Delft University of Technology
Master of Science Thesis in Electrical Engineering

Multi-domain network telemetry solution - Proof of Concept implementation

Paweł Maćkowiak

TU Delft Delft University of Technology

Multi-domain network telemetry solution - Proof of Concept implementation

Master of Science Thesis in Electrical Engineering

Networked Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Paweł Maćkowiak

24 May 2024

Author
Paweł Maćkowiak (p.mackowiak@student.tudelft.nl)
(paw.mackowiak@gmail.com)
Title
Multi-domain network telemetry solution - Proof of Concept implementation
Date
31 May 2024

Graduation Committee

Prof. dr. ir. Fernando A. Kuipers	Delft University of Technology
Dr. Zekeriya Erkin	Delft University of Technology
Dr. Piotr Zuraniewski	TNO

The work presented in this thesis has lead to a paper which has been presented at the Workshop on Transparency, Accountability and User Control for a Responsible Internet (TAURIN 2023), which was part of 28th European Symposium on Research in Computer Security.

Abstract

The inability to check how our Internet traffic is being handled and routed poses all kinds of security and privacy risks. Yet, for the typical end-user, the Internet indeed is such a black box. This thesis, adheres to the call for an Internet that is more transparent, and as a step forward proposes a mechanism that carefully balances the desire to share transparency information with the necessity to not expose all internal details of a network. The presented work realises this by building on the framework of multi-party computation. The proposed architecture and corresponding proof-of-concept is evaluated via experiments and demonstrates the feasibility of the concept to improve Internet transparency.

Preface

I would like to thank Prof. dr. ir. Fernando A. Kuipers who took the time to steer this process in the right direction.

I would also like to thank my supervisor Dr. Piotr Żuraniewski for collaborating with me on the work presented in this thesis.

I want to thank my parents for giving me the opportunity to pursue this goal of completing a master's degree at a university like TU Delft.

Finally, I am profoundly grateful to my wonderful wife, my family, the friends I have met along the way, who have shown me support when I needed it most.

Paweł Maćkowiak

Delft, The Netherlands
24th May 2024

Contents

Chapter 1

Introduction

1.1 Problem description

Nowadays, almost every aspect of modern life is connected with the Internet and its operation. Remote access to blood test results or unlimited knowledge resources that can be found on the web are just some of the possibilities that have been opened up by the Internet. With its growing impact, the Internet itself is also growing exponentially in terms of connected devices and users, resulting in an increasing demand for network resources and services available online.

The fundamental building blocks of the modern Internet, were designed over 45 years ago and therefore were not developed having in mind the demand that is experienced today. Furthermore, the model of communication with all entities participating in it is leading the network to be collection of complex and rigid segments. While many segments rely on a single vendor's equipment for better support and interoperability, a side effect of such dependencies contributes to creation of barriers and slowing down the process of implementing new and potentially better technologies, architectures and network services which aim at addressing existing deficiencies and problems.

Technologies like Virtual Reality (VR) or the Tactile Internet are extremely resource-intensive and resource-sensitive [49], which means that poor Quality-of-Service (QoS) directly affects the Quality-of-Experience (QoE). Therefore, one challenge is to have an infrastructure that would allow to satisfy the requirements set by such an application, while the other is the ability to monitor the resources assigned to this application. Current network monitoring is mainly focused on overseeing the stability of the network as a whole. Once anomalies have been identified, most cases require further investigation to determine the root cause of the problem. Typically such diagnostics require network administrator intervention, which can be labour-intensive. Diagnostics of cloud-based services often span multiple domains, making it even harder to find the root cause, if possible at all. In some cases, this results in situations in which the domains' administrators claim other parties being responsible for the service degradation, while the problem remains unresolved. In addition, the end-user who entrusts the traffic generated by them has no information on how it is being processed making them helpless in a situation where the internet-based service

does not operate in the expected way.

1.2 Motivation

The structure of the modern Internet means that the end user generates network traffic and then hands over its processing to third parties. For this reason, the Internet is treated as a black box, where users can only hope for the best possible outcome. Moreover, this structure may be described by the following characteristics of limited transparency, accountability and lack of at least partial control over how user's traffic is processed. In this context, a user can be understood in one of several ways, firstly as an end-user in the form of a person using a service on the internet or an entity that provides that service. Secondly, it can be the operator of the critical infrastructure that uses the public Internet. Lastly, it could be the Internet Service Providers (ISP). The attributes outlined above will be discussed further in this section, followed by the example of potential benefits, for the different user groups, resulting from the evolution of the current state of the Internet.

1.2.1 Limited Internet Transparency

Transparency is a feature that allows the user to see how network traffic is being processed. In the current state of affairs, the sender of data cannot obtain knowledge of the path that the packet has followed through the network, nor can obtain a guarantee that two packets with the same destination will travel along the same path. This creates dependencies, both negative and positive. The first negative one is that the sender does not know which way his network traffic is being sent, i.e. what are the intermediate 'networks' that process the traffic and whether this is the most optimal path for that traffic. In the extreme case, presented in Figure 1.1, this may mean that, as a result of the traffic hijacking event, network traffic will be routed through half the world, which the end user might only have the chance to notice by increased delay. The second negative dependency is the lack of user awareness of the details of the infrastructure that processes their traffic. This becomes a noticeable problem with the growing centralisation of the largest Internet companies [5][6]. This concern is being raised by both end users and policymakers the European Union[36][5].

1.2.2 Limited Control

Controllability describes the ability of users to determine how they want their traffic to be processed by network operators. The ability to make such a decision would result directly from the transparency described above. A few of the problems resulting from the lack of controllability may be the following: lack of route optimisation for both the sender and the receiver, poorer accessibility resulting from the lack of ability to prevent outage when e.g. the resource usage surges, which is relevant for end users, and limited load-balancing capability which is relevant for ISPs.

Figure 1.1: **Path alternation as a result of traffic hijacking event.**

1.2.3 Examples

The end user is the party that stands to gain from an improvement in the conditions related to the previously mentioned attributes. The first type of end-user is a private individual using for example video conference applications (such as Teams, Zoom), game streaming (discontinued Google Stadia) or virtual reality (VR) applications that are hosted using cloud resources. For this user, the benefit would stem from the ability to monitor network traffic and possibly detect deteriorating QoS. This ability is also relevant as application performance becomes extremely important with very high requirement concepts such as tactile internet [49] being developed.

The second type of end-users are companies whose functioning leverages the Internet, for example, Critical Infrastructure operators. With a growing number of devices that allow controlling their functions via the Internet, entities such as providers of smart city systems gain the ability to supervise their dependencies in the network, which is of great importance in the context of the security of the infrastructure itself and, consequently, of users depending on that infrastructure [51]. The use of the Internet optimises operating expenditure as opposed to maintaining and paying for private network connections used by infrastructure operators.

Another type of entity that might benefit from the developments related to the aforementioned characteristics are ISPs. The advantage can be highlighted in a following scenario in which a group of network operators may benefit is an attempt to jointly solve problems in the network whose origin is not easy to determine. Such network problems may adversely affect the QoS or availability of provided services and consequently lower QoE for users using them.

The described above presents a non-trivial challenge of utilising current technologies in the best possible way in order to provide end users and domain administrators with more network related transparency, understood as the level at which an entity is able to see how their network traffic is being processed. The system should be designed in a way that would make each domain involved capable of contributing the most to reach a common goal which in this case, is

to assist the domains administrators in their problem resolution task and give end-users more knowledge about how their traffic is being processed.

1.3 Research Objective

The main research objective is to assess feasibility of realising a multi-domain monitoring system, aimed at enhancing Internet transparency. This system using existing telemetry protocols and technologies should allow multiple domains share their telemetry data and as a result aid to solve the challenge of locating an issue within a network, that may have occurred in one of the domains through which the network traffic is being transported. From this objective the following step-by-step process is derived:

- To identify the related work and research in the field of networking, which is relating to the telemetry solutions and their implementations, as it would give perspective for the input into the multi-domain monitoring system.
- To identify the relevant, including the existing (traditional), network technologies, architectures, as well as ways of sharing data along with their major benefits, issues, and concerns.
- To propose an architecture that realises research objective while addressing identified issues and concerns over the involved technologies in the proposed multi-domain diagnostic system.
- To demonstrate a proof of concept (PoC) implementation of the proposed multi-domain diagnostic system on a testbed.
- To conduct validation and performance evaluation of proposed architecture.

1.4 Contribution

As far as this work's contribution is concerned, the proposed architecture is aimed at realising a system that enables the implementation of a multi-domain diagnostic system. This work adheres to the call for an Internet that is more transparent, and as a step forward proposes a mechanism that carefully balances the desire to share transparency information with the necessity to not expose all internal details of a network. As a result, this is a step towards improving upon the problem of lack of control in the current structure of the Internet, since knowledge of how the traffic is being processed is the starting point for taking network related control decisions.

1.5 Thesis structure

The structure of the thesis is as follows. Chapter 2 discusses the design goals for the proposed system. Chapter 3 introduces concepts and technologies needed to obtain network data from multiple domains, this section also presents related work. It is then followed by Chapter 4 that presents the proposal and PoC

implementation. In Chapter 5 performance evaluation of the proposed architecture is conducted. Finally in Chapter 6 conclusion and directions for future work are presented.

Chapter 2

Design goals

As presented in Section 1.2 , the current structure of the Internet has limitations due to the fact that its users treat the Internet like a black box. A system that allows the user to "peek" into the inside of the domains responsible for processing their traffic would improve the present situation, would have the potential to improve fault diagnosis of systems and could give users a perspective into how their traffic is processed. In section 2.1 design goals for such system are presented.

2.1 Design goals

The main design goal derives directly from the problems of the current Internet structure, described in more detail in Section 1.2.1. This objective is to provide users with greater transparency, i.e. to give them insight into how the network traffic generated by these users is processed. In addition to the main focus, the next goals are (1)to propose a solution that allows data sharing between multiple-domain while not exposing sensitive details about them, and (2)modularity - to create a solution which allows the use of variety of underlying technologies and can be expanded with additional functionalities.

2.1.1 Transparency

Transparency describes solutions' ability to provide information about the infrastructure and means used to transfer the data with regards to involved network operators, their properties and relations between them. A network operator is understood as an administrative entity that operates any network involved in transfer of the data, including access and transit networks, datacenter networks or content delivery networks (CDN). Transparency is understood in two ways. First, data plane transparency allows to gain insight into which network operators are involved in transporting data, how those data flows are being processed and with what result. For it to be possible a network operator needs to keep track of where a flow enters the network, how it is then being processed (for example how it is routed, whether it is processed by deep packet inspection systems), how long it takes to process flow in question, whether the processing is interrupted, for example by packet drops or congestion, and finally where does

the data leave the network. Well-executed transparency would allow users to determine how their traffic reaches the services they are connecting to. Second, control plane transparency describes infrastructure (e.g. sharing, without exposing, the manufacturer of the equipment, software and its versions, all to ensure that the infrastructure is free from security vulnerabilities, the same respectively for open source tools) and relations between network domains (structure of corporate capital, geolocation, jurisdiction). This type of transparency allows end users to verify if service providers involved in transporting their data meet their expectations and policies. For a user of a video conferencing application, this might mean that their data is being processed in a datacenter located within European Union (which means that the law specified for the particular jurisdiction applies). In comparison critical infrastructure operators might be inclined to verify if their network traffic is being handled by the most up-to-date infrastructure, resulting in lower risk of security incident.

2.1.2 Sensitivity of shared data

In creating a system which aims to provide third parties with access to network characteristics, the question arises of how to balance the amount of information provided against the security of the infrastructure they describe. The competitive nature of operating a digital enterprise (ISP, cloud provider, CDN provider, company offering a digital product) complicates the process of formulating a set of requirements and constraints that would convince involved parties to share their monitoring data or data describing their operation. The type of information shared or how it is analysed would have to ensure that in the process of sharing the data, the operator who chooses to do so does not take the risk of affecting its competitive edge or exposing its business model. The setting problematizes itself with more parties getting involved as the probability that an additional entity is to introduce new unmet requirements to the table rises. This makes one of the design goals the secrecy of the information shared by the entities, and the degree to which it will be realised can directly affect the value derived from the adoption of such system. For example, to increase the probability of wide adoption, the proposed solution could assume that involved parties are reluctant to share their data, as a consequence of distrust towards each other and the end user.

2.1.3 Usefulness and Usability

The usefulness of a system could be described as an ability to realise other design goals in a manner that allows for the system to be widely adopted by a broad range of users. It can come from an understanding of the technology and trust resulting from it, as described by "Attitude" in the technology acceptance model(TAM). It has been presented that compatibility, relative advantage, and complexity have the most significant relationships with adoption across a broad range of innovation types [29]. Furthermore, the production of tangible results will directly influence the system's usefulness [24]. Therefore transparency should be improved, when the system performs analysis, the results of which are easily interpretable or will allow an informed decision to be taken. The ease of adaptation of the solution has a direct impact on the desire to use it. If the provision of a better level of transparency required from operators and users to

use a particular type of equipment, the pace of adaptation of such a solution may not be sufficient to create an impact. An appropriate level of modularity would enable the collection of data from various types of devices, which would allow more users to benefit from such a solution. This means that it is necessary to design a solution that allows the collection of measurements from devices supporting different standards, so that collected data describing the same metric can contribute to a joint analysis.

Chapter 3

Network Observability System

3.1 Monitoring of networks and digital systems

Transparency as described in the earlier section is presented twofold, one of which is a control plane transparency. This type of transparency requires the operator to compose a description of its operations (mechanisms used to process traffic, tools and relationships with other operators). This information could be also gathered by an independent observer collecting data about the operator. Notion of information included in such a description is presented in the Responsible Internet [18]. This process insofar as it requires the composition of all the information that is necessary for a specific analysis may be performed once until it is required to update this description. Meaning that users wishing to obtain information derived from this description may do so without need of it being regenerated. Part of such description should also include a list of available measurements performed by the operator. The measurements themselves with resulting data are part of data plane telemetry. Depending on the scope of the system under observation and the type of operator who supplies it, measurements may concern one or more aspects of it (servers, network equipment). This means that for a flow that originates from the end user, travels towards the system handling the request and then returns to the source of the request, there are many intermediate steps in which observations can be made. For an example network application, the points of observation are the local machine that hosts the application, then the network that is used for transport and finally the system that executes the queries sent by the local machine. This general picture for each of the points indicated could be further specified, depending on the desired accuracy of the observation made. The accuracy of the measurements should be motivated by obtaining sufficient information to understand the behaviour of the application under observation, or by locating a point in the system that prevents failureless operation of that application. Given the networking nature of this work, the following section will look more closely at the network telemetry aspect of gathering measurements to monitor the system/application and to diagnose potential problems.

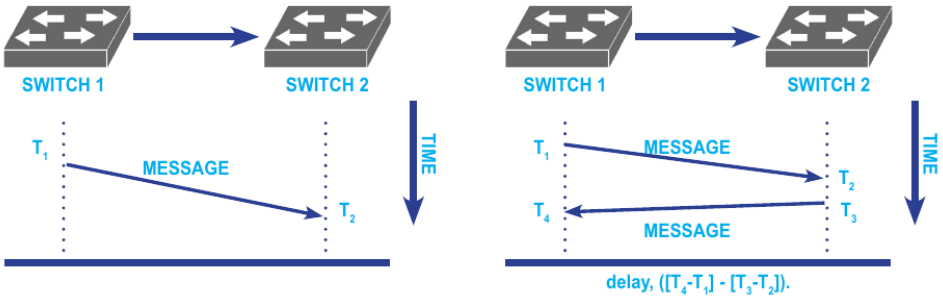


Figure 3.1: **An example of calculating Round Trip Time using active measurement.**

3.1.1 Network Telemetry

Telemetry is the process of measuring the performance and monitoring the state of a network by taking measurements at multiple points in the network. It is conducted through collecting performance metrics from points in the network called observation or measurement points, depending on the method used. Subsequently, data obtained as a result of telemetry is then collected in a central collector, where it can be further processed. The performance metric is a quantitative value that has been obtained as part of the assessment of network performance and/or reliability. This metric is defined in a precise way to ensure the unambiguous meaning of the measured value [35][20]. As mentioned, telemetry data is collected from various points in a computer network that allow packets to be observed for measurements [47]. An example of an observation point can be any medium used in a network to which a probe can be attached, such as a single interface or a set of network interfaces (both physical and logical) in a router, switch or network card.

There are two main approaches to network measurements which are active and passive methods [35].

To perform an active network measurement, synthetic packet streams are generated. Streams of packets resemble measured network traffic or have a unique structure (special probe packets) to distinguish a measurement packet from regular network traffic. An example of active measurement is presented in Figure 3.1, where special messages are sent between the switches. The timestamps that can be embedded in the message body are then used to calculate round trip time between the switches.

Passive methods, on the other hand, depend on the existence of data streams in the network traffic (traversing through single or multiple observation points) which can be used as input data for the measurement. Passive methods work by collecting information on streams of interest from observation points, analyse the collected data in order to assess the metrics of network efficiency. Worth noting is that the transfer of data to the collector may have an adverse effect on the measured metrics.

Derived from the two main types of measurements there are also hybrid methods. A measurement is considered to be using a hybrid method when it uses

Method	Description
Active	Generating a data stream for measurement
Hybrid Type I (Type II)	Modification or augmentation of a single data stream (two or more data streams), or action to change the handling of the data stream(s).
Passive	Observation of existing data stream for measurement

Table 3.1: **Summary of measurements methods.**

both elements known from passive and active methods [31]. A comparison of the general description of the methods is given in Table 3.1.

3.1.2 Operation, Administration and Maintenance

As described in section 3.1.1, active measurements use special packets, which are part of the standards and tools labelled by the term Operations, Administration and Maintenance (OAM). OAM is used for fault detection and isolation, performance measurements, network discovery and planning in computer networks [35][31]. The OAM protocols are defined at different layers of the OSI-model. Ethernet, IP, MPLS and many other network protocols use OAM for the above-mentioned purposes. The standardization bodies for the tools and standards included in the OAM are three entities: Internet Engineering Task Force (IETF), International Telecommunication Union Telecommunication Standardization Sector (ITU-T), and Institute of Electrical and Electronics Engineers (IEEE). The majority of the tools within OAM could be used as an input to a telemetry system, therefore an overview of some of them is provided in Figure 3.2. In the remainder of this section the more often used protocols from the OAM toolset will be described as to report developments in that domain, and point out the possible source of telemetry data.

Internet Control Message Protocol (ICMP) is defined as a supporting protocol for Internet Protocol (IP) as defined in standardisation [39]. As the IP protocol was not designed to provide complete reliability, the ICMP protocol provides a means by which feedback about faults in the network can be reported. For example, a network device, when communicating with another IP address, is notified if said address is unreachable. ICMP messages are used as a means to diagnose networks or can be generated as a result of an error [14]. If an error occurs the message is being directed to the original source of the packet that caused it. An example of such a situation is reaching value 0 in the time-to-live (TTL) field in the IP header. Worth noting is that a tool called traceroute [2][50] uses this mechanism to its advantage as it successively sends out packets while incrementing the value of TTL field by 1, in order to estimate the path the packet takes to its destination address.

Simple Network Management Protocol (SNMP) is a part of the Internet Protocol Suite and was defined by IETF [26]. The purpose of SNMP is to collect and organize data related to network devices on the IP network. Additionally, SNMP can be used to modify data in the device in order to change its behaviour.

NETFLOW is a vendor specific solution developed by Cisco that allows to

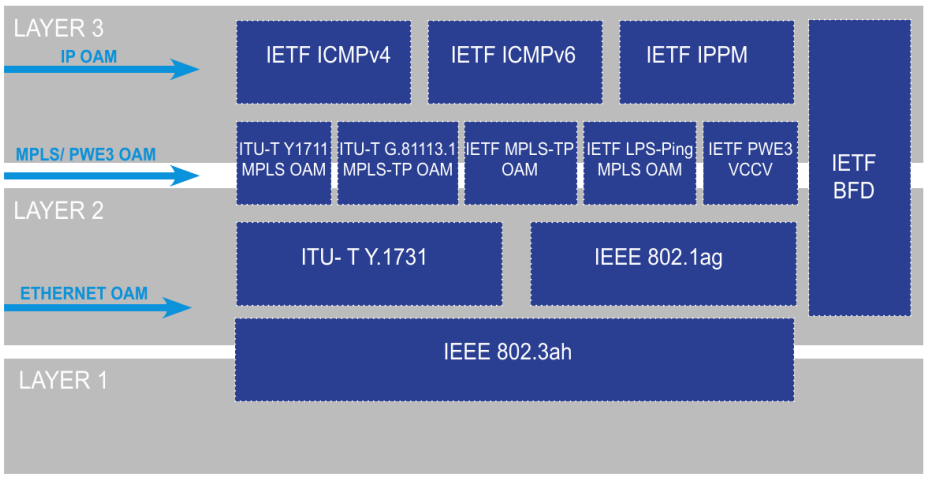


Figure 3.2: **An overview of OAM standards.**

monitor traffic traversing specific network devices [19]. Devices supporting this protocol collect statistics about IP traffic on NETFLOW enabled interfaces. Gathered data include identification of ingress interface, 5-tuple, and IP type of service. Version 9 of the NETFLOW protocol formed the foundation for Internet Protocol Flow Information Export (IPFIX) which is an IETF standard [47].

sFlow similar to NETFLOW, in a sense that it allows to export data about packets traversing network together with interface counters. The letter 's' in the name stands for sampled, since sampling is used by sFlow to achieve scalability. Depending on the sampling rate, one in n packets is sampled and exported in truncated form. While sampling does not give 100% accurate results it can provide quantifiable accuracy [37].

3.1.3 Software-defined networking

Next to traditional network management, where network devices such as switches and routers are configured individually, and network policies are enforced through device-specific configurations, there is the software-defined networking (SDN) principle that centralizes network management thus enabling new ways of conducting network measurements. This section will describe the concept of SDN and network measurements in such networks as alternative sources of telemetry data. Since SDN will be used as a building block for this work's proof-of-concept it will be described in greater detail.

Software-defined networking is an approach to network management that logically centralises the network layer responsible for traffic management (routing) in the network by separating it from the underlying data plane layer responsible for the transmission of network traffic. The main purpose of the SDN approach is to centralise network intelligence and state (global network view) while abstracting the network resources underneath. The main benefit of using SDN for telemetry is stemming from processing traffic on a per-flow basis, which allows monitoring of the network in a more granular way. Furthermore, logically pla-

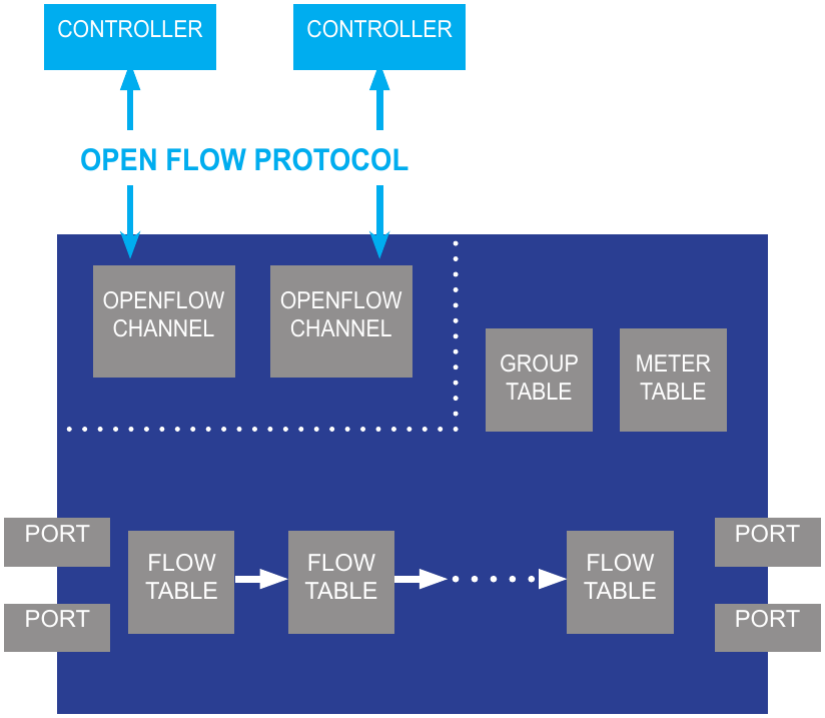


Figure 3.3: **Representation of Open Flow Switch.**

cing the software-based SDN controllers in a centralised position, gives a global view of the network. As a result, applications and policy engines communicating with the controller can use centrally located knowledge to make decisions, which the controller then redistributes to data/forwarding plane using the SDN Southbound Interface (e.g. OpenFlow).

3.1.4 OpenFlow

The OpenFlow protocol must be implemented on both sides of the interface between the controller and the network infrastructure. In OpenFlow, network traffic is referred to as flow, which is defined by rules, both static and dynamic, that are programmed by the SDN controller. These rules are reflected in one of three types of tables, according to the architecture of the OpenFlow switch [4]. As presented in Figure 3.3, the OpenFlow switch consists of one or more Flow Tables, Group Tables and Meter Tables. The rules in the corresponding tables determine how the network traffic is processed and forwarded. Communication between controller and switch consists of instructions to add, remove or update a rule. Such communication can occur reactively (upon packet/flow initial arrival when the forwarding device doesn't know how to forward traffic) and proactively (for provisioning purposes).

When a packet enters an OpenFlow switch it is processed according to the pipeline derived from the specification. Usually the processing pipeline consists

of several flow tables, which each may consist of multiple flow entries. The flow entry is described in the OpenFlow Switch specification [4], and its consists of 6 main components:

1. Match fields - contain the ingress port, packet headers, and metadata specified by the previous table.
2. Priority - defining matching precedence of the flow entry. When being matched with the flow table, only the highest priority flow entry that matches the packet is selected from all available.
3. Counter - keeps track of number of packets that matched certain flow entries.
4. Instructions - used to modify the action set or pipeline processing matched packets. Various instruction types are described in specification such as apply-, clear-, write-actions.
5. Timeouts - describes the maximum time before a flow entry is removed when it has matched no packets or the absolute time after which the rule is being removed.
6. Cookie - flow entry identifier assigned by controller.

It might happen that an incoming packet has no matching flow entry in that particular flow table. In such case, the action applied is described by a wildcard flow entry with priority equal to 0 and is called the table-miss flow entry.

One of the available methods of processing a packet in a flow table is to redirect it to a group table in order to perform an additional action available within this table (e.g. flooding, multipath, and link aggregation).

Ultimately, the OpenFlow pipeline also offers the use of meter tables. Processing performed within the meter table is done by redirection of traffic through flow entries and concerns various performance-related actions (QoS), such as rate-limiting actions on incoming traffic.

The processing pipeline in an OpenFlow Switch is presented in Figure 3.4.

The SDN architecture described above and the resulting packet processing allow for a number of new measurement methods in the network that were previously not possible due to existing constraints. The principle of processing traffic on a per-flow basis allows monitoring of the network in a more granular way. In SDN networks, the most basic measurement method is requesting per-flow counters, which are one of the fields available under flow entry. New possibilities have also resulted in new tools for measuring traffic. OpenTM [46] allows to monitor the network by creating a Traffic Matrix which is a result of querying switches on regular intervals. Another solution is called, OpenSAFE [10]. OpenSAFE uses the fact that every new flow request in an SDN network has to pass through the SDN controller. The controller has the ability to forward the creation of new flows to multiple traffic monitoring systems which record the traffic and analyse it with an IDS. A solution called OpenNetMon [48] is a software implementation capable of monitoring per-flow metrics, especially throughput, delay and packet loss, in OpenFlow networks. OpenNetMon uses statistics collected in switches per flow and uses active measurement techniques to determine values such as path delay.

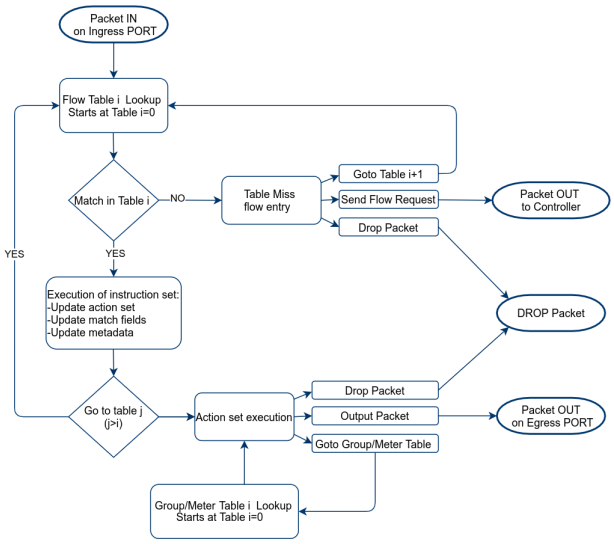


Figure 3.4: **Representation of processing pipeline in OpenFlow switch in the form of flowchart**

3.1.5 iOAM

In-Situ OAM (iOAM) is a standard developed by IETF [15], that at the time of writing has reached a RFC maturity level. iOAM describes how operational and telemetry data can be recorded in a packet as it traverses network devices. The goal of developing this standard is to complement active OAM based on ICMP probes or other types of probe packets. iOAM aims to define a common way of embedding telemetry and operational data in packets. Moreover, the iOAM also defines a domain within which it can be used. The description specifies that the standard is a domain-oriented feature of a network domain, which is defined as a set of devices administered by a single entity. Examples of such domains are the campus network or overlay network with virtual connections. As a consequence, domains defined within the iOAM must have a clearly defined perimeter and the domain manager must ensure that the measurement data contained in the packets do not leak outside the domain. In addition, the standard points out that the administrator should take into account the use of the iOAM in terms of its impact on load-balancing, path MTU (that all links in the network can handle packets with increased size caused by the iOAM headers), and ICMP message processing.

3.2 Distributed systems architectures

The standards and network monitoring tools described, depending on the technology stack of a domain, can be used to observe network stability and performance. They allow the domain administrators to collect data and carry out analysis based on it. However, the structure of the Internet, as described in the previous Chapter, makes it impossible for a user of an application to leverage insight generated by these tools, since their network traffic is handed to and

forwarded by a group of third parties. Hence, the traditional approach of monitoring individual network domains separately is no longer adequate in complex distributed systems, where information flows between different domains. Creation of multi-domain network telemetry systems may provide a more comprehensive view of the network used as a distributed system, making it easier to detect and troubleshoot possible issues.

This section will present architectural models that could be used to realise a multi-domain system for the collection and analysis of network telemetry, as well as methods to perform analysis with protection of data in mind.

3.2.1 Client-Server model

The client-server is an architectural model for a distributed application based on division between service or resource providers-servers, and clients distributed across a network that request access to said services/resource [25]. The server in a client-server model can be designed to perform a variety of functions, such as data storage, processing, and analysis. Communication is typically request-response based. In this model vertical scalability is possible by adding resources to servers, while horizontal scalability is limited due to the centralisation of services. In Client-server model single points of failure can impact the entire system fault tolerance relies on redundancy or failover mechanisms. Compared to other models it is simpler to design and manage but can become complex with increased functionality (especially in case of monolithic application). Data consistency is easier to maintain as there's a central authority for data management. In the context of network-telemetry two kinds of clients could be distinguished. The first type would be a client who is a data source, sharing measurement data collected in its domain. While the other type would be a client that can submit requests to the server to retrieve data or perform analysis on the data stored on the server.

As described above client-server model offers several advantages for distributed systems, such as scalability, and ease of maintenance. The centralised server allows for efficient data management and processing, and clients can be added or removed from the system as needed. On the other hand, the centralised server can be a single point of failure, making the entire system vulnerable to attacks or breaches. If a malicious actor were to gain access to the server, they could potentially access all the data collected. Encryption and access controls can be implemented to mitigate these risks. It's also important to ensure that the data transmitted between clients and the server is secure to provide confidentiality and integrity. Moreover, the central entity in this model (the server) would have to be managed by an entity to whom all customers have a relationship of trust. This criterion could prove unfeasible to implement due to customer focus on defending their own interests.

3.2.2 Microservices model

In the microservices model, data is collected and processed by independent, loosely coupled, and fine-grained services that are responsible for specific functions [25]. Each service is responsible for a single task, such as data collection, data storage, or data analytics. Services communication often involves lightweight protocols, such as HTTP/REST or message queues. Services communic-

ate to accomplish certain tasks. In this model horizontal scalability is possible by independently scaling services, each service can scale individually. Isolation between services can limit the impact of failures, however this model requires proper error handling and recovery strategies. Compared to other models microservices are more complex due to the distributed nature and since implementation requires additional tools for monitoring, deployment, and coordination. In the context of network-telemetry a client interested in gaining insight from a domain, would query a data analytics service which then depending on performed analysis will further request appropriate data from data storage services.

The advantage of this architecture is that it allows for easy scaling, as individual services can be scaled independently. Furthermore, the distributed nature of the microservices architecture can make it more difficult for attackers to compromise the system. However, if one or more microservices are compromised, it can lead to data leakage and integrity issues. Thorough access controls should be implemented to secure the communication between microservices. Additionally, each microservice should be properly authenticated to prevent unauthorised access.

Compared to the client-server model described earlier, components such as the data storage and data collector could reside within the domain of which they are part. This would reduce the need to share more data with a central entity, but does not eliminate the need to share the data necessary for analysis. The risk is that over time the data shared could reveal sensitive information about the domain, again raising issues of trust.

3.2.3 Publish-Subscribe model

The publish-subscribe [25] architecture is a model focusing mainly on the way messages are distributed in the system. The sender of messages, called publisher, does not send messages directly to the specific receivers-subscribers. Instead, it groups messages into classes, with no direct aim of sending them to specific subscribers. In a similar way, message recipients express their desire to receive messages of a specific class without specifying their origin. When publishers generate events or messages, they send them to the message broker along with an associated topic or category. The topic serves as a label or identifier for the type of event being published. The broker is responsible for determining which subscribers are interested in that event based on their subscriptions. The message broker then routes the event to all relevant subscribers. Consequently, subscribers receive the events that match their subscriptions. They can then process or react to these events as needed. In this model scalability depends on the scalability of the messaging infrastructure. As for fault tolerance of publish-subscribe can be resilient to failures as components can continue functioning even if one or more subscribers or publishers fail.

In the context of a multi-domain network telemetry system, this model might be unfeasible to realise the exchange of information between domains. One problem in using such a model could be the non-scalability in the way the classes of published messages are defined. Additionally, the responsibility for processing messages would be shifted to the subscriber, who may not have the necessary means to process complex measurement data. Furthermore, direct sharing of measurement data would infringe the interests of the domains from which the data originates. This model, on the other hand, can be used to

propagate measurement data within the domain, which would allow several points of presence to be created for further data analysis with other domains.

3.2.4 Peer-to-Peer model

In the peer-to-peer (P2P) architectural model [25], communication and exchange relations are established between peers that are part of the system. Peers themselves in a P2P model can be equally privileged, in other words, can be equal participants in a formed network (depending on application and type of P2P network don't have to). The data is collected and processed by distributed nodes, with each node sharing data with its peers. A different approach would be for each node to perform analysis on the data it collects, and the results can be shared with other nodes. While P2P networks increase robustness by eliminating a single point of failure, which is inevitable in a client-server model, the scalability of the system may be more difficult to achieve depending on the performed function of the network. For example, in the case of data sharing, a P2P network requires that at least one node participating in the network has the data that is being requested. This highlights that the success of a system based on a P2P network relies on the cooperation of involved nodes/parties. Moreover, similar to the models outlined above, the relationship of trust to the members who are part of the network remains a contentious issue.

3.2.5 Flexible system design

Each of the models outlined above carries both advantages and disadvantages. The structure of each has the potential to optimise interactions between parties involved, provided usage patterns of realised services leverage the advantages of a particular model. For example, the realisation of the control plane transparency described earlier in section 3.1, containing the description of the operator/-domain, requires an interaction in which a user queries the system for information about a domain's specific properties. Depending on whether the source to be queried was the operator itself or a repository describing a greater number of operators (e.g. to search for an operator meeting certain requirements), the user would send a query to a specific endpoint and receive a response from it (provided that access control allows them to do so). The description of the operator itself is a high-level overview, so the data provided is not as sensitive as the data that is published by data plane transparency. Therefore, the establishment of a trust relation and its subsequent implementation through access controls, provides greater freedom in the choice of architectural model. Consequently for this pattern, the client-server model might be the most appropriate. As already mentioned, data shared to achieve data plane transparency is more sensitive. This is due to the fact that it might reflect the state of the network devices involved in the transfer of certain flows. Such data aggregated over a lengthy period of time could pose a security risk to the domain from which it originates, e.g. due to the discovered network topology. For this reason, the very means of processing such data must be based on the protection of the data itself.

3.3 Data analysis with secrecy

As described in Section 3.2, there is a range of architectural models that can be used to create a multi-domain network telemetry system. Each model, with its advantages and disadvantages, can be used to implement specific system functions. However, in the case of an analysis aimed at achieving data plane transparency, the nature of the data to be shared makes it necessary to use data processing methods that have data protection at their core. This section will describe such methods, detailing how the data is processed, the resulting advantages and disadvantages as well as the suitability of such a method for implementing a multi-domain system with multiple stakeholders.

3.3.1 Functional Encryption

Functional Encryption is a cryptographic scheme that allows for fine-grained access control over encrypted data [22]. Using public key construction it enables different parties to compute specific functions (usually public) on encrypted data while preserving its privacy. Meaning that the input of the function is encrypted while the output is presented in clear text to the party performing the function. As a result, an entity that performs functional decryption learns only the result of the function performed on specific data, however nothing about the data itself. In the context of a multi-domain network telemetry system, functional encryption can be used to selectively grant access allowing different parties to perform specific computations without revealing the entire dataset. However, implementing and managing a functional encryption system can be complex, requiring careful design and secure key management. Furthermore, depending on the specific functional encryption scheme, there may be limitations on the types of computations that can be performed on the encrypted data allowing only to evaluate linear and quadratic functions. Moreover, the pattern of interaction in a system based on Functional Encryption makes it impossible to perform analysis on data from several domains without prior aggregation (which creates a trusted third-party dependency).

3.3.2 Fully Homomorphic Encryption

Fully Homomorphic Encryption [9] is a cryptographic technique that allows computations to be performed directly on encrypted data, without the need for its decryption. Different types of Homomorphic Encryption can be distinguished: additive, partial, leveled, somewhat (SHE) and fully (FHE). FHE is the type of homomorphic encryption in which potentially any function can be performed while in other cases limitations are present. The most practical application of this processing model is to outsource the computation. This is made possible because both the input and output of the calculation are encrypted using the client's key, while the server only operates on the encrypted data. The calculations do not require additional communication between entities, but FHE is computationally intensive, requiring significant resources and time compared to traditional computations. While progress has been made in optimising FHE schemes, they are still relatively inefficient effectively limiting their practicality in resource-constrained environments. FHE enables secure computation on encrypted data, making it suitable for secure data processing

in multi-domain telemetry systems. However, as with functional encryption, the interaction model does not correspond to a system that might aim to achieve data plane transparency, meaning that result of computation would need to be shared with interested party. This cryptographic technique is more appropriate for the owner of certain data, but not for an individual who wants to obtain information from data that is sensitive.

3.3.3 Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) is a subfield of cryptography working on protocols that enables multiple parties to jointly compute a function on their private inputs without revealing those inputs to each other [40]. In MPC, function inputs are masked or shared as secretes between parties, while the final result of the computation is revealed to a subset of them. The MPC has the advantage of being less resource-intensive, but performing the calculations successfully requires the involved entities to participate in several rounds of communication/interactions. In the context of multi-domain network telemetry systems, MPC can facilitate the collaborative processing of data while maintaining secrecy of it. In other words, it enables multiple domains to work together and leverage their data collectively, even when they cannot directly access each other's data. However, as mentioned earlier, depending on the function being evaluated MPC may require extensive communication and coordination among the participating parties, which can introduce latency and increase computational complexity. Furthermore, as the number of parties involved in the computation increases, MPC can become more challenging to implement and scale effectively.

Each of the three cryptographic techniques, similar to architectural models, has advantages and disadvantages. Functional Encryption offers fine-grained access control and privacy preservation but can be computationally intensive and lacks full data manipulation capabilities. Full Homomorphic Encryption allows for performing arbitrary computations on encrypted data but suffers from high computational overhead and may not be suitable for all applications. Secure Multi-Party Computation enables multiple parties to jointly compute a function while keeping their inputs private, offering a balance between privacy and efficiency. In the context of a multi-domain telemetry system where data collaboration across domains is essential, MPC emerges as the most suitable choice due to its ability to provide secure computation while minimizing computational costs and accommodating the diverse data sources and parties involved. In the following section, more details about data sharing, the security and efficiency of MPC are presented.

3.3.4 Secure Multi-Party Computation - data sharing, security models and efficiency

Multi Party Computation is based upon several primitives, some of which are shared between all protocols while others are only used in specific cases. This subsection describes the basic primitives to give better understanding of MPC.

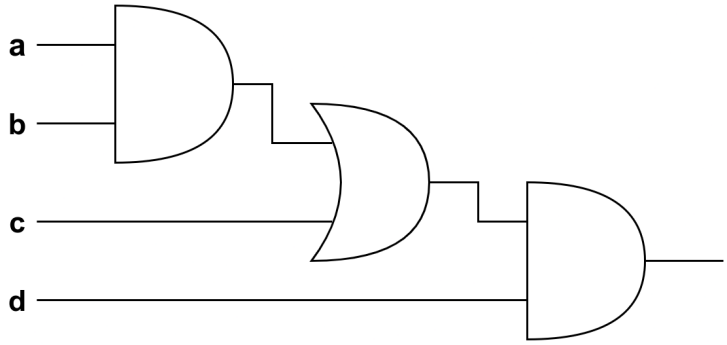


Figure 3.5: **A Boolean circuit evaluating** $((a \wedge b) \vee c) \wedge d$**.**

MPC functions

Although any computable function can be evaluated using multi-party computation, the function's 'complexity' has a significant impact on the evaluation time. MPC protocols typically employ a single computational model that defines both the operations and data formats that the protocol may work with. For instance, the Boolean circuit is a popular computational model. Data is represented as bits (0 or 1), and the operations that can be performed are OR-gates (outputs a 1 if at least one input bit is 1) and AND-gates (outputs a 1 if all input bits are 1). In essence any computable function can be represented as such a Boolean circuit. The complexity of a function is often described in terms of the size (gate count) of the corresponding circuit, together with its depth. The depth is the largest number of gates appearing sequentially in the circuit. For example, the circuit in Figure 3.5 has depth of 3, because the input for the right AND-gate depends on the output of the OR-gate, and the input of the OR-gate depends on the output of the left AND-gate. In most MPC protocols, linear gates (OR-gates) can be computed locally, whereas non-linear gates (AND-gates) require communication between the parties. For that reason, function's complexity can be evaluated by looking at the depth and the number of multiplication gates in the circuit.

Garbled circuits

Garbled circuits are a cryptographic technique used in MPC to securely evaluate a function over private inputs without revealing them to any party involved. In this method, the function to be evaluated is represented as a Boolean circuit, and each gate in the circuit is "garbled" or encrypted by the party creating the circuit. The inputs to the circuit are also encrypted, such that only the correct combinations of input values will successfully decrypt and propagate through the circuit, leading to the final output. The other party or parties involved in the computation can evaluate the garbled circuit using their encrypted inputs, ensuring that they learn nothing about the inputs or intermediate values other than the final result.

Figure 3.6: **Interaction scheme in oblivious transfer.**

Secret Sharing

Cryptographic Secret sharing [42][12] is an essential primitive which is the basis of the protocols in the MPC. The goal of secret sharing is to divide a secret value in a number of shares (n) with a threshold (t) in such a way that any subset of those shares consisting of less than t shares can not be used to reconstruct the secret, while any t shares is sufficient for the secret reconstruction. The survey about secret sharing protocols was conducted by Beimel [11].

Oblivious transfer

Oblivious transfer (OT) is a cryptographic protocol that lets two parties conduct an exchange in such a way that receiver chooses k of n secrets from party holding those secrets, without revealing which secrets were picked.

The interaction in OT is presented in Figure 3.6. As presented in Figure 3.6 a sender holds two secrets x_0 and x_1, while receiver is holding a bit allowing to choose one of the secrets $b \in 0, 1$, receiver can obtain x_b learning nothing about the other secret, while sender learns nothing at all. J.Kilian showed [27] that oblivious transfer is a building block for MPC protocols that are efficient (in a random OT, only three bits of communication are requited to compute the function).

Security of Multi Party Computation

Different MPC solutions are characterised by different threat models. In this section the differences between the models will be described, which have an impact on the decision to use the framework in the section 4.2.1.

The threat models in Multi Party Computation are based on assumption that subset of parties involved in the protocol are controlled by an adversary, while other remain honest. In general the protocol, for each party involved, specifies a function that takes set of variables, such as security parameter, party's private input, random tape (set of uniformly random bits used during execution) and list of messages a party recived so far. From all components parties produces next message together with the addressee or terminates the protocol with specific output. An adversary can corrupt one or more of the parties in which case, said parties, are considered as being an adversary. Depending on adversary be-

haviour, that is whether the adversary follows the protocol or not, two threat models may be determined: semi-honest and malicious.

The semi-honest adversary is represented by corrupted party in the protocol that follows it, however while doing so it tries to gain additional knowledge from all the messages received during protocols execution. Since semi-honest adversary does not take any additional actions other than learning from protocol execution it is also considered to be passive.

The malicious adversary, on the other hand, is represented by corrupted party that may follow the protocol, but can also deviate in an arbitrary way from it in order to attempt violating security of executed function. Therefore, malicious adversary has all the capabilities of analysing the execution that semi-honest one has, but can also take additional actions. An arbitrary way of trying to violate the security may include controlling, manipulating and injecting any message on communication network. Since, malicious adversary may take additional actions in order to breach the security of the protocol it is considered to be an active adversary.

Having described the threat models, the MPC protocol is considered to be secure if during its execution privacy and correctness are being achieved. The privacy is preserved if adversary learns nothing about the inputs of honest parties (other than what can be deduced from the computed output). The correctness is achieved when honest parties obtain the correct output of the function that is executed. Protocols that allow for the semi-honest adversary to be present while preserving said privacy and correctness are considered to have passive security, while those which allow for malicious are said to have active security. While supporting passive security might seem intuitive, worth noting is that the security against a malicious adversary is often implemented with what is known as "malicious-with-abort" scheme. Where the said scheme is in place, upon detection of malicious activity, the protocol is aborted, that is the adversary cannot cause an incorrect output, rather no output at all could be experienced.

Efficiency of Multi Party Computation

Evaluating functions securely through MPC involves additional costs compared to performing the same evaluation on a trusted machine. Implementing MPC protocols demands more resources, with some protocols causing significant computational costs and others necessitating extensive communication between parties. The communication costs can be broken down into the amount of information (e.g. bits) exchanged and the number of communication rounds required. Different scenarios have varied efficiency requirements: in high latency networks, minimizing communication rounds is crucial, while in low bandwidth networks, the volume of information exchanged can become a bottleneck. The following efficiency aspects are considered:

- Computation - which specifies the number of operations (additions and multiplications) that are performed by all the parties

- Communication - the number of bits that are communicated between the parties
- Number of rounds - how many communication rounds protocol requires

The asymptotic complexities of these three aspects can provide an initial indication of the protocol costs, showing how the computational overhead scales with factors such as the number of parties or the number of arithmetic operations required to compute a function. However, asymptotic complexity analysis omits constant factors(independent of those previously mentioned), which are manifested in a functioning implementation of the protocol.

Additionally, some MPC protocols allow a substantial portion of computations to be performed in an off-line or preprocessing phase before the private inputs are known. These schemes typically feature a highly efficient on-line phase, enabling rapid secure function evaluation once private inputs are available. Protocols using this approach are said to follow the preprocessing model of computation.

3.4 Related work

Taking into account the examples presented in Section 1.2.3, it can be seen that the creation of a system that would change the current level of transparency in the Internet would have a positive impact on its users. Improved transparency could help build trust between users and service providers, could help users to understand how their traffic is being processed and eventually empower users to take control of their data and online experiences. As described in Chapter 3.2, improving transparency should be the result of the efforts of many network domains due to the complex nature of the digital systems used on a daily basis. The remainder of this chapter presents related work on network monitoring in setting with multiple networks and on ways of sharing data between several entities.

3.4.1 Network analysis based on different approaches.

A set of papers present applications, such as alert correlation for IDS [52], a signature distraction from network traffic [34], anomaly detection [41] or collection of performance statistic, in case of NETI@home [43], that are based upon sharing private data. All of the mentioned applications use either peer-to-peer approach or trusted-third party, as a result introducing a trade-off between privacy and utility of the solution. The work presented in this report tries to use an approach that limits the trade-off, moreover, the starting point is more general and end-user oriented.

3.4.2 Privacy-preserving protocols

In [23] authors evaluate methods and products to achieve privacy-preserving protocols that are relevant when outsourcing data storage and processing of sensitive data to the public cloud. The fundamental building block presented in this work is the use of a trusted proxy, placed between the user and the public cloud, which employs methods such as anonymization, data splitting

and cryptographic techniques to preserve privacy. Workflow involving proxy is similar to the analysis node presented in section 4.1, however many of the presented methods are not applicable in the case of work presented in this report due to the nature of the analysis.

In [33] authors introduce Secure and Privacy Preserving Intrusion Detection and Prevention for UAVS (SP-IoUAV) model, dedicated to the Internet of UAVs ecosystems. In their work authors identify the sensitive nature of the data coming from drones used for surveillance and monitoring capabilities. Therefore, the goal of the model is to present privacy-preserving mechanisms of data processing between drones in smart-cities environments, as well as, to propose an intrusion detection engine that would secure UAV networks. To achieve the goal the mechanism combines federated learning, differential privacy and secure multi-party computation. The work presented in this report is similar, as it uses MPC to process sensitive data between domains.

In CRISP [17] authors present a solution for privacy and integrity preserving computations that could be applicable in use-cases such as smart metering. In their work service providers can verify the authenticity of data that has been encrypted for offloading it to them, they can provide that service while gaining useful knowledge for their business. The presented solution uses homomorphic encryption primitives for computation and MPC-in-the-head paradigm to ensure the correctness and authenticity of the data originating from the user. The work in this report, uses MPC for data processing, however, in its system model several entities provide the data for computation as well as want to gain knowledge from the result.

3.4.3 SEPIA

SEPIA [16] is a solution based on secure multiparty computation to enable correlation and aggregation of network events such as rule triggering in Intrusion Detection System. In order to create their solution, the authors propose their own protocol within which they develop a set of basic operations. The difference between SEPIA and the presented solution is that SEPIA discusses how to use its protocol in specific scenarios, such as correlation of events, but does not address the issue of increasing transparency from the end user perspective. Furthermore, presented work additionally raises the issue of the origin of information from various sources in the network.

3.4.4 GAIA-X

The goal of GAIA-X [6] is to support the development aimed at achieving trustworthiness and sovereignty of digital infrastructure in Europe. This work also emphasises on the importance of transparency as the principle. Moreover, the GAIA-X requirements underline the need for a decentralized system capable of secure data exchange, in form of protecting both the data, as well as, services. While GAIA-X creates a conceptual framework focused mainly on cloud solutions, the presented work aims to propose a modular solution for sharing descriptions and measurements originating from infrastructure processing the traffic, to users using it.

3.4.5 SCION

The purpose of SCION [7] is to allow the user to control the route that is selected by their traffic. This is achieved by introducing the Isolation Domain Concept, which is a logical presentation of a group of autonomous systems. The user, in the SCION architecture, is informed about possible paths, which allows him to choose one that he considers appropriate. As in SCION, this work aims to improve the lack of transparency in the current Internet structure. The difference is that it additionally allows the use of general descriptions not related to traffic processing, but resulting from the infrastructure used. Additionally, the implementation of the architecture presented in SCION requires agreements between the ISP in case of creating Isolation Domain in order to determine the trust root, the adaptation of the solution can be done by independent entities.

3.4.6 Senate

Authors of Sentate [38] propose system designed to facilitate collaborative execution of analytical SQL queries among multiple parties while keeping their data private. Proposed solution is based on novel MPC decomposition protocol, that breaks cryptographic MPC computation into smaller units, allowing subsets of parties to execute certain portions of the computation in parallel. Senate's workflow is divided into three stages, Agreement, Compilation and planning, the two promising to produce a more efficient Execution stage. The Senate seeks to optimise the overall execution of the analysis by decomposing the MPC into verifiable calculations that can be performed locally. The proposed solution is applicable to the execution of analysis on databases containing data accumulated over time. The Senate differs in this respect from the work presented in this report as this work explores the use of MPC for data analysis during data collection/monitoring, to achieve Internet transparency.

3.4.7 Cerebro

Cerebro [53] is a solution that leverages multi-party computation to address the demand for large amounts of data to perform machine learning, while meeting policy regulations and not affecting business competition. Authors of Cerebro present end-to-end approach to the system design, to allow parties with complex relationship to perform computation while allowing to achieve good performance of the overall execution. Similarly to the work presented in the report Cerebro uses multi-party computation to achieve its objectives, however it differs in application use-case and characteristics resulting from it.

3.4.8 Caring about Sharing

In paper titled Caring about Sharing [28] authors investigate users perception on data sharing, by clearly presenting users with details on how, with whom, and why their data is shared. Paper shows variety in users preferences resulting from the characteristics of the data sharing between parties involved. This work is relevant as it sheds light on user preferences and acceptability, which depend on the explicitness of the relationship resulting from data sharing. Furthermore the conclusion that users caring about sharing necessitates more transparent sharing practices aligns with the overarching goal of improving Internet transparency.

3.4.9 Responsible Internet

The Responsible Internet [18] is a proposal for sovereignty and transparency in the digital world. The authors postulate the need to increase the transparency of the Internet and, in order to achieve this, they create an outline architecture consisting of two distributed systems, Network Inspection Plane and Network Control Plane. The paper presents a high-level overview of these systems, discusses the dilemmas accompanying the creation of such a system and potential directions of development. Responsible Internet is the closest to the presented work, because the proposal presented in following chapters fits into the general structure defined by Network Inspection Plane.

Chapter 4

Design of a multi-domain network telemetry system

As mentioned in the previous part of the thesis, the goal is to propose a system that would make the Internet more transparent, in other words, to propose an architecture that will enable users to gain more insight into how their network traffic is being processed. Section 4.1 of this chapter will present the proposed design for such a system. It will then be followed by the description of the PoC development process.

4.1 Design proposal

This section will describe the system design process and the accompanying decisions resulting from the goals described in the Section 2. The process will be described from top to bottom, i.e. first the part of the proposal responsible for analysing and processing data between domains will be described and then the elements responsible for providing data will be discussed.

In Figure 4.1 a high-level overview of the system is presented. As shown in Figure 4.1 the initiator of the data sharing process is the end user, who can ask a domain for two types of descriptions. The first one is a general description summarising how network traffic is processed by the domain, similar to security audits conducted in the cloud context with the Cloud Security Alliance STAR framework [21]. It may include information such as a description of services that process the traffic (DPI, Encryption, DNS), a description of infrastructure processing the traffic at the data plane and control plane level (software, hardware), the ability to perform measurements on this traffic (data plane telemetry), support for additional security functions (DNSSec,DoH), peering relations that are directly related to the path of traffic, or finally, under which jurisdiction the domain operates. Such description allows the user to verify that the service provided by the domain meets their requirements/norms/standards. The second type of description is results of measurements that the domain has the ability to conduct in relation to the flow generated by the user. As a result of this description, the user can see values such as the processing time of their traffic at different levels of granularity (from a single device to a domain summary) or, for example, the use of resources.

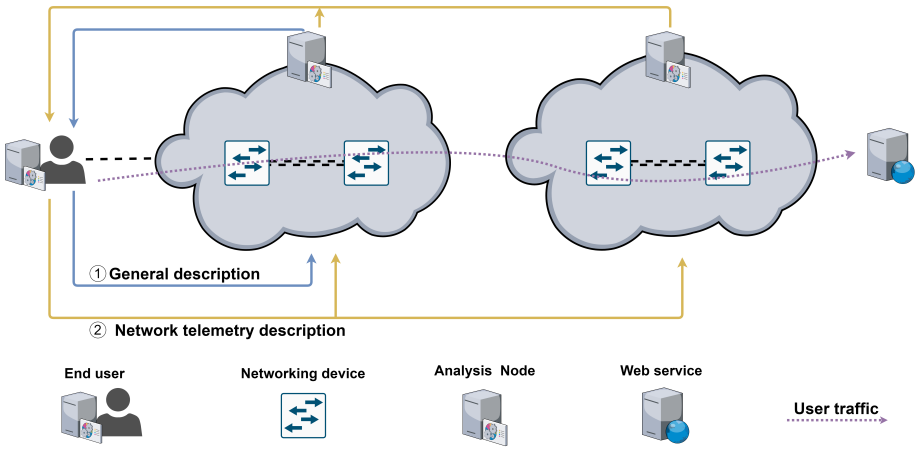


Figure 4.1: **High-level overview of the proposal with two types of description.**

As can be seen, from the two descriptions presented above, the user using such a system, would have access to a great deal of information about the functioning of the domain about which it is making the enquiry. Subsequently this may lead to negative consequences such as potential cyber attacks resulting from that knowledge. Therefore, the way in which this information is provided has an impact on the usability of the proposed system. Moreover, the user may wish to combine the data obtained by the second description in order to analyse the processing of his traffic.

In the context of providing descriptions to the user, it is necessary to define the scope of the domain from which descriptive and measurement data originate. From the point of view of the user using the system, it is important to limit the complexity of the system by minimising communication patterns (limiting number of involved parties). For example, if the end user would like to obtain a description of functioning from an operator who outsources part of his operations, it would be the operator's responsibility to obtain potential data from the entity from which he outsources operations in order to eliminate the need for the end user to make additional contact with such an entity. Therefore, it is proposed, to define domains based on the Maintenance Domain hierarchy set out in Protocol 802.1ag (Section 3.1.2). The user using the system would only communicate with a domain one degree lower in the hierarchy. Furthermore, the maximum number of entities that could participate in data exchange under the protocol would be determined by the number of domains of lower level contained within the customer level.

As for granularity of the data originating from domains, depending on the description provided, it can be different and should meet users expectations. The level of granularity could range from summaries over a whole domain or per device information. In case of some descriptors it is impossible to change the granularity of the reporting. An example is the use of device resources, which, for example, averaged over many devices in a domain, would not give any insight into the current network situation. However, the granularity could be changed for parameters such as transit delay.

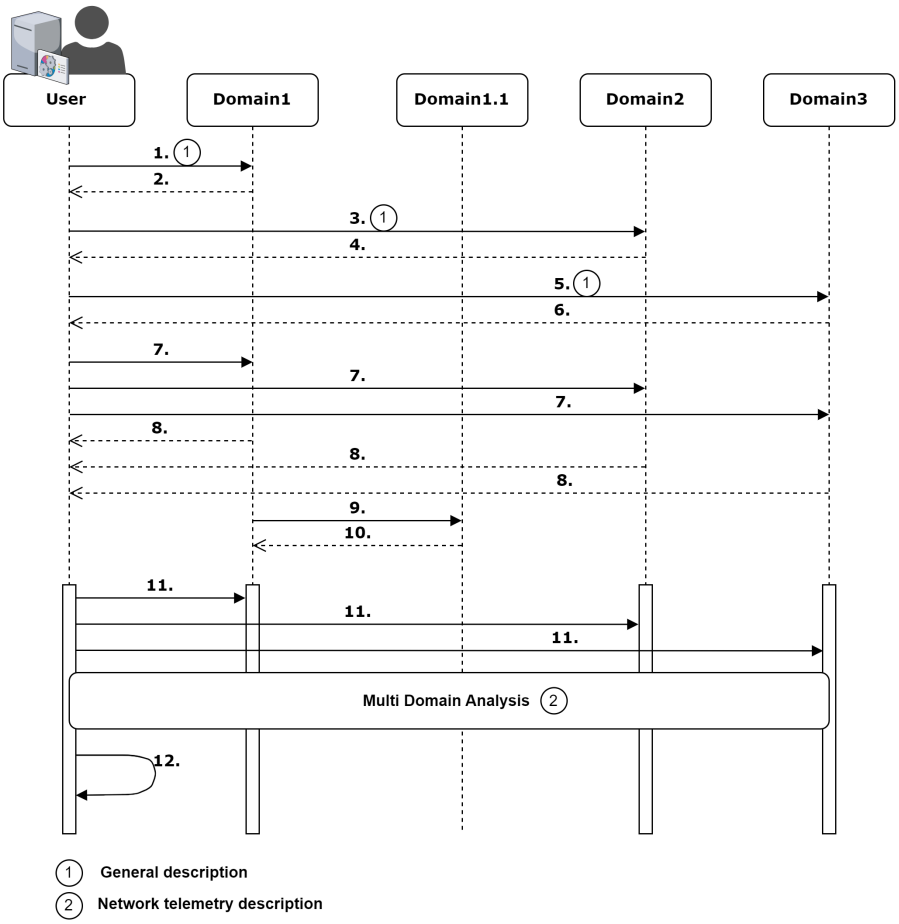


Figure 4.2: **Communication flow between user and domains aimed at performing analysis.**

Given the nature of shared data, one approach could be to design the system using a client-server architecture model, in which the user, if necessary, would obtain encrypted data directly from the server, assuming that he has appropriate permissions. This type of approach would require the establishment of trust relationship between the user and the domain. Unfortunately, in many cases, the domain providing the data cannot be sure of the user's intentions, which may make trust between them impossible to establish. Given that the trust is established there is still a risk that a malicious actor could impersonate an entity with the privileges to obtain data, consequently, accessing shared information. The challenge of trust establishment is also present when the user wants to analyse data from several domains. Then the domains not only have to trust the user, but also each other. This situation could be addressed by setting up a single entity that is trusted by the domains(trusted third party) and that performs the analysis for the user. In such case there is a risk that a said party could be compromised, as a result leaking the data.

Considering the risks and challenges described above ,it is proposed that the solution performs data sharing and analysis based on secure multi-party computation. As described in Section 3.3.3 MPC allows to carry out computation without the input data being shared with the entities participating in the analysis. As a result, entities involved in the exchange will only receive the result without the possibility of gaining access to sensitive information which could jeopardise the security of many (assuming proper design of the calculation, the result of which also does not reveal sensitive data).

Having described the types of descriptions that the user can obtain from the domains and having pre-selected the means of analysis, I would like to propose a flow of messages between the domains. In Figure 4.2 an example message flow can be seen, with following messages($M()$):

1. The user sends a message to Domain1 (e.g. his Internet provider) containing information about the network flow he wants to monitor.
   $M(Flow\ description)$

2. Domain 1 responds to the user by specifying what kind of measurement it can perform on the flow, with what accuracy, and to which domain the network flow is forwarded.
   $M(Measurement\ type, Accuracy, Next\ domain)$

3. The user, after obtaining information on the next domain, makes a query corresponding to message 1.

4. Domain 2 responds with the type of measurements it is capable of and indicates the next domain to which the network flow goes.

5. The user queries Domain 3, indicating which network flows he wants to monitor.

6. Domain 3 responds by describing its capabilities and informing the user that it is the last domain processing the network flow.

7. Being informed of all domains involved in the processing of the flow, the user wants to carry out a specific analysis on his network flow, informing the individual domains of the other entities involved in the analysis.
   $M(Flow\ description, Measurement\ type, Accuracy, Domain\ list)$

8. Each domain confirms to the user that a measurement can be performed, and provides the data necessary to initiate the relevant analysis, i.e. how the user can contact the analysis node and sends its certificate.
   $M(Acknowledgement, Node\ address, Cert)$

9. In the event that the domain delegates a part of the operation to a third party, it sends a request for the measurement of a specific flow to that third party, with the details about the flow resulting from message 7.

10. Thirds party confirms that it performs measurement and gathers telemetry data.

11. Having obtained all the details, the user provides the analysis configuration (containing a description of the monitored flow, type of analysis,

number of domains, addresses of their nodes and certificates), thereby initiating it.
$M(AnalysisConfiguration)$

12. Once the analysis has been performed for a certain period of time, the user terminates it.

As it was mentioned in the earlier part of this section, to perform successful analysis using multi-party computation, it is necessary to design the calculated functions in such a way that the result of these calculations does not include information on the system from which the data originated. In this proposal two types of functions corresponding to two types of description are described.

The first type of function is the ability to check the requirements of the user against the domain. The input data for such a function would be a list of requirements that the user has and the first of two descriptions mentioned in second paragraph. The result of such a function can help determine whether the requirements are or will not be met, which may allow the user to make a decision to change the domain being used to transfer his traffic or renegotiate the way it is processed.

The second function is to provide information on the use of the infrastructure, and also to examine measurement data that will allow to achieve the effect of end to end measurements without violating the iOAM standard (Section 3.1.5). The input data for such a function would be the measurement data and the example of result is the sum of all the delays or the argmax function, which may allow to determine the bottleneck.

The next matter is how to provide data to perform Multi Domain Analysis. The node performing the analysis could be responsible for independently querying the devices involved in the transmission of the network flow. However, such an approach would require adapting the node to "understand" variety of available measurement technologies and for it to support obtaining data from those devices. Moreover, with a large number of devices, scaling such a approach may prove problematic. I propose the introduction of an intermediate layer, which would be responsible for transmitting the request for respective data and their acquisition. Such a intermediate layer may be realised by a message broker (following model described in section 3.2.3), as it mediates communication among applications, minimising the mutual awareness which is necessary to exchange messages successfully. Message broker is used bi-directionally. Firstly, when the user expresses a desire to perform an analysis (flow 7 in Figure 4.2), the analysis node publishes in the domain information on which flow is to be monitored. Then, depending on how the network in the domain is managed, this information reaches the network devices that start the measurement. When the telemetry data is obtained, it is being published, and can be retrieved by the analysis node that subscribes to it.

4.2 Proof of Concept implementation details

The previous section aimed to bring closer the architecture that enables analysis of network measurement data in a multi domain fashion. This chapter will present details related to the implementation of the Proof of Concept and the decisions concerning the blocks on which it was based.

	Supported threat model	Supported data types	General support	Last major update
Frigate[13]	N/A	Fix and Arbitrary integer, Array	Documentation, Example code, Open source	8/2020
CBMC-GC[32]	N/A	Fixed integer, Float, Boolean, Array	Partial documentation, Example code, Open source	10/2018
SCALE-MAMBA[44]	Semi-honest, Malicious	Fixed/Arbitrary integer, Float, Array, Struct	Documentation, Online Support, Example code, Open source	03/2022
Wysteria[8]	Semi-honest	Fixed integer, Boolean, Struct	Partial documentation, Example code, Open source	10/2014

Table 4.1: **Comparison of the frameworks supporting two or more parties.**

4.2.1 Building blocks

Within the framework of the proposal described in the previous section, three main elements can be distinguished, which make up its proper functioning. These are the telemetry enabled networking device that is extended with the functionality to report that data to a message broker, the message broker itself, and a analysis node responsible for executing a specific function using MPC and obtaining data from the message broker.

The most important of these elements is the MPC framework which is the core of the whole solution. Due to the developmental character of this field and recent interest in it, many frameworks have been presented. Available frameworks can be divided into two main groups: specialised frameworks and general-purpose frameworks. The main distinguishing features of these groups are the functions offered and their performance. In the case of the former, the developers focus on the realisation of a specific function and thus try to optimise the performance of their solution due to the offered functionality [16]. The latter group aims at providing a framework that allows for any computation that can be realised within the offered features. A quality of general-purpose frameworks is more regular maintenance, which in the long run allows improving the performance of the solution based on such a framework and the protocols responsible for its operation. In addition, the protocols performed by the latter are described using high level languages. This is an important characteristic in terms of understanding the functions performed, by the involved parties, as described in section 3.3.3.

In consideration of the above, the general-purpose framework was chosen for the implementation. Among the available general-purpose frameworks, many differ in terms of the number of supported parties, the security model and the general expressibility of the high-level language that is used in the framework. As far as the number of supported parties is concerned, the description of the functions in Section 4.1 shows that the number of parties involved in the execution of these functions is greater or equal to two. In the Table 4.1 a comparison of the frameworks supporting this number of parties is presented.

As it is shown in the Table, of the available frameworks only SCALE-MAMBA supports the malicious threat model, making it the most secure framework. According to the documentation provided by the authors, this solution has no support for logical operations (resulting from lack of Boolean variables), which should eliminate it in the context of the first of the two functions described in Section 4.1. However, during the evaluation phase of that framework, it has

been found that the appropriate use of bitwise operation allows overcoming the lack of these functionalities. This meant that the function flow, which could be controlled using a Boolean variable, had to be described using operations on the basis of the supported data types and the logic operated based on the casting of the result. For example maximum value is calculated using the following expression $x - ((x - y)\& - (x < y))$, instead of returning value based on logical result of comparison. Furthermore, in case of calculating Argmax expression $((f(x_1) < f(x_2))\&x_2)\|((f(x_1) > f(x_2))\&x_1))$ is evaluated.

The next building block is the appropriate message broker. There are many open source solutions available, which differ in characteristics. Important features that should be taken into account when choosing the right solution are high availability, guaranteed delivery and delivery acknowledgement, finally how developer friendly a solution is. Among the solutions that meet these requirements, the following can be found: Apache Kafka [45], RabbitMQ [1] or ZeroMQ [3]. While these are not the only solutions that meet the requirements, they are undoubtedly the most popular (based on number of google searches for those terms), making them the most user-friendly and, consequently, lowering the adaptation threshold of the described solution. However, out of the listed, Apache Kafka differs in one feature that stands out. Namely, it has the ability to reproduce messages again. In case of many message brokers, once a message is consumed it cannot be repeated. This is an important feature for the validation of authenticity of the data presented for the computation. This characteristic allows establishing the chain events leading to a possible incident. Moreover, Kafka uses a Pull-based approach. This allows for on-demand data analysis, which increases the flexibility of the proposal. Finally, many programming languages have dedicated modules to communicate with Kafka broker, which significantly lowers the adaptation.

Final building block is telemetry enabled networking device. As described in Section 4.1, the user can request on-demand monitoring of his network flow. Technology allowing for such monitoring is SDN (Section 3.1.3), and the network in the PoC leverages that characteristic. The PoC implements a SDN controller with logic for routing traffic appropriately. In addition, the controller has been extended with a module responsible for receiving monitoring requests from message broker, translating them into OpenFlow rules, and instructing the switches to monitor the specified flows. Since network controller is also responsible for gathering the telemetry data, module's functionality is extended by another function, which sends the measurement results to the broker. An overview of a domain that has all building block is presented in Figure 4.3. At the top is Analysis Node that triggers monitoring and performs execution of MPC based multi domain calculation. Below it, is intermediate layer in the form of Apache Kafka broker. Finally at the bottom of the Figure is SDN with its controller and additional modules.

The PoC development process was carried out in the TNO Research Cloud which is based on the OpenStack platform. As can be seen in Figure 4.4, the PoC consists of three independent domains and a separate network used for the MPC communication (which is not a compulsory but was done to achieve separation). Each domain includes several telemetry enabled network devices, a message broker and analysis node that initiates on demand measurement, retrieves measurement results and is supervising MPC protocol execution. In each of the domains during its operation, when requested, the delay between switches

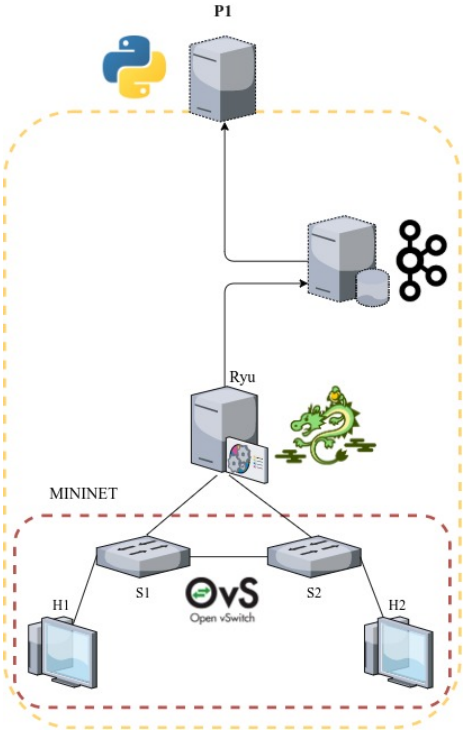


Figure 4.3: **Detailed view of domain.**

is monitored (the delay value includes the delay caused by port queuing, packet processing and link delay). This delay is then reported and made available for the multi-domain analysis. In the PoC three types of analysis were conducted on the gathered data. These types are further discussed in next section. In the PoC the two edge domains (i.e. 1 and 3) are the same in sense of used telemetry technology, while middle domain differs from them. This design step was taken to show that domains using different networking technologies can perform analysis together, highlighting modularity. Edge domains consist of Mininet network with two switches and two hosts, where switches in the network are OpenFlow-enabled Open vSwitch. The RYU SDN controller manages the network. At the beginning of the network functioning, a flow traversing through the switches is initialised, which then triggers monitoring of the link between them. The latency of the link between the switches is varying and ranges from 15 to 50 ms, as can be seen in the later analysis. Details about the delay measurement method are available in appendix B.

The middle domain is based on the FD.io VPP telemetry solution, that was developed by Mauricio Solis as a part of his graduation project [30]. In this domain the telemetry data is aggregated to the packet using an iOAM hop-by-hop header extension as it enters network namespace. This action is executed by the iOAM encapsulation node. As the packet traverses the network the iOAM transit nodes add additional telemetry information. The telemetry data is removed as it is leaving namespace with iOAM decapsulation node. The data is then polled by the telemetry collector allocated in the network and reported

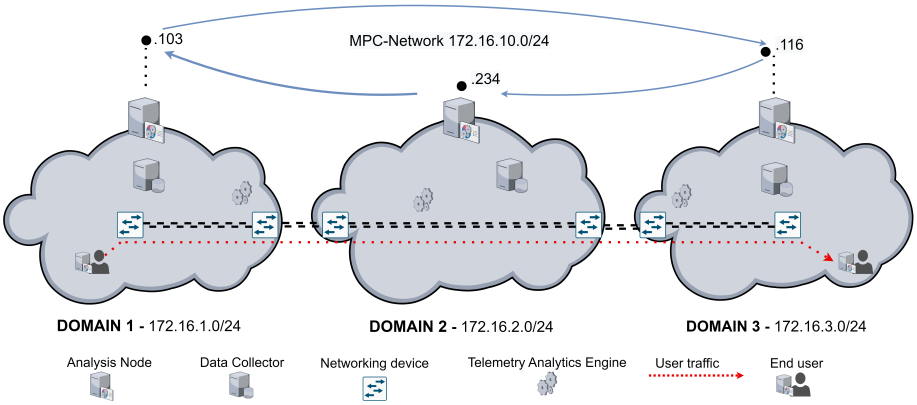


Figure 4.4: **Overview of the PoC implementation.**

to the broker. The delay is being deducted from Timestamp Trace type, which is a 32-bit value that represents timestamp in ms accuracy. The use of one of the functions described in the next section made it possible to perform a collective analysis of the data obtained from the network. What is more, the implementation allowed to place the modules responsible for sending messages with measurements in the right observation points, which allows to carry out the analysis of the system functioning in an end to end fashion, which goes beyond the data collected in the network. This means that if the MPC function provides for such an analysis, it is possible not only to monitor network conditions but also to monitor the use of resources utilised to generate network traffic (e.g. VR application).

Figure 4.5 presents the initialisation of domain1. In Figure 4.6 a console view of the MPC execution is presented. The red boxes highlight, the first iteration of the multi domain analysis. It can be also seen in the Figure that only one of the three domain participating in the calculations gets the results of it. It is intended behaviour described in the code of the protocol execution available in appendix A.

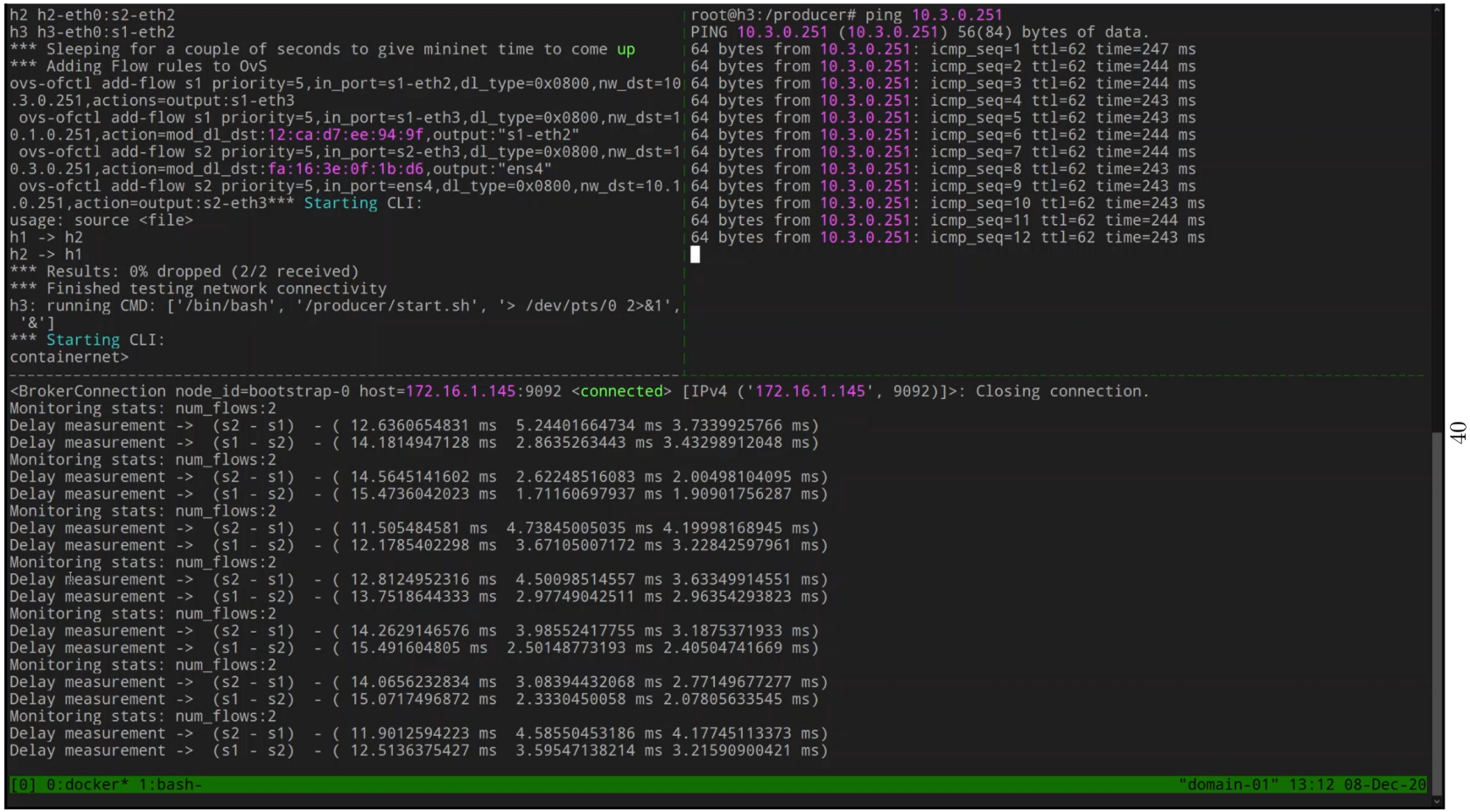


Figure 4.5: **Console view of Domain 1 initialisation. The view includes three windows. Top-left represents output of the mininet, top-right the ping initialised by end-user while the bottom view presents the results of the network telemetry.**

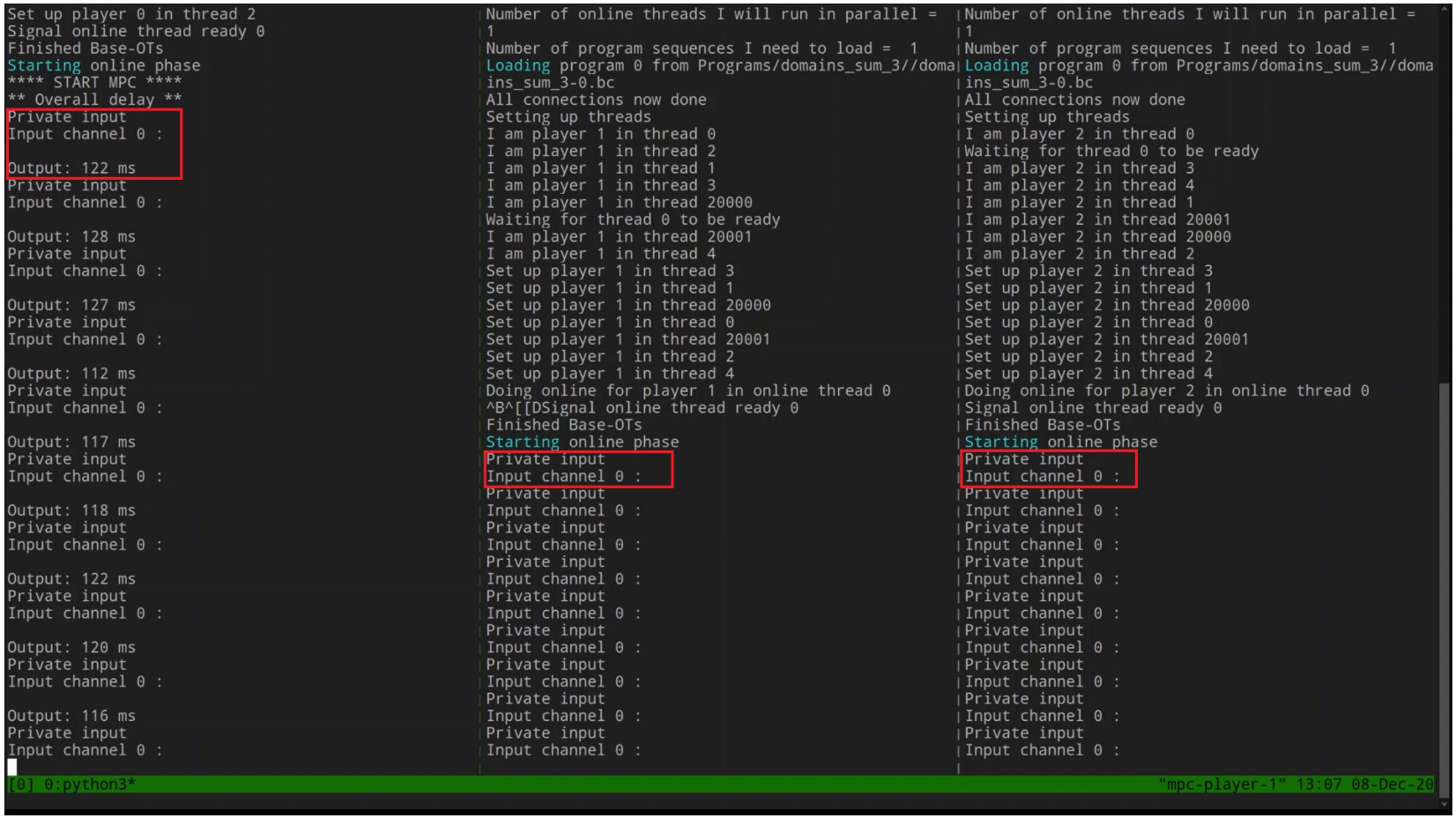


Figure 4.6: **Console view of the data analysis execution. The view includes three windows that correspond to to the domains. In the red box the first iteration of analysis is marked.**

Chapter 5

Performance Analysis

5.1 Testbed

While the original PoC was developed using Docker containers, the performance analysis was conducted with Virtual Machines that provide better resource separation. This decision was made on account of the fact that SCALE MAMBA is a framework with a high demand for resources [44]. In order to check the scalability of the architecture, a reliable resource allocation solution is required. For the test, a set in a range of three to eight virtual machines were used. The hardware resources allocated for each of the virtual machines were equal: four virtual cores E5-2683 of Intel Xeon CPU running at 2.1 GHz with 16384 KB cache and 8 GB of RAM. The network is organised in a star topology: the Virtual Machines are all connected to an instance of OvS that operates on the hypervisor. All hosts were connected to each other with a 15Gbit emulated networking interface (the bandwidth of the link was measured using iperf), and the default link latency is around 0.461 ms with a standard deviation of 0.081 ms (link latency was measured using ping). The operating system of choice on each machine was Ubuntu 18.04.4 LTS with a 4.15 Linux Kernel. Additionally, all nodes were connected to Network Time Protocol (NTP) Server - Stratum-1 resulting in a time synchronisation between the host ranging from -0.042 us to 0.142 us in relation to NTP server.

5.2 Test application

In order to conduct performance analysis, three types of data analysis, were proposed to present possible scenarios for the analysis of network data. These three types directly corresponds to the types of descriptions presented in Section 4.1. These functions are:

- Calculation of the aggregated sum of delays measured in each domain. It corresponds to the second type of description mentioned.
- Comparison of data against predetermined values. It corresponds to the first type of description, where the requirements are checked against provided data. It also represent the scenario where the measured data is checked against system-level agreement (SLA).

- Defining which of the domains participating in the protocol gives a maximum value of the targeted function (argmax).

An important element of the analysis is to determine the scalability of the proposed solution. The test applications available within SCALE-MAMBA assume that while the protocol can be executed between multiple parties, only one of them is responsible for the introduction of input data. To make the functions more realistic and correspond to the situation of analysing network telemetry data, each of the domains involved in the execution of the protocol input own data during its execution. This means that one iteration of the function first takes data from each of the parties and then conducts a calculation defined within it. Sample code of the third function is included in listing A-2.

In the analysis, the influence of the following parameters on the execution of the protocol was checked:

- Number of peers
- Network latency
- Transmission rate
- Parallelization of the input data

Their impact was determined using the following performance indicators:

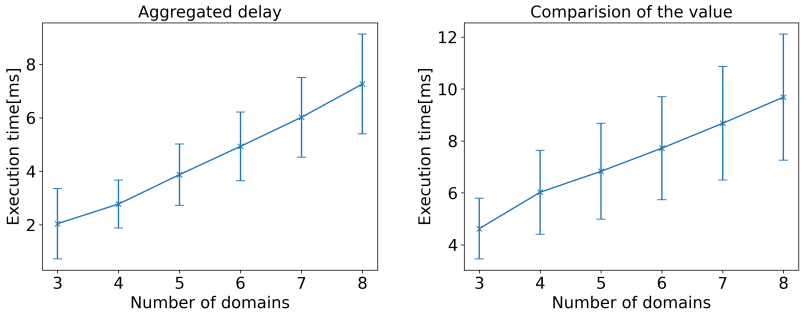
- Time to execute one protocol iteration
- Use of CPU resources
- Use of RAM

The purpose of the analysis is to determine the performance and scalability of the part of the architecture responsible for performing computation on the data, but not the part responsible for the network measurements. For this reason, in order to determine the upper boundary of the performance, each of the control nodes responsible for initialising the function and providing data to the MPC protocol will produce a synthetic data point each time the protocol asks for the next value to the function. This achieves the necessary separation between the performance of the multi domain network analysis and the performance of the underlying telemetry technology.

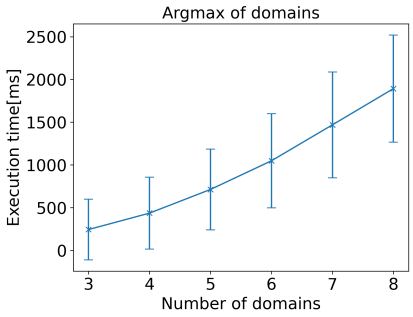
5.3 Results

This section will present the results of performance measurements. To the best of the author's knowledge, at the time of writing this thesis, the results of a similar nature allowing for comparison of the results obtained were not published. The results that are published, concerning the performance of the MPC frameworks, refer to a different nature of calculations, for example the multiplication of matrices, which makes direct comparison impossible.

Presented results are divided as follows. First, the execution times of each of the three functions are shown, depending on the number of domains involved in the computation. Then one of the functions shows the impact of network delay, transmission rate and parallelization of the input data. The final part of this section will present the online phase of the protocol data generated by the compiler and the analysis of resource usage.



(a) (b)



(c)

Figure 5.1: **The execution time in milliseconds in relation to the number of domains in case of three functions. The vertical bar represents the standard deviation.**

5.3.1 Number of domains

For the test a range of 3 to 8 domains were used in order to check change in execution time. The data presented on the figures were obtained as a result of 10000 consecutive iterations of the function. The results were then averaged.

The effect of the increasing number of domains on the execution time of three functions is presented in Figure 5.1. On each of the plots, the x-axis shows the number of domains, while the y-axis corresponds to the execution time in seconds. In case of all three functions, a higher number of domains results in a greater number of inputs and as a result, increase in number of communication rounds between the domains. The influence of these factors is accountable for increasing the execution time of a single function iteration. In case of first two functions ((a) and (b)), the relation between the number of domains and execution time is linear(in case of (a) $R^2 = 0.995$, and in case of (b) $R^2 = 0.994$, which is a positive behaviour in terms of scalability). Moreover, execution time also presents the potential for real-time analysis of the data. For the first two functions, the execution time of a single iteration is between 2 and 10 ms. This time shows that for less complex functions, MPC allows making fast calculations. On the other hand, in case of more complex functions such as argmax(c) calculation, execution time of single iteration ranges from around 250ms to

1800ms, depending on the number of domains involved. Significant increase in execution time is caused by the amount of intermediate operations performed to calculate this function. This comes as a consequence of the lack of support for boolean variables as described in section 4.2.1. As a consequence, one iteration requires a single comparison to determine the maximum value (which is then stored for the next iteration), and two comparisons and three bitwise operations to determine the argument for which the value is greater. This highlights that certain tasks can be performed using suboptimal implementation. Finally, such result shows that in case of a real system for telemetry data analysis, very complex functions can be used for the quasi-real time analysis of events for certain applications, e.g. media playback.

5.3.2 Network Latency

Next parameter, as described in section 5.2 is network latency. The test aims to present the effect network latency has on the execution time of the single iteration of the function calculating the aggregated delay. As in the previous test, the test results were obtained by repeating 10000 iterations of data analysis for the number of domains from 3 to 8, and then averaged. In order to obtain predictable values of the delay Linux tool *tc* was used. The latency in the test ranges from 0, which represents the initial latency on the testbed to 50 ms. The value of the latency corresponds to the round trip network delay between the domains. In Figure 5.2, the effect on the execution time in relation to network latency is presented. As presented in the figure, the execution time grows significantly with the increasing network latency. The execution time with an additional latency of 10 ms ranges from 16 ms to 48 ms, while a latency of 50 ms results in execution time from 77 ms to 210ms. Increase in network latency of factor five results in an execution time increase around factor four and a half. An important conclusion is that the network latency plays a key factor role in a successful deployment of MPC based solution. With the increasing latency, the ability to perform the near-real-time analysis is becoming less and less possible. For functions with much greater complexity, this effect can be even more visible.

5.3.3 Transmission rate

The transmission rate is the next parameter whose influence was tested during the analysis. As in the case of network latency, the *tc* program was used to limit the transmission rate. The transmission rate was then verified using the *iperf* tool. The impact of the transmission rate limitation is presented in Figure 5.3 (as the standard deviation of the values presented ranges from approximately 1 ms to 4 ms, a decision was made not to include them in the Figure as to not decrease its readability). As can be seen in the graph for all the number of domains involved, the execution time of a single iteration was measured for the transmission rate at 100 Mbit/s, 1 Gbit/s and 10 Gbit/s. 5.3 illustrates that there is no significant difference in the execution time of the single function iteration depending on the transmission rate.

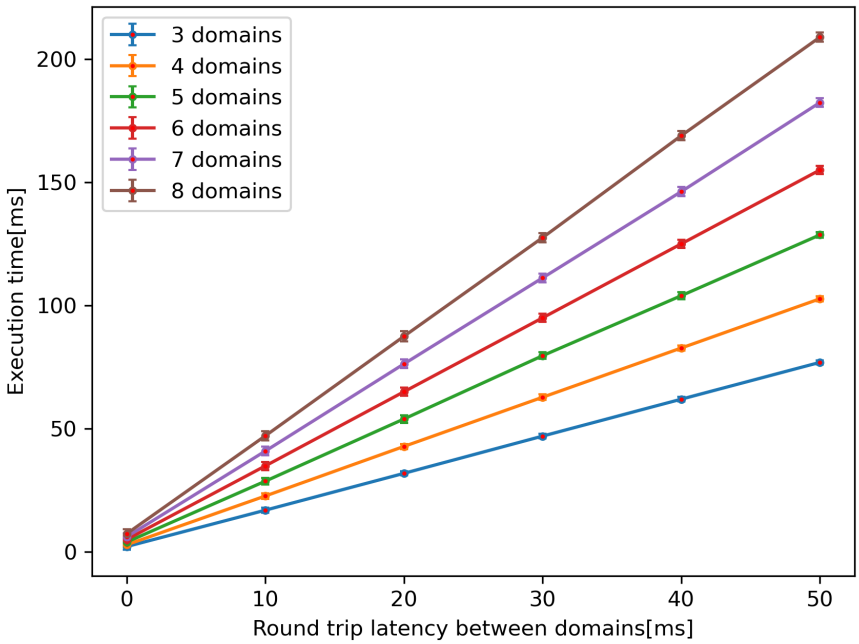


Figure 5.2: **The execution time of single iteration of function in relation to the number of domains and network latency.**

5.3.4 Parallelization of the input data

The final parameter, whose impact on the execution time of the function is examined is data parallelization. For this test, the function aggregating the delay over domains was modified in such a way that during single function iteration instead of conducting calculations on one data point it is executed on five and ten data points. The test results were obtained by repeating 10000 iterations of data analysis for the number of domains from 3 to 8, and then averaged. The effect of data parallelization in relation to function execution is presented in Figure 5.5 and Figure 5.4. As seen on the figures while the parallelization of the input data increases the execution time of a single iteration, it results in more data points being analysed during that time. For example, when compering no parallelization with the case of inputting 10 data points at the same time it can be seen that for the no parallelization case it would take on average twice as much time to perform computation on 10 data points than in the latter case. However, in cases of very time-sensitive cases, this trade-off may not be possible. However, as can be seen from the graphs, this is one form of optimisation of function execution that can be considered for environments with a negative impact on function execution time, as described in the previous sections.

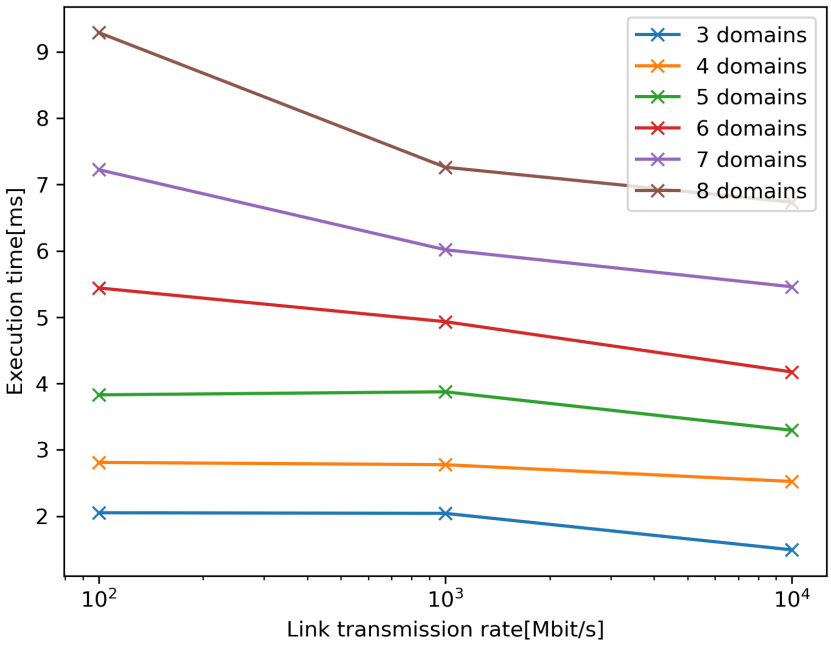


Figure 5.3: **The execution time of a single iteration of function in relation to the number of domains and transmission rate.**

5.3.5 Resource scalability

The last element examined in the analysis is the scalability in terms of resources, that is RAM and CPU usage. In case of the former, the amount of used memory is strictly defined by the compiler at the function compilation stage. For all previously performed tests, for each function the compiler returned information that the maximum amount of memory used during calculation is 8192 MB. As for CPU (4 virtual cores used), for this purpose, during the test described in section 5.2 additionally measured value was the processor usage. The results of the measurement are presented in Figure 5.6 (the large error bars presented in the Figure comes from the fact that presented value is an averaged utilisation of 4 used cores. For some operations performed during function execution only one core can be used, while other stay at 0% utilisation. This causes the standard deviation to get larger).

As shown in the figures for the first two functions, as the number of domains increases, processor usage increases with it. This is due to more communication between the domains, which takes place during function execution. For the third function, as the number of domains increases, CPU usage decreases. In order to understand this dependence, one should refer back to Figure 5.1. In the case of the argmax function, a large number of domains significantly affects the execution time of a single iteration, which also affects the amount of time when one of the domains is waiting for the others. As a result, the processor spends

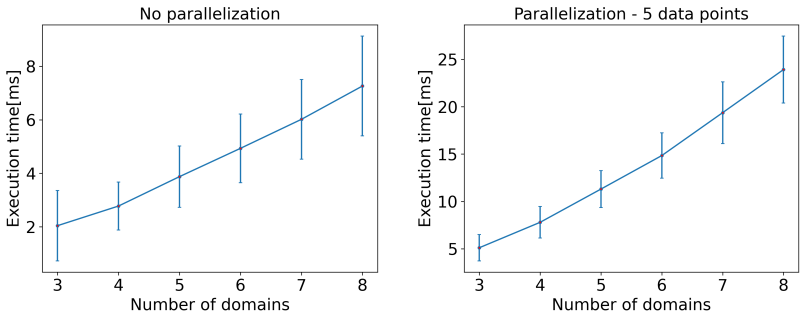


Figure 5.4: **The execution time in milliseconds of a single function iteration in relation to the number of domains in case of three types of data parallelization. The vertical bar represents the standard deviation.**

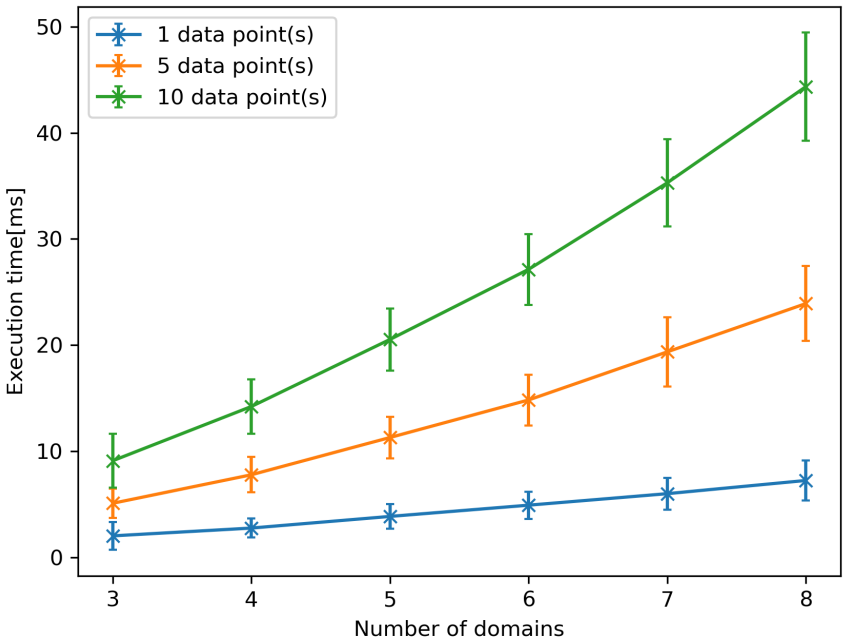


Figure 5.5: **The execution time of a single function iteration in relation to the number of domains in case of three types of data parallelization presented on a single plot.**

more time in idle lowering overall usage number. When analysing these graphs, it should be remembered that the presented values concern the execution of a function which analyses a single data stream involving certain domains. Analysis of many data streams simultaneously (assuming for each a dedicated function is used) would require significantly more resources. This is due to the fact that, for each function, the involved domains must go through a protocol initialisation step and maintain in memory the necessary data during its execution. Such dependence shows that the presented solution comes at the price of limited scalability. As a result, there is a risk that this solution will only be viable for those for whom obtaining the knowledge resulting from the analysis is justifiable in relation to the potentially high costs incurred by the high use of the resources.

5.3.6 Conclusion

As shown in the previous sections, the performance of an MPC-based solution varies depending on the prevailing network conditions and the complexity of the analysed functions. For less complex functions it is possible to use them in real systems, assuming that network conditions do not have too negative impact on performance (significant delays). As a consequence, at the time of writing this thesis, the presented solution could be utilised by entities for which the trade-off between the use of resources and, consequently, the costs incurred, is justified

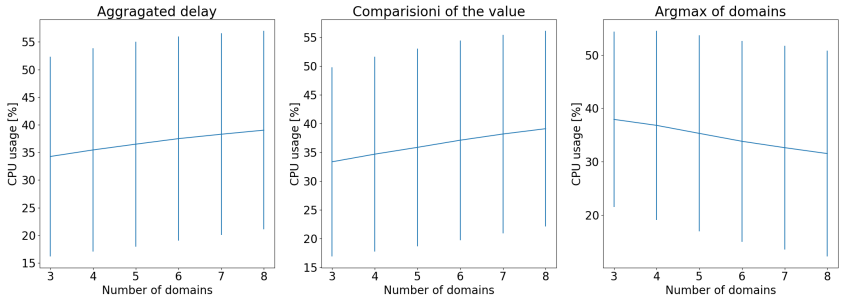


Figure 5.6: **The CPU usage, averaged over time of execution, in relation to the number of domains in case of three functions: (a) aggregated delay, (b) comparison of the value, (c) argmax of domains.**

by the insight gained from the analysis. However, for scenarios with a different threat model, it could be possible to propose a different, more resource scalable approach.

Chapter 6

Conclusions and Future Work

With the growing demand for fast deployable, customizable and quickly adaptable services, it is becoming necessary to supervise them and the resources associated with them. The presented proposal realizes a multi-domain diagnostic system, and the conclusions of this project, as well as, future work are presented in this section.

6.1 Conclusion

In order to draw conclusion the step-by-step process derived from research objective presented in introduction is going to be revisited.

- As presented in section 3.4 there are many solutions showing the benefits of joint analysis of network data. Some of them use TTP or P2P approach. However, a multi-domain diagnostic system based on these approaches introduces tradeoffs in terms of usability vs security/privacy. SEPIA [16] is an MPC based solution. The solution presented by the authors uses a custom-made framework sewn to specific needs, which significantly limits the possibility of extending this solution with additional functionality.

- The issues presented in the paper are researched by the public. GAIA-X [6], Responsible Internet [18] sketch the need to improve transparency on the Internet in the context of trustworthiness and sovereignty. Presented work is complementary in view of the efforts made to improve transparency in the Internet while maintaining privacy and competitive balance.

- The presented solutions put emphasis on maintaining competitiveness, which allows to convince potential users to adopt it. In this work, an important role is played by the flexibility of the solution, which allows the acquisition of measurement data from multiple observation points, allowing to achieve the effect of end to end monitoring and allowing to customize the solution according to the needs.

- In order to present a working prototype, the containerization available in Linux using Docker was used. This prototype was later expanded by creating independent domains within the Research Cloud provided by TNOs. The multi-domain nature of the solution was presented by creating independent networks in which different technologies were used for the purpose of network telemetry that were then used as input data for analysis.

- In order to analyze the upper performance limit of the solution, tests were performed which checked the time of one iteration of data analysis with variable factors such as: variable number of domains, various functions analyzing data and variable network parameters used for communication between the domains. The performance analysis showed that the proposed solution comes at the price of high resource utilization.

In conclusion, the presented work shows the possibility of implementing a multi-domain diagnostic system, whose main objective is to maintain the privacy of the involved systems and of analyzed data. The efficiency of this solution is limited depending on the function used to analyze data and the prevailing conditions in the network used for communication between domains. The presented work shows one of the possible approaches to the implementation of a multi-domain system and related aspects.

6.2 Future work

On the outcome and the course of the work carried out, the following activities were identified as part of future work.

- The performance analysis of the presented PoC was used to determine the upper performance boundry. In order to analyse the performance in real life system it would be necessary to conduct tests based on real telemetry solutions.

- As with the previous point, tests involving different physical networks (representing different domains), where the network conditions would not be static, would have to be carried out in order to check the actual performance of the solution.

- As presented in the results, the resource-intensive nature of the proposed solution means that increasing the number of domains in the system can be a costly. Hence, ways to improve the scalability of the system to apply it to complex analyses would need to be explored.

- The presented work indicates specific ways of processing data in order to analyse information from multiple domains. The type of data analysed is different from typical MPC based solutions. In view of the above, a potential direction could be to further examine the optimisation of the data analysed in the context of reducing resource use.

Furthermore current efforts of the academic community in the MPC field are partly focused on addressing the shortest path problem on a graph whose representation is private according to the concepts available in the MPC. The

solution presented is a step that will enable future contributions to the protocols for solving this problem. As a consequence, one of the possible future directions is to extend the functionality with new analysis function, which will be based on future protocols.

Bibliography

[1] Rabbitmq. https://www.rabbitmq.com/. Last accessed: Apr. 15, 2020.

[2] traceroute(8) - linux manual page. https://man7.org/linux/man-pages/man8/traceroute.8.html. Last accessed: Aug. 18, 2020.

[3] Zeromq. http://zeromq.org/. Last accessed: Apr. 17, 2020.

[4] OpenFlow switch Specification. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf, 2015. Last accessed: Aug. 10, 2020.

[5] Eu ict industrial policy: Breaking the cycle of failure. https://www.enisaeuropa.eu/publications/enisa-position-papers-and-opinions/eu-ict-industry-consultation-paper, 2019.

[6] Gaia-x. https://www.data-infrastructure.eu/GAIAX/Redaktion/EN/Publications/gaia-x-the-european-project-kicks-of-the-next-phase.pdf, 2020.

[7] R.Reischuk-L.Chuat A.Perrig, P.Szalachowski. *SCION: a secure Internet architecture*. Springer, 2017.

[8] A.Rastogi. Wysteria: A programming language for generic, mixed-mode multiparty computation. https://bitbucket.org/aseemr/wysteria/wiki/Home, 2014. Last accessed: Mar. 26, 2020.

[9] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A. Reuter, and Martin Strand. A guide to fully homomorphic encryption. Cryptology ePrint Archive, Paper 2015/1192, 2015. https://eprint.iacr.org/2015/1192.

[10] Jeffrey R Ballard, Ian Rae, and Aditya Akella. Extensible and scalable network monitoring using opensafe. *Inm/wren*, 10, 2010.

[11] Amos Beimel. Secret-sharing schemes: a survey. In *International Conference on Coding and Cryptology*, pages 11–46. Springer, 2011.

[12] George Robert Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318. IEEE, 1979.

[13] B.Mood. Frigaterelease. http://www.powercastco.com, 2020. Last accessed: Sept. 30, 2020.

[14] R. Braden. Requirements for internet hosts - communication layers, 10 1989.

[15] Frank Brockners, Tal Mizrahi, and Shwetha Bhandari. Data fields for in-situ oam, 07. Last accessed: Aug. 13, 2020.

[16] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1(101101), 2010.

[17] Sylvain Chatel, Apostolos Pyrgelis, Juan Ramón Troncoso-Pastoriza, and Jean-Pierre Hubaux. Privacy and integrity preserving computations with CRISP. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2111–2128. USENIX Association, August 2021.

[18] R.Holz-F.Kuipers Fernando J.Xue M.Jonker J.de Ruiter A.Sperotto R.Rijswijk-Deij G.Moura others C.Hesselman, P.Grosso. A responsible internet to increase trust in the digital world. *Journal of Network and Systems Management*, 28(4):882–922, 2020.

[19] B. Claise. Cisco systems netflow services export version 9, 10. Last accessed: Aug. 15, 2020.

[20] Alan Clark. Guidelines for considering new performance metric development, 10 2011. Last accessed: Aug. 15, 2020.

[21] Cloud Security Alliance. Csa star framework. https://cloudsecurityalliance.org/star/levels/. Last accessed: Dec. 17, 2020.

[22] B. Waters D. Boneh, A. Sahai. Functional encryption: Definitions and challenges. *TOC/CIS groups, LCS, MIT*, 1996.

[23] Josep Domingo-Ferrer, Oriol Farràs, Jordi Ribes-González, and David Sánchez. Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Computer Communications*, 140-141:38–60, 2019.

[24] I. Benbasat G.C Moore. Development of an instrument to measure the perceptions of adopting an information technology innovation. *Information Systems Research*, 2(3):192–222, 1991.

[25] Tim Kindberg George Coulouris, Jean Dollimore. *Distributed systems: concepts and design*. Pearson Education, 2013.

[26] W. Hardaker. Transport layer security (tls) transport model for the simple network management protocol (snmp), 07. Last accessed: Aug. 17, 2020.

[27] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer–efficiently. In *Annual international cryptology conference*, pages 572–591. Springer, 2008.

[28] Bailey Kacsmar, Kyle Tilbury, Miti Mazmudar, and Florian Kerschbaum. Caring about sharing: User perceptions of multiparty data sharing. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 899–916, Boston, MA, August 2022. USENIX Association.

[29] R. J. Klein L. G. Tornatzky. Innovation characteristics and innovation adoption-implementation: A meta-analysis of findings. *IEEE Transactions on Engineering Management*, EM(29):28–45, 1982.

[30] Solis Jr Mauricio. Further implementation of ioam using ipv6 in fd.io vector packet processor. Master's thesis, Department of Electrical and Computer Engineering, Technische Universität Kaiserslautern, 2020.

[31] Al Morton. Active and passive metrics and methods (with hybrid types in-between), 05 2016. Last accessed: Aug. 22, 2020.

[32] N.Buescher. Powercast hardware. `http://www.powercastco.com`, 2018. Last accessed: Mar. 24, 2020.

[33] Ernest Ntizikira, Wang Lei, Fahad Alblehai, Kiran Saleem, and Muhammad Ali Lodhi. Secure and privacy-preserving intrusion detection and prevention in the internet of unmanned aerial vehicles. *Sensors*, 23(19), 2023.

[34] Janak J Parekh, Ke Wang, and Salvatore J Stolfo. Privacy-preserving payload-based correlation for accurate malicious traffic detection. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*, pages 99–106, 2006.

[35] V. Paxson, J. Mahdavi, M. Mathis, and G. Almes. Framework for ip performance metrics, 05 1998. Last accessed: Aug. 16, 2020.

[36] Peter Strempel. New report on european digital infrastructure and data sovereignty. `https://www.eitdigital.eu/newsroom/news/article/new-report-on-european-digital-infrastructure-and-data-sovereignty/`, 2020.

[37] Peter Phaal and Sonia Panchen. sflow - packet sampling basics.

[38] Rishabh Poddar, Sukrit Kalra, Avishay Yanai, Ryan Deng, Raluca Ada Popa, and Joseph M. Hellerstein. Senate: A Maliciously-Secure MPC platform for collaborative analytics. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2129–2146. USENIX Association, August 2021.

[39] J Postel. Internet control message protocol, 09 1981.

[40] O. Goldreich M.Naor R. Canetti, U. Feige. Adaptively secure multi-party. *Proceedings of Theory of Cryptography Conference*, TCC, 2011.

[41] Haakon Andreas Ringberg and Jennifer Rexford. *Privacy-preserving collaborative anomaly detection.* Citeseer, 2009.

[42] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[43] Charles Robert Simpson and George F Riley. Neti@ home: A distributed approach to collecting end-to-end network performance measurements. In *International Workshop on Passive and Active Network Measurement*, pages 168–174. Springer, 2004.

[44] Smart, N. Scale-mamba software. `https://homes.esat.kuleuven.be/~nsmart/SCALE/`. Last accessed: Aug. 15, 2020.

[45] The P4 Language Consortium. Apache kafka documentation. `https://kafka.apache.org/documentation/`. Last accessed: Aug. 13, 2020.

[46] Amin Tootoonchian, Monia Ghobadi, and Yashar Ganjali. Opentm: traffic matrix estimator for openflow networks. In *International Conference on Passive and Active Network Measurement*, pages 201–210. Springer, 2010.

[47] Brian Trammell and Benoit Claise. Specification of the ip flow information export (ipfix) protocol for the exchange of flow information, 09 2013. Last accessed: Aug. 12, 2020.

[48] Niels LM Van Adrichem, Christian Doerr, and Fernando A Kuipers. Opennetmon: Network monitoring in openflow software-defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8. IEEE, 2014.

[49] Daniël Van Den Berg, Rebecca Glans, Dorian De Koning, Fernando A Kuipers, Jochem Lugtenburg, Kurian Polachan, Prabhakar T Venkata, Chandramani Singh, Belma Turkovic, and Bryan Van Wijk. Challenges in haptic communications over the tactile internet. *IEEE Access*, 5:23502–23518, 2017.

[50] Yaacov Weingarten, Nurit Sprecher, Elisa Bellagamba, and Tal Mizrahi. An overview of operations, administration, and maintenance (oam) tools, 06 2014. Last accessed: Aug. 20, 2020.

[51] W.Neelen, R.van Duijn. Hacking traffic lights defcon 2020. `https://media.defcon.org/DEF%20CON%2028/DEF%20CON%20Safe%20Mode%20presentations/DEF%20CON%20Safe%20Mode%20-%20Wesley%20Neelen%20%26%20Rik%20van%20Duijn%20-%20Hacking%20Traffic%20Lights.pdf`, 2020. Last accessed: Sep. 28, 2020.

[52] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global intrusion detection in the domino overlay system. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2003.

[53] Wenting Zheng, Ryan Deng, Weikeng Chen, Raluca Ada Popa, Aurojit Panda, and Ion Stoica. Cerebro: A platform for Multi-Party cryptographic collaborative learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2723–2740. USENIX Association, August 2021.

Appendix A

MPC details

This appendix contains the code which is part of the proof of concept implementation.

```
4
RootCA
7
10.0.1.192
Player0.crt
10.0.1.141
Player1.crt
10.0.1.105
Player2.crt
10.0.1.175
Player3.crt
10.0.1.196
Player4.crt
10.0.1.29
Player5.crt
10.0.1.201
Player6.crt
2
9223372036855103489
1
```

Listing A.1: **A sample setup file used to configure the SCALE-MAMBA framework**

```
print_ln('**** START MPC ****')
print_ln('**** Max delay ****')

def maximum(a,b):
    return a - ((a - b) & (a < b))

def argmax (a,b):
    return (a[0] - ((a[0] - b[0]) & (a[0] < b[0])),
        ((a[0] < b[0]) & b[1]) | ((a[0] > b[0]) & a[1]))
```

```
@while_do(lambda x: x < 1, 0)
def cal_max_delay(i):

    a = (sregint(sint.get_private_input_from(0)),sregint(0))
    b = (sregint(sint.get_private_input_from(1)),sregint(1))
    c = (sregint(sint.get_private_input_from(2)),sregint(2))

    # max = maximum(a[0],b[0])
    # max = maximum(max,c[0])

    max = argmax(a,b)
    max = argmax(max,c)

    print_ln('Output: value of max delay  is caused
             by Player %s' , max[1].reveal())


    return i + 1

if __name__ == '__main__':
    cal_max_delay(0)
```

Listing A.2: **A code of a MPC function calculating argmax of presented values. This function is writen for three domain environment**

Appendix B

Measurement method

This appendix describes the active measurement method used to acquire data in domains 1 and 3 of PoC, that served as an input to the MPC protocol.

As seen in Figure B.1, making a single measurement requires three activities. First, the controller measures the time between sending a packet and receiving a return message when requesting statistics from switches that are the beginning and the end of a measured link. This measurement allows to estimate the delay between the controller and each of the switches. Then a special measurement packet is sent to the network, which differs by a specific value in the header. This packet goes from the controller to the first switch, then to the second through the measured link, finally reaching the controller after being redirecting by the second switch. The time measured between sending the packet and receiving it again, after subtracting the previously mentioned delays to the switches, allows to

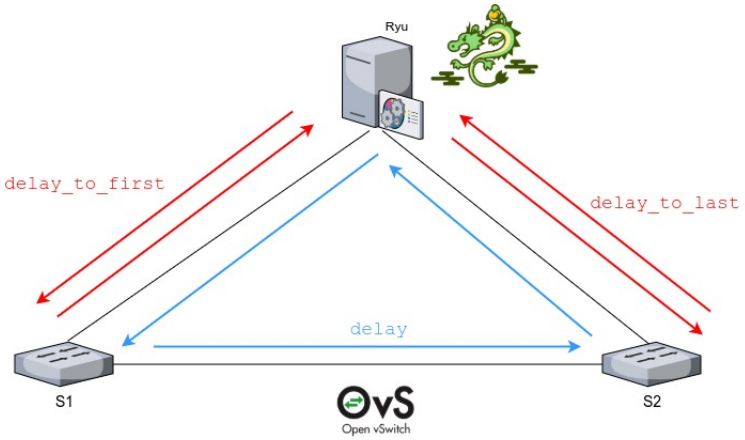


Figure B.1: **Active measurement method using probe packet in SDN based network.**