) *MITTRE ATT&CK*

This feature allows the user to customize the alert information to include specific information related to MITRE ATT&CK techniques. MITRE ATT&CK matrix stores all possible attacks that can be made and what to do to detect them and mitigate the risk [9]. This can be useful when an attack is detected through an alert and a user wants to know more about it. MITRE ATT&CK assigns each attack technique an ID (identification). These techniques are grouped by tactics (Defense Evasion, Privilege Escalation, etc.) although some of them belong to more than one tactic.

D. *Regulatory Compliance*

The Wazuh platform is often used to meet the technical aspects of regulatory compliance standards. Wazuh not only provides the necessary security controls such as host intrusion detection, configuration assessment, log analysis, and vulnerability detection, among others, to meet compliance requirements but also uses its SIEM (Security Information and Event Management) capabilities to centralize, analyse and enrich security data. In order to provide regulatory compliance support, the Wazuh rules have been mapped against compliance requirements [10]. This way, when an alert is generated (a rule condition has been matched), it automatically includes compliance information. The following standards are supported: Payment Card Industry Data Security Standard (PCI DSS), General Data Protection Regulation (GDPR), NIST Special Publication 800-53 (NIST 800-53), Good Practice Guide 13 (GPG13), Trust Services Criteria (TSC SOC2), Health Insurance Portability and Accountability Act (HIPAA)

IV. EXPERIMENT, RESULTS AND SYSTEM PERFORMANCE

As already mentioned, one of the attacks analyzed in more detail is the SSH brute force attack. a brute-force attack is performed by an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly. The attacker systematically checks passwords and passphrases from the database until the correct one is found. Also, if a user tries to connect via SSH using a random username, they will be considered to have attempted an attack and that alert will be recorded. Data processing on the Wazuh manager is in real-time and a potential attack is detected and alerted almost immediately. Table I provides very detailed information about the alerts when a potential attacker tried to connect as a non-existent user ‘dunja’.

The basic geo locations based on the source IP address are followed by information about the user trying to connect and the name of the decoder that analyzes the data. This is followed by a description of the full log and an abbreviated name as well as the original log file location.

TABLE I
SSH failed login overview

Parameter	Data
GeoLocation.city_name	Belgrade
GeoLocation.country_name	Serbia
GeoLocation.location	{ "lon": 20.4721, "lat": 44.8166 }
GeoLocation.region_name	Belgrade
_id	s2K6-H8Bao9qD9NQsksr
_index	wazuh-alerts-4.x-2022.04.05
data.srcip	217.24.19.131
data.srcport	11847
data.srcuser	dunja
decoder.name	sshd
decoder.parent	sshd
full_log	Apr 5 07:57:36 wazuh sshd[233488]: Invalid user dunja from 217.24.19.131 port 11847
id	1649145457
input.type	log
location	/var/log/auth.log
manager.name	wazuh
predecoder.hostname	wazuh
predecoder.program_name	sshd
rule.description	sshd: Attempt to login using a non-existent user
rule.gdpr	IV_35.7.d, IV_32.2
rule.gpg13	7.1
rule.groups	syslog, sshd, invalid_login, authentication_failed
rule.hipaa	164.312.b
rule.id	5710
rule.level	5
rule.mail	FALSE
rule.mitre.id	T1110
rule.mitre.tactic	Credential Access
rule.mitre.technique	Brute Force
rule.nist_800_53	AU.14, AC.7, AU.6
rule.pci_dss	10.2.4, 10.2.5, 10.6.1
rule.tsc	CC6.1, CC6.8, CC7.2, CC7.3
timestamp	Apr 5, 2022 @ 09:57:37.807

A. *System performance*

The Wazuh manager described in this paper is installed on an Ubuntu virtual machine which is assigned 8GB of RAM (Random Access Memory), 4 cores and 30GB of storage. It aggregates data from a total of 5 servers in the network. Of the allocated resources, the system uses about 4.7GB of RAM, a

 ISBN 978-86-7466-930-3

negligible percentage of CPU load, and about 11G of storage is filled. The system uptime is 2 months.

V. Conclusion

The aim of this paper is to present the functionality of the Wazuh tool in detecting attacks demonstrated on web servers. Web servers are components that are very exposed to the Internet and if they are not well protected, they are very susceptible to attacks of various kinds. In order to prevent attacks, they must first be detected and that is why Host Intrusion Detection systems are used. One of the solutions that can help is Wazuh. It is a powerful tool that displays all detected attacks in great detail and in real-time. Although this paper demonstrates the analysis of detected attacks on web servers, Wazuh is a tool used to analyze attacks across the entire infrastructure. This paper presents the basic principle of operation of Wazuh tools based on the agent-manager system. Agents installed on hosts send the log data for processing to the manager. Statistics of different types of attacks are presented and special attention is paid to details concerning some well-known attacks such as SSH brute force. The attack was successfully detected and shown almost immediately. As further work, the installation of agents on all infrastructure devices is proposed, without limiting on the type of device. It would be desirable to integrate the Wazuh tool with an antivirus software in order to get deep inspection on viruses, worms trojans and other malicious content. Some security issues are most successfully detected by inspecting a server's actual network traffic, which is generally not accounted for in logs. This is where a Network Intrusion Detection System can provide additional insight into security. One of those systems is Suricata. Because Suricata is capable of generating JSON (JavaScript Object Notation) logs of events, it has very good integration option with Wazuh, so this is also a proposal to future work.

ACKNOWLEDGEMENT

This paper was done at Vlatacom Institute on project P158 and partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia (grant number 2022/200103).

References

[1] M. Moh, S. Pininti, S. Doddapaneni, T.S. Moh "Detecting Web Attacks Using Multi-Stage Log Analysis", *6th IEEE International Conference on Advanced Computing*,2016, doi: 10.1109/IACC.2016.141

[2] I. Ghafir, V. Prenosil, J. Svoboda, M. Hammoudeh, "A survey on Network Security Monitoring Systems", 4th International Conference on Future Internet of Things and Cloud Workshops, 2016, DOI: 10.1109/W-FiCloud.2016.30

[3] R. M. Muhammad, I. D. Irawati, M. Iqbal "Integrated Security System Implementation for Network Intrusion", *Journal of Hunan University*, vol. *48*, no. *6*, pp. *183-188*, June, 2021.

[4] O. Negotia, M. Carabas "Enhanced Security Using Elastic Search and Machine Learning", *Advances in Intelligent Systems and Computing*, vol. *1230*, July, 2020.

[5] Wazuh documentation overview, https://documentation.wazuh.com/current/, last visited 28.4.2022

[6] Wazuh documentation components, https://documentation.wazuh.com/current/getting-started/components/index.html last visited 28.4.2022.

[7] F. Mulyadi, L. A. Annam, R. Promya and C. Charnsripinyo, "Implementing Dockerized Elastic Stack for Security Information and Event Management", 5th International Conference on Information Technology, 2020. DOI: 10.1109/InCIT50588.2020.9310950

[8] J. Chandler, "Evaluating Open-Source HIDS with Persistence Tactic of MITRE att&ck", SANS Institute, 2021.

[9] Wazuh documentation, mitre, available at https://documentation.wazuh.com/current/user-manual/ruleset/mitre.html, last visited 29.04.2022.

[10] Wazuh documentation regulatory compliance, available at https://documentation.wazuh.com/current/getting-started/use-cases/regulatory-compliance.html, last visited 29.04.2022.

 ISBN 978-86-7466-930-3

Infrastructure for Simulating n-Dimensional Simplicial Complexes

Dušan Cvijetić, Nenad Korolija, and Marko Vojinović

***Abstract*—We present an infrastructure for simulating simplicial complexes. The classes for storing the structure of simplicial complexes and simplices are explained in detail, for arbitrary dimension.**

The implementation is tested using functions for seeding simplicial complexes and for printing them on the screen. Beside these functions, the supporting classes and the function for assigning unique identifiers and screen coordinates is also explained.

Results of simulation show that there are potentials for the simulator to be used for big data problems, although appropriate experimental results are still being collected. Future work includes parallelizing the execution of the simulator using supercomputing architectures.

***Index Terms*— Simplicial complex; triangulation; manifold; algebraic topology.**

I. Introduction

A manifold is one of the fundamental concepts in mathematics [1], and its importance in applications in physics, technology and engineering cannot be overstated. Virtually all modern physics describes the world using *field theory* [2], in which all physical quantities (fields) are represented as functions over some manifold (for example, spacetime). In technology, manifolds appear in all forms and guises, whenever one needs to deal with curved surfaces --- from civil engineering to graphics in video games.

While most of the interest in science and engineering revolves around *smooth* manifolds, for the purpose of studying manifolds using numerical techniques, the attention focuses on the so called *piecewise-linear* manifolds [3], which can intuitively be imagined as a structure made out of small flat cells called *simplices*, arranged like bricks into a structure which models a manifold. The procedure of approximating a smooth manifold with a piecewise-linear one is commonly called *triangulation*, see Fig. 1.

Within the framework of algebraic topology, the formal mathematical structure which describes piecewise-linear manifolds is called a *simplicial complex*. For the purpose of this article, we provide an informal descriptive definition of a simplicial complex, without mathematical rigour. A simplicial complex is a combinatorial structure, containing the information about *simplices* of various dimensions that make up a complex, and the information about how simplices are connected to each other. A *k-simplex* is an elementary building block of a simplicial complex. It is an elementary geometrical "cell" of dimension k, which is being used to build simplices of higher dimension, and the entire simplicial complex. For k = 0, the simplex is called a *vertex*, it is represented geometrically as a single point, and has no internal structure. The k = 1 simplex is called an *edge*, geometrically represented as a single straight line, having two vertices at its boundary. For k = 2, the simplex is a *triangle*, having three boundary edges and three vertices. The case k = 3 describes a *tetrahedron*, having four boundary triangles, six edges and four vertices. One can go further into higher dimensions: k = 4 represents a simplex called *pentachoron* – it is a 4-dimensional figure, having five boundary tetrahedra, 10 triangles, 10 edges and five vertices. In general, one can introduce a k-simplex for arbitrary dimension k, also called *level*.

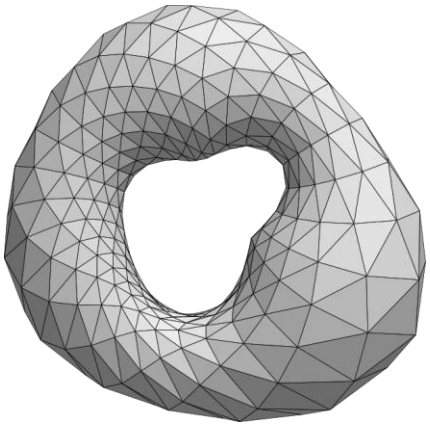


Fig. 1. Simplicial complex of a torus (source: Wikipedia).

Given a set of simplices, one can "glue them up" into a bigger geometrical structure, called simplicial complex. In order to describe a manifold of dimension D, a simplicial complex is constructed by gluing a set of D-simplices by identifying their common boundary (D-1)-simplices. Naturally, this implies the identification of all corresponding sub-simplices of level k < D-1 as well. The resulting simplicial complex is homeomorphic to a piecewise-linear manifold of dimension D.

The most important information about the simplicial complex, aside from its dimension D, is the data that tells one

Dušan Cvijetić is a student of the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: dusancvijetic2000 @ gmail.com).

Nenad Korolija is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: nenadko @ etf.bg.ac.rs).

Marko Vojinović is with the Institute of Physics, University of Belgrade, Pregrevica 118, 11080 Pregrevica, Serbia (e-mail: vmarko @ ipb.ac.rs).

 ISBN 978-86-7466-930-3

which simplices are glued to which. This gives rise to a notion of a *neighborhood* of a k-simplex, which is a set of all simplices which contain a given simplex as its sub-simplex (called super-neighbors) and simplices which are contained in a given simplex (called sub-neighbors). Each k-simplex (for $0 \leq k \leq D$) in the complex has its set of neighbors, where by definition a simplex is not a neighbor of itself (this is convenient to avoid infinite loops when traversing a complex). The neighborhood structure of the entire complex determines the *topology* of the corresponding manifold.

While manifolds of various topologies are important in their own right in mathematics, the applications in physics and engineering typically introduce functions over manifolds, such as distances, areas and volumes, temperature, electric and magnetic fields, etc. In the language of simplicial complexes, these functions are commonly called *colors*, and are assigned to simplices of various level k within the complex. Given a k-simplex, one can assign to it multiple colors, representing the value of a given function when evaluated on the k-simplex. A prototype example of colors is the geometry of a simplicial complex: each k-simplex is assigned its "size" according to its geometry --- each 1-simplex (an edge) is assigned a real number representing its length, each 2-simplex (a triangle) is assigned a real number representing its area, tetrahedra are assigned volumes, and so on. Other examples are abound --- vertices can be assigned a temperature, edges can be assigned vectors of electric field, and so on. Depending on the problem at hand, one may or may not impose relationships between various colors, such as that the area of a triangle is consistent with the length of its edges, or similar. These relationships are collectively called *constraints*.

In most everyday applications, one is interested in manifolds of dimension 1 and 2 (curves and surfaces). However, within the context of theoretical physics, one often needs to deal with manifolds of higher dimension – most commonly 3, 4, 5, 10, 11 and 26, while more sporadically anything in between and above. One of the typical scenarios is *quantum gravity* [4,5], a vast research area of fundamental theoretical physics, where the notion of spacetime is described as a piecewise-linear manifold of dimension D=4 or higher [6,7]. In order to apply numerical techniques to study the manifolds in such research disciplines, it is necessary to formulate and implement structures and algorithms which describe colored simplicial complexes of arbitrarily large dimension, in a uniform and optimal way. In what follows, we describe one such implementation, which is purposefully designed to mimic the mathematical structure of a simplicial complex as close as possible, while simultaneously providing efficient numerical techniques for the manipulation and study of such structures.

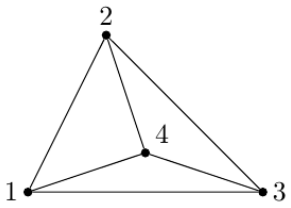
I. N-Dimensional Simplicial Complexes

This section describes the structure of simplicial complexes, and explains an example C++ implementation of classes for storing simplicial complexes.

Simplicial complexes consist of k-simplices at different levels. Given a simplicial complexes of dimension D, these elements include k-simplices for each level from zero to D. Elements at level zero are vertices, elements at level one are edges, elements on level two are triangles, etc. Finally, there are elements of highest level D. The representative source code of class for simplicial complexes is given in Algorithm 3 from the Appendix. The source code is pruned from comments and unnecessary functionalities for the presentation of the simulator.

K-simplex stores the level it has, the dimension of the simplicial complex it belongs to, neighboring elements and colors assigned to it.

Neighboring elements of a k-simplex are defined as k-simplices that this k-simplex is touching. Since these can be on various levels, the structure of neighbors is the same as for the simplicial complex. Therefore, the two main classes are mutually connected.




```
Printing SimpComp tetrahedron, D = 3
Simplices k = 0:
1, 2, 3, 4
Simplices k = 1:
(1-2), (1-3), (1-4), (2-4), (2-3), (3-4)
Simplices k = 2:
(1-2-3), (1-3-4), (1-2-4), (2-3-4)
Simplices k = 3:
(1-2-3-4)
```

Fig. 2. Tetrahedron and a corresponding output of the simplicial complexes simulator.

One possible implementation of the neighboring elements is to store only neighbors from one level above, and one level beneath (first sub-neighbors and first super-neighbors). The lower- and higher-level neighbors can be deduced following the structure of the first neighbors. However, we have opted for storing neighbors from all levels, giving us the opportunity to divide the structure onto multiple computing nodes and run the code in parallel. At current state, the simulator is running on a single CPU.

The instructions a CPU is executing are repeated over and over again, which makes this simulator suitable for acceleration using the dataflow paradigm [8,9]. The effort required for programming such architectures is higher than for conventional von Neumann architectures [10], but the simulator is suitable for transforming the C++ source code automatically [11]. Executing multiple simplicial complex operations in parallel requires appropriate scheduling

 ISBN 978-86-7466-930-3

techniques [12].

Each k-simplex (including all vertices, edges, triangles, etc.) can be colored with different types of color. Example colors include:

- k-simplex name,
- unique identifier of k-simplex,
- boundary color of k-simplex,
- screen coordinates.

These colors are included in our simplicial complex simulator, but the structure of the simulator allows adding additional user defined colors.

The representative source code of the class for k-simplices is given in Algorithm 4 from the Appendix. Just like it is the case with simplicial complexes, this source code is pruned for better clarity.

For simulation purposes, we have developed functions for seeding simplicial complexes at various levels, as it will be explained in the following section. In addition, coloring and printing simplicial complexes is also implemented. Pretty printing (or compact printing) prints k-simplices at all levels, where k-simplices of level higher than zero are printed as tuples consisting of unique identifiers (IDs) of their vertices. Fig. 2 shows an example tetrahedron (i.e. simplicial complex of dimension D = 3 consisting of a single 3-simplex and its sub-simplices) whose vertices are colored with unique identifiers that auto-increment after each assignment of the unique color to a vertex. Details of the implementation of compact printing is also explained in this manuscript.

Screen coordinates can be attached to vertices of the tetrahedron. Therefore, it can be drawn on the screen. However, there is no need to assign coordinates. They are just a convenient way to show an object on a screen. Similarly, there is no need to assign unique ID to any vertex. In the previous example, if a vertex with unique ID four would not have a unique ID assigned to it, the tetrahedron could still be printed out, but with word “Simplex” being printed out in place of number four.

II. Seeding Simplicial Complexes

This section describes seeding simplicial complexes using C++ implementation of function *seed_single_edge()*. The example source code for seeding a single edge is used for demonstrating purposes.

The process of seeding simplicial complexes will be explained using the source code shown in Algorithm 1. The source code is pruned from comments and unnecessary statements. Seeding a simplicial complex consists of the following steps, and statements in Algorithm 1 follow the same principle in the same order:

- creating an empty simplicial complex of given dimension,
- creating k-simplices for storing vertices and simplices of higher levels,
- connecting vertices at each level with vertices on higher and lower levels.

Adding a neighbor to a k-simplex is a symmetric operation. This means that both k-simplices (the calling one and the one given as an argument) are neighbors to each other. All functions of the simulator are written in a robust manner, checking the validity of input parameters.

Note that multiple colors can be assigned to each k-simplex, which is left out of consideration in this algorithm for better clarity.

III. Coloring and Pretty Printing K-simplices

This section describes coloring and pretty printing simplicial complexes. These functions might work in pair, but are not necessarily connected.

A. Coloring K-simplices

Coloring k-simplices will be explained using Algorithm 2 by coloring vertices of an edge with boundary colors. First, vertices have to be created as k-simplices of level zero. Then, colors have to be created for all vertices. Finally, colors need to be pushed back to the vector of colors that each k-simplex has.

Algorithm 1: Seeding a single edge.

```
SimpComp* seed_single_edge(string name){
    SimpComp *edge = new SimpComp(
            name, 1);
    KSimplex *v1 =
            edge->create_ksimplex(0);
    KSimplex *v2 =
            edge->create_ksimplex(0);
    KSimplex *e1 =
            edge->create_ksimplex(1);
    v1->add_neighbor(e1);
    v2->add_neighbor(e1);
    return edge;
}
```

Algorithm 2: Coloring vertices with boundary color.

```
KSimplex *v1 =
        edge->create_ksimplex(0);
KSimplex *v2 =
        edge->create_ksimplex(0);
Color *c1 = new BoundaryColor(true);
Color *c2 = new BoundaryColor(true);
v1->colors.push_back(c1);
v2->colors.push_back(c2);
```

Following colors are currently available:

- unique ID colors
- boundary colors
- screen coordinate colors.

Additionally, user is allowed to construct a custom color and use it within the simulator. The source code of the simulator is organized as a library, and user is allowed to extend it by using the library.

Unique ID colors are predominantly used for pretty printing simplicial complexes. They are implemented by a class inherited from the basic color class. Two main fields include

 ISBN 978-86-7466-930-3

static integer number, and an integer number. The first represents the current maximum of a unique color ID that is in use, and the second one is the color of a given k-simplex.

Unlike unique ID colors, boundary colors have special meaning. Each k-simplex may contain boundary color, but it does not have to. A simplicial complex can have boundaries on k-simplices of one level lower than the dimension of the simplicial complex. For example, a triangle can have edges as boundaries.

Screen coordinate colors are used for drawing simplicial complexes on a screen. The basic graphical user interface is under development.

B. Pretty Printing K-simplices

Printing k-simplices includes printing of all of the fields that *KSimplex* class contains. This includes printing all of the neighborhood elements the k-simplex has. This is usually overwhelming for a user. Therefore, pretty printing is designed to print unique ID colors of each k-simplex in most readable way authors could think of.

Function *KSimplex::print_compact()* is responsible for pretty printing. It assigns to the pointer to the unique ID a value returned by a function *get_uniqueID()* that returns either nullptr if a k-simplex doesn't have a unique ID, or a pointer to the color.

If there is no unique ID color assigned to a k-simplex, the output consists solely of word "Simplex". Otherwise, *print_compact()* function is called for a color that the pointer points to. Further, the following procedure is repeated, if level k is greater than zero and there are neighboring elements for all neighbors. A set of integer values is constructed, and then function *print_vertices_in_parentheses(s)* is called for neighbors, adding unique IDs to the set. This way, printing sorted values is achieved, along with avoiding duplicate values. Sample output of a simplicial complex pretty printing is shown in Fig. 1.

IV. Conclusion

We have demonstrated how one can implement in code the structure of a simplicial complex of arbitrary dimension, in a way that is faithful to its combinatorial definition, and perform the most basic operations on it, like instantiating, coloring and printing.

The implementation of the basic classes of the code described in this work represents a fundamental basic building block for a more versatile software collection that aims to construct, manipulate and study the properties of simplicial complexes of arbitrary dimension. Future extensions of the software library will include the functions which implement attaching additional simplices to a boundary of a complex, performing Pachner moves [13] which transform a given complex into a different one without changing its topology, and functions for manipulating the colors and evaluating various mathematical constructions that include them. Note that the experimental data regarding the parallelization is yet to be collected (see the accompanying paper [14]).

The resulting software collection will feature the generality and versatility that aim for applications both in pure mathematics (algebraic topology research) and theoretical physics (quantum gravity, field theory), but also with potential applications in other disciplines of engineering and industry, wherever the analysis and the study of geometry of manifolds and curved surfaces may be relevant.

Appendix

Algorithm 3: Declaration of SimpComp class.

```
class SimpComp{
public:
    SimpComp(int dim);
    SimpComp(string s, int dim);
    ~SimpComp();
    int count_number_of_simplexes(
            int level);
    void print(string space = "");
    bool all_uniqueID(int level);
    void collect_vertices(set<int> &s);
    void print_set(set<int> &s);
    void print_vertices_in_parentheses(
            set<int> &s);
    void print_compact();
    // Creating new KSimplex at level k:
    KSimplex* create_ksimplex(int k);
    void print_sizes();

    string name;
    int D;
    // An element at each level
    // is a list or vector
    // of KSimplex pointers
    // to KSimplex on that level:
    vector< vector<KSimplex *> >
            elements;
};
```

Algorithm 4: Declaration of KSimplex class.

```
class KSimplex{
public:
    KSimplex();
    KSimplex(int k, int D);
    ~KSimplex();
    bool find_neighbor(KSimplex *k1);
    void add_neighbor(KSimplex *k1);
    void print(string space = "");
    UniqueIDColor* get_uniqueID();
    void print_compact();

    int k; // level
    int D; // dimension
    SimpComp *neighbors;
    vector<Color *> colors;
};
```

Acknowledgment

DC and NK were partially supported by the School of

 ISBN 978-86-7466-930-3

Electrical Engineering, University of Belgrade, Serbia. NK was partially supported by the Institute of Physics Belgrade, contract no. 0801-1264/1. MV was supported by the Science Fund of the Republic of Serbia, grant no. 7745968, "Quantum gravity from higher gauge theory" – QGHG-2021. All authors were partially supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

REFERENCES

[1] M. W. Hirsch, *Differential Topology*, New York, USA: Springer Verlag, 1976.

[2] A. Hobson, "There are no particles, there are only fields", *Amer. Jour. Phys.* **81**, 211-223 (2013).

[3] E. H. Spanier, *Algebraic Topology*, New York, USA: Springer Verlag, 1966.

[4] C. Rovelli, *Quantum Gravity*, Cambridge, UK: Cambridge University Press, 2004.

[5] C. Rovelli and F. Vidotto, *Covariant Loop Quantum Gravity*, Cambridge, UK: Cambridge University Press, 2014.

[6] T. Radenković and M. Vojinović, "Higher Gauge Theories Based on 3-Groups", *JHEP* **10**, 222 (2019).

[7] A. Miković and M. Vojinović, "Standard Model and 4-Groups", *Europhys. Lett.* **133**, 61001 (2021).

[8] B. Lee and A. R. Hurson, "Issues in dataflow computing," *Advances in computers*, Elsevier, **37**, 285-333 (1993).

[9] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, "Transforming applications from the control flow to the dataflow paradigm," *Dataflow supercomputing essentials*, Springer, Cham, 107-129 (2017).

[10] J. Popovic, D. Bojic, and N. Korolija, "Analysis of task effort estimation accuracy based on use case point size," *IET Software*, **9(6)**, 166-173 (2015).

[11] N. Korolija, J. Popović, M. Cvetanović, and M. Bojović, "Dataflow-based parallelization of control-flow algorithms," *Advances in computers*, Elsevier, **104**, 73-124 (2017).

[12] N. Korolija, D. Bojić, A. R. Hurson, and V. Milutinovic, "A runtime job scheduling algorithm for cluster architectures with dataflow accelerators," *Advances in computers*, Elsevier, **126** (2022).

[13] U. Pachner, "PL homeomorphic manifolds are equivalent by elementary shellings", *Eur. Jour. Combinat.* **12**, 129-145 (1991).

[14] D. Cvijetić, N. Korolija and M. Vojinović, "Possibilities for Parallelizing Simplicial Complexes Simulation", IcETRAN 2022, Novi Pazar, Republic of Serbia, June 6-9, 2022, Belgrade: Društvo za ETRAN, Beograd: Akademska misao (2022).

 ISBN 978-86-7466-930-3

Possibilities for Parallelizing Simplicial Complexes Simulation

Dušan Cvijetić, Nenad Korolija, and Marko Vojinović

***Abstract*—This manuscript presents potentials for parallelizing simulation of simplicial complexes. The implementation of most important fields and methods of classes for storing simplicial complexes and k-simplices is followed by wrapper classes for simplicial complexes and k-simplices respectively. Infrastructure for communication between Message Passing Interface (MPI) processes along with helper functions is explained further in the manuscript. Once multiple data are prepared to be sent from each MPI process to other MPI processes, sending and receiving is performed in the background. Because of the stall introduced by using MPI directives, the amount of data to be transmitted is maximized by processing multiple operations over simplicial complexes in parallel. This requires the method for locking simplicial complexes and k-simplices by the owner MPI process until all the requests are processed. Locking mechanism and supporting simplicial complex class actions regarding locking is not in the scope of this manuscript.**

***Index Terms*—Simplicial complex; k-simplex; triangulation; manifold; MPI; parallelization.**

I. Introduction

In modern theoretical physics, a lot of problems are too complicated for study using analytical methods, and one needs to resort to numerical techniques. Among those problems, an especially important class deals with evaluation of functions over simplicial complexes. A simplicial complex [1] is a piecewise-linear approximation of a smooth spacetime manifold [2] and is typically 4-dimensional or higher. Functions over a simplicial complex represent physical fields on spacetime, and one commonly employs path integral evaluations of such structures to extract expectation values of observables. For example, in Lattice Quantum Chromo-dynamics, one employs such numerical techniques to predict the theoretical values for the masses of elementary particles called hadrons [3]. Also, in Causal Dynamical Triangulations approach to quantum gravity [4,5], one uses these techniques to evaluate spectral dimension of spacetime, and study various properties of phase space of triangulated manifolds. Finally, in the Regge Quantum Gravity approach [6,7,8] one can study the entanglement properties of matter fields and gravity described by the Hartle-Hawking wavefunction [9,10], again using the techniques of numerical evaluation of path integrals over simplicial complexes.

It goes without saying that all such calculations are exceptionally expensive in computation time. Typically, one develops custom-made code, heavily optimized to solve precisely one specific problem, and executes it over months-long periods on hardware dedicated for high performance computing (HPC), usually clusters with thousands of work nodes. Such enormous calculational efforts are usually unavoidable due to the nature of the problems that need to be solved.

Nevertheless, at least for one class of such problems, it may be possible to construct a more general algorithm and structures which would provide a common basis for solving an all-encompassing class of problems using the same underlying software, while intrinsically exploiting the parallelization possibilities of the code itself and the distributed nature of the underlying hardware. Our aim is to develop such a generic software library, which could be used to solve a whole host of physics problems in the same way and optimize it for parallelized HPC environments. In this work we present the first steps towards the construction of such a library. This approach of developing common code for a whole class of problems has not been attempted so far because research teams are usually concentrated on solving only one specific problem and opt to construct custom code for that problem. However, in our opinion, a generic software library, which would provide support for a whole class of problems simultaneously, would open new avenues for numerical research, since one could use the same code to study new, yet unexplored problems as well as old well-known ones.

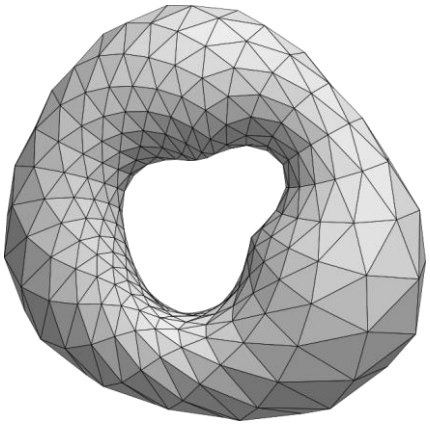


Fig. 1. Simplicial complex of a torus (source: Wikipedia).

The fundamental structure which lies at the core of the whole numerical method is the notion of a *simplicial complex*. A simplicial complex is a combinatorial structure which is easiest to understand as a generic lattice-like mesh,

Dušan Cvijetić is a student of the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: dusancvijetic2000 @ gmail.com).

Nenad Korolija is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: nenadko @ etf.bg.ac.rs).

Marko Vojinović is with the Institute of Physics, University of Belgrade, Pregrevica 118, 11080 Pregrevica, Serbia (e-mail: vmarko @ ipb.ac.rs).

 ISBN 978-86-7466-930-3

whose cells are called *simplices*, and are connected to each other along their boundaries to form the simplicial complex of a given dimension. The purpose of the whole structure is to approximate the smooth spacetime manifold with a discrete structure which is more convenient for numerical methods.

The most elementary simplex is a simplex of level zero, often called *0-simplex* or *vertex* – it is just a dimensionless point with no structure. Next is the *1-simplex*, also called an *edge* – it is a one-dimensional line with two vertices at its boundaries. At level two we have the *2-simplex* or *triangle*, whose boundary are three edges and their vertices. The *3-simplex*, also known as the *tetrahedron*, has the boundary made of four triangles and their edges and vertices. The procedure of constructing simplices can be done for arbitrary dimension, giving rise to the notion of a *k-simplex*, whose level (i.e. natural dimension of space in which it is defined) is equal to any positive integer k. The most commonly used example is the *4-simplex*, also called *pentachoron* – a 4-dimensional figure whose boundary consists of 5 tetrahedra, 10 triangles, 10 edges and 5 vertices. In most applications in physics, the spacetime manifold is considered to be 4-dimensional, and it is cut into a lattice-like structure made of 4-simplices, which are glued together along their boundary tetrahedra. The resulting structure is a simplicial complex of dimension 4. Fig. 1 depicts an intuitive example of a 2-dimensional simplicial complex of a torus.

Given a simplicial complex, one typically wants to introduce functions that are evaluated on it. These are commonly called *colors* and are assigned via their values to each k-simplex within in the complex. In other words, some colors live on vertices, some on edges, some on triangles, and so on. The colors are a natural discretization of the notion of a *field* over a manifold. For example, just like electric and magnetic fields have a value at each point of a smooth spacetime, analogously the colors have values at each k-simplex in the simplicial complex.

Depending on the type of the problem at hand, algorithms that are used to evaluate required quantities on a simplicial complex can vary in complexity, from conceptually simple Monte Carlo integration techniques, to vastly complicated traversal and ray-tracing algorithms, to various methods for solving functional partial differential equations. Due to the variability of the complexity of all these algorithms, dictated by the nature of the problem at hand, it is helpful to develop the underlying software simulator to exploit the parallelization avenues that are intrinsic to the simplicial complexes and k-simplices themselves, so that the simulator can exploit parallel hardware environments even for algorithms that are themselves hard to parallelize. This helps the code developer with overall optimization and application to HPC hardware architectures. In what follows, we shall demonstrate a set of possible approaches to these intrinsic parallelization techniques.

II. N-Dimensional Simplicial Complexes

This section describes data structures used in the simulator of simplicial complexes from the point of view of their suitability for parallelizing the simulator execution. Data demanding structures are of main interest for optimizing the communication between processing units. Along with those, data that describes the structure and needs to be updated on multiple processing units will be described in detail. Further, the amount of data that needs to be exchanged and the frequency of expected changes will be compared to the pyramid, where top elements demand less memory, but require more often communication.

The parallelization is simulated using the MPI framework. The simulator is implemented in C++, and, as a result, the parallelization framework is built on top of the simulator. As improving the simulator of simplicial complexes is an ongoing process, the possibility for accelerating the computation is simulated based on the requirements.

Simplicial complexes are formed out of k-simplices at various levels. Simplicial complexes at level zero represent vertices. The structure of each vertex is stored in *KSimplex* class. Simplicial complexes at level one represent edges. Each edge consists of two vertices. As it is the case with vertices, information about edges are also kept in a *KSimplex* class. However, while vertices can be independent of other vertices, representing separate simplicial complexes, each edge must have at least two vertices defined as neighbors. Neighbor of an k-simplex is defined also as a k-simplex that the first k-simplex relies on. Neighboring relation is symmetrical. Therefore, if two vertices are neighbors of an edge, edge is also the neighbor of both vertices. Further, edges can form a triangle. By analogy, neighbors of triangle are three edges, but also the triangle is neighbor of these edges. The neighboring relation spans more than one level up or down. The triangle has also three vertices as neighbors and the opposite.

Simplicial complex representing a triangle consists of a k-simplex representing a triangle along with all neighbors of the triangle. Simplicial complex class is used for storing information about simplicial complexes. As it has elements field that is a pointer to pointer of k-simplices, it is also used for keeping neighbors of each k-simplex.

III. Parallelizing Simplicial Complexes Simulation

Parallelizing operations over simplicial complexes is implemented by splitting the structure over multiple MPI processes. First, we can consider a single simplicial complex system, as the most general approach. If no screen coordinates for k-simplices are assigned, we can artificially assign this type of color, so that we can present k-simplices in 2D space. Further, we can imagine multiple planes, where each plane is responsible for keeping k-simplices of one dimension. This way, we can consider n-dimensional simplicial complex as a pyramid that we observe from the bird's eye view. Now we could have a bottom-up approach, where k-simplices of dimension zero are divided onto MPI processes based on their screen coordinates. Going up, each MPI process would store higher dimensional k-simplices that have those that are one level below as their neighbors. When a k-simplex has neighbors on one level below that belong to multiple MPI processes, this k-simplex gets copied to all MPI processes involved. Finally, all MPI processes would keep the highest-level k-simplex. In the case of multiple simplicial complexes, they could be split over MPI processes based on the same bottom-up approach.

 ISBN 978-86-7466-930-3

The notion of determining the MPI process where a k-simplex is located is hidden by using wrapper functions, so that the calculation operations are performed as if all k-simplices would have been on the same MPI process, i.e. as if the simulation was executed serially. Each wrapper function can keep either a pointer to the structure, if it exists on the same MPI process, and the ID used for finding the structure on the owner MPI process.

Algorithm 1 describes the most important aspects of simplicial complex classes. First, a basic *SimpComp* class is given, followed by the wrapper class *VirtualSimpComp* used for parallelization.

Algorithm 1: Declaration of simplicial complex classes.

```
class SimpComp{
public:
    SimpComp(int dim);
    SimpComp(string s, int dim);
    ~SimpComp();
    // Creating new KSimplex
    // at level k:
    VirtualKSimplex* create_ksimplex(
        int k);
    void update_owner(int owner);

    string name;
    int D;
    vector< vector<
        VirtualKSimplex *> > elements;
};
class VirtualSimpComp{
public:
    SimpComp *find_simpcomp;

    int id;
    int ownerRank;
    SimpComp *simpComp;
};
```

Algorithm 2 describes the most important aspects of k-simplices classes. A basic *KSimplex* class is followed by the wrapper class *VirtualKSimplex* used for parallelization.

Algorithm 2: Declaration of k-simplex classes.

```
class KSimplex{
public:
    KSimplex();
    KSimplex(int k, int D);
    ~KSimplex();
    bool find_neighbor(
            VirtualKSimplex *k1);
    void add_neighbor(
            VirtualKSimplex *k1);

    int k; // level
    int D; // dimension
    VirtualSimpComp *neighbors;
    vector<Color *> colors;
};
class VirtualKSimplex{
public:
    KSimplex *find_ksimplex();
    int id;
    int ownerRank;
    KSimplex *ksimplex;
};
```

In both algorithms, wrapper functions store a pointer to the base class object, if such exists on a local MPI process. Otherwise, the value is *nullptr*, and the data is searched for on the so called *ownerRank* based on unique identifier called *id*. Owner of this k-simplex can issue multiple requests while it holds a lock.

IV. Infrastructure for Communication Between MPI Processes

The communication between MPI processes is organized as follows. Each MPI process is preparing the data to be sent to other MPI processes. Order of operations prepared for other MPI processes is not important. All requests to other MPI processes for processing are packed in to_rank vector of vectors of unsigned char.

Each type of primitive data is serialized into the array of unsigned characters as it will be explained in the following section. Each prepared byte is pushed to the back of the vector of unsigned characters. Once all the data is prepared, the data is sent to other MPI processes in the background using *MPI_Isend* directive. If a reference to the vector of array of unsigned characters is called *vec*, the pointer to the array is obtained by calling member function *data()* of vector class from standard template library. After issuing all *MPI_Isend* directives, waiting for each of sending to finish is achieved using *MPI_Wait*.

Similarly receiving the data from other MPI processes is implemented in the background using *MPI_Irecv*, followed by *MPI_Wait*, once the data is needed for the processing. The data is received into array of unsigned characters, that is further packed into vector of vectors of unsigned characters called from_rank for simple processing.

V. MPI Supporting Functions

As already mentioned, variables are serialized into the array of unsigned characters using the following syntax:

```
*( (__typeof__ (variable) *) (array + nArray) ) = variable;
nArray += sizeof(variable);
```

Here, *array* is array of unsinged characters where the data stored in the variable is serialized, and *nArray* is the number serialized bytes in the array.

Similarly, a variable is read and prepared into the to_rank using the following syntax:

```
__typeof__ (variable) temp_var = variable; \
int nBytes = sizeof(temp_var); \
for(int iByte = 0; iByte < nBytes; iByte++) \
   to_rank[rankNumber].push_back(
        ((unsigned char *) &temp_var) [iByte] );
```

This can be further optimized, but the optimization is out of the scope of this research.

 ISBN 978-86-7466-930-3

The communication between MPI processes is continued for as long as any MPI process requires further communication with other MPI processes. This is achieved using the following source code, where the MPI process that requires further communication sets variable *to_send* to one:

```
int to_receive = 0; // A rank required communication
MPI_Allreduce(&to_send, &to_receive, 1, MPI_INT,
      MPI_SUM, MPI_COMM_WORLD);
```

After *MPI_Allreduce* is executed, all MPI processes will have the information whether they have to communicate further in *to_receive* variable.

VI. Parallelization Possibilities Using Dataflow Paradigm

This simulator issues the same set of computer architecture instructions repeatedly. As in majority simulator of physical phenomena, the number of instructions is dependent on the precision of the model and is limited by the computing resources and the total simulation time requirement. These conditions are exactly what is required for a program to be suitable for acceleration using the dataflow paradigm [11]. Programming dataflow architectures requires programming skills that are higher than those needed for programming conventional von Neumann architectures. One of the possibilities is to write a program in a VHDL. More suitable solution to most of the programmers would be to exploit the framework that enables writing source code in a Java-like language, which gets automatically translated into the FPGA image [12,13]. Even in this case, the effort needed for programming such architectures is higher [14]. Besides programming dataflow architecture for the simplicial complex simulator, appropriate scheduling scheme is also needed for efficient running of multiple jobs simultaneously [15].

As the number of operations that can be applied to simplicial complexes can lead to several days' simulation time or even more, having in mind the aging and the probability of failure of supercomputing nodes [16], we have decided to write restarts after given number of simulations defined by the user, so that the calculation can continue from the last stored state.

VII. Conclusion

In this work we have presented the basics of the paralellization techniques that can be applied to the structure of a simplicial complex, which underlies a host of research problems in theoretical physics (see also our accompanying paper [17]). These problems tend to be computationally extremely expensive, and the common underlying software that enables parallelization at the level of the basic data structure can possibly go a long way towards optimization of code for numerical study using heavily parallel hardware platforms such as HPC clusters. In particular, the simplicial complex naturally allows for various aspects of parallelization, and we have described the basic classes, corresponding MPI communication infrastructure, supporting functions and the dataflow paradigm employed for the construction.

One should note that our work represents just a first step towards a full working software implementation, and much more effort is needed to properly implement, optimize and test the resulting code in real world environments. All that is the topic for future work. In particular, the data regarding the experimental evaluation, which would compare the proposed parallelization method to ordinary sequential methods still needs to be gathered and analyzed. Nevertheless, this first step is fundamental, and it is conceptually important since it represents a paradigm in which parallelization is implemented dominantly at the level of the simplicial complex as the underlying data structure, rather than at the level of the particular algorithm that aims to solve some particular problem using these data structures.

Finally, we note that our code, once properly developed, may possibly find applications not just in theoretical physics, but also in other disciplines of science, technology and engineering.

Acknowledgment

DC and NK were partially supported by the School of Electrical Engineering, University of Belgrade, Serbia. NK was partially supported by the Institute of Physics Belgrade, contract no. 0801-1264/1. MV was supported by the Science Fund of the Republic of Serbia, grant no. 7745968, "Quantum gravity from higher gauge theory" – QGHG-2021. All authors were partially supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

References

[1] E. H. Spanier, *Algebraic Topology*, New York, USA: Springer Verlag, 1966.

[2] M. W. Hirsch, *Differential Topology*, New York, USA: Springer Verlag, 1976.

[3] S. Durr, Z. Fodor, J. Frison, C. Hoelbling, R. Hoffmann, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, K. K. Szabo and G. Vulvert, "Ab-initio Determination of Light Hardon Masses", *Science* **322**, 1224-1227 (2008).

[4] J. Ambjorn, A. Goerlich, J. Jurkiewicz and R. Loll, "Nonperturbative Quantum Gravity", *Phys. Rep.* **519**, 127 (2012).

[5] M. Vojinović, "Causal dynamical triangulations in the spincube model of quantum gravity", *Phys. Rev. D* **94**, 024058 (2016).

[6] T. Radenković and M. Vojinović, "Higher Gauge Theories Based on 3-Groups", *JHEP* **10**, 222 (2019).

[7] A. Miković and M. Vojinović, "Standard Model and 4-Groups", *Europhys. Lett.* **133**, 61001 (2021).

[8] A. Miković and M. Vojinović. "Quantum gravity for piecewise flat spacetimes", *SFIN* **XXXI**, 267 (2018).

[9] N. Paunković and M. Vojinović, "Gauge protected entanglement between gravity and matter", *Class. Quant. Grav.* **35**, 185015 (2018).

[10] J. B. Hartle and S. W. Hawking, "Wave function of the Universe", *Phys. Rev. D* **28**, 2960 (1983).

[11] B. Lee and A. R. Hurson, "Issues in dataflow computing," *Advances in computers*, Elsevier, **37**, 285-333 (1993).

[12] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, "Transforming applications from the control flow to the dataflow paradigm," *Dataflow supercomputing essentials*, Springer, Cham, 107-129 (2017).

[13] N. Korolija, J. Popović, M. Cvetanović, and M. Bojović, "Dataflow-based parallelization of control-flow algorithms," *Advances in computers*, Elsevier, **104**, 73-124 (2017).

[14] J. Popovic, D. Bojic, and N. Korolija, "Analysis of task effort estimation accuracy based on use case point size," *IET Software*, **9(6)**, 166-173 (2015).

 ISBN 978-86-7466-930-3

[15] N. Korolija, D. Bojić, A. R. Hurson, and V. Milutinovic, "A runtime job scheduling algorithm for cluster architectures with dataflow accelerators," *Advances in computers*, Elsevier, **126** (2022).

[16] K. Huang, Y. Liu, N. Korolija, J. M. Carulli, and Y. Makris, "Recycled IC detection based on statistical methods," *IEEE transactions on computer-aided design of integrated circuits and systems*, **34(6)**, 947-960 (2015).

[17] D. Cvijetić, N. Korolija and M. Vojinović, "Infrastructure for Simulating n-Dimensional Simplicial Complexes," IcETRAN 2022, Novi Pazar, Republic of Serbia, June 6-9, 2022, Belgrade: Društvo za ETRAN, Beograd: Akademska misao (2022).

 ISBN 978-86-7466-930-3

The Evolution of Big Data Analytics Solutions in the Could

Danko Miladinović, Jovan Popović, and Nenad Korolija

***Abstract*—Big data analytics is a very important topic both for enterprises, science, and government institutions. The amount of data that is generated is exponentially increasing and the need to analyze data is more important every year. Big data analytics evolved over the past decade from a large on-premises infrastructure for storing and processing data to modern cloud environments. In this paper we discuss how big data analytics evolved over the years and what are the future trends in this area.**

***Index Terms*—Big data; cloud; analytics; machine learning; databases.**

I. Introduction

BIG data analytics is one of the most important topics in the IT industry. There is a well-known "Data is the new oil" expression that points out the importance of data analytics that can have a huge effect on modern businesses and economy.

Most of the enterprises either leverage information from their data or plan to extract information from the data they own. Data science and analytics became more important for strategic growth of many organizations.

The organizations and software systems are continuously increasing the amount of data that is generated. Relatively big organizations must face the large amount of data that contains the information important for business decisions. Globally, we are now talking about the Exabyte to Zettabyte scale of data that needs to be processed. The global estimates are that the amount of data to be processed would reach multiple Zettabytes in this decade [1].

In this paper we discuss the industry trends and standards for big data analytics with a focus on data analytics in the cloud. This manuscript describes the solutions offered by the open-source community and the biggest commercial data analytics vendors that pave the way that will be followed by companies. The rest of the document is organized in the following sections:

- In the first section, we will talk about the main problems that impact big data analytic solutions.
- The cloud analytics section describes what are the main benefits of the cloud environments for big data analytics.
- Data analytic solutions section describes the two mainstream approaches for storing and analyzing data: Datawarehouse and Datalakehouse solutions.
- In the data format section, we will discuss the most important aspect of big data analytics – the format that is optimized for storing data.
- The conclusion section summarizes the trends for modern big data analytics.

Danko Miladinović is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: danko@etf.bg.ac.rs).

Jovan Popović is with the Microsoft Research and Development Center, Belgrade. Španskih boraca 3/3, 11000 Belgrade, Serbia (e-mail: jovanpop@microsoft.com).

Nenad Korolija is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: nenadko@etf.bg.ac.rs), (https://orcid.org/1234-1234-1234-123X).

II. Problem statement

The main problem in big data analytics is the size of data. There are many problems that can be solved by analyzing data stored in files and spreadsheets containing gigabytes of data, or even the relational or NoSQL databases that can contain and process terabytes of data. However, there are many domains where the data contains petabytes of data that cannot be stored in a limited set of files or classic database systems.

The researchers and engineers tried to solve the problem of big data processing using the following approaches:

- Datawarehouses that try to stretch the capabilities of the relational databases by applying distributed processing over large data.
- File-based processing systems that try to build an infrastructure for storing a large amount of data. One of the most widely used solutions is Hadoop with HDFS file system [2].

The big data analytic solutions must ensure that users can store Exabyte scale data and ensure that there is enough compute power to process the data when needed. The infrastructure teams must ensure that they have enough hardware (processors and disk storage) to fulfil the user needs, but at the same time to ensure that the resources are not underutilized or constantly over-provisioned.

Solving the problem of ensuring the required resources for big data analytics, but at the same time not over-provisioning them, appeared to be too hard for the on-premises infrastructure. Planning for the resources could not be both cost effective and ensure enough capacity that will be utilized for most of the time.

Therefore, most of the organizations tried to solve this resource management problem in the cloud making the cloud analytics the mainstream in the big data analytics space.

III. The cloud analytics

Over the past years, the clouds became a very important

 ISBN 978-86-7466-930-3

choice for modern big data analytics. There are three main benefits of cloud infrastructure that make them important for big data analytics:

- Large amount of storage that could be used to store any amount of data. The "Data Lake" [3] is a commonly used term for virtually unlimited storage where the organizations might store petabytes of data. As the amount of data rapidly rises, the organizations need to have an infrastructure that will guarantee that they can easily store Exabytes of data, and with the possibility to scale to Zettabytes in the future.
- Large amount of available compute power that could be used for data processing [4]. The compute power needed to analyze data is proportional to the data size and might easily span to the thousands of CPU cores needed to complete the data analytic tasks.
- Cloud resources can be used on-demand and released when they are not needed. This is one of the main reasons for choosing the cloud environment. Most of the data analytic jobs are not continuously processed and might require thousands of CPU cores for data processing, and then suddenly release all the resources. Cloud providers solve this problem using the economy of scale – with the large number of customers there is a high probability that someone will use them once others release them.

We should note that the could environment is not an absolute requirement to have an infrastructure for storing a large amount of data and use thousands of computer cores to process the data. The organizations might use their own data centers, supercomputers, and any custom-built architecture that will organize hundreds or thousands of computers that process the data. This was a common solution for the organizations who built their own Hadoop/HDFS infrastructure [2] for in-house analytics. However, the cost of management includes the need to maintain and replace the hardware, ensure that the infrastructure has enough compute power to satisfy the peak processing, but also to make sure that the capacity is not too underutilized during the period when nobody is executing analytic jobs.

The current trend is that most of the organizations are deciding to delegate the resource management to the cloud providers and utilize the resources on-demand when they need to run some analytics.

The data analytics solutions should not be misguided with the infinite scale claims of the cloud vendors. Cloud environments are built as many computers that are working together to process tasks. The applications see the sum of the compute power, memory, and storage allocated to the computers that are executing the tasks. In many scenarios, this setup can provide "the infinite scale" promise for a variety of applications such as web applications, easily parallelizable functions or jobs, or the classic databases that might not require a constant compute power of up to 128 cores. This kind of infrastructure is the ideal choice for scaling out the large number of small compute units such as microservices, functions that might need to quickly replicate to. These kinds of solutions made of a large number of micro-compute units are perfect for the cloud environments where each unit can be deployed on some available compute node in the cloud [5]. However, these solutions will rarely require an atomic compute unit between 64 and 128 cores. The big data analytics solutions with the demand to store and process petabytes of data might require compute power that can challenge the infinite scale promise of the clouds. In the big data analytics solutions, we can see the impact of the physical infrastructure where the components that process data might require a large amount of CPU or memory needed to decompress the large files and process the information. Therefore, migrating data to cloud and running the analytical functions in the compute provided by the cloud are not enough. In practice, data analytics solutions require specialized services such as cloud Datawarehouse [6] or cloud Datalakehouse [7] solutions, that are able to efficiently combine the physical resources in the cloud and optimally process data.

IV. THE DATA ANALYTIC SOLUTIONS

Data analytics solutions provide infrastructure and tools for the analysts and the business users that enable them to store and analyze data. There are two main classes of data analytic solutions:

- The Datawarehouse solutions that represent centralized data storage with API for analyzing data and implementing the business intelligence solutions [6].
- The Lakehouse solutions represent the analytical solution running on the storage that is detached from the analytical engine [7].

Both classes of the solutions are aware of the underlying infrastructure and designed to optimally process large amounts of data. The main differences between Data Warehouse and Data Lakehouse solutions are given in Table I.

TABLE I
THE KEY DIFFERENCES BETWEEN DATA WAREHOUSE AND DATA LAKEHOUSE

	Warehouse	Lakehouse
Data format	Proprietary and highly optimized.	Based on open specifications.
Data location	Internal – data is ingested from the external source.	External – data is placed on the original locations.
Data access	Through the predefined API or protocol (SQL)	Direct file access using the storage API

The main trade-off between Datawarehouse and Data Lakehouse solutions is the choice between the interactive and the real-time analytics. The Datawarehouses store data in the data format optimized for analytics, which enables them to

 ISBN 978-86-7466-930-3

complete the queries in second-to-minute time span. However, they require data engineers to load data from the actual locations into the Datawarehouse, meaning that the data analysts work with the snapshot of the data taken at the load time. The Lakehouses reference original data in the lakes without a need to ingest the data. However, the raw format of the data cannot guarantee sub-second or even sub-minute performance. Despite the differences, both Datawarehouse and Datalakehouse solutions are used in practice. The following sections describe the main characteristics of these solutions.

A. Warehouse solutions

The traditional database systems containing data used for the analytics are Datawarehouse solutions. Datawarehouse solutions have existed for decades, have well defined techniques for designing Datawarehouse schema [8, 9], and a variety of tools available for advanced data analytics. For a very long time, Datawarehouses were the mainstream solutions for storing and analyzing huge data volumes. The top vendors such as Oracle, Teradata, and Exadata are still enabling enterprises to store and analyze large amounts of data.

The main idea with the Datawarehouse solution is that the data required for analytics must be ingested into the internal data format that is highly optimized for analytics. The advantage of this approach is the performance. Internal and in many cases proprietary format contains optimizations that are not available in the open-source solutions. In the past, the proprietary format gave the Datawarehouse performance advantages compared to other analytic systems.

The main issues in the Datawarehouse solutions are the facts that the underlying infrastructure must enable Datawarehouse to store all required data, which puts a burden on the administration teams, and the cost of the solution that includes the resources that are always allocated to the Datawarehouse system even if it is not used. The Datawarehouse solutions were the first choice for most of the analytic teams who need interactive analytics, but the total cost of ownership (TCO) in many cases does not justify the solution.

Amazon made a breakthrough in the Datawarehousing technology with the release of Redshift [10] – a cloud-native Datawarehouse service that provides full data warehousing experience exposed as a cloud service. The customers got the ability to provision the Datawarehouse service, load the data, and define the compute needed to analyze the data. The biggest advantage of cloud Datawarehouse is the resource elasticity that solved the main drawback of the classical Data warehousing solutions – the TCO. Unlike the on-premises Datawarehouse that had pre-build resources, the cloud Datawarehouse enabled organizations to scale up and down the resources depending on the needs. The organizations could load a large amount of data without worrying about the underlying storage capacities, scale-up the compute power of the Datawarehouse when needed, and scale it down to reduce the cost when there is no need for processing the data. The cloud Datawarehouse provided by Amazon fulfilled the main requirements of the classical warehouses and added elasticity. Other vendors followed this approach and now we have many cloud data warehousing solutions such as Azure Synapse Datawarehouse [11], Google BigQuery [12], Snowflake [13], etc. All vendors are trying to combine all the benefits of the classical Datawarehouse with the elasticity and scale of the cloud.

The modern cloud Datawarehouses solve the problem of elasticity and scale on-demand that cannot be easily solved in the on-premises Datawarehouses. However, there is still one downside – the data must be ingested from the external data sources into the Datawarehouse internal data format, which causes a lag between the latest data and the data available for the analytics.

B. Lakehouse solutions

In cloud environments the data is stored in the Data lakes. The Data lake is a logical storage space where the organizations can store the exabytes of data, get the best throughput for reading raw files, ensure redundancy that can span across multiple geographical regions and data centers. The Data lake seems like a perfect solution for storing data. The only drawback of Data lake is that they do not provide the ability to analyze the data in the lake.

The first successful attempt to provide analytic capabilities over large storage was Hadoop – a distributed system that enabled the analyst to analyze a large amount of data stored in the distributed system called Hadoop file system (HDFS). This setup enabled the analyst to analyze data, but the performance of Hadoop is far from interactive. Apache Spark [14] is one of the most popular platforms that enabled analysts to do efficient analytics on the lake. Apache Spark became mainstream in the data lake solutions. One of the most common query engines used to implement the Lakehouse pattern is Apache Spark. Apache Spark is an open-source distributed query system licensed under Apache License 2.0. Spark enables advanced data analytics, management, and updates, and provides a rich and powerful set of APIs to analyze data.

The main idea of Lakehouse architecture is the separation of compute and storage. The compute is a query engine or data processing engine that is detached from the storage, and the data is placed in Data lake where it can be accessed by any query. The compute engine fetches data from the remote Data lake and return data to the analysts once the processing is completed.

Many commercial vendors offered their own implementation of Data Lakehouse services. Nowadays, we have many Lakehouse-type solutions that are offered on different clouds such by Databricks (proprietary version of Apache Spark code implemented by the founders of Spark), Azure Synapse, etc. The main characteristics of these solutions is that they are always referencing the externally stored data, and don't require data to be ingested to start analytics. This enables real-time insight into the latest version of data without the need to wait for the daily data loads to

 ISBN 978-86-7466-930-3

finish before starting the analytics.

One of the main concerns in the Lakehouse solution is whether they would be able to match the performance of Warehouse solutions. Traditionally, the proprietary format used in Datawarehouse solutions was the main competitive advantage compared to the original data formats stored in Data lake. Databricks announced that they have set a new world record in 100TB TPC-DS, the gold standard performance benchmark for data warehousing [15]. The test was performed in Barcelona supercomputing center and officially submitted as TPC-DS result that outperformed the previous record by 2.2x. This was the first Lakehouse-class solution that set the record in an official Datawarehouse benchmark and proved that the Lakehouse architectures can compete with the modern Datawarehouse solutions. One of the key reasons for this kind of success of Lakehouse solution is that they are using the optimized storage file format that matches the proprietary internal formats used in the Datawarehouse solutions.

More insights into data lake solutions and current trends can be found in the literature, including what steps are needed to adopt the cloud concept in data analytic solutions [16].

V. DATA FORMAT

One of the most important design decisions that will impact the efficiency of the data analytics is the choice of the file format that will be used to store the data. Most of the data used for analytic purposes is stored in a plain textual format represented as a delimited text (for example comma separated values – CSV, tab-separated values – TSV, etc.). The documents containing tabular data are represented as Open Office or Microsoft Office formats. Although these data formats are very common, they are inefficient for big data analytics.

There is an additional class of plain textual data represented in JSON format. The JSON format is the standard format in many Internet of Things (IoT) applications where the IoT devices send the messages in JSON format, or the messages that will be eventually stored in JSON format [17]. JSON format provides flexibility for changing the structure of data but complicates the analytics because it requires a parser that is more complex than the plain delimited text parser.

In the practice, the data analysts could analyze data stored in CSV, JSON and other commonly used formats. However, since these formats are not optimized for analytics, it was very hard for data analysts to extract valuable information with performance that matches performance of database systems.

The proprietary data formats that are used to store data in Datawarehouse solutions were the biggest competitive advantage of the Datawarehouse systems compared to the open-source solutions. The optimized formats with high compression, columnar organization of data, and vectorized processing was the main reason why the data analytics teams used the Datawarehouse solutions.

In the open-source community multiple advanced formats are proposed that are designed to optimize the storage format and improve the performance of analytics. Examples are Row Columnar (RC)[18] or Optimized Row Columnar (ORC)[19] file format. The idea of these formats was to define binary representation of data prepared for analytics and optimize access for the analytical jobs. However, the open-source format that took most of the market share became Parquet format [19]. Parquet format is an open-source format that introduces most of the benefits that exist in the proprietary Datawarehouse formats, such as:

- Column organization – the data is physically separated into column segments instead of rows. The column segments contain all cell values from the same column. The columnar organization enables the analytical queries that read 2 columns out of 100 columns to read only 2% of data on average. Since the analytical queries aggregate the measures and summarize them by few columns, the columnar organization introduces most of the performance benefits for the analytical queries.
- Row-groups – columns are divided into row–groups (for example 100.000 rows represent a row group that will be split into the columns) The column segments within the row groups contain some statistical information about cells such as min/max values.
- Compression – There are some compression techniques such as run-length encodings (RLE) [20] that can be applied on the Parquet files to achieve excellent 10-100x compression, which matches the compression in the proprietary Datawarehouse formats. The main impact is not just storage savings. Compressed storage decreases IO requests sent to the storage and improves data throughput that analytical tools can use to fetch the data.
- Non-relational types – Parquet is not limited to strongly defined types and enables storing objects and arrays. The organization of complex types in Parquet format is described in [21].

The Parquet format became the mainstream in data analytics. Although there are other formats that are used in practice, the Parquet format is getting the highest market share and we can expect that the majority of data will be stored in the Parquet format. Therefore, any modern big analytical solution must be based on the Parquet format, or some enhancement based on Parquet. Even if the new file format arrives in the future, there is a high chance that most data will be stored in the Parquet format.

Although the Parquet format is designed to store analytical data that should be read-only (or append-only), there is a need to enable data engineers to make updates to the data. There are several updateable formats (such as Delta Lake [22]), that combine the excellent storage format for analytics and provide ACID guarantees of the operations that managed data. Another possible advancement includes parsing big data using the dataflow paradigm [23] by transforming automatically the parsing software [24] and using appropriate scheduling techniques [25] for the dataflow supercomputing architecture.

 ISBN 978-86-7466-930-3

VI. Conclusion

Big data analytics solutions evolved from the original on-premises based big Datawarehouse solutions to modern cloud based Datawarehouse solutions. The original on-premises Datawarehouse solutions enabled organizations to store large amounts of data and analyze data with acceptable performance. However, these kinds of solutions failed to enable scalability and elasticity. Cloud computing can scale resources on-demand. Data lake that can store Exabytes of data with guaranteed replication, and advances in the opensource file formats disrupted the Datawarehouse solutions in big data analytics. Although Datawarehouse solutions evolved and have been adapted for the modern cloud environments, architectures with full separation of compute power and storage, where the compute can scale when needed, have a direct access to the latest version of data in the lake, and the performance that match modern Datawarehouses. In addition to matching all features that were historically considered as the advantage of Datawarehouse, the Lakehouse solves the issue that fundamentally cannot be solved with Datawarehouse solutions – data ingestion. Lakehouse solutions are able to access the original data in the Data lake and don't require an explicit process to load external data. Without performance degradation compared to internal data formats used in Datawarehouse solutions, direct access simplifies data management process by avoiding the additional processes that constantly move the data and also enable analyst to get the data without any delay.

By looking at the modern trends, we can conclude that the future of big data analytics will be based on the cloud environments and Lakehouse architectures. The cloud Data Lakehouse solutions leverage all benefits of the cloud and match performance of the Datawarehouse solutions. The cloud Data Lakehouse solutions can be considered as a primary solution for most of the future research and as the mainstream and preferred technology for development projects.

Acknowledgment

DM and NK are partially supported by the School of Electrical Engineering, University of Belgrade, Serbia and by the Ministry of Education, Science, and Technological Development of the Republic of Serbia. NK is partially supported by the Institute of Physics Belgrade, contract no. 0801-1264/1.

References

[1] P. Chauhan, M. Sood, "Big Data: Present and Future," The IEEE Computer Society, (2021), DOI: 10.1109/MC.2021.3057442.

[2] S. G. Manikandan, S. Ravi, "Big Data Analysis Using Apache Hadoop," International Conference on IT Convergence and Security (ICITCS), pp. 1-4, (2014). doi: 10.1109/ICITCS.2014.7021746.

[3] E. Zagan, M. Danubianu, "Cloud DATA LAKE: The new trend of data storage," 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, 1-4 (2021), doi: 10.1109/HORA52670.2021.9461293.

[4] I. Hashem, I. Yaqoob, N. Anuar, S. Mokhtar, A. Gani, A., and S. Khan, "The rise of "big data" on cloud computing: Review and open research issues", Information Systems, Volume 47, 2015, Pages 98-115, ISSN 0306-4379.

[5] R. Han, L. Guo, M. M. Ghanem and Y. Guo, "Lightweight Resource Scaling for Cloud Applications," 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 644-651, (2012). doi: 10.1109/CCGrid.2012.52.

[6] G. Garani, A. Chernov, I. Savvas and M. Butakova, "A Data Warehouse Approach for Business Intelligence," 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, 70-75 (2019). doi: 10.1109/WETICE.2019.00022.

[7] D. Oreščanin and T. Hlupić, "Data Lakehouse - a Novel Step in Analytics Architecture," 44th International Convention on Information, Communication and Electronic Technology, 1242-1246 (2021). doi: 10.23919/MIPRO52101.2021.9597091.

[8] R. Kimball, M. Ross, "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling," 2nd. edition. John Wiley & Sons, Inc., USA, (2002).

[9] W. Inmon, "Building the Data Warehouse," John Wiley & Sons, Inc., USA, (1992).

[10] A. Gupta, D. Agarwal, D. Tan, J. Kulesza, R. Pathak, S. Stefani, and V. Srinivasan, "Amazon Redshift and the case for simpler data warehouses," SIGMOD (2015).

[11] J. Aguilar-Saborit, R. Ramakrishnan, K. Srinivasan, K. Bocksrocker, I. Alagiannis, M. Sankara, M. Shafiei, J. Blakeley, G. Dasarathy, S. Dash, L. Davidovic, M. Damjanic, S. Djunic, N. Djurkic, C. Feddersen, C. Galindo-Legaria, A. Halverson, M. Kovacevic, N. Kicovic, G. Lukic, D. Maksimovic, A. Manic, N. Markovic, B. Mihic, U. Milic, M. Milojevic, T. Nayak, M. Potocnik, M. Radic, B. Radivojevic, S. Rangarajan, M. Ruzic, M. Simic, M. Sosic, I. Stanko, M. Stikic, S. Stanojkov, V. Stefanovic, M. Sukovic, A. Tomic, D. Tomic, S. Toscano, D. Trifunovic, V. Vasic, T. Verona, A. Vujic, N. Vujic, M. Vukovic, M. Zivanovic, "POLARIS: The distributed SQL engine in Azure Synapse" PVLDB, vol. 13, issue 12, (2020).

[12] K. Sato, "An inside look at Google BigQuery," Technical report, Google. https://cloud.google.com/files/BigQueryTechnicalWP.pdf.

[13] B. Dageville, T. Cruanes, M. Zukowski, V. Antonov, A. Avanes, J. Bock, J. Claybaugh, D. Engovatov, M. Hentschel, J. Huang, A. W. Lee, A. Motivala, A. Q. Munir, S. Pelley, P. Povinec, G. Rahn, S. Triantafyllis, P. Unterbrunner, "The Snowflake Elastic Data Warehouse," SIGMOD (2016).

[14] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, M. Zaharia, "Spark SQL: Relational data processing in Spark," Armbrust, Melbourne, Victoria, Australia, ACM, SIGMOD, (2015).

[15] R. Xin, M. Mokhtar, "Databricks Sets Official Data Warehousing Performance Record," November, 2021, Databricks company blog.

[16] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang, "Leveraging the data lake: Current state and challenges," International Conference on Big Data Analytics and Knowledge Discovery, Springer, Cham, 179-188, (2019, August).

[17] N. Nikolov, "Research of the Communication Protocols between the IoT Embedded System and the Cloud Structure," 2018 IEEE XXVII International Scientific Conference Electronics - ET, 1-4 (2018). doi: 10.1109/ET.2018.8549604.

[18] Y. He, R. Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang, and Z. Xu, "RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems," Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, IEEE Computer Society, USA, 1199–1208, (2011). DOI:https://doi.org/10.1109/ICDE.2011.5767933

[19] T. Ivanov, M. Pergolesi, "The impact of columnar file formats on SQL-on-Hadoop engine performance: A study on ORC and Parquet. Concurrency," Computat Pract Exper. (2020), https://doi.org/10.1002/cpe.5523.

[20] A. Ishtiaq, S. Ahmad, and D. S. Shukla, "Fast Retrieval with Column Store using RLE Compression Algorithm," International Journal of Computer Applications, 111. 30-34, (2015). 10.5120/19537-1193.

[21] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive Analysis of Web-Scale Datasets", Proc. of the 36th Int'l Conf on Very Large Data Bases, 330-339 (2010).

[22] M. Armbrust, T. Das, L. Sun, B. Yavuz, S. Zhu, M. Murthy, J. Torres, H. van Hovell, A. Ionescu, A. Łuszczak, M. Świtakowski, M.

Szafrański, X. Li, T. Ueshin, M. Mokhtar, P. Boncz, A. Ghodsi, S. Paranjpye, P. Senster, R. Xin, and M. Zaharia, "Delta lake: high-performance ACID table storage over cloud object stores," Proceedings of the VLDB Endowment, vol. 13, issue 12, 3411–3424 (2020) DOI:https://doi.org/10.14778/3415478.3415560.

[23] N. Korolija, J. Popović, M. Cvetanović, and M. Bojović, "Dataflow-based parallelization of control-flow algorithms," *Advances in computers*, Elsevier, **104**, 73-124 (2017).

[24] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, "Transforming applications from the control flow to the dataflow paradigm," *Dataflow supercomputing essentials*, Springer, Cham, 107-129 (2017).

[25] N. Korolija, D. Bojić, A. R. Hurson, and V. Milutinovic, "A runtime job scheduling algorithm for cluster architectures with dataflow accelerators," *Advances in computers*, Elsevier, **126** (2022).

 ISBN 978-86-7466-930-3

Hybrid Manycore Dataflow Processor

Danko Miladinović, Miroslav Bojović, Vladisav Jelisavčić, and Nenad Korolija

***Abstract*— This work addresses high performance computing architectures, presenting a hybrid processor that includes multiple computing architectures on a single chip die.**

Beside commonly used multicore processor, a personal computer might include manycore graphical processor. This work advocates for a combination of these two architectures along with a dataflow processor that usually appears in the form of a FPGA chip able to perform parallel tasks at the same time.

A quick overview of these computer architectures and appropriate programming paradigms is followed by the comparison based on flexibility and speed, price and development time, and speed and power consumption. Finally, the proposed hybrid processor is analyzed against computationally demanding algorithms that are often executed on high performance computing architectures.

Future work will include the comparison of the proposed computer paradigms and the comparison of the proposed hybrid architecture with existing ones.

Index Terms— **High performance computing; manycore processors; dataflow programming**.

I. Introduction

Many high performance computing algorithms are scalable, and as such, suitable for execution using:

- computer architectures the include graphical processing units capable of executing algorithms,
- dataflow computer architectures.

Both of these exist in the form of:

- personal computers,
- computer clusters,
- cloud computers.

Besides the differences in dataflow and conventional computing architectures, their computing paradigms also differ, as well as their suitability for executing high performance computing algorithms.

This work presents recently exploited computer architectures from the point of view of their usability for executing high performance computing algorithms. These architectures are compared based on programming flexibility that they offer, algorithm execution speed, scalability, software development effort, constraints of each of architecture type, price, and power consumption. The presentation of the work is based on the proposed method for presenting the results [1].

Danko Miladinović, Miroslav Bojović, and Nenad Korolija are with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mails: danko @ etf.bg.ac.rs, mbojovic @ etf.bg.ac.rs, and nenadko @ etf.bg.ac.rs).

Vladisav Jelisavčić is with the Mathematical Institute of the Serbian Academy of Sciences and Arts, Kneza Mihaila 36, Belgrade 11001 (e-mail: vladisavj @ gmail.com).

Following sections describe these computer architectures in a uniform manner. A brief overview is followed by the estimation of effectiveness. Each computing architecture is subjected to the following criteria:

- order of number of transistors per number of instructions that can run in parallel,
- speed of the hardware,
- suitability for high performance computing algorithms,
- independence from other computer architectures,
- price performance ratio,
- power consumption performance ratio,
- required space for computer architecture per performance ratio.

On the top of the work, authors propose a hybrid processor that includes multiple computing paradigms on a single chip.

II. Multicore architectures

Most of personal computer architectures are based on von Neumann paradigm. Programs written in programming languages are compiled and linked, and the resulting instructions are stored on the disk. Once a program starts, instructions are loaded into the RAM memory, from where they get copied into the cache memory, so that they could be read faster.

The processor is responsible for executing instructions. Long ago, processors included a single arithmetical logical unit (ALU) for performing arithmetical and logical operations. The speed of processors was increasing for decades by approximately doubling each second year. However, once the speed reached around 3GHz, the trend stopped. The constraint was and still is that the wave length became around 10 cm. Given the fact that the clock cycle must be stable during the whole instruction execution, and the signal has to travel with the speed of light multiple times in different directions, options for further acceleration were:

- to decrease the size of the chip,
- to decrease the size of transistors,
- to decrease the number of transistors per logical gates,
- to implement multiple ALUs that would work in parallel.

Decreasing the size of the chip die implies reducing the number of transistors, leading to deteriorating performances. Reducing the size of transistors would affect their functions considerably (i.e. more failures would appear). Logical gates are already optimized so that further reduction in the number of transistors would affect their capacities for producing output to multiple logical gates.

As a result, both research community and the industry opted for multiplying ALUs on a single chip die.

Multicore architectures can execute an order of 10

 ISBN 978-86-7466-930-3

operations simultaneously. The transistor count is around 1 billion. Clock cycle is order of GHz. As such, their usability for high performance computer algorithms is limited in terms of utilizing algorithms in the real time applications. One of the most important characteristics of multicore architectures is their independence from other computer architectures. As a result, personal computers usually contain only one processor of this type. Typical power consumption is around 500 watts. When used in computer clusters, a single node could include multiple processors of this type.

III. MANYCORE ARCHITECTURES

Graphical processing units are often referred to as manycore computer architectures, as the number of processing units is larger than those of multicore processors.

So-called manycore architectures are logical successor of multicore processors. Certain fields, like computer graphics, require more instructions per second than multicore architectures produce, so that the picture can be refreshed many times per second (e.g. 60 times). With shading effects and full-hd or even 4k resolutions this implies updating millions of pixel colors based on performed operations with appropriate matrices. As a result, companies started producing chips for graphical cards that include thousands of cores that have lower number of available instructions supported by their architectures but are capable of executing more instructions per second. Soon after, utilization of the processing power of new graphical cards started. Many algorithms are implemented using the CUDA programming model.

Manycore architectures have an order of 10 billion transistors with an order of 1000 processor cores, and an order of 1000 instructions that can run in parallel. The clock cycle has an order of 1 GHz. This makes it suitable for high performance computing algorithms. Although each core is based on von Neumann paradigm, being capable of executing any instruction defined by the architecture at any given moment, it relies on the multicore processor as the one that assigns jobs to cores. The manycore architectures proved to be efficient for high performance computing algorithms, including data mining and coin mining. Power consumption is of the same order of magnitude as of multicore processors, while manycore processor offer superior performance. They usually come in the form of a PCIe card that attaches to the mainboard.

IV. DATAFLOW ARCHITECTURES

Dataflow architectures are based on a separate programming paradigm called dataflow paradigm. Data flows through a hardware in terms of electrical signals, resulting in transforming an input to the output [2, 3]. One of the main advantages over the previously mentioned computer architectures is that there is no need for an order of 1 billion transistors to execute only around 10 instructions in parallel. Dataflow hardware can execute even 1000 instructions simultaneously. As such, it is capable of accelerating many algorithms [4, 5]. The main disadvantage is that the multicore processor is needed for preparing the data to be processed using a dataflow hardware and for handling results.

The order of number of transistors is comparable to previously mentioned computer architectures, while the number of instructions that can run in parallel can be much higher. The speed of the hardware is around 0.1 GHz. Since they do not support executing any instruction defined by the architecture at any given moment, they are suitable for only a portion of high performance computing algorithms and more programming effort is needed for creating programs [6]. A mitigating circumstance is that there is a way to automatically translate certain algorithms from the control-flow into the dataflow paradigm [7]. Having significantly lower space occupation and power consumption per instruction, dataflow hardware has good price performance ratio.

V. HYBRID ARCHITECTURES

While each of the available computer architectures has its own advantages and disadvantages, it is a logical step but also a challenge to try to merge multiple programming paradigms into a single computer architecture. The task is even harder to achieve if they are to be put on the same chip die. However, the need for computing justifies the effort needed to merge existing computing paradigms.

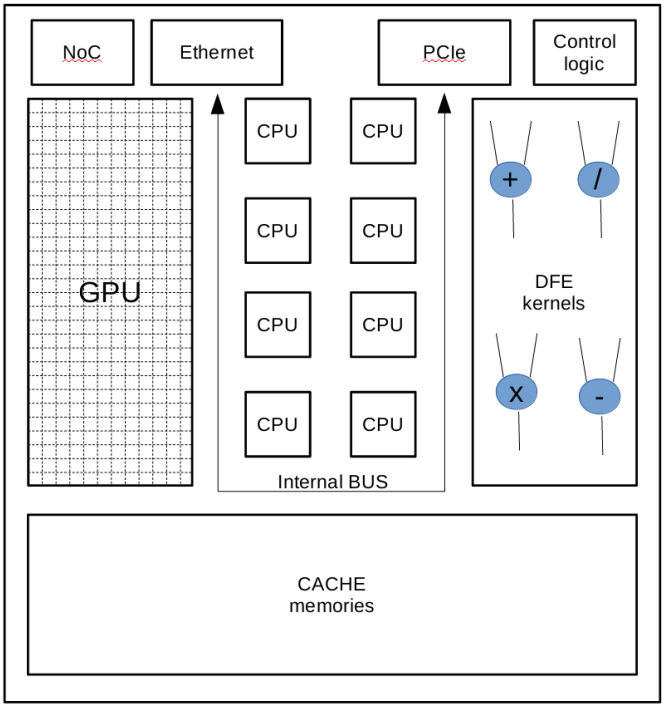


Fig. 1. Hybrid control-flow dataflow architecture.

Fig. 1 depicts a hybrid control-flow dataflow architecture on a single chip. Compared to the typical multicore processor architecture, the proposed hybrid architecture includes graphics processing unit (GPU) cores and dataflow kernels beside central processing unit (CPU) cores and cache memories. Additionally, network on chip (NoC) is suggested as a good way of handling the communication between GPU

 ISBN 978-86-7466-930-3

cores. Compared to the Maxeler dataflow hardware PCIe cards, proposed dataflow engine (DFE) kernels are connected directly to the slowest internal cache memory, because of the necessity for dataflow hardware to execute at the same speed. Constraint that has to be taken into account is that cache locking mechanism has to be implemented for cache memory connected to the dataflow, which would enable granted access to DFE kernels, but also to GPU if it is required by the application that is executed using both computing paradigms.

This architecture offers the best that any of the paradigms offer. On the other side, the complexity is equal to the sum of complexities of incorporated architectures. As such, the proposed hybrid architecture is suitable for executing multiple jobs simultaneously, where some of them are suitable for executing using the dataflow paradigm, while others achieve better performance when executed using manycore architecture. Finally, there are software applications that are not based on scalable algorithms, making them suitable solely for the multicore processor.

The emerging problem on heterogeneous computer architectures that include both dataflow and control-flow hardware is scheduling program execution. Authors have presented their novel algorithm for optimal scheduling of both dataflow and control-flow jobs [8]. The algorithm is general but is limited in number of jobs it can schedule due to the computational complexity. Based on this optimal algorithm, two heuristic algorithms for optimizing the throughput and minimizing total execution time are derived, producing near-optimal schedules for both dataflow and control-flow jobs at large job counts at the cost of negligible scheduling penalty. The heuristic algorithms performance gain decreases slightly as job count increases and only at the beginning, proving that the performance of existing cluster structures with appropriate dataflow accelerators can be considerably improved.

The drawback of combining multiple computing architectures on a single chip is that it increases the probability of failure. However, the probability is relatively high at the very beginning, and once a chip enters the so-called wear-out phase, and is relatively low in the meanwhile [9].

The order of number of transistors per number of instructions that can run in parallel depends on the type of job. If a job is suitable for dataflow architectures, the acceleration would be similar to those of dataflow architectures, while the number of transistors would be around three times higher, assuming that included architectures consume the same number of transistors. The same applies for jobs suitable for manycore architectures, and jobs that cannot be efficiently accelerated using neither manycore architecture, nor a dataflow architecture. However, this applies solely to scheduling a single job that can be executed using one of these three types of architectures. If we combine multiple jobs, those suitable for dataflow architectures would be executed there, based on their hardware requirements and on acceleration they achieve using the dataflow architecture. Jobs suitable for manycore architectures could run in parallel, while jobs that are based on algorithms that are not scalable could run in parallel on the multicore processor. Therefore, achieving fair comparison of the number of instructions that can be run on the proposed architecture is not a straightforward task. As a result, new benchmarks are needed to compare the proposed architecture to the existing ones. For now, one could consider the worst-case scenario, where the acceleration of a job is the same as for the best suited architecture for the job, and the number of transistors is equal to the sum of the number of transistors included on each part of the proposed hybrid processor [10].

The speed of the proposed hardware is also hard to define. As the processor includes different architectures that naturally run on different clock rates, the proposed processor would have a clock speed for the multicore processor that is a multiple of the clock speed of a dataflow architecture and manycore architecture. Therefore, the multicore would be able to efficiently communicate with other parts of the processor using input and output communication buffers.

The proposed computer architecture is not only suitable for high performance computing algorithms, but it is more efficient that aforementioned architectures, as it offers the most suitable hardware type for any particular job. The proposed processor includes multicore processor on the same chip die, making the architecture independent from other computer architectures. The price performance ratio for any given job is lower than those of the best suited of the aforementioned architectures. However, given a set of jobs, where each is suited for one particular architectures, the proposed architecture exploits advantages of all three types of architectures and could achieve the best speed-up in all categories of jobs. The price performance ratio is also lower for a particular job suited for a single computer architecture but is better than any of the aforementioned architectures if there are jobs that could approximately equally occupy all resources of the proposed processor. If completely utilized, the power consumption performance ratio is better than any of the three underlying architectures. For a single job, it is around three times lower, as it is estimated that this processor would consume as much electrical power as all three underlying architectures combined. The space required for the proposed computer architecture is one of its main advantages. Having in mind that it would be able to execute any type of jobs that any of the three computer architectures can, the space per performance ratio cannot be outperformed by any of the existing architectures.

This is not the unique case of proposing combination of manycore and dataflow architectures. Similar tries have been by researchers in the past [11, 12, 13].

VI. COMPARISON

This section summarizes the advantages and disadvantages of described computer architectures and compares them based on various criteria. For each of the given criteria, the proposed hybrid computer architecture is compared to all three underlying computer architectures.

The research [2] summarizes in their Table 2 the achieved

 ISBN 978-86-7466-930-3

speedups of algorithms implemented using the same type of the dataflow hardware that includes a memory on a chip comparing to the control-flow implementations of the same algorithms. The Lattice-Boltzmann algorithm is presented in detail, along with the dataflow code. Authors have compared execution time of Lattice-Boltzmann algorithms using the MAX2 card with 6GB of RAM and using Intel i5 650 processor with the clock speed of 3.2GHz. The computer used 4GB RAM memory at the speed of 1333MHz. The conclusion that can be drawn is that the speed-up of all observed dataflow algorithms ranges from 25% up to the multiplication factor of 150. Based on the comparison of these algorithm implementations for control-flow and for dataflow paradigms, we can summarize the advantages and disadvantages of both programming paradigms.

Table I presents a comparison of computing architectures in terms of flexibility to execute any instruction at any given moment and the speed of execution measured in number of instructions per second. As it can be seen, the multicore computing paradigm offers the highest flexibility by being able to execute any type of job, as it can execute any instruction defined by the architecture at any given moment. Although manycore architectures work on the same principle, they are considered to be utilized if many processing units may work in parallel. Therefore, their flexibility is limited to scalable algorithms suitable for manycore architectures. Dataflow architectures introduce new constraints by not being able to adapt to new needs before re-configuring the hardware. Hybrid architectures offer the advantages of both paradigms.

The speed of execution measured in number of instructions is lowest for the multicore architectures, as they are limited to processing up to an order of 10 instructions simultaneously. Any other of presented computer architectures can execute two or three orders of magnitude more instructions simultaneously.

TABLE I
FLEXIBILITY AND SPEED COMPARISON OF VARIOUS COMPUTING ARCHITECTURES

Type of architecture	Flexibility	Speed
Multi core	+++	+
Many core	++	++
Dataflow	+	++
Hybrid	+++	++

Table II presents a comparison of computing architectures in terms of price and the development time measured by effort needed for producing the software. The price raises as we lean towards more optimized computer architectures. The same applies to software development time. The only exception is that hybrid computer architecture, which is suitable for executing any of the given type of software, which means that the development time depends on the type of software being executed on the architecture.

As dataflow architectures are not as utilized as multicore and manycore computer architectures, the price tag of dataflow architectures is higher that it would be if each personal computer would include dataflow engines as well.

TABLE II
PRICE-DEVELOPMENT TIME COMPARISON OF COMPUTING ARCHITECTURES

Type of architecture	Price	Development time
Multi core	+	+
Many core	++	++
Dataflow	+++	+++
Hybrid	++++	+ - +++

Table III shows a speed to power consumption comparison of these computing architectures. The speed of the multicore computer architecture is slower than others, as it can run a smaller number of instructions in parallel.

When it comes to power consumption, it is similar for all types of architectures, which leads us to the following conclusion. The power consumed per a single instruction is the highest in the case of the multicore computer architecture, while it is the lowest for dataflow and hybrid computer architectures, assuming that they are not underutilized.

TABLE III
SPEED-POWER CONSUMPTION COMPARISON OF COMPUTING ARCHITECTURES

Type of architecture	Speed	Power consumption
Multi core	+	++
Many core	++	++
Dataflow	+++	++
Hybrid	+++	++

Based on these comparisons, we could conclude that the proposed architecture has the potential for achieving better results in terms of speed, flexibility, and power consumption comparing to the existing computer architectures, while the programming effort might be higher in the case of the algorithms implemented using the dataflow paradigm.

Comparing to the research available in the open literature, the proposed architecture is more high performance computing oriented than the Ultimate dataflow processor [12], while it doesn't support internet of things. Authors of SambaNova [14] also recognized the potentials of dataflow computing paradigm. Their Reconfigurable Dataflow Unit (RDU) enables accelerating algorithms with the flexibility to build custom dataflow pipelines as well as large memory capacity to run big models such as Natural Language Processing (NLP) and high-resolution computer vision efficiently. However, it is dedicated to algorithms that consist predominantly of the source code that can be most efficiently accelerated using the dataflow paradigm.

Research [15] exploits the opportunities from digital Processing-in-memory (PIM) bit-serial processing and in-memory customization, to tackle the above challenges by co-designing sparse algorithm, multiplication dataflow, and PIM

 ISBN 978-86-7466-930-3

architecture.

VII. Conclusion

Along with a multicore processor, a personal computer might include manycore graphical processor and a dataflow processor on the same chip die. This work advocates for a combination of these two architectures in order to create the type of the computer architecture that is able to execute jobs suitable for any of these three types of architectures in parallel.

Presented comparison between computing architectures suggests what kind of algorithms are suitable for execution using existing computing paradigms.

The proposed hybrid processor is analyzed against computationally demanding algorithms that are often executed on high performance computing architectures. As it includes multiple computing architectures on a single chip die, it could achieve the best acceleration for a job suitable for any of aforementioned computer architectures. At the same time, if the amount of jobs suitable for these three computer architectures matches the amount of resources of the proposed processor, the proposed processor with appropriate job scheduling can achieve the performance of combined architectures.

Future work includes the simulation comparison of the paradigms and the comparison of the proposed hybrid architecture with existing ones.

Acknowledgment

DM, NK, and MB are partially supported by the School of Electrical Engineering, University of Belgrade, Serbia. VJ is partially supported by the Mathematical Institute of the Serbian Academy of Sciences and Arts. NK is partially supported by the Institute of Physics Belgrade, contract no. 0801-1264/1. All authors are partially supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

References

[1] V. Milutinovic, "The best method for presentation of research results," IEEE TCCA Newsletter, 1-6 (1996).

[2] N. Korolija, J. Popović, M. Cvetanović, and M. Bojović, “Dataflow-based parallelization of control-flow algorithms,“ Advances in computers, Elsevier, 104, 73-124 (2017).

[3] N. Trifunovic, V. Milutinovic, J. Salom, A. Kos, “Paradigm shift in big data super-computing: dataflow vs. controlflow,” J. Big Data, vol. 2, issue 4, 1–9 (2015).

[4] N. Trifunovic, V. Milutinovic, N. Korolija, G. Gaydadjiev, "An AppGallery for dataflow computing," Journal of Big Data, vol. 3, issue 1, 1-30 (2016).

[5] N. Korolija, T. Djukic, V. Milutinovic, and N. Filipovic, “Accelerating Lattice-Boltzman method using Maxeler dataflow approach,” The IPSI BgD Transactions on Internet Research, 34 (2013).

[6] J. Popovic, D. Bojic, and N. Korolija, “Analysis of task effort estimation accuracy based on use case point size,” IET Software, 9(6), 166-173 (2015).

[7] V. Milutinovic, J. Salom, D. Veljovic, N. Korolija, D. Markovic, and L. Petrovic, “Transforming applications from the control flow to the dataflow paradigm,” Dataflow supercomputing essentials, Springer, Cham, 107-129 (2017).

[8] N. Korolija, D. Bojić, A. R. Hurson, and V. Milutinovic, “A runtime job scheduling algorithm for cluster architectures with dataflow accelerators,” Advances in computers, Elsevier, 126 (2022).

[9] K. Huang, Y. Liu, N. Korolija, J. M. Carulli, and Y. Makris, “Recycled IC detection based on statistical methods,” IEEE transactions on computer-aided design of integrated circuits and systems, 34(6), 947-960 (2015).

[10] V. Milutinović, N. Trifunović, N. Korolija, J. Popović, and D. Bojić, “Accelerating program execution using hybrid control flow and dataflow architectures,” 25th Telecommunication Forum, IEEE, 1-4 (2017).

[11] V. Milutinović, E. S. Azer, K. Yoshimoto, G. Klimeck, M. Djordjevic, M. Kotlar, M. Bojovic, B. Miladinovic, N. Korolija, S. Stankovic, N. Filipović, Z. Babovic, M. Kosanic, A. Tsuda, M. Valero, M. de Santo, E. Neuhold, J. Skorucak, L. Dipietro, I. Ratkovic, "The ultimate dataflow for ultimate supercomputers-on-a-chip, for scientific computing, geo physics, complex mathematics, and information processing," 10th Mediterranean Conference on Embedded Computing, IEEE, 1-6 (2021, June).

[12] V. Milutinović, M. Kotlar, I. Ratković, N. Korolija, M. Djordjevic, K. Yoshimoto, and M. Valero, "The Ultimate Data Flow for Ultimate Super Computers-on-a-Chip, " Handbook of Research on Methodologies and Applications of Supercomputing, IGI Global, 312-318 (2021).

[13] V. Milutinović, B. Furht, Z. Obradović, and N. Korolija, “Advances in high performance computing and related issues,” Mathematical problems in engineering, (2016).

[14] R. Prabhakar, S. Jairath, and J. L. Shin, “SambaNova SN10 RDU: A 7nm Dataflow Architecture to Accelerate Software 2.0,” 2022 IEEE International Solid-State Circuits Conference (ISSCC), IEEE, vol. 65, 350-352 (2022).

[15] F. Tu, Y. Wang, L. Liang, Y. Ding, L. Liu, S. Wei,... and Y. Xie, “SDP: Co-Designing Algorithm, Dataflow, and Architecture for in-SRAM Sparse NN Acceleration,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (2022).

Service-Oriented Communication Between ADAS and IVI Domains in Automotive Solutions

Dušan Kenjić and Marija Antić, *Members, IEEE*, Dušan Živkov

***Abstract*—As the complexity of automotive systems has grown, it has become necessary to cluster various vehicle components into several domains, based on a specific function they perform. This approach has facilitated the development of domain-specific features, as it allows to create communication standards and common libraries that meet the requirements of the particular domain. On the other hand, it has created the redundancy in the resource consumption required to perform similar tasks in different domains, which leaves the room for further optimizations. This is most notable if we analyze the functionalities of the two fastest growing domains: autonomous driving assistance (ADAS) and in-vehicle infotainment (IVI), which are both developing simultaneously, and may benefit from the option of providing features and services to each other. This paper will examine and propose a solution for interconnection between ADAS and IVI domains by utilizing state-of-the-art mechanisms of the service-oriented architecture (SOA) paradigm. The examination of SOA utilization rationale will be presented, as well as the crucial challenges and limitations of the possible approaches, derived mainly from the discrepancy of service-oriented architecture implementation and mapping in different standards. Various features and use-cases will be discussed, that would be good candidates for cross-domain implementation.**

***Index Terms*— in-vehicle domains, ADAS, IVI, interconnection, SOA in automotive.**

I. Introduction

The transition to centralized domains in the automotive system design and development was necessary due to the increasing number of Electronic Control Units (ECUs) in the modern vehicle. With this approach, the system is organized into several domains based on the features and the tasks ECUs within the domain perform. By splitting the whole system into a set of specialized domains, it was possible to create standards and abstractions that facilitate the development of features specific to the particular domain, without the need for developers to constantly solve the problems of connectivity and resource sharing. Two domains which are constantly improved and require powerful resources are ADAS - Advanced Driver System Assistant domain and IVI - In-vehicle Infotainment domain. The ADAS domain is responsible for safety-critical features and algorithms using various types of sensors in order to enable safe, comfortable and cost-effective driving. On the other hand, the IVI domain is oriented towards passenger entertainment, as well as towards providing useful information about the driving conditions and the state of the vehicle. Although these two domains perform different tasks, there is a set of features and sensors of the same type which are commonly used in both of them. However, in the current architecture of modern vehicle, these two domains do not share any of the hardware resources nor results of the data processing algorithms. This creates an implementation overhead, as similar functionalities need to be implemented in both of the domains, and the hardware cost is constantly increasing. This represents the main motivation to design an approach for resource sharing between domains as a first step towards the unified platform which shall control the entire system.

In this paper, we will present the results of the initial phase of a research project aiming to create the solution for the inter-domain communication and resource sharing in the automotive solutions. First, we will present the summary of the state-of-the-art research and commercially used approaches for the inter-process communication and service-oriented architecture in automotive industry. Then, we will propose the architecture of the solution connecting ADAS and IVI domains, provide some practical details and discuss the examples of the use-cases which would benefit from the resource sharing between these two domains. Finally, we will discuss the implementation challenges of the proposed approach.

II. IPC and SOA in Automotive Systems

A. Service-Oriented Middleware in Automotive

Traditionally, automotive systems use a conventional signal-based communication approach, which provides a deterministic data transfer, and enables the processes to run in the predefined schedule [1][2]. However, such an approach does not support the desired scalability of the system, which is required to satisfy the requirements of emerging applications and scenarios. Therefore, in order to provide the flexibility and a more dynamic and scalable system, service-oriented architecture (SOA) was introduced to the automotive system design. Service-oriented communication approach has been adopted from the domains of web applications, cloud and information systems, where it has already proven its flexibility for functional services implementation [3]. SOA represents an efficient way to encapsulate the job done by the specific component into a service. This way, resources can be

Dušan Kenjić is currently working toward the Ph.D. degree with the University of Novi Sad, Serbia, (e-mail: dusan.kenjic@uns.ac.rs).

Marija Antić is currently the Assistant Professor with the University of Novi Sad, Serbia, (e-mail: marija.antic@rt-rk.uns.ac).

Dušan Živkoc is with the RT-RK Institute for Computer based Systems, (e-mail: dusan.zivkov@rrt-rk.com).

 ISBN 978-86-7466-930-3

distributed to the clients interested in the information which the service provides, the service implementation can remain obscured from the clients and modularity and repetitiveness can be achieved. Additionally, the unified communication mechanism facilitates the interoperability between heterogenous system components, which otherwise represents a time consuming and challenging problem that needs to be solved during the application development.

The first step towards the integration of SOA principles within an automotive system is to create a platform and define a protocol which can support this integration. Such platform must be compatible with other automotive solutions and protocol must be suitable with the automotive requirements [4]. Scalable service-Oriented Middleware over IP (SOME/IP) is a Remote Procedure Call (RPC) mechanism [5] specialized for the usage in the automotive systems. It consists of three modules: SOME/IP, SOME/IP Service Discovery (SD) and SOME/IP Transformer. SOME/IP fundamental module is managing the serialization and deserialization of transmitting data, SD module enables the connection establishment and service discovery procedure and the Transformer module specifies automotive/embedded data serialization [4].

There are multiple protocols which can be used for in-vehicle cross-domain communication such as DDS, HTTP, MQTT, web sockets, etc. Besides the abovementioned fact that the SOME/IP is created for the automotive industry there are several functional benefits that made it our choice for such use-case. First, the mandatory configuration of the communication over SOME/IP enables somewhat more deterministic behavior in contrary to another protocols and mechanisms used to implement SOA in the web and cloud computing such as the HTTP for example. Additional benefit is that SOME/IP provides multiple types of communication. Comparing to the HTTP, which allows only request-response communication initiated from clients, SOME/IP provides both request-response and publish-subscribe approaches. Furthermore, SOME/IP does not require communication establishment for each data exchange, but only for initial client-service connection and it can rely on both, TCP and UDP protocols in contrary to the communication implementing Representational State Transfer (REST) principles.

Although having different architectures, diverse software platforms use the SOME/IP communication stack based on the similar concept as depicted in Fig. 1.

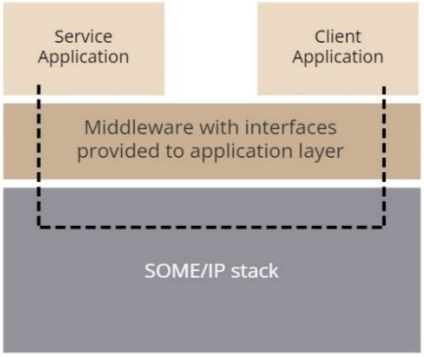


Fig. 1. Concept of SOME/IP implementation in automotive platforms

Usually, the middleware which provides the applications with the particular interfaces based on the determined configuration parameters is implemented by the standard. Depending on whether it is event, method or field that is defined by the configuration the data exchange would be performed by publishing the information to subscribed client when a logic on the service side determines so, client requesting the execution of a method on service side and getting the response if needed and getting, setting or notifying about the changed state of a field, i.e. attribute on the service side respectively.

Services in automotive SOA need to meet strict requirements regarding the service discovery and startup latency time [6]. Authors in [3] even propose dividing and isolating secured and exposed subnetworks in order to accomplish more reliability, since the service discovery mechanisms cannot guarantee that the service will be provided at the needed time. However, this aspect will not be examined in this paper, but another one instead – how SOA is implemented within available architectures and how it can be used for inter-communication between ADAS and IVI domains.

B. IPC Standards in IVI Domain

Modern vehicles are currently competing to meet the requirements driven by the consumer technology, especially in the infotainment domain. Inside the vehicle, the passengers expect the experience they have when using everyday portable devices, such as tablets and mobile phones. They are used to being able to install and use various types of applications developed by different vendors. In order to meet these requests, it is necessary to utilize the globally accepted standards for building scalable and portable platforms.

1) GENIVI approach

The former GENIVI (currently COVESA) alliance drives the development of open standards and technologies used in automotive systems. Their goal is to address the challenges which the in-vehicle infotainment components are facing when reaching to the outside world (cloud services, other vehicles, etc.) and communicating with other in-vehicle infotainment components as well. They offer the CommonAPI [8] – an inter-process communication middleware based on the FRANCA framework, which provides service-oriented mechanisms. It is designed to split the applications implementation apart from the communication mechanisms used between the implemented application components.

Since the only purpose of this middleware is to provide interfaces between lower (platform services and protocols) and upper (applications) layers, its implementation is generated mainly from the FRANCA Interface Definition Language (FIDL) to make its utilization easier. Applying the specified interface definition language – FIDL, it enables flexible deployment models. This way, the dynamic behavior of an API is specified by defining client/server interaction interfaces, states and transitions between them [7]. The communication itself is performed by using the generated Stub and Proxy classes relying on the CommonAPI middleware within the Service and Client applications respectively. This way, the concept from Fig. 1 is kept since the entire CommonAPI stack

 ISBN 978-86-7466-930-3

including the Stub and Proxy provides applications with interfaces for usage of the SOME/IP mechanisms.

Additionally, COVESA semantically differentiates between the two realms: Common-API Core, which does not depend on the communication protocol itself, and CommonAPI Binding which is protocol-specific [8]. Currently, the CommonAPI support two RPCs, D-Bus and the SOME/IP. In order to set deployment parameters for chosen protocol, the FRANCA Deployment (FDEPL) files are used along with the FIDL.

2) Android approach

Android is an open-source operating system mainly utilized for mobile devices. It enables deployment on wide range of hardware platforms and supports third-party applications development [9]. Currently, the automotive industry is facing a similar requirement for the possibility for third-party application development and utilization, therefore the automotive community is more interested in the Android platform [10].

Android platform has the mechanisms for feasible handling of the Inter-Process Communication (IPC) via its proprietary interface definition language called AIDL. It provides a programming interface utilized by both the client and the service using the IPC to communicate with each other [11]. Although AIDL has similar functionality as other IDLs, its utilization does not rely on the same paradigm as it is the case with the FIDL and COVESA's Common-API service-client communication model. Additionally, the SOME/IP had not been supported in Android until vsomeip version 3 was released. The possible correlation between CommonAPI and Android and more details about the AIDL paradigm and its communication mapping to other mechanisms will be addressed in the Section 4.

C. IPC standards in ADAS domain

The previously described standards are used for the implementation of the application for the in-vehicle infotainment part of the automotive system. On the other hand, ADAS domain is faced with the challenges driven by different requirements, as it considers safety-critical algorithms and modules. Nevertheless, ADAS domain implies the integration of functionalities provided by machine vision and sensor fusion. Lots of these algorithms are used in consumer technologies, i.e., in the IVI domain also. Therefore, the benefit of exchanging resources between two mentioned domains is obvious, since there is a set of functionalities they share. A standard that has become a convention for the implementation of ADAS domain functionalities is AUTOSAR.

The AUTOSAR standard considers both safety host and performance host implementations. Safety hosts are referring to ECU's cores with safety and security control features specialized for the automotive industry. Classic AUTOSAR platform is designed for the fully deterministic, deeply embedded standardization of safety hosts. Furthermore, the Adaptive AUTOSAR platform is offering more flexibility by addressing operability and communication mechanisms more suitable for high-performance computing devices called performance hosts. Since the performance host resources and algorithms complexity are more similar to the ones in the IVI domain, we will focus on the sharing resources and features of the Adaptive AUTOSAR platform.

User applications are running on the top level, right on top the AUTOSAR Runtime Environment for Adaptive Applications (ARA). The main component of ARA is ara::com, a middleware controlling the communication within a system. It provides the interfaces to the user applications which allow data exchange with both local and remote applications and ARA services [12].

Equivalent to the FIDL, ara::com interfaces in ARA-API are defined by the ARXML. Interfaces are provided to applications with the exact same purpose as it is the case with CommonAPI, to decouple the applications development from the communication mechanism. It is done by utilizing two artifacts - Skeleton and Proxy which implement the SOA paradigm, i.e., the service-client communication, likewise it is the case with the Stub and Proxy in CommonAPI. Skeleton represents the generated instance which provides service calls functionalities. On the other hand, Proxy is a generated instance which provides the client calls functionalities.

III. CURRENT CROSS-DOMAIN RESOURCE SHARING SOLUTIONS

Most of the research in the field of interconnecting different automotive domains focuses on the modelling and implementation of multi-ECU system using a single standard. Since meeting the safety and latency requirements for ADAS is critical, it dictates the approach to use Adaptive AUTOSAR for both the ADAS and IVI realm. This way, for the sake of connectivity between different domains, neither the CommonAPI nor Android are used, although they are a better fit for IVI domain, since the development is forced to a single standard approach which must fit ADAS requirements. The authors then try to deal with the shortcomings that the AUTOSAR standard provides in terms of UI as an important aspect of in-vehicle infotainment [13]. Authors in [14] presented challenges of modelling ADAS components for camera resource sharing. However, it is needed to perform further research on the most suitable communication channel for the transmission of sensing data and data streams along with the research on the most suitable communication mechanisms by considering the entire, end-to-end communication context for such resources sharing between domains in automotive, the SOME/IP is not the most effective solution for such use-cases. Furthermore, taking into consideration the variety of operating systems on the other side, such implementation cannot be taken "as is".

COVESA alliance recognized this challenge and tried to attain the adaptation between Adaptive AUTOSAR and the CommonAPI by creating FARACON generator [15]. This generator is used to translate the interface definition files from one standard to another. This can be considered as a first step towards the mapping of the features between standards. However, it does not solve the cross-domain heterogeneity issue, which is somewhat more complex.

There are not many papers that provide the actual proposition for interconnection between ADAS and IVI domains utilizing different standards. i.e., following AUTOSAR on ADAS and CommonAPI or Android on IVI side. The existing solutions have recognized the need for such

 ISBN 978-86-7466-930-3

binding, but are also typically reduced to simple utilization of socket-based communication with no actual research background on the available protocols and state-of-the-art SOA principles [16]. Additionally, the inter-process communication paradigms diversity when considering the various platforms standards is not actually covered even in papers which provide the extensive solution for heterogenous in-vehicle environments [17]. Hence, there is no comprehensive project dealing with all aspects of this topic.

Since this topic is substantive and our project is still in the development, some of the challenges will not be covered by this paper but will be addressed in future work instead.

IV. Proposed Solution

In this section, we will discuss the possible approaches that allow remote procedure calls and exchange of data between ADAS and IVI domains of the vehicle. Our goal is to provide the connectivity, without compromising the functionality of the IVI domain offered by the CommonAPI or Android, or the safety features provided by the AUTOSAR in ADAS domain. We will design our solution using the service-oriented architecture principles, which fit perfectly into the scenarios we want to support. Our focus is on allowing IVI domain applications to use raw measurement data from ADAS sensors, as well as the results of some of the algorithms that run on the ADAS side. The opposite direction of integration is not possible, due to potential safety issues.

There are several examples of use-cases where the proposed cross-domain inter-connection can be beneficial. For example, inputs from cabin camera which is commonly used for driver monitoring on ADAS side can be shared for video calls and other applications using camera in IVI domain. This way, the cost of providing redundant hardware components would be avoided. On the other hand, data from sensors monitoring tire pressure, engine temperature and other crucial components of the vehicle could be easily transferred and handled by the applications in the IVI domain. These applications could then not only inform the driver, but also provide the better user experience by searching for the recommendations and manuals on the Internet, or help by finding the route to the nearest mechanic service. The results of the data processing algorithms such as traffic sign detection and recognition or driver drowsiness monitoring could also be used by the IVI domain applications, to propose rest stops, provide tourist information, etc.

To connect the components of the two domains in the proposed solution, Ethernet-based communication will be used. Recently, Ethernet has taken on the role of the vehicle communication backbone because of its bandwidth, scalability, flexibility and prevalence. In all of the aforementioned terms, Ethernet is generally superior to other in-vehicle buses, which are designed and optimized to fit only specific use-cases. For example, CAN provides the reliability which Ethernet cannot achieve because of the different transmission media access strategies. On the other hand, CAN is the automotive specific technology which means that Android, as a standard that was not created solely for the automotive industry, does not support CAN bus module natively. Similarly, other in-vehicle buses are created to meet

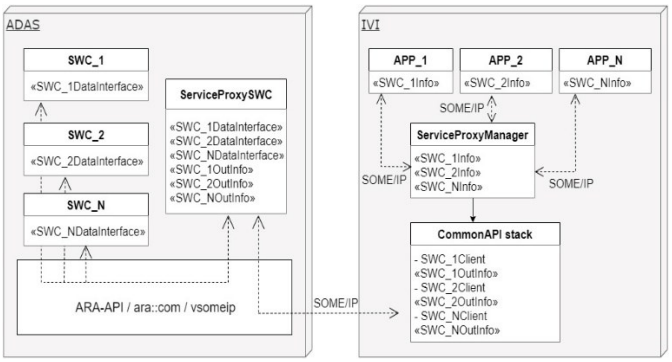


Fig. 2. Centralized interconnection approach with POSIX OS on IVI side

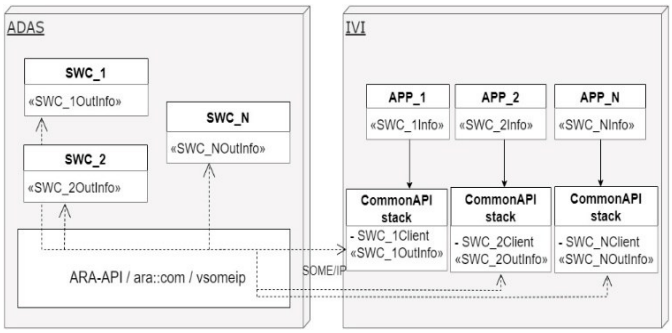


Fig. 3. Distributed interconnection approach with POSIX OS on IVI side

the requirements of automotive signal-based communication, where priority is the price and the determinism of the communication mechanism, not the bandwidth itself. On the other hand, Ethernet is widely used technology which makes it suitable for interconnection of different domains. To exchange data between the domains, we will use SOME/IP, from the reasons already discussed in Section 2, and it can be used over the Ethernet network.

Typically, IVI solutions can either run on Linux operating system and use CommonAPI mechanisms for the inter-process communication, or they can be Android-based. For both of these cases, we will propose the solution architectures in the following sections.

Since Adaptive AUTOSAR and CommonAPI both implement the SOME/IP communication interfaces, this is the easiest way to establish the communication between the two domains in the SOA manner. The ADAS side is implemented by following Adaptive AUTOSAR standard and the IVI domain uses the CommonAPI middleware running on the native operating system such as Linux. This scenario is depicted in Fig. 2 and Fig. 3. Communication in Adaptive AUTOSAR is handled by ara::com which natively supports vsomeip as a library that implements SOME/IP standard. The same vsomeip implementation is utilized in CommonAPI SOME/IP stack. This means that serialization and deserialization of data shall be handled in the same way, so both sides will be able to interpret data properly.

Information from components on ADAS side are initially given to the Service Proxy SWC via SOME/IP implemented within the ara::com module. This data is furtherly forwarded to the corresponding CommonAPI clients grouped together on the IVI side in a single Service Proxy Manager instance (Fig. 2). Such inter-domain transfer is performed over SOME/IP on

 ISBN 978-86-7466-930-3

demand of IVI applications or when the event/change is captured.

The type of communication between the ADAS and the IVI does not necessarily have to match the communication between the ADAS Service Proxy and other SWCs which means that IVI applications can request the data through the Service Proxy Manager instance over method mechanism, but the sharing information on the ADAS side can be sent to the Service Proxy SWC from the actual service component as an event for example.

Another approach is to implement separate services for each CommonAPI client (Fig. 3) so the Service Proxy components on both sides are unneeded and the information will be provided from ADAS ara::com services to the IVI CommonAPI clients included in particular application. The first approach is easier to scale and can be used with the variable number of application instances. Also, it can be favorable from the safety perspective since it can contain mechanisms to protect from other SWCs from being jeopardize by IVI applications. On the other hand, the second is superior in terms of reliability, because there is no single central node which distributes the data between the applications. This way, the malfunction of one service does not affect the operability of others. Furthermore, the monolithic design is harder to maintain, as even minor changes require the entire integration cycle. The speed of access to information is also one of the factors that is on the side of the distributed approach.

As already said, Android has recently become the operating system of choice for IVI applications, as most of the users are familiar with it and it is available on a very large variety of hardware. The interconnection of the ADAS domain with the IVI domain running on the Android platform is a bit more challenging for the implementation. Namely, Android itself does not have mechanisms to implement SOME/IP client which can communicate with ADAS side. Therefore, the CommonAPI must also be used in this scenario in the exact same way it was the case when non-Android OS was examined, as it is presented in Fig. 4 and Fig. 5.

The CommonAPI clients are included within an Android native service and provided information can be transferred to both, custom applications and HAL modules over AIDL. The entire CommonAPI stack can be built within an AOSP (Android Open Source Project maintained by Google) with the soong build system. Still, the vsomeip itself has some dependencies, such as boost library, which can cause issues while building within the AOSP. Further options are to build CommonAPI client beyond the AOSP, with the Native Development Kit – NDK, or even to use another implementation of SOME/IP standard instead of vsomeip, which would eliminate the dependencies such as the afore-mentioned boost library. Nevertheless, CommonAPI clients must be included in Android services so the data from ADAS can be provided to applications or other services in IVI domain.

Additionally, the mapping of SOME/IP service-client communication paradigm from CommonAPI/AUTOSAR to Android represents a challenge. Namely, AIDL files used for

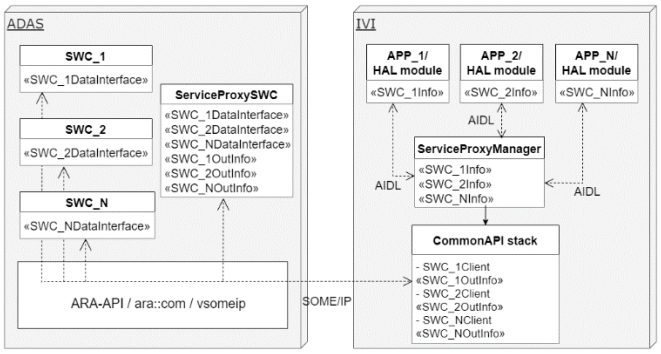


Fig. 4. Centralized interconnection approach with Android on IVI side

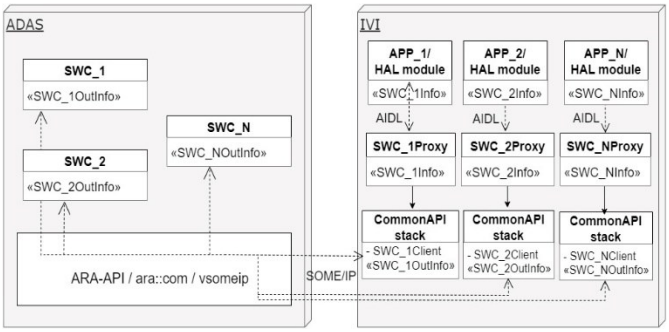


Fig. 5. Distributed interconnection approach with Android on IVI side

interface generation provide the inter-process communication by marshaling the object instances through the binder. This is not suitable for the event-triggered traffic. Event-triggered communication from service to clients within a SOA is performed in a way that the client itself is only subscribed to the events from service. This specific case cannot be covered by using regular AIDL, because AIDL always assumes that the communication is initiated from the client side

Our approach was to incorporate the receiving (client) side for broadcasts and events in the Android native service, and further distribute this information to the interested applications. The easiest way to achieve this is to set properties based on the information received by the Android native service. The interested applications can then read that particular property. This approach has a big limitation since the data can only be used to transfer flags and states since properties do not exist to be used as IPC mechanisms.

We went for the another, a slightly more demanding way for implementation. It assumes the creation of a helper AIDL, which will pass the interface object as a parameter from the applications to Android native service, in order to enable the Android native service to react on event-trigger signals from CommonAPI by invoking methods from the passed object like it is a client to the application. Furthermore, in order to avoid forming a list of registered applications which methods will be invoked when event-triggers we created additional helper service in Java with which it will be communicated via that helper AIDL and which will furtherly provide Intents to the applications. Additionally, it is even possible to have stand-alone Java service which will use the CommonAPI via Java Native Interface – JNI. JNI is necessary in this scenario to enable inter-operability between Java and C/C++ code, since the COVESA provides CommonAPI middleware in C++

 ISBN 978-86-7466-930-3

programming language.

V. Conclusion and Further Work Directions

This paper presented both, the theoretical and the practical aspects of proposal for service-oriented communication between ADAS and IVI domains. Background and motivation for such binding are provided, along with the key challenges and limitations as it is summarized in Table 1. Several approaches were elaborated in order to satisfy system heterogeneity. Additionally, the beneficial use-cases are discussed in order to emphasize the value of bonding itself.

TABLE I
Major challenges and limitations

Challenges	Limitations
SOA paradigm mapping	Implementing broadcast/events with AIDL principles
Centralized or distributed approach	Prioritization, robustness, bandwidth
Data transfer channel	Performance evaluation
Safety	Enable safety solution for Android
Generation of inter-communication	Verification and tool qualification

In our future work, we will focus on the evaluation of the latency, bandwidth and robustness in order to present comprehensive comparison of the centralized service proxy manager approach with the distributed approach, to determine the optimal design.

The performance of data transfer channel shall be furtherly examined too by considering the Audio Video Bridging (AVB) and other mechanisms for big data integration. It is needed to determine the exact use-cases where the data shall be transferred only within SOME/IP request/response, and where it is more suitable to open additional channel for data transfer. Several aspects regarding data size and safety shall be analyzed in order to define the optimal approach.

Safety requirements are maybe the most complex of all challenges that we plan to address. Safety analysis implies the detail examination on the system level too. It is not enough only to implement mechanisms for Android native service to control which applications can use it based on the given permissions and to properly handle dead listeners and multiple registrations which is done by now. Hazard analysis on the system level involves hardware and OS safety competence and certain communication determinism (Time-Triggered Ethernet or Time-Sensitive Networking). Android itself currently cannot have any Safety integrity level but QM [18]. From that reason, it is mandatory to involve the hypervisor if the communication must be initiated from the Android [19].

The final challenge will be to automate the entire process of providing resources from ADAS to IVI. This means that our goal will be to generate the translation between ARXML, FIDL and AIDL, as well as the generation of Android service along with the code that is responsible for providing resources from service on ADAS side to the IVI realm.

Acknowledgment

This research (paper) has been supported by the Ministry of Education, Science and Technological Development through project no. 451-03-68/2022-14/ 200156 “Innovative scientific and artistic research from the FTS (activity) domain”.

References

[1] P. Bajaj, M. Khanapurkar, “Automotive networks based intra-vehicular communication applications. New Advances in Vehicular Technology and Automotive Engineering”, pp. 207-230, (2012).

[2] B. Glas, J. Guajardo, H. Hacioglu, M. Ihle, K. Wehefritz, A. Yavuz, “Signal-based automotive communication security and its interplay with safety requirements.”, In Proceedings of Embedded Security in Cars Conference, 2012

[3] M. Bellanger, E. Marmounier, E, “Service Oriented Architecture: impacts and challenges of an architecture paradigm change”, In 10th European Congress on Embedded Real Time Software and Systems, (2020).

[4] G. L. Gopu, K. V. Kavitha, J. Joy, “Service oriented architecture based connectivity of automotive ecus”, In 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), pp. 1-4, (2016).

[5] “Example for a Serialization Protocol (SOME/IP)”, [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-1/AUTOSAR_TR_SomeIpExample.pdf”, last accessed 2022/10/1.

[6] J. R. Seyler, T. Streichert, M. Glaß, N. Navet, J. Teich, “Formal analysis of the startup delay of SOME/IP service discovery”, In 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 49-54, (2015).

[7] "Welcome to FRANCA!" [Online]. Available: https://github.com/franca/franca, last accessed 2022/10/1.

[8] “CommonAPICppUserGuide”, [Online]. Available: https://usermanual.wiki/Document/CommonAPICppUserGuide.1126244679/html, last accessed 2022/10/1.

[9] G. Macario, M. Torchiano, M. Violante, M., “An in-vehicle infotainment software architecture based on google android”, In 2009 IEEE International Symposium on Industrial Embedded Systems, pp. 257-260, (2009).

[10] N. Pajic, M. Bjelica, “Integrating Android to Next Generation Vehicles”, In 2018 Zooming Innovation in Consumer Technologies Conference (ZINC), pp. 152-155, (2018).

[11] “Android Interface Definition Language(AIDL)” [Online]. Available: https://developer.android.com/guide/components/aidl, last accessed 2022/10/1.

[12] S. Fürst, M. Bechter, “AUTOSAR for connected and autonomous vehicles: The AUTOSAR adaptive platform”, In 2016 46th annual IEEE/IFIP international conference on Dependable Systems and Networks Workshop (DSN-W), pp. 215-217, (2016).

[13] S. Aust, ”Paving the way for connected cars with adaptive AUTOSAR and AGL”, In 2018 IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops), pp. 53-58, (2018).

[14] M. Kotur, N. Lukić, M. Krunić, G. Velikić, “One solution of camera service in AUTOSAR ADAPTIVE environment”, In 2020 IEEE 10th International Conference on Consumer Electronics, pp. 1-5, 2020.

[15] “Franca/ ARA:COM Interoperability”, [Online]. Available: https://at.projects.genivi.org/wiki/download/attachments/16026116/GENIVI%20Franca-ARA-COM-tech-brief-20181219.pdf, last accessed 2022/10/1.

[16] K. Omerovic, J. Janjatovic, M. Milosevic, T. Maruna, “Supporting sensor fusion in next generation android In-Vehicle infotainment units”, In 2016 IEEE 6th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), pp. 187-189, (2016).

[17] M. Milosevic, M. Z. Bjelica, T. Maruna, N. Teslic, “Software platform for heterogeneous in-vehicle environments”, In IEEE Transactions on Consumer Electronics, pp. 213-221, (2018).

[18] L. Perneel, H. Fayyad-Kazan, M. Timmerman, “Can Android be used for real-time purposes?”, In 2012 International Conference on Computer Systems and Industrial Informatics, pp. 1-6, (2012).

[19] M. Bjelica, Z. Lukac, “Central vehicle computer design: software taking over”, IEEE Consumer Electronics Magazine, 8(6), 84-90, 2019.

 ISBN 978-86-7466-930-3

Утврђивање сличности софтверског кода

Захарије Радивојевић, Милош Цветановић

***Апстракт*—Утврђивање сличности софтверског кода представља област истраживања у оквиру софтверског инжењерства. Поред великог броја домена у којима налази примену оно представља кључни елемент за утврђивање постојања софтверских клонова, а самим тим утиче на софтвер током читавог његовог животног циклуса, током дизајна, развоја и одржавања. У овом раду је, на основу искустава аутора, дат преглед домена и карактеристике кодова који се у датим доменима пореде. Описане су и технике које се примењују у овој области независно од домена примене, као и карактеристике кодова које се датим техникама пореде. Такође је изложено пет сукцесивних поступака које су аутори развили за примену у домену откривања неовлашћеног коришћења лиценци. Поступци обухватају описе техника за утврђивање сличности бинарног кода, утицаја архитектуре рачунара на утврђивање сличности бинарног кода, утицаја трансформација преводиоца на утврђивање сличности бинарног кода, могућности за коришћење неуралних мрежа за утврђивање сличности бинарног кода, као и коришћење секвенци операција за утврђивање сличности бинарног кода.**

***Кључне речи*—Софтверски клонови, сличност кода, бинарни код, софтверске метрике, неуралне мреже.**

I. Увод

Анализа изворног кода програма има широки спектар примена, које укључују од рангирања и категоризације програма [1], преко откривања преписивања у домаћим задацима [2], до откривања недостатака у програмима [3]. У овој анализи као један од важних корака јесте утврђивање сличности делова кода. Постоје домени примене код којих је утврђивање сличности кода од пресудног значаја. Ови домени обухватају откривање злонамерног кода [4], реверзно инжењерство софтверских закрпа [5], откривање неовлашћеног коришћења лиценци [6], и слично. Ова анализа се у већини случајева обавља над изворним кодом програма, али се у неким случајевима може обављати и над асемблерским кодом програмима као и директно над бинарним записом извршног кода програма. Без обзира да ли се ради анализа изворног или извршног кода, као и без обзира на домен примене поступак се заснива на откривању софтверских клонова.

Овај рад има за циљ да представи област истраживања која се бави анализом сличности софтверског кода. Рад даје сажетак истраживања у овој области и нема за циљ да да систематичан преглед литературе у овој области. Рад је организован тако да ће у другој глави бити представљени домени примене утврђивања сличности кода. У трећој глави објашњене постојеће технике утврђивања сличности кода, док ће у четвртој глави бити представљени резултати истраживања у области анализе софтверског кода датог у бинарном облику које су аутори спровели током више година. Последња, пета, глава представља закључак овог рада.

Захарије Радивојевић, доктор електротехнике и рачунарства – Електротехнички факултет, Универзитет у Београду, Булевар краља Александра 73, 11000 Београд, Србија (e-mail: zaki@etf.bg.ac.rs).

Милош Цветановић, доктор електротехнике и рачунарства – Електротехнички факултет, Универзитет у Београду, Булевар краља Александра 73, 11000 Београд, Србија (e-mail: cmilos@etf.bg.ac.rs).

II. Домени примене одређивања сличности кода

У овој глави ће бити описани карактеристични домени у којима се може применити утврђивање сличности софтверског кода. За сваки од домена биће наведена сврха са којом се утврђивање сличности користи. Такође, за сваки од домена биће наведен и угледни примерак алата и описан његов начин функционисања. Преглед основних карактеристика појединих домена је дат у табели 1.

A. Детекција клонова

У домену детекције клонова упоређивање сличности кода се користи са циљем упаривања делова кода који могу потицати од сличног изворног кода или од сличног бинарног кода који потиче од истог изворног кода. Два дела кода која представљају софтверски клон поседују исту семантику, али се разликују по нивоу синтаксне сличности и на основу ње су класификовани у четири типа. Два синтаксно идентична кода се класификују као клонови типа 1, а могу се разликовати само по форматирању и пратећим коментарима. Уколико између два кода постоји и разлика у називима литерала онда се класификују као клонови типа 2. Убацивање малог броја додатних инструкција или промена редоследа инструкција која не утиче на резултат извршавања кода се класификује као клон типа 3. Све остале промене које чувају семантику али у већој мери мењају синтаксу оригиналног кода се класификују као клонови типа 4 [7].

Један од алата који се може користити у поступку откивања софтверских клонова је ACD алат [8]. Овај алат покушава да упари секвенце инструкција највеће могуће дужине. За упаривање инструкција потребно је да редослед упарених инструкција буде идентичан у оба кода који се упарују и да се утврди да се инструкције нису раније упариле. Као додатни услов се користи да је циљна адреса скокова иста у односу на остале упарене инструкције. Ако се током упаривања секвенци деси да се мали број инструкција не може упарити оне се игноришу. За сваке две упарене секвенце израчунава се тежина. Сваки пут, када су две инструкције упарене, тежина се

 ISBN 978-86-7466-930-3

ТАБЕЛА 1. ПРЕГЛЕД ДОМЕНА ПРИМЕНЕ

Домен	Тип кода	Тип клона	Циљ потраге	Узрок разлика
Детекција клонова	Изворни	1, 2, 3, 4	Сличне процедуре	Програмер
Откривање неовлашћеног коришћења лиценци	Изворни/Бинарни	1-Изворни, 4-Бинарни	Еквивалентне процедуре	Преводилац
Откривање злонамерног кода	Бинарни	1, 2, 3	Сличан код	Програмер/Преводилац
Откривање рањивости кода	Бинарни	1, 2, 3	Сличан код	Програмер

повећава за одређену вредност. Када се у некој од секвенци наиђе на инструкцију која се занемарује, да би се наставило упаривање, тежина се умањује. Процес упаривања две секвенце почиње са две инструкције које се могу упарити и наставља се све док тежина претходно упарених секвенци не буде једнака нули. Након тога, покушава се са побољшањем упаривања две секвенце откривањем нових инструкција које се могу упарити унутар ових секвенци а које би могле повећати укупан број упарених инструкција.

B. Откривање неовлашћеног коришћења лиценци

Све већи број кршења ауторских права у софтверској индустрији доводи до огромних економских губитака за носиоце ауторских права [9]. Једна од ситуација у којој може доћи до кршења је лиценцирање софтвера са двоструком лиценцом. Овај тип пословног модела се обично користи да би се подржао софтвер отвореног кода у комерцијалне сврхе. У таквом пословном моделу, власник ауторских права софтвера нуди изворни код бесплатно за некомерцијалну употребу, али остварује профит продајом ауторских права компанијама које желе да користе изворни код у својим производима. Повреда настаје када се изворни код власника користи у производу без лиценце добијене од власника. За разлику од домена откривања клонова, где је изворни код углавном доступан, одређивање сличности кода се у домену кршења лиценци примењује углавном над извршним кодом. Разлике које у извршним кодовима постоје последице су употребе различитих преводиоца над истим изворним кодом.

Један од алата који се може користити у поступку откивања кршења лиценце је BAT алат који је заснован на приступу описаном у раду [10]. Приступ користи три технике за ублажавање негативних ефеката потенцијалних разлика између кодова који се пореде. Прва техника прикупља стринг литерале који се појављују у коду који се пореди. Након тога, покушава да пронађе исте литерале у коду за који се сумња да крши лиценцу. Друга техника претпоставља да ће алгоритам компресије података успети да компримује боље два кода заједно ако међу тим кодовима постоје исте секвенце инструкција. Успех се мери односом између величине архиве, која се састоји од два кода, и укупне величине архива које се појединачно састоје од само једног од два кода које се пореде. Што је однос мањи, то се очекује већа сличност између упоређених кодова. Трећа техника се заснива на израчунавању бинарних разлика између кодова. Што су разлике мање, вероватноћа да посматрани код крши лиценцу је већа. Алат BAT користи се за откривање да ли посматрани код крши лиценцу неког од кодова из репозиторијума, а за ту сврху користи само прву технику.

C. Откривање злонамерног кода

Злонамерни код је сваки код који има могућност да оштети било који рачунарски систем [4]. Количина злонамерног кода се сваке године све брже повећава и представља озбиљну безбедносну претњу. Отуда је откривање злонамерног кода критична тема у рачунарској безбедности. У оквиру комерцијалних антивирусних програма откривање злонамерног кода је засновано на поређењима отисака метода са познатим отисцима метода. Недостатак овог начина откривања злонамерног кода је да не успева да открије нове варијације већ познатог злонамерног кода. У циљу превазилажења тог недостатка може се употребити одређивање сличности злонамерног кода са циљем откривања дела кода који показује слично понашање као што је понашање које има код из већ познате колекције злонамерног кода.

Један од алата који се може користити у поступку откивања злонамерног кода који користи одређивање сличност кодова је алат који упоређује секвенце кодова операција [4]. Током поређења посматрају се низови кодова операција фиксних дужина и узима се у обзир да различити кодови операција имају различиту релевантност током поређења. Код овог алата се најпре обавља анализа великог броја злонамерног кода и регуларног кода и утврђује се релевантност кодова операције. Затим се израчунавају фреквенције појављивања свих секвенци изабране дужине у кодовима који се упоређују. За сваки од кодова формира се вектор чије су компоненте бројеви појављивања појединачних секвенци. Свака од компоненти вектора се затим множи са производом релевантности свих кодова операција који се појављују у посматраној секвенци како би се елиминисао шум који уноси ирелевантан код. На крају сличност између кодова који се пореде се израчунава коришћењем косинусне сличности ових вектора. Приступ такође предлаже комбинацију добијених резултата за неколико различитих дужина посматраних секвенци.

 ISBN 978-86-7466-930-3

D. Откривање рањивости кода

У домену рањивости софтверског кода могу се разматрати анализа рањивости и откривање рањивости. Анализа рањивости има за циљ да формалним методама потврди или оповргне хипотезу да је софтвер рањив. Приступи анализи рањивости се могу класификовати у три категорије: статичка анализа, динамичка анализа и хибридна анализа кода. Откривање рањивости има за циљ да мање формалним поступцима лоцира конкретну рањивост у коду. У овој области постоји више различитих приступа који између осталог обухватају: тестове пенетрације софтвера, насумично тестирање, и статичку анализу токова података [11]. За разлику од претходно разматраних домена у овом домену се одређивање сличности софтверског кода користи за откривања разлика између две различите верзије истог кода.

Један од алата који се може користити за потребе откривања рањивости је алат који покушава да упари делове кода односно процедура које су скоро идентичне [12]. Први предуслов за упаривање процедура је да су идентичне према свим датим критеријумима (тзв. селекторима): броју основних блокова, броју ивица у контролном графу тока и броју позваних потпрограма. Други предуслов за упаривање је да не постоји друга процедура која је идентична са процедурама које се пореде на основу коришћених критеријумима. Да би се смањило ограничење другог предуслова у свим корацима алгоритма, посматрају се подскупови процедура из кодова који се пореде а који су дати у бинарном облику. Приликом формирања првих парова, подскупови се формирају издвајањем процедура које задовољавају неку карактеристику. Неке од карактеристика које се користе су број улазних и излазних грана процедура у графу позива, исти називи процедура, референце на исте стрингове и број пута када су се неке од инструкција појавиле. Када даље упаривање по датим карактеристикама више није могуће, формирају се нови подскупови процедура које се позивају из упарених процедура. Подскупове такође формирају процедуре које позивају упарене процедуре. Алгоритам се понавља за сваке две упарене процедуре све до тренутка када даље упаривање више није могуће. Унутар упарених процедура врши се даље упаривање основних блокова и инструкција.

III. Технике утврђивања сличности кода

У овој глави ће бити описане најзаступљеније технике утврђивања сличности софтверског кода описане у прегледним радовима [7], [13], [14], [15], [16], [17]. За сваку од техника биће наведен кратак опис, применљивост, могућности проширења, као и на које типове софтверских клонова се најчешће примењује. Такође, биће наведен и угледни примерак алата и описане његов начин функционисања за сваку од техника. Поред техника које су описане у наставку постоје и хибридне технике код којих се може јавити синергистички ефекат тако да могу детектовати додатне типове клонова у поређењу са типовима клонова који се могу детектовати било којом од коришћених техника детекције одвојено.

A. Технике засноване на поређењу текста

Технике засноване на поређењу текста посматрају секвенцу линија изворног кода. Како би се пронашла секвенца истоветних линија, стрингова или лексема пореде се делови два изворна кода или њихови отисци. Када се у датим деловима кода открије да су поједини делови слични они се проглашавају за клонове одређене класе. Ове технике се углавном користе за откривање клонова код виших програмских језика и развијени алати често подржавају више од једног програмског језика. У случају да их је потребно проширити тако да подрже неки нови програмски језик или није потребно ништа додатно имплементирати или је потребно имплементирати лексички анализатор. Ова техника се углавном може применити за откривање клонова типа 1 и типа 3, а може се применити и за откривање клонова типа 2. Ова техника се у имплементацијама показала као техника средње сложености.

Један од алата који користи технику засновану на поређењу текста је SimCad алат [18]. Овај алат подржава више програмских језика и то: C, C#, Java, Python. Процес откривања клонова се заснива на техникама откривања скривеног знања и рударења података и користи алгоритам за груписање података као и претрагу података засновану на вишенивојским индексима. Овај алат извршава три фазе обраде: прелиминарну обраду, откривање клонова и генерисање резултата. У фази прелиминарне обраде користећи Simhash алгоритам се генеришу отисци кода, након чега се обавља индексирање формирањем вишенивојски индекси за потребе брже претраге. У другој фази се обавља откривање клонова тако што се фрагменти кода групишу у кластере користећи Хемингово растојање између генерисаних отисака који одговарају тим фрагментима кода. Парови фрагмената који не прелазе одговарајући ниво сличности бивају елиминисани из кластера, а такође мора постојати и одређен број фрагмената кода у кластеру. У трећој фази се врши чишћење добијених резултата као и њихов приказ у одговарајућем формату.

B. Технике засноване на поређењу токена

Технике засноване на поређењу токена посматрају секвенцу токена издвојених из изворног кода. Издвајање токена се обавља у поступку лексичке анализе изворног кода. Секвенца токена се формира као скуп токена на одређеном нивоу гранулације. Срж ове технике представља поређење токена који припадају суфиксним стаблима или суфиксним низовима састављених од секвенци токена. Ова техника откривања користи посебан објекат који представља апстракцију конкретних вредности идентификатора и литерала и који води рачуна о очувању њиховог међусобног редоследа. Ове технике се углавном користе за откривање клонова код виших

 ISBN 978-86-7466-930-3

ТАБЕЛА 2. ПРЕГЛЕД ТЕХНИКА

Техника	Тип кода	Тип клона	Проширивост	Сложеност извршавања
Поређење текста	Изворни	1, 2, 3	-/Лексички	Средња
Поређење токена	Изворни	1, 2, 3	Лексички	Мања
Поређење метрика	Изворни/Бинарни	1, 2, 3, 4	Лексички, парсер	Средња
Поређење апстрактног синтаксног стабла	Изворни/Бинарни	1, 2, 3	Парсер	Средња
Поређење графа зависности	Изворни/Бинарни	1, 2, 3, 4	Парсер	Велика

програмских језика, али је за разлику од техника заснованих на поређењу текста зависност од програмског језика већа. У случају да је потребно проширити их подршком за нови програмски језик неопходно је имплементирати лексички анализатор. Ова техника се углавном може применити за препознавање клонова типа 1, 2 и 3, и представља једну од најцитиранијих техника за одређивање сличности. Ова техника се у имплементацијама показала као техника мање сложености.

Један од алата који користи технику засновану на поређењу текста је CCFinder алат [19]. Овај алат подржава више програмских језика и то: C/C++, C#, Cobol, Java, VB. Обрада коју спроводи овај алат се извршава у четири фазе: лексичка анализа, трансформација међурезултата, одређивање сличности и форматирање добијених резултата. У фази лексичке анализе се свака линија изворног кода дели у низ токена у складу са лексичким правилима датог програмског језика. У другој фази се обавља трансформација добијених низова токена користећи правила трансформације и правила замене одређених типова токена. У трећој фази се обавља упоређивање свих подстрингова како би се идентификовали парови клонова. На крају, у четвртој фази, се препознати клонови пресликавају на линије изворног кода из кога потичу.

C. Технике засноване на метрикама

Технике засноване на метрикама посматрају карактеристике изворног програма издвојене користећи скуп метрика. Издвајање карактеристика се обавља полазећи од кода који је трансформисан у одговарајућу структуру. Метрике се могу израчунавати на основу имена, размештаја елемента, израза, контроле токе функција, и сличних елемената структуре кода. Одређивање сличности се обавља израчунавањем удаљености између одговарајућих чланова у формираном метричком простору. Ове технике се углавном користе за откривање клонова код виших програмских језика и зависност од програмског језика је велика, али се такође могу примењивати и код нижих програмских језика. У случају да је потребно проширити их подршком за нови програмски језик неопходно је имплементирати не само лексички анализатор, већ и одговарајући парсер. Техника се може применити за препознавање свих типова клонова. Ова техника се у имплементацијама показала као техника средње сложености.

Један од алата који користи технику засновану на метрикама је CLAN алат [20]. Овај алат подржава програмске језике C/C++. Обрада коју спроводи овај алат се извршава у четири фазе: препроцесирање директива, парисирање и идентификација фрагмената, екстракција метрика и идентификација клонова. Како је алат намењен језицима C/C++ у првој фази се обавља препроцесирање директива датог језика како би се добио изворни код који се даље може обрађивати. У другој фази се обавља екстракција декларација и дефиниција самих функција користећи наменски парсер. У трећих фази се издвајају метрике коју укључују бројање функција, локалних променљивих, дељених и глобалних променљивих, параметара, исказа скокова и петљи. У четвртој фази се обавља идентификација клонова у поступку квантизација метрика коришћењем прагова за елиминацију шума и спектралне анализе резултата.

D. Технике засноване на апстрактним синтаксним стаблима

Технике засноване на апстрактним синтаксним стаблима трансформишу изворни код у структуру стабла које се касније може упоређивати. За поређење стабала, или њихових подстабала, се може користити више различитих метода које укључују хеширање, трансформацију у префиксна стабла, одређивање најдуже заједничке секвенце употребом динамичког програмирања. Ове технике се углавном користе за откривање клонова код виших програмских језика и зависност од програмског језика је велика, али се такође могу примењивати и код нижих програмских језика. У случају да је потребно проширити их подршком за нови програмски језик неопходно је имплементирати одговарајући парсер. Ова техника се углавном може применити за препознавање клонова типа 1, 2 и 3. Ова техника се у имплементацијама показала као техника средње сложености.

Један од алата који користи технику засновану на метрикама је DECKARD алат [21]. Овај алат подржава програмске језике: C, Java, Php. Обрада коју спроводи овај алат се извршава у пет фаза: генерисање парсера, формирање стабла, генерисање вектора, кластеровање вектора и додатна обрада. У првој фази се на основу формалне граматике језика генерише парсер који се користи и даљој обради. У другој фази се на основу генерисаног парсера и улазног изворног кода формира апстрактно синтаксо стабло које ће се даље обрађивати. У

 ISBN 978-86-7466-930-3

трећој фази се обрађује синтаксно стабло како би се формирали вектори карактеристика фиксне дужине. У овој фази се додатно може обавити и спајање делова који имају заједничко подстабло. У четвртој фази се обавља груписање вектора у кластере користећи еуклидско растојање са циљем одређивања клонова. На крају се као додатна, пета, фаза обавља додатна обрада како би се користећи одређене хеуристике елиминисали погрешно идентификовани клонови.

E. Технике засноване на коришћењу графа зависности

Технике засноване на коришћењу графа зависности које постоје у графу тока контроле и у графу тока података. Графовска репрезентација изворног кода се дели на мање делове у зависности од понашања фрагмента изворног кода. За деобу и поређење се може користити неки од постојећих алгоритама за рад са подстринговима и откривање сличности. Ове технике се углавном користе за откривање клонова код виших програмских језика и зависност од програмског језика је велика. У случају да је потребно проширити их подршком за нови програмски језик неопходно је имплементирати одговарајући парсер. Ова техника се, као и рад са метрикама, може применити за препознавање свих типова клонова. Ова техника се у имплементацијама показала као техника велике сложености.

Један од алата који користи технику засновану на коришћењу графа зависности је PDG-DUP алат [22]. Овај алат подржава програмске језике C/C++. Обрада коју спроводи овај алат се извршава у четири фаза: формирање графа зависности, проналажење парова клонова, уклањање обухваћених клонова и груписање клонова у веће групе. У првој фази се формира граф зависности код кога чворови представљају наредбе програма и предикате, а ивице представљају зависности података и контроле. Друга фаза се обавља у неколико корака. На почетку се обавља деоба свих чворова графа зависности у класе еквиваленције на основу синтаксне структуре исказа и предиката који ти чворови представљају, занемарујући имена променљивих и литерале. У наредном кораку се обавља најбитнији део, срж, алгоритма који проналази изоморфне подграфове коришћењем технике одсецања уназад (*backward slicing*) која датом програмском сегменту додаје претходни повезани део. Поред повратног одсецања користи се и техника одсецања унапред која додаје наредни повезани део. У трећој фази се обавља уклањање свих парова клонова који су већ садржани у неким другим паровима клонова идентификованим у претходном кораку. У четвртој фази се обавља груписање идентификованих парова у веће групе користећи особину транзитивности.

IV. Доприноси утврђивању сличности кода

Откривање неовлашћеног коришћења софтверске библиотеке је проблем детекције клонова који у случају комерцијалних производа има додатну сложеност због чињенице да је код доступан само у бинарном облику. Циљ аутора је био да предложи приступ за процену нивоа сличности између процедура који потичу из различитих бинарних кодова. Основна претпоставка у истраживањима је била да клонови у бинарним кодовима потичу од употребе заједничке софтверске библиотеке, односно истог изворног кода, која се може преводити користећи различите алате. За потребе истраживања бинарни код је дисасемблиран, а за потребе коришћења других алата и декомпајлиран. Детаљан преглед различитих приступа у препознавању сличности бинарног кода дат је у прегледном раду [17] док ће у овој глави бити описана истраживања аутора овога рада спроведена стремећи ка наведеном циљу у области утврђивања сличности бинарног кода. За свако од спроведених истраживања биће наведен кратак опис истраживања, кораци предложеног приступа а потом дати и остварени резултати. Преглед основних карактеристика појединих истраживања је дат у табели 3.

A. Утврђивање сличности бинарног кода

Прво истраживање је имало за циљ да утврди применљивост постојећих техника и алата описаних у претходним поглављима за утврђивање сличности бинарног кода [6]. Имајући у виду да су постојеће технике и алати превасходно намењени раду са вишим програмским језицима у овом истраживању су нека одабрана решења прилагођења раду са бинарним кодом ради евалуације. У истраживању је такође предложен нови приступ који је заснован на метрикама.

Предложени приступ за процену нивоа сличности између две процедуре које се пореде је процес који се обавља у четири фазе: издвајање метрика, упоређивање метрика, трансформације метрика, израчунавање нивоа сличности. У првој фази се спроводи статичка анализа процедура и издвајање метрика тако да се за сваку процедуру формира вектор метрика, при чему елементи тог вектора могу бити скаларне или векторске вредности. Ове метрике укључују одређивање броја и учестаности свих инструкција, појединих типова инструкција (аритметичких, логичких и трансфер података), скокова (условних и безусловних скокова, позива процедура и петљи) и које се представљају скаларним вредностима. Поред скаларних вредности метрика коришћене су и нескаларне, векторске, метрике која укључују бројање сваке појединачне инструкције, сваке адресе доскока, и сваке адресе позива процедуре. У другој фази се обавља поређење сваке о издвојених метрика користећи одговарајући функције за поређење. У трећој фази се обавља трансформација и нормализација добијених резултата поређења на основу претходног знања. У последњој, четвртој, фази комбинују се одабране метрике користећи одговарајућу формулу за формирање једне вредности која представља сличност између две процедуре.

Резултати истраживања су дали одговоре на три истраживачка питања и довела до следећих закључака. Прво питање се односило на разматрање додавања нове

 ISBN 978-86-7466-930-3

ТАБЕЛА 3. ПРЕГЛЕД РЕЗУЛТАТА ИСТРАЖИВАЊА

Приступ	Метрике	Архитектура	Мера сличности	Одзив
1.	Скаларна(13) векторска(6)	ARM	Формуле (7)	35,8%-73,1%
2.	Скаларна(3) векторска(1)	x86	Тежинска сума	37,6%-63.0%
3.	Скаларна(13) векторска(6)	ARM	Формуле (7)	52,7%-81,8%
4.	Векторска(1)	ARM	Неуралнa мрежа	49,2%-82,8%
5.	Векторска(1)	ARM	Левенштајнова удаљеност	57,9%-88,9%

метрике у скуп метрика предложних у постојећим решењима и утврђивање доприноса приликом рангирања процедура. Резултати спроведеног експеримента показују да увођење нових метрика доприноси рангирању процедура за све посматране позиције приликом рангирања. Штавише, са новим метрикама предложени приступ је постигао 1,44 пута бољи одзив (*recall*) за прву посматрану позицију у поређењу са случајем када се користе само постојеће метрике. Друго питање се односило на проверу да ли рангирање процедура обављено предложеним приступом зависи од преводиоца, нивоа оптимизације програма и контекста проблема. Према резултатима експеримента утврђено је да постоји значајан утицај изабраног преводиоца на рангирање процедура који је остварен у предложеном приступу. Међутим, контекст проблема и опције превођења имају мањи утицај на резултате. Треће питање се односило на проверу да ли предложени приступ постиже боље резултате од постојећих алата у смислу прецизности, одзива и F2 мере. Одговарајући експеримент је показао да предложени приступ постиже други најбољи резултат у смислу прецизности и најбољи резултат у смислу одзива. У случају посматрања F2 мере, предложени приступ постиже резултате боље од осталих алата када се посматра до првих шест позиција, док се максимална вредност постиже када се посматрају само прве две позиције при рангирању.

B. Утицај архитектуре рачунара на утврђивање сличности бинарног кода

Друго истраживање је имало за циљ да утврди применљивост приступа предложеног у првом истраживању на другу архитектуру [23]. С обзиром да је у првом истраживању коришћена ARM архитектура, за потребе другог истраживања је коришћена x86 архитектура. Имајући у виду разлике у архитектури било је потребно поновно креирање скупа података на коме ће се обавити тестирање. Овом приликом је коришћен сличан скуп програмских преводилаца, али не потпуно исти, јер ниси сви преводиоци подржавали обе архитектуре.

У оквиру другог истраживања је предложен нови приступ који представља подскуп приступа предложеног у првом истраживању. У овом новом приступу број различитих корака који су спровођени током испитивања у појединим фазама је редукован, и такође је редукован број фаза. Предложени приступ уместо четири фазе има само три фазе. У првој фази је смањен број издвојених метрика, тако да су коришћене метрике које одређују дужину процедуре, број скокова, број позиваних процедура и броје инструкција. Иако је смањен број метрика сачувано је постојање и скаларних и векторских метрика. У другој фази се обавља поређење сваке о издвојених метрика користећи одговарајући функције за поређење, на истоветан начин као и првом истраживању. Трећа фаза се може посматрати као обједињење треће и четврте фазе из првог истраживања и у њој се израчунава сличности користећи хармонијску средину, тежинску суму и наивни Бајес.

Резултати за x86 архитектуру показују да је најбољи одзив постигнут користећи приступ заснованим на тежинској суми. Одзив се креће од 37% до 63% када се посматра прва односно првих десет позиција. Поређење резултата остварених на x86 и ARM архитектурама показује да су постигнућа добијена за првих десет позиција на обе архитектуре упоредива.

Ово показује да се технике поређења бинарних кодова у извесној мери могу користити за поређење кодова који су преведени за исту архитектуру рачунара. Истраживања која су спровели други истраживачи након овог истраживања, почев од 2016. године а која се баве поређењем сличностима бинарног кода између архитектура рачунара [17], су показала да се технике за поређење могу примењивати не само на једној архитектури рачунара, већ се могу примењивати и на кодове код којих су једни преведени за једну архитектуру рачунара а други за други архитектуру рачунара. Овиме се проширује оно што је другим истраживањем утврђено а то је да се технике поређења бинарних кодова у извесној мери могу посматрати независно од архитектуре рачунара.

C. Утицај трансформација преводиоца на утврђивање сличности бинарног кода

Као што је претходно описано, резултати првог истраживања су показали да постоји значајан утицај изабраног преводиоца на рангирање процедура. Из тог разлога треће истраживање је имало за циљ да детаљније испита утицај трансформација програмског преводиоца на утврђивање сличности бинарног кода [24]. У истраживању је такође предложен и нови приступ који дефинише нове технике за смањење утицаја процеса превођења на перформансе поступка утврђивања сличности бинарног кода.

У односу на приступ предложен у првом истраживању у овом приступу је у првој фази, у којој се спроводи

 ISBN 978-86-7466-930-3

статичка анализа, додато пет техника које се спроводе као независни кораци. Прва техника предлаже игнорисање инструкција за рад са стеком. Ова техника је уведена је уведена зато што је приликом анализе уочено да различити преводиоци на различите начине интерагују са стеком и да инструкције које том приликом користе не носе информације о семантици кода. Друга техника предлаже игнорисање инструкција за трансфер података. Ова техника је уведена из сличног разлога као и прва техника јер различити преводиоци користе различите приступе приликом алокације регистара и трансфера између њих или меморијске локације на којој се неки податак налази. Трећа техника предлаже измену неких од разматраних векторских метрика. Овде се уместо посматрања статистике о појединим инструкцијама посматра статистика о секвенцама инструкција одређене дужине. Четврта техника предлаже уграђивање позваних процедура у оквиру позивајуће процедуре. Ова техника је уведена како би се елиминисала разлика између бинарних кодова која може настати усред одлуке преводиоца да обави аутоматско уграђивање у време превођења. Пета техника предлаже увођење прага сличности између процедура. Ова техника је уведена како би смањила број лажно позитивно идентификованих клонова, с обзиром да преводиоци могу различиту логику да имплементирају користећи сличне инструкције.

Резултати истраживања су дали одговоре на три истраживачка питања и довела до следећих закључака. Прво питање се односило на одређивање постигнућа постојећих решења зависе од тога да ли се користи произвољни преводилац и произвољни ниво оптимизације. Према резултатима експеримента утврђено је да постоји значајан утицај изабраног преводиоца на рангирање процедура, и за разлику од првог истраживања овом приликом су тачно измерени утицаји који имају промена преводиоца и промена нивоа оптимизације. Друго питање се односило на проверу да ли додавање сваке од предложених техника једне по једне основном приступу повећава постигнуће ако се упоређене процедуре преводе са произвољним преводиоцима и нивоима оптимизације. Према резултатима експеримента утврђено је да све технике, изузев друге технике које филтрира инструкције трансфера података, доприносе побољшању резултата утврђивања сличности. Треће питање се односило на проверу да ли додавање свих предложених техника заједно основном приступу повећава постигнуће и да ли резултира синергистичким ефектима ако се упоређене процедуре преводе са произвољним преводиоцима и нивоима оптимизације. Резултати експеримента су показали да додавање свих техника има синергистички ефекат који доноси 6,7% до 9,3% повећања одзива за првих пет посматраних позиција приликом рангирања.

D. Коришћење неуралних мрежа за утврђивање сличности бинарног кода

Четврто истраживање је имало за циљ да испита могућност употребе неуралних мрежа за утврђивање сличности бинарног кода [25]. За разлику од приступа описаном у претходним истраживањима који се заснивају на посматрању метрика у овом истраживању су посматране само инструкције. За потребе евалуације је искоришћен исти скуп података који је коришћен у првом и трећем истраживању.

Предложени приступ обавља поређење процедура у пет фаза: издвајање инструкција, кодирање инструкција, коришћење једног првог слоја неуралне мреже, коришћење другог слоја неураллне мреже, поређење. У првој фази се спроводи издвајање токена који одговарају појединим инструкцијама. У другој фази се обавља кодирање инструкција у целобројну вредност. У поступку кодирања су примењене три технике: лексикографско кодирање, семантичко мапирање и мапирање засновано на класи. У трећој фази се полазећи од секвенце кодираних инструкција формирана секвенца карактеристика исте дужине користећи неуралне мреже. Коришћена је Дуготрајно-краткотрајна меморија LSTM (*Long Short-Term Memory*). У четвртој фази се обавља издвајање одређеног броја карактеристика користећи неуралну мрежу која занемарује одређен број улазних карактеристика DNN (*Dropout Neural Network*). У последњој, петој, фази комбинују се рачунање сличности између две процедуре користећи издвојене карактеристике и поступак агрегације и нормализације.

Резултати истраживања су дали одговоре на три истраживачка питања и довели до следећих закључака. Прво питање се односило на разматрање какве резултате даје предложени поступак у односу на поређење процедура само по њиховој дужини. Резултати спроведеног експеримента показују да предложени приступ даје бољи одзив на свакој посматраној позицији. Друго питање се односило на проверу да ли рангирање процедура обављено предложеним приступом зависи од преводиоца. Према резултатима експеримента утврђено је да постоји неки утицај изабраног преводиоца на рангирање процедура од 0,55% до 1,64% када се посматра одзив. Овај утицај је знатно мањи у односу утицај који је присутан код првог и трећег приступа. Треће питање се односило на проверу да ли предложени приступ постиже боље резултате од постојећих алата у смислу одзива. Одговарајући експеримент је показао да предложени приступ постиже незнатно боље резултате, просечно 1,14%. Овај приступ је дао бољи одзив на вишим позицијама, док у првих 9 позиција даје лошије резултате у односу на трећи предложени приступ.

E. Коришћење секвенци операција за утврђивање сличности бинарног кода

У првом истраживању је примећено да метрика која узима у разматрање секвенцу инструкција највише доприноси успешном одређивању сличности. Из тог разлога пето истраживање је имало за циљ да детаљније испита могућности те метрике и да је учини робуснијом на промене које чини преводилац. Сходно томе

 ISBN 978-86-7466-930-3

предложен је нови приступ заснован на испитивању секвенце кодова операција [26].

Предложени приступ обавља поређење процедура у четири фаза: издвајање инструкција, кодирање инструкција, одређивање Левенштајнове удаљености, рачунање релативне Левенштајнове удаљености. У првој фази се спроводи издвајање токена који одговарају појединим инструкцијама. У другој фази се обавља кодирање инструкција у целобројну вредност на истоветан начин ономе описаном у четвртом приступу. У трећој фази се обавља рачунање Левенштајнове удаљености користећи *Needleman–Wunsch* алгоритам. У последњој, четвртој, фази се коришћењем релативне Левенштајнове удаљености добија сличности између две процедуре.

Ово истраживање је требало да да одговор на једно истраживачко питање о томе какве резултате даје предложени приступ у односу на остале приступе које су аутори развили за ARM архитектуру. Резултати спроведеног експеримента показују да предложени приступ даје бољи одзив на свакој посматраној позицији од свих претходно разматраних приступа. У односу на приступ из четвртог истраживања даје од 3,59% до 9,45% бољи одзив, као и од 0,17% до 4,73% бољу прецизност када се посматрају првих 20 позиција. У односу на приступ из трећег истраживања даје од 3,45% до 9,53% бољи одзив, као и од 0,45% до 1,72% бољу прецизност када се посматрају првих 20 позиција.

V. Закључак

Утврђивање сличности кода представља област истраживања са дугом историјом, доста запажених резултата али и перспективом за даљи развој у будућности. Овај рад је имао за циљ да ову област приближи истраживачима и да да осврт на истраживања која су аутори спровели. У раду је најпре објашњен значај и домени у којима се резултати истраживања примењују. Након тога је дат сажет преглед најважнијих техника у овој области. На крају су изложена истраживања и резултати до којих су аутори дошли у претходним годинама.

Захвалница

Рад на овом пројекту је делимично био финансиран од стране Министарства просвете, науке и технолошког развоја Републике Србије (2022/200103), као и Фонда за науку Републике Србије (АВАНТЕС).

Референце

[1] L. Li, T. Bissyandé, M. Papadakis, S. Rasthofer, A. Bartel, D. Octeau, J. Klein, Y. Le Traon, “Static analysis of android apps: A systematic literature review,” *Information and Software Technology*, vol. 88, pp. 67-95, Aug. 2017, 10.1016/j.infsof.2017.04.001.

[2] M. Misic, Z. Sustran, and J. Protic, “A comparison of software tools for plagiarism detection in programming assignments,” *International Journal of Engineering Education*, vol. 32, no. 2, pp. 738-748, 2016.

[3] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, and M. di Penta, “How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines,” in IEEE International Working Conference on Mining Software Repositories, 2017, doi: 10.1109/MSR.2017.2.

[4] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, and P. G. Bringas, “Idea: Opcode-Sequence-Based Malware Detection,” In: F. Massacci, D. Wallach, N. Zannone, (eds) Engineering Secure Software and Systems. ESSoS 2010. Lecture Notes in Computer Science, vol 5965. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-11747-3_3.

[5] T. Dullien and R. Rolles, “Graph-based comparison of executable objects (english version),” Sstic, 2005, doi: 10.1.1.96.5076.

[6] S. Stojanović, Z. Radivojević, and M. Cvetanović, “Approach for estimating similarity between procedures in differently compiled binaries,” *Information and Software Technology*, vol. 58, pp. 259-271, 2015, doi: 10.1016/j.infsof.2014.06.012.

[7] C. K. Roy, J. R. Cordy, R. Koschke, “Comparison and evaluation of code clone detection techniques and tools: a qualitative approach,” *Sci. Comput. Program*, vol. 74, pp. 470–495, 2009.

[8] Davis, I.J. and Godfrey, M.W, “From Whence it Came: Detecting Source Code Clones by Analyzing Assembler,” Proc. WCRE 10, Beverly, MA, October 13–16, pp. 242–246. IEEE, Los Alamitos, 2010.

[9] P.E.Chaudhry, and A. Zimmerman “Protecting Your Intellectual Property Rights: Understanding the Role of Management, Governments, Consumers and Pirates,” Springer, New York, 2012.

[10] A. Hemel, K.T. Kalleberg, R. Vermaas, and E. Dolstra, “Finding Software License Violations through Binary Code Clone Detection,” Proc. MSR 11, Honolulu, HI, May 21–22, pp. 63–72. ACM, New York, 2011.

[11] S. M. Ghaffarian and H. R. Shahriari. “Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey,” ACM Comput. Surv. 50, 4, Article 56 (July 2018), 36 pages. DOI:https://doi.org/10.1145/3092566

[12] T. Dullien, and R. Rolf, “Graph-based Comparison of Executable Objects (English version),” Proc. SSTIC 05, Rennes, France, June 1–3, pp. 1–13. STIC, Paris, 2005.

[13] D. Rattan, R. Bhatia, and M. Singh, “Software clone detection: a systematic review,” *Inform. Softw. Technol.*, vol. 55, pp.1165–1199, 2013.

[14] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, E. Merlo, “Comparison and evaluation of clone detection tools,” *IEEE Trans. Software Eng*. vol. 33, pp. 577–591, 2007.

[15] C. K. Roy, J. R. Cordy, “A Survey on Software Clone Detection Research,” Technical Report 2007-541, Queen’s University, Canada, 2007.

[16] Q. U. Ain, W. H. Butt, M. W. Anwar, F. Azam and B. Maqbool, “A Systematic Review on Code Clone Detection,” *IEEE Access,* vol. 7, pp. 86121-86144, 2019, doi: 10.1109/ACCESS.2019.2918202.

[17] I. Haq, J. Caballero, “A Survey of Binary Code Similarity,” *ACM Computing Surveys,* vol. 54, no. 3, pp. 1-38, April, 2022.

[18] M. S. Uddin, C.K. Roy, K.A. Schneider, A Hindle, “On the effectiveness of Simhash for detecting near-miss clones in large scale software systems,” in: Proceedings of 18th Working Conference on Reverse Engineering (WCRE 2011), Limerick, Ireland, 2011, pp. 13–22.

[19] T. Kamiya, S. Kusumoto, K. Inoue, “CCFinder: a multilinguistic token-based code clone detection system for large scale source code,” *IEEE Trans. Software Eng*., vol. 28, pp. 654–670, 2002.

[20] E. Merlo, “Detection of plagiarism in university projects using metrics-based spectral similarity,” in: Proceedings of Dagstuhl Seminar 06301: Duplication, Redundancy, and Similarity in Software, Dagstuhl, Germany, 2006, pp. 1–10.

[21] L. Jiang, G. Misherghi, Z. Su, S. Glondu, “DECKARD: scalable and accurate treebased detection of code clones,” in: Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, USA, 2007, pp. 96–105.

[22] R. Komondoor, S. Horwitz, “Using slicing to identify duplication in source code,” in: Proceedings of the 8th International Symposium on Static Analysis (SAS’01), vol. LNCS 2126, Paris, France, 2001, pp. 40–56.

[23] K. Berta, S. Stojanović, M. Cvetanović, Z. Radivojević, “Estimation of Similarity between Functions Extracted from x86 Executable Files,” *Serbian Journal of Electrical Engineering*, vol. 12, no. 2, pp. 253 - 262, Jun, 2015.

 ISBN 978-86-7466-930-3

[24] Z. Radivojević, M. Cvetanović and S. Stojanović, "Comparison of Binary Procedures: A Set of Techniques for Evading Compiler Transformations," *The Computer Journal*, vol. 59, pp. 106–118, 2015.
[25] N. Pejić, M. Cvetanović, Z. Radivojević, "Estimating similarity between differently compiled procedures using neural networks," ТЕЛФОР XXVII, Belgrade, Nov, 2019.
[26] N. Pejić, M. Cvetanović, Z. Radivojević, "Comparing Assembler Procedures by Analyzing Sequences of Opcodes," Telfor Journal, vol. 12, no. 1, pp. 46 - 49, Jul, 2020.

 ISBN 978-86-7466-930-3

Softversko rešenje za akviziciju i vizuelizaciju moždanih talasa

Ivan Tot, Boriša Jovanović, Dušan Bogićević, Tamara Gajić, Jordan Atanasijević

***Apstrakt*—U današnjem društvu veoma značajnu ulogu imaju sistemi za identifikaciju korisnika. Složeni bezbednosni zahtevi naterali su eksperte da istraže načine na koje se biometrijski podaci mogu iskoristiti u utvrđivanju identiteta korisnika. U ovom radu je prikazano softversko rešenje za akviziciju i vizuelizaciju moždanih talasa (EEG) kao biometrijskih podataka.**

***Ključne reči*—biometrija; moždani talasi; akvizicija; vizuelizacija**

I. UVOD

Kontrola pristupa informacionim sistemima predstavlja jedan od najznačajnijih aspekata zaštite, posebno kod sistema koji sadrže bezbednosno osetljive podatke. Ona onemogućava osobama koja nemaju pravo da pristupe nekom sistemu i samim tim zloupotrebe podatke koje se nalaze u njemu.

Da bi se došlo do faze identifikacije korisnika, u smislu zaštite informaciono-komunikacionih sistema, neophodno je da se putem biometrijskih senzora prikupe biometrijski podaci koji će se uporediti sa već postojećim podacima registrovanim u samom sistemu [1] [2] [3]. Spajanje više biometrijskih podataka smanjuje stepen sistemske greške prilikom identifikacije korisnika. Samo prikupljanje podataka sa više biometrijskih senzora stvara kompletniju sliku o korisniku. Metod fuzije biometrijskih podataka uključuje sekvencijalnu obradu biometrijskih modaliteta dok se ne dobije prihvatljivo podudaranje u slučaju identifikacije korisnika [4] [5].

Ovaj rad daje prikaz softverskog rešenja koje omogućava akviziciju i vizuelizaciju moždanih talasa (EEG) koje predstavlja prvi korak u realizaciji rešenja za autentifikaciju korisnika.

Ivan Tot – Vojna akademija, Univerzitet odbrane u Beogradu, Generala Pavla Jurišića Šturma 33, 11000 Beograd, Srbija (e-mail: ivan.tot@va.mod.gov.rs).

Boriša Jovanović – Vojna akademija, Univerzitet odbrane u Beogradu, Generala Pavla Jurišića Šturma 33, 11000 Beograd, Srbija (e-mail: borisa.jovanovic@vs.rs).

Dušan Bogićević – Vojna akademija, Univerzitet odbrane u Beogradu, Generala Pavla Jurišića Šturma 33, 11000 Beograd, Srbija (e-mail: dusan.bogicevic@vs.rs).

Tamara Gajić – Vojna akademija, Univerzitet odbrane u Beogradu, Generala Pavla Jurišića Šturma 33, 11000 Beograd, Srbija (e-mail: tamara.gajic@vs.rs).

Jordan Atanasijević – Vojna akademija, Univerzitet odbrane u Beogradu, Generala Pavla Jurišića Šturma 33, 11000 Beograd, Srbija (e-mail: jordan.atanasijevic@vs.rs).

.

II. PRIMENJENI HARDVERSKI UREĐAJ

Akvizicija predstavlja prikupljanje podataka iz spoljašnje sredine u određeni električni uređaj, to jest senzor. Kada se govori o biometrijskom podatku, onda je potreban biometrijski uređaj za akviziciju takve vrste podataka [6]. Tako prikupljeni podatak može biti upotrebljen za identifikaciju ljudi. Pojedini biometrijski podaci su jedinstveni za svaku osobu i mogu služiti za identifikaciju osoba, kako u civilnom sektoru npr. u zdravstvu, obrazovnim ustanovama, firmama, tako i u vojnim, policijskim i državnim ustanovama u cilju zaštite sopstvenih resursa. Biometrijska autentifikacija (odnosno realna autentifikacija) se koristi u informacionim tehnologijama kao oblik identifikacije korisnika i kontrole pristupa zaštićenim resursima [7].

EEG autentifikacija koristi elektrofiziološki sistem za praćenje aktivnosti mozga. Ova tehnologija je vrlo popularna i može se koristiti bez ikakvih sporednih efekata na mozak. Do sada je izvršeno nekoliko istraživanja o mogućnosti primene EEG signala za autentifikaciju korisnika [8] [9].

Postoji nekoliko komercijalnih uređaja sa različitim brojem elektroda koje se koriste za prikupljanje EEG podataka. Neki od senzora koriste suve elektrode, a neki senzori koriste mokre elektrode. Moždani režnjevi emituju EEG signale kao odgovor na različite stimulanse i mentalna stanja. Pretpostavlja se da postoji promenljiva razlika uzoraka EEG talasa dok se vizuelizuje lozinku u mirnim uslovima u odnosu na prinudu, zbog različitih mentalnih stanja, mozak proizvodi različite obrasce analognog EEG talasa [10]. Na slici 1 prikazan je uređaj koji je korišćen u ovom radu.

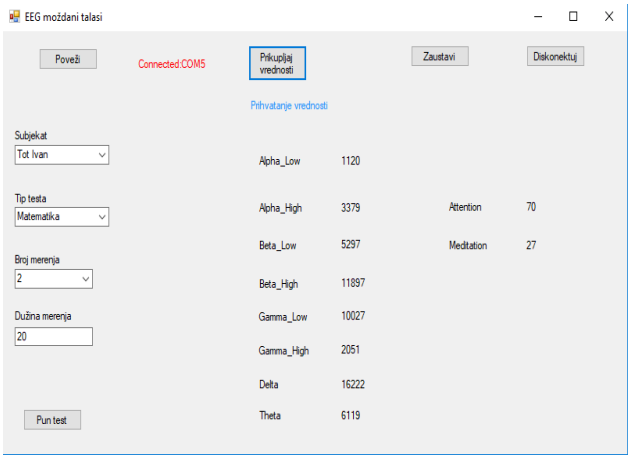
Sl. 1. Neuro Sky Mind Wave 2-EEG biometrijski uređaj [11]

 ISBN 978-86-7466-930-3

Uređaj prikazan na slici 1 vrši merenje alfa, beta, gama, delta i teta moždanih talasa kao i trenutno stanje usredsređenosti – fokusa (attention) i opuštenosti (meditation) subjekta čiji se moždani talasi mere [12].

III. PRIKAZ SOFTVERSKOG REŠENJA

Izgled ekrana softverskog rešenja za akviziciju podataka sa senzora prikazan je na slici 2. Rešenje je razvijeno u alatu Microsoft Visual Studio 2017 korišćenjem programskog jezika C# i biblioteke ThinkGear.dll proizvođača uređaja. Sama biblioteka nudi funkcije za ostvarivanje komunikacije sa uređajem. Dalji rad je podrazumevao prihvatanje vrednosti koje dolaze sa uređaja u određenim vremenskim intervalima definisanim i samom programskom kodu softverskog rešenja. Radi usrednjavanja vrednosti korišćena je logaritamska funkcija. Tako izračunate vrednosti su čuvane u realizovanoj bazi podataka.



Sl. 2. Izgled ekrana za akviziciju podataka

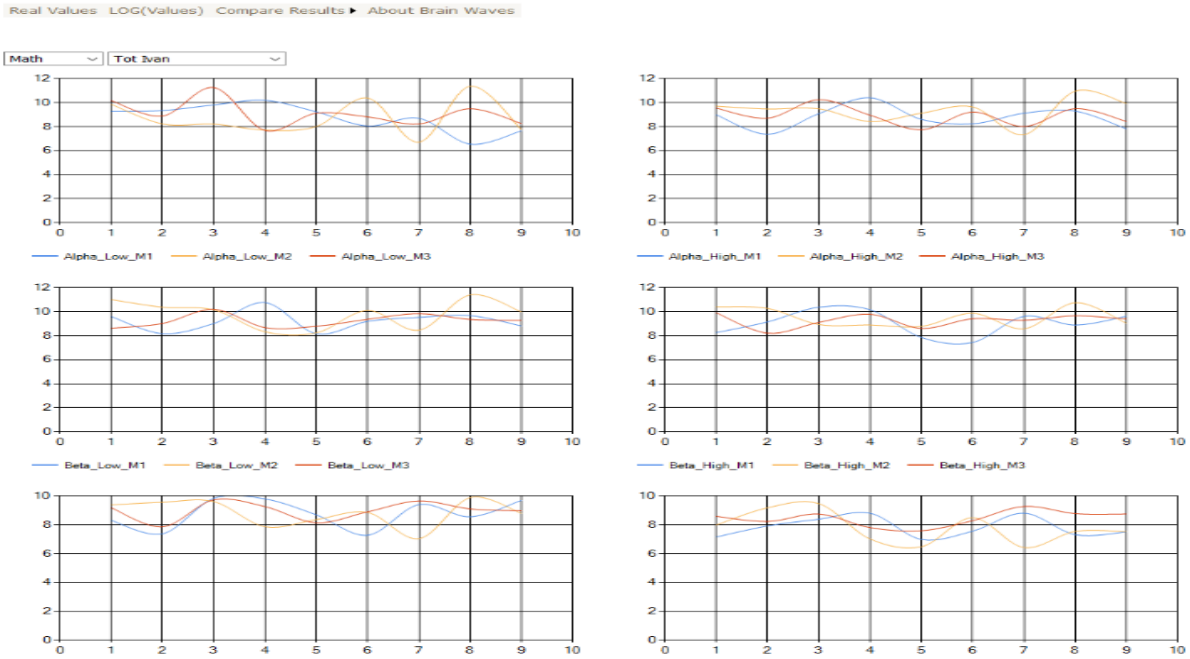
Klikom na komandno dugme Poveži vrši se uspostavljanje bluetooth konekcije sa senzorom. Nakon uspešno ostvarene konekcije, klikom na dugme Prikupljaj vrednosti aplikacija će prihvatati vrednosti sa uređaja i prikazivati ih na ekranu. Komandno dugme Zaustavi privremeno prekida prihvatanje vrednosti sa uređaja, a komandno dugme Diskonektuj prekida konekciju sa uređajem.

Prihvaćene vrednosti mogu se čuvati u bazi podataka realizovanoj u Microsoft SQL Server Express 2014 klikom na komandno dugme Pun test svake 2 sekunde. U tom slučaju neophodno je izabrati subjekta čiji se moždani talasi mere, tip testa, broj merenja kao i zadati dužinu svakog testa u sekundama. Tip testa može biti:

1. *Opušteno* – subjekat se potpuno opusti i zatvori oči,
2. *Čitanje* – subjekat dobija tekst koji treba da čita u sebi,
3. *Lepe slike* – subjekat posmatra slike koje u njemu bude lepe emocije,
4. *Matematika* – subjekat dobija matematički zadatak koji treba da reši,
5. *Uznemiravajuće slike* – subjekat posmatra slike koje treba da izazovu ružne emocije (slike ratnih zločina, iskasapljenih životinja i slično).

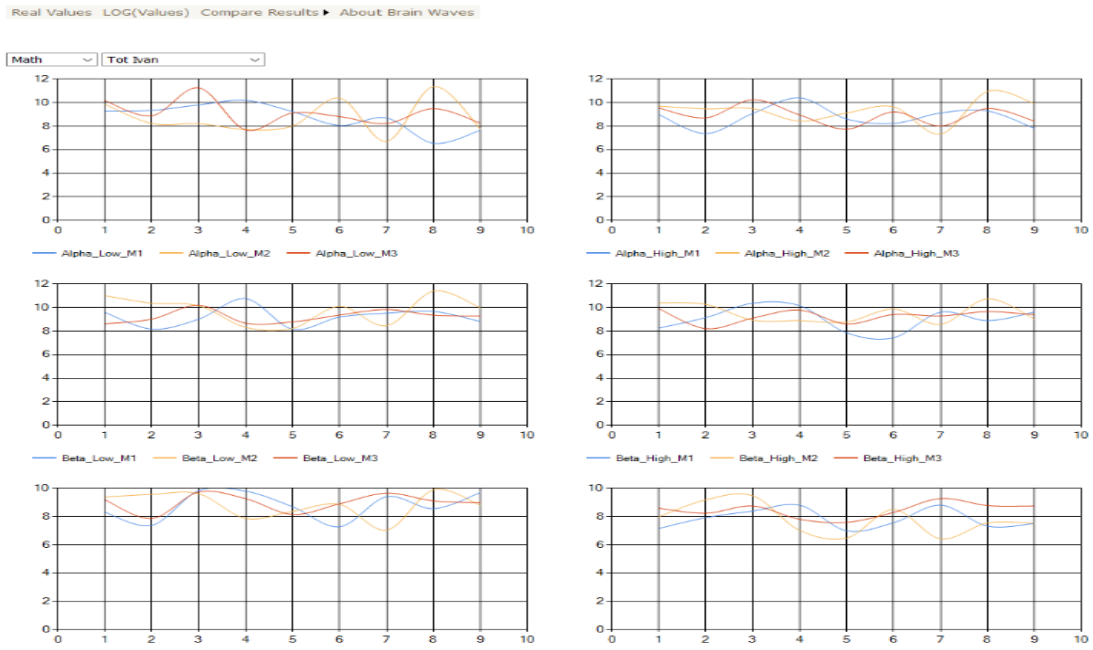
U ovoj fazi istraživanja predviđeno je da se radi do 5 merenja svih navedenih testova u različite dane. Nad uzorkom od 60 testiranih lica (različitih polova, godina i nivoa obrazovanja) uočeno je da postoje određena poklapanja moždanih talasa naročito prilikom primene testa „Čitanje“. Takođe, uočeno je da prilikom primene testa „Uznemiravajuće slike“ značajan faktor imaju godine testiranih lica. Na mlađe osobe slike ratnih zločina slabije izazivaju ružne emocije, ali zato značajno reaguju na slike iskasapljenih životinja. U slučaju starijih osoba, reakcije su suprotne.

Za vizuelizaciju podataka razvijena ja web aplikacija u alatu Microsoft Visual Studio 2017 korišćenjem ASP.NET tehnologije. Na slikama 3 i 4 prikazane su delovi web stranica realizovane aplikacije koje omogućavaju poređenje dobijenih podataka.



Sl. 3. Web stranica za vizuelizaciju podataka za izvršena tri merenja

 ISBN 978-86-7466-930-3



Sl. 4. Poređenje moždanih talasa

Web stranica prikazana na slici 3 omogućava vizuelizaciju i poređenje podataka dobijenih iz uređaja za izabranog subjekta i izabrani tip testa. Za svako merenje se po kolonama prikazuje zbirni grafikon sa svim talasima (na x-osi su odbirci vremena, a na y-osi logaritamske vrednosti izmerenih talasa), a zatim i pojedinačni grafikon za svaki izmereni moždani talas.

Web stranica prikazana na slici 4 takođe omogućava vizuelizaciju i poređenje podataka dobijenih iz uređaja za izabranog subjekta i izabrani tip testa, ali po pojedinačnim talasima. Svaki grafikon prikazuje rezultate svih izvršenih merenja po vrstama talasa (na x-osi su odbirci vremena, a na y-osi logaritamske vrednosti izmerenih talasa).

IV. Zaključak

Bezbednost informacionih sistem je jedno od najaktuelnijih pitanja današnjice. Kontrola pristupa je pogotovo bitna jer služi da spreči korisnike koji nemaju prava da pristupe i koriste sistem. Pristup nepoželjnih korisnika je veoma opasan skoro kod svih sistema zbog postojanja velike mogućnosti zloupotrebe samog sistema i informacija koje se nalaze u njemu.

Zbog značaja utvrđivanja identiteta potrebno je stalno raditi na usavršavanju sistema za preciznu identifikaciju, odnosno na poboljšavanju njihovih performansi, bilo kroz razvoj biometrijskih uređaja, bilo kroz unapređenje metoda akvizicije biometrijskih podataka. Jedan od potencijalnih načina je i primena moždanih talasa za autentifikaciju korisnika.

Do sada su po ovom projektu razvijene dve aplikacije. Jedna za akviziciju podataka sa senzora i druga za vizuelizaciju podataka.

U daljem istraživanju planirana je detaljna analiza dobijenih podatka primenom MATLAB softverskog paketa i statističkih alata da bi se utvrdilo postojanje korelacije moždanih talasa na odgovarajući tip testa kao i testiranje značajno većeg broja lica kako bi rezultati planirane analize bili tačniji. Ukoliko bi se potvrdilo postojanje korelacije, onda bi se moždani talasi mogli koristiti pouzdano za autentifikaciju korisnika.

Zahvalnica

Rad je nastao kao rezultat rada na naučno-istraživačkom projektu “Upravljanje pristupom zaštićenim resursima računarskih mreža u Ministarstvu odbrane i Vojsci Srbije na osnovu multimodalne identifikacije korisnika”, pod brojem VATT/3/18-20, od 2018. do 2020. godine na Vojnoj akademiji Univerziteta odbrane u Beogradu.

Literatura

[1] I. Jayarathene, M. Cohen, S. Amarakeerthi, “BrainID: Development of an EEG-Based Biometric Authentication System“, 2016.

[2] K. Lalović, I. Tot, A. Arsić, M., “Security Information System, Based on Fingerprint Biometrics”, Acta Polytechnica Hungarica, Volume 16, Issue Number 5, 2019.

[3] N. Maček, B. Đorđević, J. Gavrilović, K. Lalović, “An Approach to Robust Biometric Key Generation System Design”, Acta Polytechnica Hungarica, Volume 12, Issue Number 8, 2015.

[4] K. Lalović, M. Milosavljević, I. Tot, N. Maček, “Device for Biometric Verification of Maternity”, Serbian Journal of Electrical Engineering-Vol. 12, No. 3, 2015.

[5] A. K. Jain, A. A. Ross, K. Nandakumar, “Introduction to Biometrics”, Springer, 2011.

[6] J. Ashbourn, “Biometrics: Advanced Identity Verification”, Springer, 2014.

[7] L. Feng, „Brain password: A secure and truly cancelable brain biometrics for smart headwear“, International conference on Mobile Systems, Applications and Services, ACM, 2018.

[8] S. Soni, S. B. Somani, V. V. Shete, „Biometric user authentication using brain waves“, International Conference on Inventive Computation Technologies (ICICT), India, 2016.

 ISBN 978-86-7466-930-3

[9] A. A. Alariki, A. W. Ibrahimi, M. Wardak, J. Wall, „A Review Study of Brian Activity-Based Biometric Authentication“, Journal of Computer Science, 2018.
[10] https://www.mdpi.com/journal/biosensors/special_issues/wearable_biosensors
[11] https://cdn.sparkfun.com//assets/parts/1/2/9/9/4/14758-NeuroSky_MindWave_Mobile2-04.jpg
[12] M. McDowell, “Brainwaves: The Nature Of Brain Waves & Their Frequencies”, 2015.

ABSTRACT

Nowadays, user identification systems play a very important role in modern society. Complex security requirements have led experts to explore ways in which biometric data can be used to identify user identities. This paper presents a software solution for acquisition and visualization of brain waves (EEG) as biometric data.

Software Solution for Acquisition and Visualisation of Brain Waves

Ivan Tot, Boriša Jovanović, Dušan Bogićević, Tamara Gajić, Jordan Atanasijević

 ISBN 978-86-7466-930-3

Platforma za praćenje kvaliteta vazduha u gradu Čačak

Nikola Kukrić, Božidar Popović, Slobodan Lubura, Zorana Mandić

Elektrotehnički fakultet, Univerzitet u Istočnom Sarajevu, Istočno Sarajevo, Bosna i Hercegovina

***Apstrakt*—Izloženost PM2.5 česticama izdvaja se kao vodeći zdravstveni problem na globalnom nivou, a problem zagađenosti vazduha predstavlja jedan od glavnih uzroka smrtnosti na svijetu. Kako ovo predstavlja problem današnjice, zahtjevi da se mjerenje kvaliteta vazduha vrši što češće na što više lokacija, doveli su do razvoja niskobudžetnih senzora. Uz pomoć široko dostupnih senzora grade se senzorski čvorovi, koji pribavljaju podatke o trenutnim koncentracijama suspendovanih čestica, a ovi podaci kroz kasniju analizu dovede do kasne informacije o kvalitetu vazduha preko AQI - indeksa zagađenosti vazduha. U ovom radu osim razvoja senzorskih čvorova, predstavljena je i realizacija jedinstvene web platforme za prikazivanje, arhiviranje i analizu kvaliteta vazduha, čime je realizovan jedinstven i potpun sistem za prikupljanje i obradu podataka o kvalitetu vazduha. Iako je platforma realizovana za područje grada Čačka, gdje je izvršena integracija postojećih senzorskih čvorova sa novo realizovanim čvorovima, može biti prilagođena i drugim geografskim područjima te je široko upotrebljiva.**

***Ključne riječi*—kvalitet vazduha, suspendovane čestice, PM2.5, PM10**

I. Uvod

Kvalitet vazduha je jedan od vodećih problema u mnogim gradovima i ima direktan uticaj na kvalitet života. Izloženost suspendovanim česticama PM10 i PM2.5 (eng. *Particular Matter*) predstavlja jedan od vodećih uzroka smrtnosti u svijetu i prema podacima Svjetska zdravstvena organizacija (eng. *World Health Organization* - WHO) na svjetskom nivou tokom godine dovede do između 4,2 i 8,9 miliona smrtnih slučajeva [1]. Suspendovane čestice su sastavni dio prašine i manje su od 10μm (PM10), odnosno 2.5μm (PM2.5) te predstavljaju smješu čađi, dima, kiseline, uz prisustvo teških metala, a izloženost PM2.5 česticama je prepoznat kao glavni globalni zdravstveni problem [2]. Podjela na osnovu izvora iz kojih potiču suspendovane čestice data je u Tabeli I, što predstavlja glavni kriterijum pri izboru lokacije senzorskih čvorova koji prate njihovu koncentraciju.

TABELA I
IZVORI PM2.5 ČESTICA [3]

Izvor	Procentualno (%)
Saobraćaj	25%
Neodređenog ljudskog porijekla	22%
Ogrjev u domaćinstvu	20%
Prirodna prašina i soli	18%
Industrijska djelatnost	15%

Dugoročnim praćenjem koncentracije suspendovanih čestica moguće je detektovati u kojim oblastima gradova je najveća koncentracija čestica te djelovati na izvore zagađenja kako bi se smanjenjem koncentracije čestica unaprijedio kvalitet života.

U gradu Čačku su ranije instalirana tri senzorska čvora za mjerenje zagađenosti vazduha, koji nisu bili povezani u jedinstvenu mrežu, niti su informacije o mjerenjima bile dostupne korisnicima tj. građanima. U pitanju su senzorski čvorovi *Davis AirLink*® kompanije *Davis Instruments* [4]. Navedeni senzorski čvorovi mjere količine suspendovanih čestica PM1.0, PM2.5 i PM10 i u na osnovu njih izračunavaju indeks kvaliteta vazduha (eng. *Air Quality Index* - AQI).

U proteklom periodu, realizacijom projekta prekogranične saradnje „*Transport related Air Pollution and Health impacts in the Čačak city – AIRPOLISCA*“, u decembru 2021. godine [5], ovi senzorski čvorovi su povezani na jedinstven server sa odgovarajućom bazom podataka gdje se čuvaju podaci o izvršenim mjerenjima kvaliteta vazduha, obezbijeđen je domen za Web aplikaciju i kreirana Web aplikacija za prezentaciju podataka. Na taj način, broj građana Čačka koji imaju pristup informacijama o kvalitetu vazduha nije ograničen.

U nastavku aktivnosti vezanih za proširenje mreže senzorskih čvorova za mjerenje kvaliteta vazduha u saradnji sa gradom Čačkom, zaključno sa 08.02.2022. godine postavljeno je osam novih senzorskih čvorova koji će biti predstavljeni u nastavku.

Grad Čačak je pokriven sa senzorskom mrežom od jedanaest senzorskih čvorova za prikupljanje podataka o koncentraciji suspendovanih čestica i pristup podacima u realnom vremenu je omogućen svima preko Web stranice https://cacak.vazduh.net.

U nastavku rada, poglavlje II, predstavljen je problem kvaliteta vazduha te standardi i referentne vrijednosti za ocjenu kvaliteta. Poglavlje III daje detaljnu analizu realizovanog senzorskog čvora, a prezentacija prikupljenih podataka je predstavljena u poglavlju IV.

II. Kvalitet Vazduha

Prema uredbi za monitoring i zahtjevima kvaliteta vazduha [6] kvalitet ambijentalnog vazduha se određuje pomoću koncentracije sumpor dioksida SO_2, azot monoksida NO i azot dioksida NO_2, ozona O_3, ugljen monoksida CO i masene koncentracije PM10 i PM2.5 čestica. Za mjerenje

 ISBN 978-86-7466-930-3

koncentracije navedenih čestica propisane su referentne metode i standardi koju su dati u Tabeli II.

TABELA II [6]
VRSTE MJERENJA I REFERENTNE METODE

Vrsta mjerenja	Metode/Specifikacije
Koncentracija sumpor dioksida SO2	EN 14212:2013 EN 14212/Cor1:2015 Ultraljubičasta fluoroscencija
Koncentracija azot monoksida NO i azot dioksida NO2	EN 14211:2013 Hemiluminiscencija
Koncentracija ozona O3	EN 14625:2013 Ultraljubičasta fotometrija
Koncentracija karbon monoksida CO	EN 14626:2013 Nedisperzivna infracrvena spektroskopija
Masena koncentracija PM10 i PM2.5	EN 12341:2015 Gravimetrijska metoda

Ministarstvo zaštite životne sredine - Agencija za zaštitu životne sredine Republike Srbije je 2015 pokrenula projekat „Objedinjeni prikaz automatskog monitoringa kvaliteta vazduha u Republici Srbiji“ gdje se prate navedeni parametri [7]. Za ocjenu kvaliteta vazduha korišten je indeks kvaliteta vazduha EAQI (eng. *European Air Quality Index*), koji je 2017. godine usvojila Evropska agencije za životnu sredinu (eng. *European Environment Agency*) zajedno sa Upravom za ekologiju Evropske komsije (*European Commission's Directorate General for Environment*). Pomenuti indeks kvaliteta vazduha prikazan je u Tabeli III.

TABELA III
SKALA ZA OCJENU KVALITETA VAZDUHA

Oznaka	Zagađivač (satna koncentracija u µg/m^3)	
	PM10	PM2.5
Odličan	0 – 25	0 – 15
Dobar	25 – 50	15 – 30
Prihvatljiv	50 – 90	30 – 55
Zagađen	90 – 180	55 – 110
Izuzetno zagađen	>180	>110

S obzirom na to da je raspoređivanje i postavljanje akreditovanih stanica sa referentnim metodama za praćenje navedenih parametara vazduha veoma skupo te da najveći problem u gradovima uglavnom predstavljaju PM10 i PM2.5 čestice odlučeno je da se realizuju senzorski čvorovi za mjerenje PM10 i PM2.5 čestica. Za senzore koji prate koncentraciju suspendovanih čestica korišćeni su nisko budžetni (eng. *low ost*) laserski senzori.

Prema standardu EN 12341:2015 PM česticu su „čestice suspendovane u zraku koje su dovoljno male da mogu proći kroz otvor za odabir veličine sa 50% efikasnosti pri aerodinamičkom promjeru x µm.“ [8]. Referentna metoda je gravimetrijska motoda koja podrazumijeva prikupljanje prašine na uzorku u periodu od 24h te određivanje koncentracije u opsegu od 1 – 150 µg/m^3 za PM10 i 1 – 120 µg/m^3 za PM2.5 čestice.

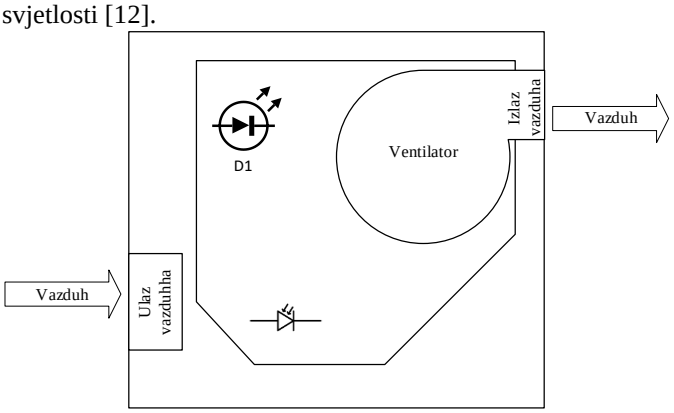
III. SENZORSKI ČVOR

Tokom istraživanje korišćena su tri niskobudžetna laserska senzora i to: SDS011 od proizvođača *Nova Fitness Co*., PMS5003 i PMS7003 od proizvođača *Beijing Plantower Co., Ltd, China*. Pregled karakteristika senzora dat je u Tabeli IV.

TABELA IV
PREGLED KARAKTERISTIKA POJEDINIH NISKOBUDŽETNIH LASERSKIH SENZORA SUSPENDOVANIH ČESTICA [9,10,11]

Karakteristika	Senzori		
	SDS011	PMS5003	PM7003
Dimenzije (mm)	71x70x23	50x38x21	48x37x20
Napajanje (V)	4.7~5.3	4.5~5.5	4.5~5.5
Potrošnja (mA)	70 ± 10	<100	<100
Temperaturni opseg (°C)	-20 ~ +60	-40 ~ +80	-40 ~ +80
Opseg mjerenja čestica (µg/m^3)	0 – 999.9	0 – 500 efktivno, >1000 maks	0 – 500 efktivno, >1000 maks

Navedeni optički senzori mjere intenzitet svjetlosti koja se raspršuje pod uticajem prašine koja se uvlači u senzor. Senzori se sastoje od svjetlosne diode (eng. *Light Emitting Diode* – LED), fotodiode, ventilatora koji uvlači vazduh sa česticama prašine i većeg broja sočiva za fokusiranje snopa svjetlosti [12].



Sl. 1. Struktura optičkog senzora suspendovanih čestica

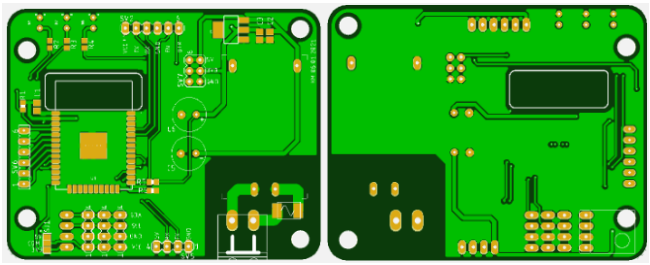
Prilikom realizacije senzorskog kao najbolji omjer kvalitet/cijena odabran je senzor PMS7003.

Senzorski čvor se sastoji od mikrokontrolerske jedinice, senzora i jedinice za napajanje. Mikrokontroler korišćen u realizaciji senzorskog čvora je ESP-WROOM-32, koji

 ISBN 978-86-7466-930-3

pomoću *UART* modula komunicira sa PMS7003 senzorom i pomoću *Wi-Fi* modula prenosi očitane podataka sa senzora brzinom do 150 Mbps, sa snagom signala od 20 dB. Za napajanje je korišćen Hi-Link naponski modul. Navedeni modul ima naponsku konverziju sa 90-245 VAC na 5VDC sa maksimalnom izlaznom snagom od 3W [13]. Napon od 5VDC je potreban za napajanje senzora PMS7003. S obzirom da je za napajanje mikrokontrolera potrebno 3.3V korišćen je naponski regulator AMS1117-3.3V koji za ulazni napon u opsegu 4.8VDC – 6.5VDC daje na izlazu 3.3VDC.

Za crtanje šeme i realizaciju štampane ploče korišćeno je programsko okruženje *Eagle*. Izgled štampane ploče senzorskog čvora sa prednje i zadnje strane prikazan je na Sl. 2.



Sl. 2. Izgled štampane ploče senzorskog čvora

Tokom realizacije senzorskog čvora realizovano je kućište u kojem se smještaju pomenute elektronske komponente, koja treba da omogući protok vazduha prema senzoru PMS7003 te da elektronske komponente budu zaštićene od vremenskih uslova. Izgled senzorskog čvora i sastavljene ploče sa senzorom PM10, PM2.5 prikazani su na Sl. 3.

Sl. 3. Izgled senzorskog čvora i sastavljene ploče

Tokom instalacije senzorskog čvora, potrebno je senzoru proslijediti parametre za pristupnu *Wi-Fi* tačku preko kojeg će senzor slati očitane podatke. Za te potrebe kreirana je Android aplikacija *VazduhNet* koja omogućava jednostavno prosljeđivanje pristupnih podataka i praćenje očitanja sa svih dodatih senzora. Prilikom dodavanja uređaja svakom se dodjeljuje jedinstveni *Token* koji omogućava autorizaciju prilikom slanja podataka, a na osnovu *MAC* adrese i lokacije uređaja radi se autentifikacija senzorskog čvora.

IV. PREZENTACIJA SENZORSKIH OČITANJA

A. Za administraciju senzorskih čvorova i prezentaciju očitanih vrijednosti kreirana je Web aplikacija koju možemo podijeli na administratorsku i klijentsku.

Administratorska aplikacije je kreirana pomoću *Yii2* okruženja, a njene osnovne funkcionalnosti su: administracija senzorskih čvorova (dodavanje, brisanje, izmjene), prijem podataka od senzorskih čvorova, obrada podataka, tabelarni prikaz svih očitanja i generisanje izvještaja koji sadrže očitane podatke. *Yii2*, čiji naziv potiče od kineske riječi *Yii* - "jednostavan i evolucijski", je PHP okvir visokih performansi, baziran na komponentama za brzi razvoj modernih web aplikacija [14]. Komunikacija između senzorskih čvorova i web aplikacije se odvija preko programskog interfejsa aplikacije (eng. *Application Programming Interface* – API). Pristup administratorskoj aplikaciji je dozvoljen samo autentifikovanim korisnicima.

B. Klijentska aplikacija

Klijentska aplikacija je dostupna svim korisnicima i omogućava korisnicima da prate zagađenost vazduha. Prilikom otvaranja Web stranice prvo se prikazuje poruka o trenutnom kvalitetu vazduha na području grada i odgovarajući „emitikon“ u zavisnosti od kvaliteta vazduha. Prikaz kvaliteta u slikovnoj formi sa emotikonima omogućava korisnicima jednostavnije i brže dobijanje informacija. Takođe uz emotikone stoje i preporuke građanima i informacije o uticaju kvaliteta vazduha na zdravlje stanovništva.

Sl. 4. Emotikoni i preporuke građanima u zavisnosti od kvaliteta vazduha

Na stranici je omogućeno mapiranje senzora pomoću biblioteke *Leaflet*. *Leaflet* je vodeća JavaScript biblioteka otvorenog koda za interaktivne mape prilagođene mobilnim uređajima. Težak je samo oko *32KB*, i ima sve funkcije za rad sa mapama koje će većini programera biti potrebne, a ako postoji potreba za dodatnim stvarima tu su i razni dodaci i efekti [15]. *Leaflet* je dizajniran za jednostavnu upotrebu, brzinu i kvalitet. Radi lako i efikasno na gotovo svim desktop i mobilnim platformama, postoji i mogućnost ekstenzija sa dosta dodataka. Podaci o mapama koje se koriste su dobijeni od strane *OpenStreetMap* platforme. Svaki senzor na mapi je obojen odgovarajućom bojom u zavisnosti od trenutne zagađenosti. Klikom na jedan od senzora na mapi otvara se novi prozor na kojem se prikazuju detalji kvaliteta vazduha odnosno trenutni kvalitet, tačne koncentracije PM10 i PM2.5 čestica i vrijeme posljednjeg mjerenja, dok klikom na naziv senzora otvaramo grafik kvaliteta vazduha u vremenskom rasponu od montiranja senzora pa do posljednjeg mjerenja. Na dnu Web stranice nalazi se i sumirani grafik kvaliteta vazduha svih senzorskih čvorova i dodatne informacije šta prestavljaju PM10 i PM2.5 čestice. Izgled mape prikazan je na Sl. 5.

 ISBN 978-86-7466-930-3

Sl. 5. Mapa sa označenim senzorskim čvorovima

V. ZAKLJUČAK

U radu je prezentovan senzorski čvor sa mogućnošću praćenje količine PM10 i PM2.5 čestica. Realizovana je platforma za praćenje, koja omogućava jednostavnu prezentaciju podataka grafički, prikaz na mapi i tabelarno. U narednom periodu je planirano da se dobijena mjerenja upoređuju sa mjerenjima koja su dobijena referentnim metodama i da se odredi koeficijent kalibracije. Takođe, planirano je unapređenje mobilne aplikacije da pomoću lokacije korisnika ispisuje poruku kvaliteta vazduha.

ZAHVALNICA

Zahvaljujemo se gradu Čačak, koji je obezbijedio novčana sredstva za realizaciju projekta praćenja kvaliteta vazduha.

LITERATURA

[1] OECD, The Economic Consequences of Outdoor Air Pollution, 2016 dostupno na: https://www.oecd.org/env/the-economic-consequences-of-outdoor-air-pollution-9789264257474-en.htm.

[2] Richard Burnett, Hong Chen, Mieczysław Szyszkowicz, i drugi, *Global estimates of mortality associated with long-term exposure to outdoor fine particulate matter*, 115 (38) 9592-9597, 2018.

[3] Bulot, F.M.J., Johnston, S.J., Basford, P.J. i drugi. *Long-term field comparison of multiple low-cost particulate matter sensors in an outdoor urban environment*. Sci Rep 9, 7497 (2019). https://doi.org/10.1038/s41598-019-43716-3J

[4] Davis AirLink® Quality Monitor, Davis Instruments, dostupno na https://www.davisinstruments.com/pages/airlink .

[5] Rezultati projekta prekogranične saradnje, https://banjaluka.net/na-elektrotehnickom-fakultetu-predstavljeni-rezultati-projekta-prekogranicne-saradnje/.

[6] Službeni glasnik Republike Srbije br, 11/2010, 75/2010 i 63/2013

[7] Objedinjeni prikaz automatskog monitoringa kvaliteta vazduha u Republici Srbiji, http://www.amskv.sepa.gov.rs/

[8] Ambijenatlni zrak – Standardna gravimetrijska metoda za određivanje masene koncentracije PM10 ili PM2.5 u suspendovanoj čestičnoj tvari (EN 12341:2014)

[9] Laser PM2.5 Sensor specivation SDS011, https://cdn-reichelt.de/documents/datenblatt/X200/SDS011-DATASHEET.pdf

[10] Digital universal particle concentration sensor PMS5003, https://aqmd.gov/docs/default-source/aq-spec/resources-page/plantower-pms5003-manual_v2-3.pdf

[11] Digital universal particle concentration sensor PMS7003, https://download.kamami.pl/p564008-PMS7003%20series%20data%20manua_English_V2.5.pdf

[12] Marek Badura, Piotr Batog, Anetta Drzeniecka-Osiadacz, Piotr Modzel, *Optical particulate matter sensors in PM2.5 measurements in atmospheric air*, 2018, https://doi.org/10.1051/e3sconf/20184400006

[13] 3W Ultra-small PowerModule PM03/PM01/PM09/PM12, Hi-Link, https://datasheet.lcsc.com/szlcsc/1909111105_HI-LINK-HLK-PM24_C399250.pdf

[14] Yii2 dokumentacija, https://www.yiiframework.com/doc/guide/2.0/en

[15] Leaflet dokumentacija, https://leafletjs.com/reference.html

ABSTRACT

PM2.5 exposure stands out as a leading health problem globally while the problem of air pollution is one of the leading causes of death in the world. Due to the poor air quality, the requirement for frequent measures in as many locations as possible has led to the development of low-cost sensors. Sensor nodes are being developed by utilizing widely accessible sensors, their use is to acquire the current concentration of particle matters. This raw data is later processed and used to calculate the air quality index – AQI. This paper presents the development of sensor nodes, as well as the development of a unique web platform. The web platform is equipped with a graphic representation of the date and history of archived data that had been collected. While this platform has been developed for the city of Čačak, it can be adapted to other geographical areas and it can be broadly used.

Air Quality Monitoring Platform in the City of Čačak

Nikola Kukrić, Božidar Popović, Slobodan Lubura, Zorana Mandić

 ISBN 978-86-7466-930-3