COoperative Cyber prOtectiON for modern power grids

D1.1 Control, Measurement and Monitoring Properties

Distribution Level	PU
Responsible Partner	University of Glasgow (UGLA)
Prepared by	Filip Holik (UGLA), Mingwei Li (UGLA), Awais Aziz Shah (UGLA), Dimitrios Pezaros (UGLA), Georgios Kryonidis (AUTH), Hymanshu Goyel (TU Delft), Vetrivel S. Rajkumar (TU Delft), Alfan Preseka (TU Delft), Alex Stefanov (TU Delft), Luna Moreno Diaz (ING), David Senas Sanvicente (ING)
Checked by WP Leader	Angelos Marnerides (UCY)
Verified by Reviewer #1	Angelos Marnerides (UCY) 15/07/2024
Verified by Reviewer #2	Jose Maria Maza Ortega (USE) 15/07/2024
Approved by Project Coordinator	Angelos Marnerides (UCY) 15/07/2024

Co-funded by the European Union

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Directorate General for Communications Networks, Content and Technology. Neither the European Union nor the Directorate General for Communications Networks, Content and Technology can be held responsible for them.

Deliverable Record

Planned Submission Date	17/07/2024
Actual Submission Date	15/07/2024
Status and version	FINAL

Version (Notes)	Date	Author(s)	Notes
0.1 (Draft)	17/06/2024	Filip Holik (UGLA), Awais Aziz Shah (UGLA), Dimitrios Pezaros (UGLA)	ToC, initial structure, work allocation
0.2 (Draft)	26/06/2024	Filip Holik (UGLA), Mingwei Li (UGLA), Georgios Kryonidis (AUTH), Himanshu Goyel (TU Delft), Vetrivel S. Rajkumar (TU Delft), Alfan Presekal (TU Delft) Alex Stefanov (TU Delft)	EPES communication protocols, chapters 5, 6
0.3 (Draft)	30/06/2024	Filip Holik (UGLA), Mingwei Li (UGLA), Luna Moreno Diaz (ING), David Senas Sanvicente (ING)	First draft
0.4 (Draft)	04/07/2024	Filip Holik (UGLA), Mingwei Li (UGLA)	Second draft
0.5 (Draft)	07/07/2024	Filip Holik (UGLA)	Final draft for reviewers
0.6 (Draft)	12/07/2024	Filip Holik (UGLA), Jose Maria Maza Ortega (USE), Awais Aziz Shah (UGLA), Angelos Marnerides (UCY)	Pre-final version
1.0 (Final)	15/07/2024	Filip Holik (UGLA)	Final version

Contents

List of Figures

List of Tables

Definition of Acronyms

μNF Micro Network Function. 12, 17, 27–29, 31, 32, 43–45, 47

AD Anomaly Detection. 10–13, 22, 28, 29, 35, 36, 39, 45, 47

APDU Application Protocol Data Unit. 38, 39

API Application Programming Interface. 19, 24, 43, 44

AS Ancillary Services. 20, 21

ASDU Application Service Data Unit. 39

ASIC Application-Specific Integrated Circuit. 25

AUTH Aristotle University of Thessaloniki. 13, 47

BMv2 Behavioral Model version 2. 26

CNN Convolutional Neural Networks. 44, 45

COCOON COoperative Cyber prOtectiOn for modern power grids. 10–13, 19, 21, 28, 32, 33, 35–37, 44–47

COMML Control Measurement and Monitoring Layer. 10, 12–14, 17, 19, 20, 22, 27, 28, 35, 43–47

CPN COCOON Programmable Node. 10–14, 17, 19, 20, 22–24, 26–35, 38, 43–45, 47

CPU Central Processing Unit. 27, 35

CSL Cybersecurity Services Layer. 10–14, 19, 22, 28, 29, 32, 33, 35, 43–47

DDoS Distributed Denial of Service. 46

DL Deep Learning. 45

DNN Deep Neural Network. 46

DoS Denial of Service. 39

DPDK Data Plane Development Kit. 30

DRL Deep Reinforcement Learning. 10, 45

DSO Distribution System Operator. 18, 20

eBPF extended Berkeley Packet Filter. 12, 26–32, 43, 47

EPES Electrical Power and Energy Systems. 10, 12–20, 33, 35, 37, 38, 43, 44, 46, 47

EU European Union. 20

FDI False Data Injection. 39

FDII False Data Injection Identification. 10, 13, 22, 28, 35, 42, 47

GA Genetic Algorithm. 44

GOOSE Generic Object-Oriented Substation Event. 13, 17, 18, 20, 35, 36, 38, 39, 45

GPDU GOOSE Protocol Data Unit. 38

HMI Human Machine Interface. 14

I/O Input / Output. 43

IDPS Intrusion Detection Prevention System. 27

IDS Intrusion Detection System. 23, 45, 46

IEC104 IEC 60870-5-104. 13, 18, 20, 36, 37, 40–42

IED Intelligent Electronic Device. 14, 17, 30, 34

ING Ingelectus. 13

IOL Instrumentation and Orchestration Layer. 12, 13, 19, 27, 28, 44, 47

IP Internet Protocol. 15, 23, 24, 34–36, 40, 42

ISO/OSI International Organization for Standardization / Open Systems Interconnection. 10, 14–17, 47

IT Information Technology. 10, 17

LAN Local Area Network. 15, 16, 22, 23, 36

LSTM Long Short Term Memory. 44, 45

MAC Medium Access Control. 12, 15, 18, 23–25, 31, 36

ML Machine Learning. 10, 12, 44, 45

MU Merging Unit. 14, 30

NFV Network Function Virtualization. 12, 19, 26, 27, 47

NIC Network Interface Cards. 30

OT Operational Technology. 10, 11, 14, 16, 17, 19, 25, 27, 30, 34, 47

OvS Open vSwitch. 24, 30

P4 Programming Protocol-independent Packet Processors. 12, 23, 25–27, 30, 31

PDP Programmable Data Plane. 12, 19, 22, 24, 31, 32, 34–39, 41–43, 47

PDU Protocol Data Unit. 18

PMU Phasor Measurement Units. 44

POI Point of Interconnection. 22

PPC Power Plant Controller. 21, 22

PV Photovoltaic. 20–22

QoS Quality of Service. 16

RAM Random-Access Memory. 27

RL Reinforcement Learning. 44–46

RNN Recurrent Neural Networks. 45

RTT Round Trip Time. 46

RTU Remote Terminal Unit. 18, 21, 30

SARSA State-Action-Reward-State-Action. 46

SBC Single Board Computer. 12, 26, 30

SCADA Supervisory Control and Data Acquisition system. 10, 16, 18, 36, 44

SDN Software Defined Networking. 12, 19, 23–25, 27, 30, 47

SV Sampled Values. 13, 17, 18, 20, 35, 39

SVM Support Vector Machine. 44

TCP Transmission Control Protocol. 16, 18, 21, 22, 35, 37, 38, 40, 42

TCP/IP Transmission Control Protocol / Internet Protocol. 10, 14–16, 18, 21, 36, 37, 42, 47

TTL Time To Live. 23

TUD TU Delft. 13, 47

UDP User Datagram Protocol. 16, 37

UGLA University of Glasgow. 13, 46

USE University of Sevilla. 13, 47

VLAN Virtual Local Area Network. 23, 44

VM Virtual Machine. 27

VNF Virtual Network Functions. 27, 28

VRRP Virtual Router Redundancy Protocol. 36

WAN Wide Area Network. 15, 23, 36

XDP Express Data Path. 30

Executive Summary

The continuing digitalization of Operational Technology (OT) networks, including Electrical Power and Energy Systems (EPES), has brought significant advantages in reducing the deployment and operational costs, ease of maintenance, and improved management, but at the same time it also introduces new challenges in terms of network security, due to the fact that cyber resilience has not been the consideration in the initial design stages. Moreover, network operators are reluctant to retrofit state-of-the-art cyber security solutions in their existing OT network infrastructures, realizing associated risks, especially the increased complexity of deploying the cyber security solutions that could negatively influence network availability, which is the prime priority. This could also result in a lack of Information Technology (IT) network monitoring as Supervisory Control and Data Acquisition system (SCADA) is devoted specifically to monitor the physical processes and not the underlying IT network.

The COoperative Cyber prOtectiOn for modern power grids (COCOON) project is addressing this lack of measurement, monitoring and control with the aim of improving the protection of OT networks, specifically EPES networks. This is achieved by implementing a programmable networking device i.e., COCOON Programmable Node (CPN), which will run customized network functions in a safe and secure manner. Such a solution allows implementation of various high-level services such as Anomaly Detection (AD), False Data Injection Identification (FDII) and Deep Reinforcement Learning (DRL)-based attack mitigation transparently in the network. The CPN implemented on a single networking device, such as a switch, does not require any modifications in the network topology and therefore retrofit the security features in the existing OT networks, while its programmability makes the solution future proof with the ease to modify or replace the supported network services.

This deliverable, D1.1: *Measurement, Monitoring & Control*, focuses on the lowest layer of the project i.e., the computer network, which fundamentally forms the foundation for the CPN. Firstly, the baseline concepts of International Organization for Standardization / Open Systems Interconnection (ISO/OSI) and Transmission Control Protocol / Internet Protocol (TCP/IP) models are highlighted in general and in relation with EPES communication protocols used in the project. Terminology of these models will be used thorough the rest of the project, since all services implemented in the CPN needs to interact with network parameters from various ISO/OSI layers. Thus, ensuring that the project partners will use a common terminology which will improve team effectiveness and reduce errors.

Secondly, the architecture and properties of the lowest CPN layer, the Control Measurement and Monitoring Layer (COMML), are presented based on the requirements gathered from the project partners. The detailed description of the technologies to be used for COMML implementation are also described herein.

Finally, the last two sections are devoted on describing the COMML measurement, monitoring and instrumentation properties. The section focusing on the COMML measurement and monitoring properties presents concepts of passive and active network measurement and sheds light on the general and EPES communication protocols in terms of their structure and parameters associated with the COMML. COMML parameters are crucial for the formation and instrumentation of high-level services present within the COCOON Cybersecurity Services Layer (CSL). Hence, the CPN algorithmics section elaborates on how these services can be used for managing the network and examples of Machine Learning (ML) and DRL-based algorithms suitable for the COCOON scenarios.

1 Introduction

The COCOON project follows a bottom-up, systems-oriented methodology, and this deliverable, first within WP1, is laying out the necessary background information and concepts related with the lowest layer - the OT network. The network is responsible for providing reliable and fast communication service between all devices, including smart power grid devices. Unfortunately, computer networks were not designed with security in mind and retro fitting security requires various middle box devices and tedious configuration. The COCOON project takes a different approach by introducing a programmable network device called the CPN.

CPN will provide an abstraction layer between low-layer network operations and high-layer cyber security services. The original high-level CPN architecture as described in D4.1: *COCOON Development Blueprint* is shown in Figure 1.1. This architecture shows the conceptual functionality of the CPN divided into three layers. Note that this architecture does not reflect a real implementation on devices - top two layers can be either deployed locally on the CPN, or remotely on a server.

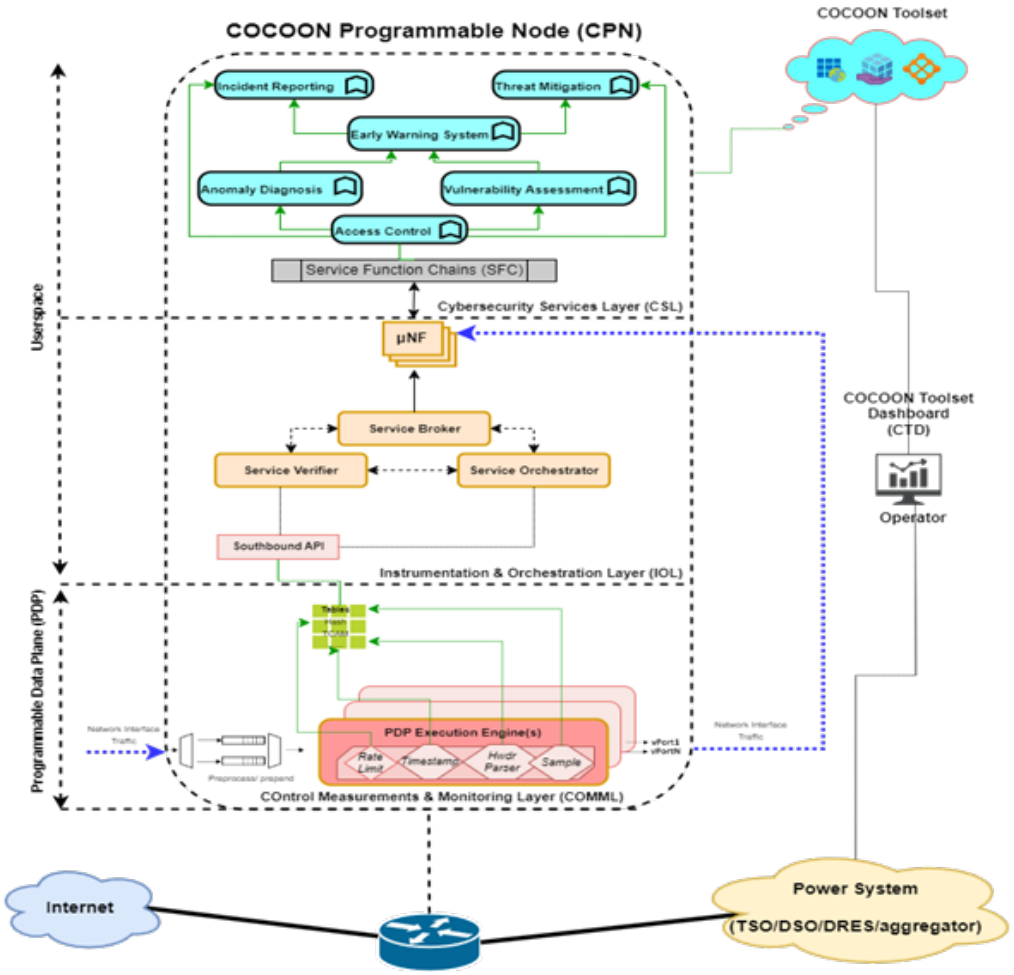

Figure 1.1: Initial architecture of CPN

The top layer, CSL, hosts services responsible for network monitoring and control. This will include services such as AD and threat mitigation. Services can be written in any programming

language and can run in different virtual machines, or containers. The next layer, Instrumentation and Orchestration Layer (IOL), translates network operation requirements (intents) from CSL to Micro Network Function (μNF). For example, the AD service will need real-time information about traffic flows including source and destination Medium Access Control (MAC) addresses. The IOL will decompose this requirement into a micro network function(s) which will log this information. Depending on the logic complexity, the service might need to be decomposed to multiple functions. In the AD example, it could be one for collecting timestamps and MAC addresses, another one for collecting data payload values, and finally the last one for forwarding the message further in the network.

Finally, the lowest layer, COMML receives and installs the μNF into its Programmable Data Plane (PDP) pipeline and performs corresponding network traffic operations such as send and receive traffic. This will be implemented with the use of flexible PDP technology which will support deployment on diverse set of devices including whitebox programmable switches and low performance Single Board Computer (SBC).

This deliverable describes COMML functions and operations including its architecture, network functions chaining, and basic packet level primitives such as interaction with communication protocols (values parsing) and storing and providing network data.

1.1 Scope of the deliverable

The scope of this deliverable is focused at control, measurement and monitoring properties of the COMML. Section 2 explains general and specific concepts critical for COMML functionality, such as computer networks operations.

Section 3 describes PDP paradigms - mainly Software Defined Networking (SDN), Programming Protocol-independent Packet Processors (P4), and extended Berkeley Packet Filter (eBPF). This section explores all the main approaches for the data plane programmability and describes the technology selected for the CPN implementation - eBPF utilizing the SDN architecture. Next, related concepts which will support the implementation are described, specifically Network Function Virtualization (NFV) and network function chaining. The final part describes COMML properties: eBPF features and eBPF packet level primitives. This provides context for the following work.

The COMML measurement and monitoring properties presented in Section 4 describes header structures for all common and EPES communication protocols used in the project. This includes composition of various protocol structures and high-level services requirements to interact with specific protocol header fields.

Section 5 presents CPN algorithmics and COMML control instrumentation on several examples of ML-based mitigation services.

This deliverable does not cover any aspects connected with detailed implementation of the CPN architecture including COMML and IOL. This will be the main objective of the deliverable D4.2: *COCOON System Architecture*.

1.2 Relation with other work packages and tasks

This section provides the deliverable's purpose in relation to other tasks, deliverables and WPs within the project. This deliverable utilizes outputs from the deliverable D4.1: *COCOON Development Blueprint*, and tasks T1.2: *Threat Models* and T1.3: *Vulnerability Assessment* as shown in Figure 1.2. The system requirements defined in D4.1 play the main role for this deliverable as they formed a basis for selecting the appropriate technology for CPN implementation.

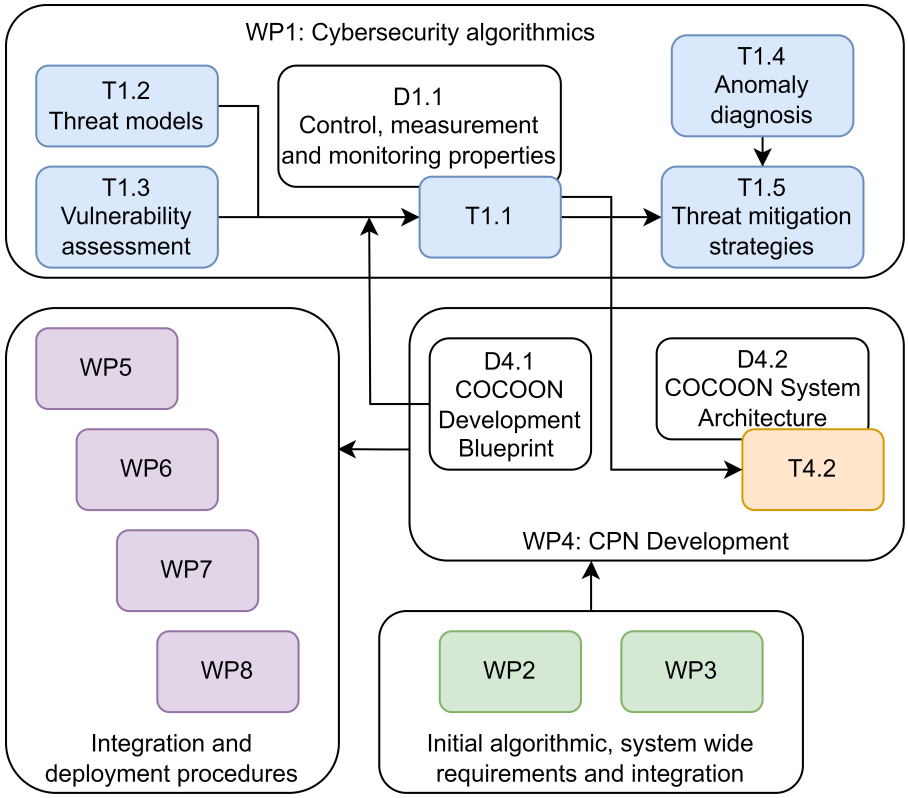

Figure 1.2: The relationship of D1.1 with other tasks, deliverables and WPs

The output of this deliverable will be utilized in task T1.5: *Threat Mitigation Strategies* and more importantly in T4.2 leading to the next deliverable - D4.2: *COCOON System Architecture*, which will describe CPN architecture with details of both COMML and IOL.

Moreover, this deliverable will provide meaningful operational properties for control, measurement and monitoring in intra-domain EPES setups to be used in the context of vulnerability assessment and risk profiling (T1.2, T1.3).

1.3 Methodology

The methodology used for this deliverable combines offline and online discussions between partners with the goal of identifying network requirements of CSL services such as AD and FDII. Furthermore, comments from partners responsible for pilots implementations were considered for selecting appropriate technology for the CPN implementation.

Partners were highly involved in developing communication protocols section and for identifying header data fields needed for the CSL services as prepared by University of Glasgow (UGLA). Specifically, TU Delft (TUD) provided expertise on Generic Object-Oriented Substation Event (GOOSE) and Sampled Values (SV) protocols, Aristotle University of Thessaloniki (AUTH) provided expertise on the IEC 60870-5-104 (IEC104) protocol, and Ingelectus (ING) and University of Sevilla (USE) provided expertise on the Modbus TCP/IP protocol. In summary, this deliverable was written in a collaborative effort following an iterative development approach with several online meetings for feedback.

2 CPN networking fundamentals

This section describes the basic operation of computer networks and EPES communication protocols used in the project. The ISO/OSI and TCP/IP models are introduced as they provide conceptual and realistic explanation of computer network operations. The EPES communication protocols described in this section will be further expanded upon in Section 4.3 which will explain how data fields of these protocols will be used in the COMML layer of the CPN by services from the CSL.

2.1 Introduction to computer networks

A computer network is an interconnection of computers and other end devices such as servers and printers that can communicate with each other using common communication protocols. Computer networks are now being utilized in digitalized OT networks including digital substations where they are interconnecting devices such as Human Machine Interface (HMI), Intelligent Electronic Device (IED) and Merging Unit (MU). These devices are then considered to be end devices from the network perspective, as they represent the analog/digital boundary. Similarly, OT networks are also used in other EPES applications such as renewable power plants and energy communities where field devices, e.g. PV inverters, communicate with centralized controllers, e.g. Power Plant Controllers (PPC) or control center.

This section briefly introduces some basic concepts of computer networks which will be utilized in the remaining chapters of this deliverable as well as throughout the project.

2.1.1 ISO/OSI model

The ISO/OSI model [1] is a reference model which describes the functionality of computer networks. This model also applies to digitalized OT networks including digital substations, renewable power plants and energy communities. The seven layers architecture of the model is shown in Figure 2.1. The left part presents all the layers, numbered from the lowest one, while the right part depicts how data are structured on every layer with the correct datagram terminology for three main layers.

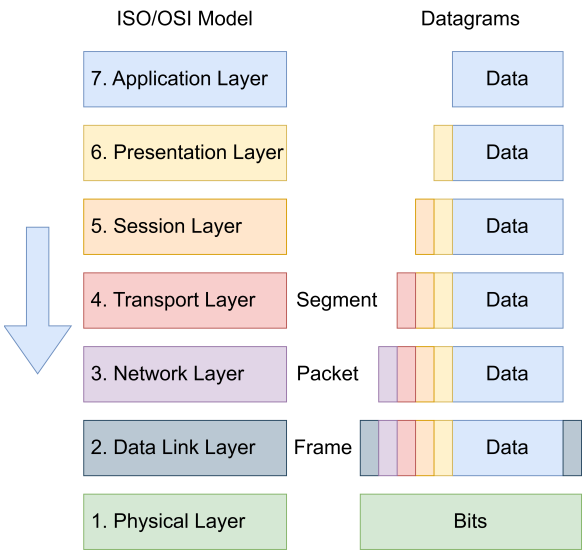

Figure 2.1: The relationship between ISO/OSI model layers and datagram structures

The main functions of each layer are as follows:

7. Application layer is responsible for providing an interface for user interaction.
6. Presentation layer translates data from an application to a network format. This might also include encryption and decryption.
5. Session layer establish and maintains a communication session between the endpoints.
4. Transport layer provides flow and error control. This includes data transfer at optimal speeds based on current link capacity.
3. Network layer is responsible for routing messages between source and destination based on logical Internet Protocol (IP) addresses.
2. Data Link layer forwards messages on a local network based on physical MAC addresses. It also checks if the received message was not damaged during the transit.
1. Physical layer represents data in form of zeros and ones and is responsible for physical transmission.

Communication between two applications on separate devices follows a strictly defined process of traversing through the seven layers (some layers can be skipped, depending on the application requirements). This process is called encapsulation and decapsulation. Encapsulation is done on the sender device, and it involves appending every layer header field information to the original data as shown by colored boxes appended in front of data on the right side of the figure. Decapsulation (unwrapping) is a process executed on the receiver device. Header fields are sequentially removed before the data is delivered to the correct application.

EPES communication protocols are using the same ISO/OSI model and they are inserted as a payload (data) into one of the layer headers. Protocols operating on the Local Area Network (LAN) use the data link layer as they do not require functionalities of the layers above, but are time sensitive. Protocols communicating over the Wide Area Network (WAN) use the transport layer as they need to be routed outside LAN and require reliable delivery.

2.1.2 TCP/IP model

The ISO/OSI model is only a conceptual model and for practical implementation, TCP/IP model is being used instead. It abstracts from presentation and session layers as their functionalities are handled by the application layer. Figure 2.2 shows the mapping between these two models and examples of general protocols for every layer. Finally, EPES communication protocols are shown on the right side aligned to appropriate layers of the models.

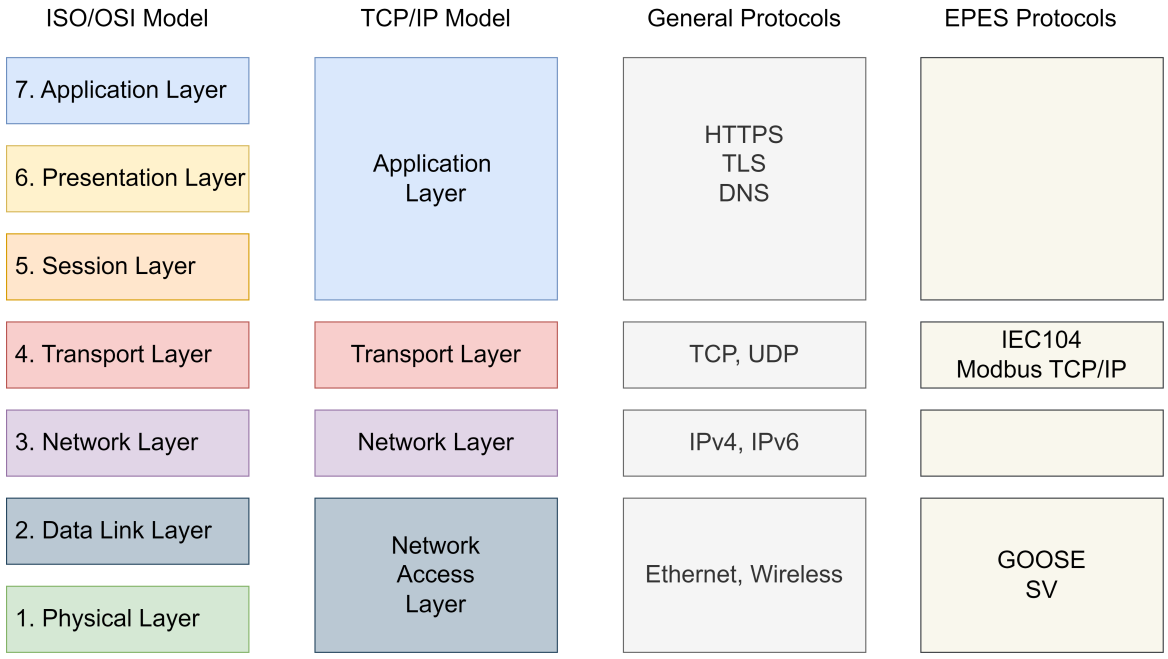

Figure 2.2: Mapping between ISO/OSI and TCP/IP models and their protocols

The main functionality of the TCP/IP layers is as follows:

1. Network access is composed from physical and data link layers of the ISO/OSI model. Datagrams on this layer are called frames. This layer is responsible for delivery on the LAN. It uses physical MAC addresses for forwarding between devices. The layer performs a frame check sequence to verify that the frame was not damaged during the transit.

2. Network layer is composed from the same layer in the ISO/OSI model. Datagrams on this layer are called packets. The layer is primarily responsible for delivery on the global scale. It uses logical IP addresses for routing between destinations. It is also responsible for discarding looped packets and providing Quality of Service (QoS).

3. Transport layer is composed from the same layer in the ISO/OSI model. Datagrams on this layer are called segments. The layer is responsible for delivering data to correct applications on a single device. It uses port numbers to distinguish between different applications. The most widely used protocols on this layer are Transmission Control Protocol (TCP) for reliable delivery and User Datagram Protocol (UDP) for time sensitive applications.

4. Application layer is composed from session, presentation and application layers of the ISO/OSI model. This layer handles the applications which need to interact with applications on another devices.

2.2 EPES communication protocols

EPES is a complex system composed of electricity generation, transmission and distribution power networks. EPES relies on various digital OT communication protocols which are all implemented in specific layers of the ISO/OSI model - most often either layer two (data link), or layer four (transport). Data of these protocols then represents the application layer. This can be for example value reporting to the SCADA. Note that when these protocols are talked

about, they are always considered to be protocols on the corresponding ISO/OSI layer (data link or transport) and never application layer protocols.

CPN converges IT and OT as long as the communication protocols follow the ISO/OSI model architecture and from COMML perspective, any such communication protocol can be used for measurement, monitoring and control.

This section is focused on communication protocols used within the COCOON pilots, which represents frequently utilized EPES communication protocols in real infrastructures. Note that the CPN architecture will be generic and able to operate with any standardized Internet-enabled EPES communication protocol, provided that an appropriate μNF will be developed.

2.2.1 IEC 61850 GOOSE

The GOOSE protocol is part of the IEC 61850 standard [2] that forms the backbone of digital substations. GOOSE is used to communicate critical intra-substation events in real time, e.g., tripping commands between two or more protective relays or IEDs using Ethernet layer-2 multicast. The payload of a GOOSE message typically contains circuit breaker statuses, switch controls, block commands, etc. Under normal operating conditions, all GOOSE messages are communicated within a predefined time range of 100 to 5000 milliseconds (about 5 seconds). Since GOOSE is a non-acknowledgement-based protocol, this regular heartbeat is to ensure that the communications are healthy and can be used in case of events. With every outgoing GOOSE message, the sequence number field is incremented by 1 and has a maximum value of 2^{32} (unsigned 32-bit integer) before rolling over to 1. When a substation event occurs, e.g., a trip signal, the update rate of the new GOOSE messages is statistically increased to ensure that the message is successfully delivered. This event mode has a time range of 0.5 to 5 milliseconds. Furthermore, the status number is incremented by one and sequence number is reset to zero. These fields and the data payload of a GOOSE message will be summarized in Section 4.

GOOSE protocol follows the publisher-subscriber communication mode which is based on layer 2 multicast for delivering the message from the source (publisher) to multiple destinations (subscribers). This mode is mostly used for communication between IEDs. The mode is detailed in Figure 2.3, which shows that three subscribers are receiving the published traffic, while two devices depicted in red color are not. CPN in this example acts only as a forwarding device - a network switch.

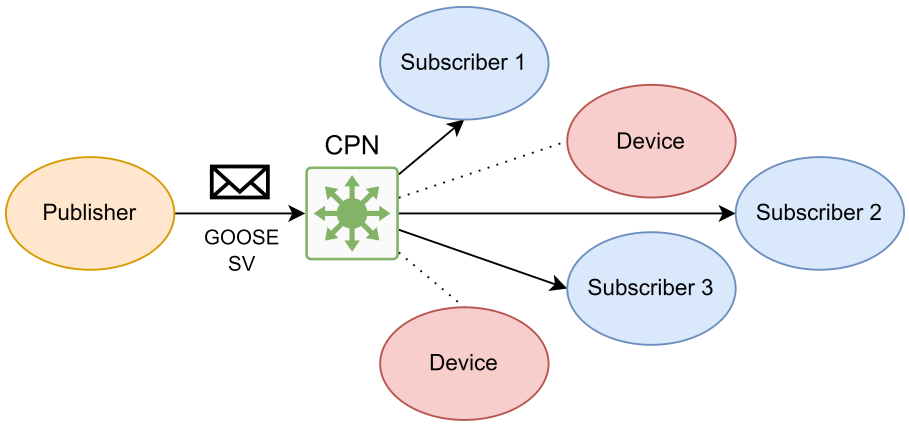

Figure 2.3: IEC 61850 publisher-subscriber communication mode

2.2.2 IEC 61850 Sampled Values (SV)

The SV protocol, defined in IEC 61850 9-2 [3], is used for reporting physical measurements, e.g., voltages and currents, which are converted from analog to digital values at the source. Like

GOOSE, SV follows the same publisher-subscriber mode based on layer-2 Ethernet communications, where each subscriber is listening to a particular publishing MAC address. A typical SV frame contains 8 signals - 3-phase voltages and currents along with the neutral voltages and currents. The typical update rate of an SV stream is 80 samples/cycle. This corresponds to a data-exchange rate of 4.8 and 4 kHz for a 60 and 50 Hz system, respectively.

2.2.3 IEC 60870-5-104

IEC104 is a widely used standard in the field of industrial automation, introduced to facilitate reliable and efficient communication between various control and monitoring systems within EPES, such as Remote Terminal Unit (RTU), metering infrastructure, and electrical line switches [4]. Its distinct characteristic is the ability to support various types of exchanged messages, including monitoring, control, and configuration data, thus ensuring interoperability and seamless integration across different systems and vendors. Furthermore, it leverages the TCP/IP suite to provide robust and scalable real-time data exchange over wide-area networks with reduced infrastructure requirements. For these reasons, several Distribution System Operator (DSO) of EPES are using this protocol in their modern SCADA systems.

The protocol follows the client-server communication mode shown in Figure 2.4. This mode establishes a connection between the source (client) and the destination (server) and uses bi-directional communication for data exchange.

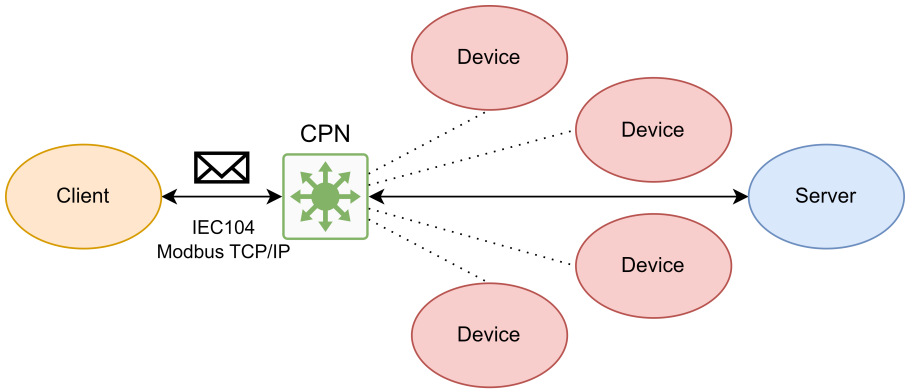

Figure 2.4: Client-server communication mode

2.2.4 Modbus TCP/IP

Modbus is one of the communication protocols widely used in the industry for the transmission of information between electronic devices [5]. The original protocol, known as Modbus RTU, operates over RS-232 or RS-485 serial lines, allowing point-to-point communication or multipoint networks in industrial environments. Later, through the Ethernet technology, this protocol evolved to Modbus TCP/IP, used for communications over TCP/IP networks.

Same as in case if IEC104, Modbus TCP is a client-server (master-slave) mode, where the client initiates communication requests and the server responds to these requests. Each Modbus TCP message contains a Protocol Data Unit (PDU) that is encapsulated within a TCP/IP packet. This encapsulation allows Modbus data to be transmitted over Ethernet networks, benefiting from its high-speed capabilities and flexibility.

The Modbus TCP protocol is addressed based on the IP addresses of devices communicating over an Ethernet network, using port 502 as the default port for data exchange. This protocol allows the transmission of Modbus messages encapsulated in TCP/IP packets.

3 COMML architecture and properties

COMML is part of the three layer CPN architecture as shown in Figure 3.1. As it is the layer responsible for physical manipulation with the OT network traffic, it is the only layer which has to be implemented on the CPN. IOL and CSL can either be implemented on the node, or on a remote device where they can manage multiple nodes.

The main functionality of COMML is traffic measurement, monitoring and control, which is done based on the instructions from higher layers. For this purpose, two different Application Programming Interface (API) are used in the architecture: northbound and southbound.

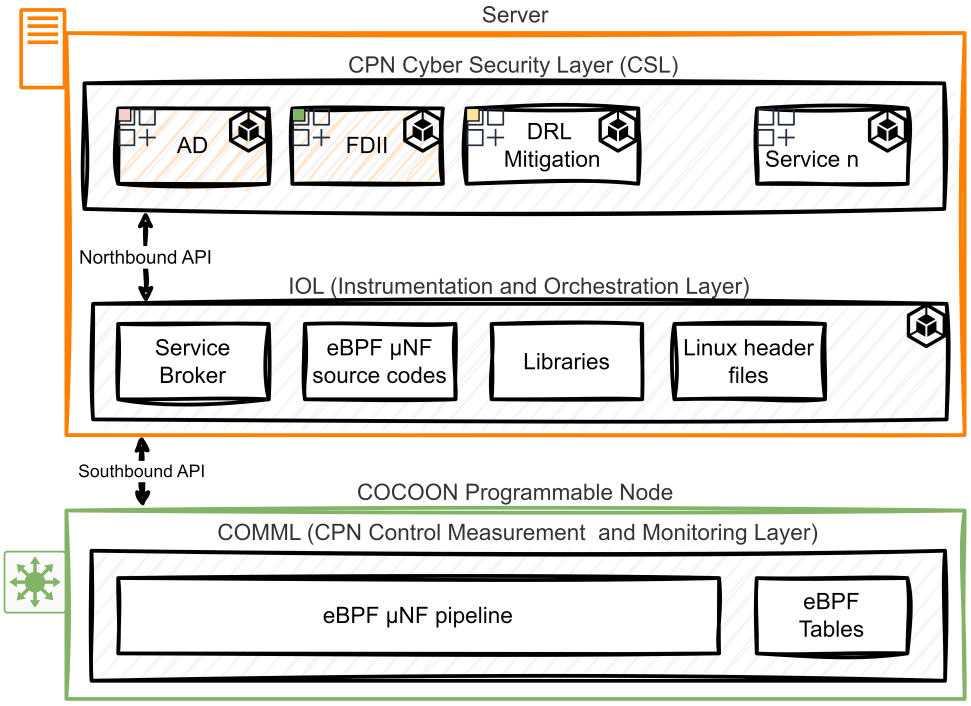

Figure 3.1: COMML architecture in relationship with other layers

This section describes architectural requirements for the COMML, technologies which will be used to implement the architecture and its main properties. This includes PDP, SDN, NFV and network function chaining.

3.1 COMML architecture requirements

This section summarizes architecture requirements from the information exchange and application perspectives. Information exchange requirements are defined by used EPES protocols, while the application requirements are defined by supported services in CSL.

3.1.1 Information exchange requirements

Information exchange requirements are defined by the communication protocols used within the COCOON project, but they are generalized enough to be also applicable to other EPES communication protocols.

Communication modes

COMML will need to support various communication modes including the publisher-subscriber explained in Section 2.2.1 and client-server explained in Section 2.2.3. While the client-server mode support requires a simple forwarding functionality, the publisher-subscriber modeutilizes layer 2 multicast delivery. This is different from layer 3 multicast, where IP addresses in a range 224.0.0.0 to 239.255.255.255 represents multicast addresses. In layer 2 multicast, MAC addresses have to be used instead of IP addresses and their range is 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff.

Most layer 2 devices of traditional networks - switches - treat layer 2 multicast as broadcast and simply forward such traffic to all the ports except the receiving one. COMML can implement either this functionality, or complete multicast forwarding.

IEC 61850 GOOSE

The main requirement of the GOOSE protocol on the COMML is minimum latency of processing which must not exceed 3 ms end-to-end. This is measured between a publisher and every subscriber. For this reason, all the COMML operations has to be executed well below this boundary, as processing by the sender and receivers counts to the limit too.

IEC 61850 Sampled Values (SV)

The SV protocol is not as latency sensitive as GOOSE, but the messages are sent with high frequency of up to 4.8 kHz. This corresponds to 0.2083 ms interval between every SV message. With the average SV message length of 120 bytes [6], this generates 0.576 Mbps traffic. COMML must be able to process multiple of such streams simultaneously while having enough reserve for additional protocols used in the network.

IEC 60870-5-104

The CPN node will be equipped to handle the IEC104 communication protocol, as this standard is used in the majority of European Union (EU) DSOs and therefore is a strong requirement within the CPN COMML. This is also reflected in the pilot related to the energy community that is located in Chalkidiki, Greece, where the Greek DSO, HEDNO, monitors and controls the corresponding Photovoltaic (PV) plants and adjacent network elements, e.g., line switches, of the energy community via SCADA using the IEC104 communication protocol.

The EPES data that will be monitored by the CPN node include the following:

1. Voltage magnitudes at the point of interconnection (POI) of each PV plant with the distribution grid.
2. Current magnitudes injected by the PV plants to the distribution grid.
3. Active and reactive power at the POI of each PV plant with the distribution grid.
4. Active and reactive power flows though specific monitored elements within the distribution grid, e.g., line switches, as well as the corresponding voltages and currents.
5. Active and reactive flows at the primary substation, as well as the corresponding voltages and currents.

The delivery frequency of the above data depends on the timescale of the Ancillary Services (AS) that the energy community provides to the distribution network. In the frame of

COCOON, two AS types will be examined, i.e., voltage-related and frequency-related AS. The former operates on a relatively slow timescale, e.g., every 5 min, while the latter requires a fast response, e.g., less than 1 s. Therefore, the delivery frequency shall be less than 5 min and 1 s (ideally less than 0.5 s) for the voltage-related and frequency-related AS.

Modbus TCP/IP

The purpose of this section is to provide a comprehensive guide on the use of the Modbus TCP/IP communication protocol in the management and operation of PV plants. It provides a detailed and practical view of how this communication protocol facilitates the interaction between the various equipment that makes up a PV plant, such as the Power Plant Controller (PPC), PV inverters, power meters, and monitoring and control systems.

A) Equipment involved. Different devices using Modbus TCP exist in PV power plants. The most relevant ones are the following:

- PPC.
- SCADA.
- Power Meter.
- Weather station.
- PV inverters.
- PV inverter logger.
- Control Center.

Depending on the type of PV inverter installed in the plant, communication between the equipment may vary as shown in Figure 3.2. PV inverters can be widely classified into central and string inverters. A central PV inverter is usually characterized by a unique power electronics stack that injects the power to the system. In this case, the central PV inverter directly communicates with the PPC. On the contrary, string inverters are composed of several power electronic stacks injecting the power to the system. In this case, the communication front-end of the set of string inverters is the so-called PV inverter logger, which aggregates the power injections and coordinates the individual actions of each inverter. Note that the communication between the PV inverter logger and the PPC is Modbus TCP but, internally, the communication between the string inverters and the PV inverter logger is Modbus RTU.

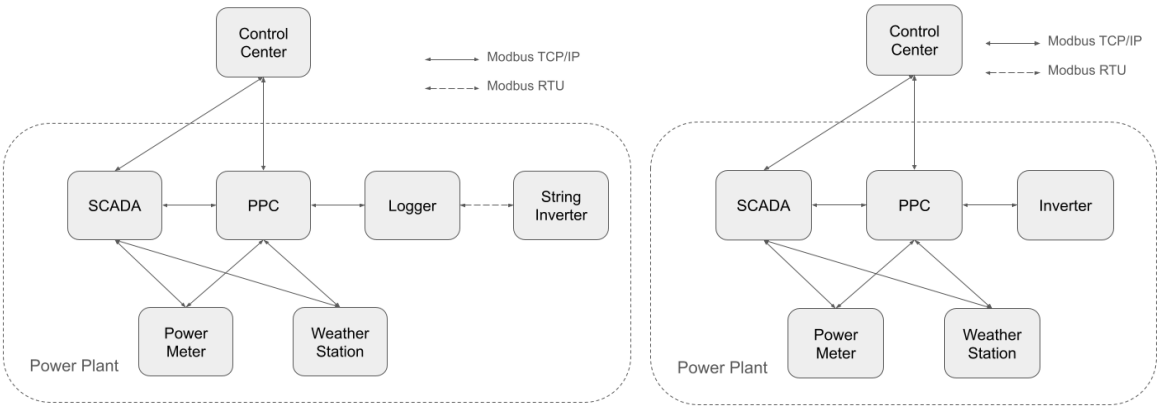

Figure 3.2: Required communication to be addressed by the CPN between PV plant components depending on the PV inverter type: central inverter (left), string inverter (right)

B) Information types. The information exchanged between the different equipment in a PV plant through the Modbus TCP protocol can be classified into the following types:

- Measurements: values of physical quantities measured in the plant: active power, reactive power, apparent power, voltage, current, frequency and atmospheric measurements such as temperature and irradiance.
- Setpoints: control references sent from the PPC to the PV inverters for the management of the PV plant as a single controllable unit seen from the PV plant Point of Interconnection (POI).
- Status signals: status of the different elements of the plant, indicating, for example, alerts or status of the elements.

Read or write frequency times depend on the type of information sent or requested. For example, the average read/write times for different types of messages are approximately around the following values:

- POI measurements: 50-100 ms.
- PV inverter setpoints: 0.5-1 s.
- Atmospheric measurements: 1 s.
- Status: 1 s.
- Alerts: 1 s.

3.1.2 COMML application-level requirements

Application level requirements are defined by the services used in CSL. These will include AD, FDII and threat mitigation. These services requirements on COMML will be in the form of: (i) data measurement and monitoring, such as to collect specific protocol values; (ii) traffic control, such as to drop a defined traffic flow. Specific measurement and monitoring requirement for every service are described in Section 4 while the expected traffic control requirements are shown in Section 5.1, but they will be defined in more detail in the following work.

3.2 COMML architecture: PDP paradigms

The most important architectural decision of the CPN is the technology for the PDP. This section presents the basic terminology around PDP and explains how the main PDP paradigms relate to the selected technology for the COMML implementation.

3.2.1 Traditional computer networks

Traditional computer networks are composed of dedicated hardware devices with fixed-function architecture. These are typically mainly focused on a single function such as filtering, routing or forwarding. Nowadays, the more advanced devices can perform more functions at the same time, but due to the software and architecture limitations, they will always have to compromise between performance and functionality. The most common examples of computer network devices are:

- Switch performs line-rate forwarding on a LAN.

- Router performs routing between different networks such as LAN to WAN.
- Firewall filters traffic.
- Intrusion Detection System (IDS) analyses traffic and searches for anomalies.
- Gateway connects networks of different technologies (for example wired to wireless).

The architecture of every networking device is composed from two layers - data plane and control plane. The control plane provides logical decisions about how to process traffic while the data plane does the actual processing. Basic functions of these layers are explained below.

Control plane functionality

The control plane is a layer responsible for making decisions about how to process and forward traffic. It typically includes functions such as routing protocols, topology discovery, security, etc. The configuration of these functions is then translated to instructions for the data plane layer.

Data plane functionality

The data plane, also called the forwarding plane, is a network layer responsible for fast and efficient processing and forwarding of network traffic. Every device connected to a network has a data plane. The data plane contains of processing pipeline which consists from packet level primitives defining the steps for traffic handling. These can include:

1. Parsing - reading of the message header fields, such as MAC and IP addresses, used protocols, etc.
2. Classification - comparison of the parsed fields with rules provided by the control layer and handling the traffic according to the result, for example dropping traffic from an unknown network.
3. Modification - any message modification, such as decreasing the Time To Live (TTL) or changing Virtual Local Area Network (VLAN).
4. Deparsing - writing the message header fields to the output buffer.
5. Forwarding - message transmission.

The data plane is implemented in a sequence of so-called match-action tables. The match corresponds to the classification step and the action corresponds to the modification and (or) forwarding step. The match-action concept represents low-level primitives and it is easy to implement on hardware, thus providing high performance [7]. The concept is explained in more detail in Section 3.2.2.

Network programmability

Network programmability is a concept originating in 1990s [8], but becoming more widely used around 2010 with the success of the OpenFlow protocol in backbone networks [9]. Traditional network devices have fixed control and data planes which do not allow functions modification beyond software configuration. Network programmability, on the other hand, enables modification of control and data planes to achieve custom traffic processing which can dynamically adapt to changing (network) conditions.

The network programmability types, ignoring the unsuccessful attempts such as active networking, can be classified into three paradigms: SDN, P4 and eBPF, which are briefly described below as they are all relevant for the CPN.

3.2.2 Software Defined Networking (SDN)

SDN is based on the separation of control and data planes, where the control plane is moved to a centralized device called SDN controller. It became popular with the adoption of the OpenFlow protocol which defines the data plane, control plane and the communication between them via a so-called southbound interface. The OpenFlow protocol became a norm, and it is included in Open vSwitch (OvS) [10], which is part of the Linux kernel from version 3.3 [11], making it usable on any Linux device.

SDN does not support full data plane programmability as the processing pipeline must follow pre-defined structures (flow tables) and processing logic must correspond to the match-action fields defined by the OpenFlow protocol.

While the flow tables allow relatively enough flexibility as the tables can be chained, searched in parallel, executed in groups or placed in specific locations (ingress and egress tables), the strictly defined match fields pose a more significant restriction as only well-used network protocol header fields are defined. This for example means, that while MAC and IP addresses can be matched, no header fields of industrial protocols could be used for classification or modification as they are not included in the OpenFlow standard.

On the other hand, SDN can be easily implemented on any device running Linux via a software switch like OvS (although without the hardware support, performance will be lower). There are also already existing industrial proprietary solutions utilising OpenFlow for configuration and management of critical infrastructure such as the SEL-5056 Flow Controller [12].

The CPN architecture will follow the three layer SDN architecture with southbound and northbound APIs as shown in Figure 3.3, but will not utilize the OpenFlow protocol for the implementation of PDP due to its limitations.

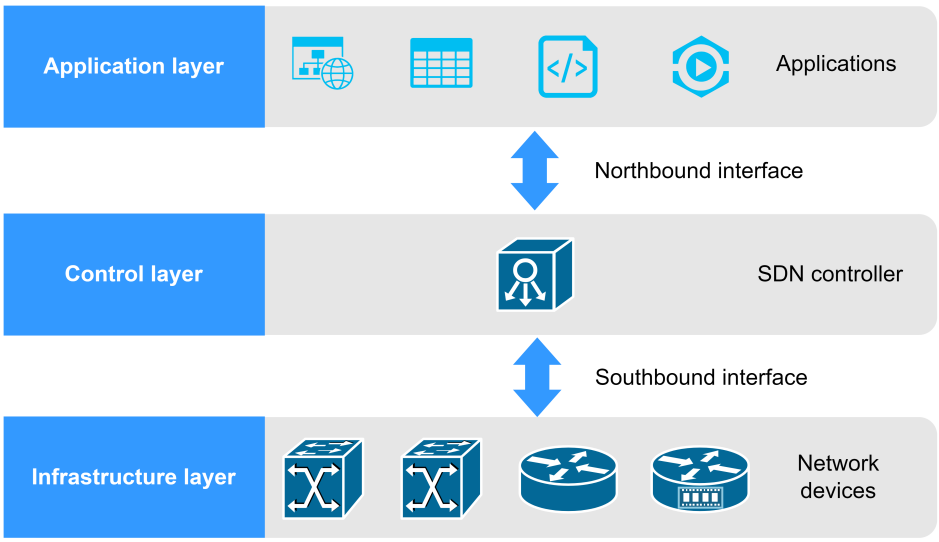

Figure 3.3: SDN architecture

The match-action concept in CPN

SDN introduced the concept of "match-action" which refers to the fundamental mechanism used by the SDN controller to direct network switches on how to handle packets. OpenFlow uses match-action tables to implement programmability using a simple logic which can be easily translated into the networking device architecture and hardware accelerated. This concept is in some form used in all network programmability paradigms.

A set of match-action entries specify how packets matching an entry should be handled. This set will be present in the table at run time. Every entry consists of:

- Match - a pattern that encodes a predicate on packet headers, for example based on source MAC address and incoming port number.
- Action - a list of actions that encodes a function on packet headers, for example forward to port number 5).
- Priority (optional) - a number that is used to differentiate between rules with overlapping match patterns.
- Statistics (optional) - a set of counters that log various parameters, for example the total number and total size of packets processed using the entry.

The match-action mechanism in SDN compares incoming packets against predefined criteria and executes specific actions based on the comparisons. This process allows for dynamic, programmable network management by directing how packets are handled, such as forwarding, dropping, or modifying them, based on real-time or policy-driven decisions. The workflow of match-action table in SDN is as depicted in the Figure 3.4.

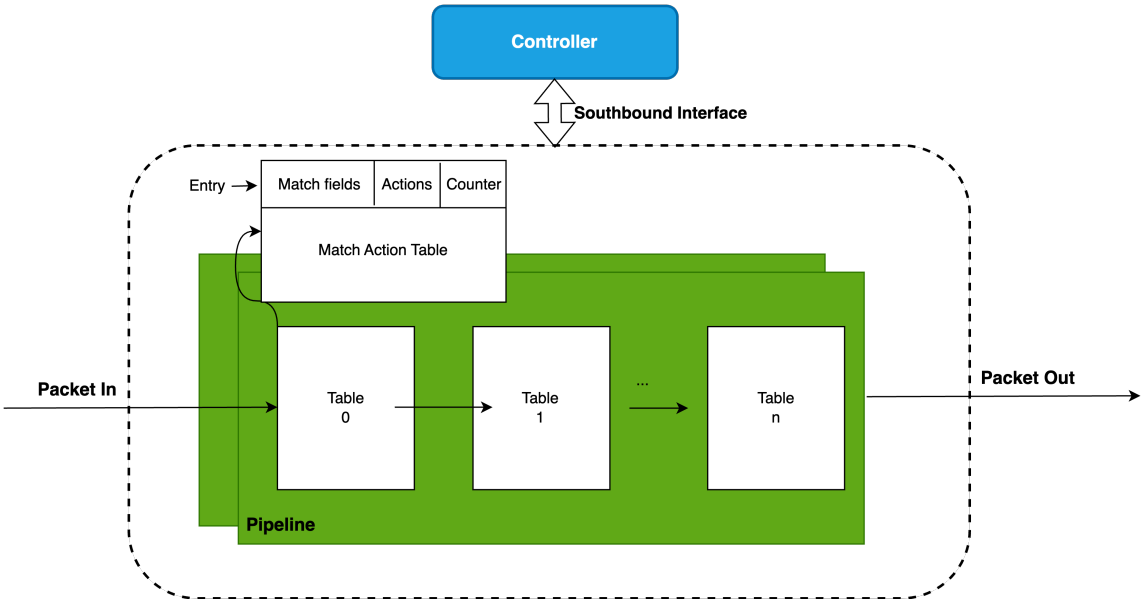

Figure 3.4: Workflow of the match-action concept in the CPN

3.2.3 Programming Protocol-independent Packet Processors (P4)

P4 is a language for data plane processing definition which brought full programmability of the match-action tables via custom parser and deparser, while keeping the data and control plane separation [13]. P4 architecture also utilizes parallelism and support of hardware chips such as Application-Specific Integrated Circuit (ASIC) to achieve high performance. P4 is suitable for implementing highly specific functionality including processing of OT protocols, but it requires support of the underlying hardware and is currently supported only on a limited number of networking devices (switches) and cannot be deployed on end devices (there is no Linux kernel support). The limitation in pre-defined set of actions like in the OpenFlow protocol is valid for P4 as well [14].

P4 will not be utilized for the CPN implementation due to the hardware requirements which makes P4 unsuitable for relatively low resources OT networks. While there is an experimental software implementation of P4 - Behavioral Model version 2 (BMv2) [15], which could be deployed even on SBCs, it lacks most P4 features and is not suitable for high availability production deployments.

3.2.4 eBPF

eBPF is a new paradigm of running a virtual machine-like construct with specific instruction set inside the Linux kernel. eBPF was created as an extension of BPF (now called classic or cBPF while eBPF is called simply BPF) and the current instruction set supports flexible use cases well beyond just networking. eBPF is particularly suitable technology for OT networks as it offers verifiable safety, security and time constrained execution required in those scenarios. Moreover, it allows unprecedented flexibility (not bound by the match-action concept) and support on any Linux-based device as shown in the following section. For these reasons, it was selected as the main implementation technology for the CPN.

General eBPF architecture is shown in Figure 3.5. The eBPF code is first verified before being compiled and loaded into the selected eBPF implementation. This will solve the CPN requirement on code verification. The remaining parts of the figure will be explained in Section 3.5.1 as they are related to implementation flexibility.

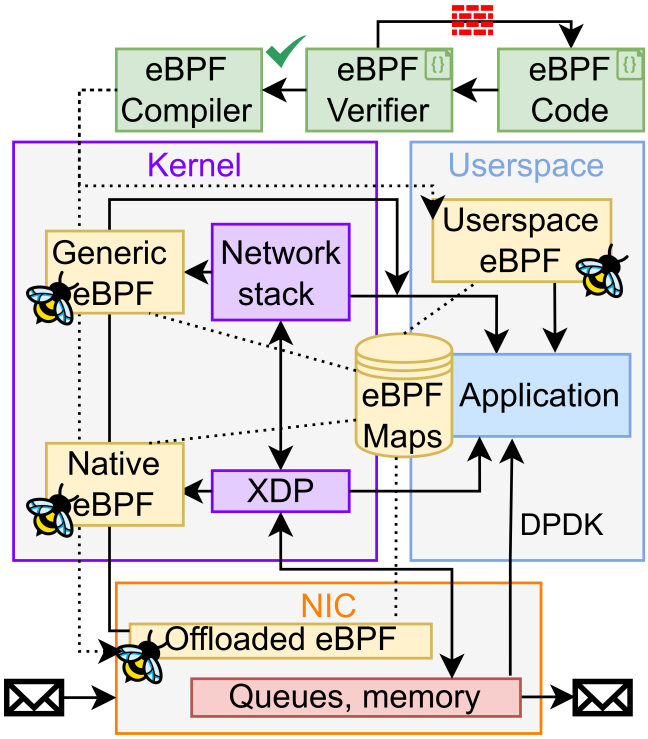

Figure 3.5: eBPF architecture concept used within the CPN

3.3 Network Function Virtualization

NFV is a concept utilizing virtualization technologies to manage networking functions via software, instead of relying on dedicated hardware. This approach transforms the way networking services are deployed and managed, offering several key benefits such as flexibility, scalability, and cost efficiency.

NFV is mainly used in data centers and cloud deployments within high-performance networks. Most of the underlying hardware for NFV is optimized for these environments and not for OT network requirements. Both SDN and P4, which can be looked upon as enablers of NFV, are only very slowly finding way into OT networks. Only very recently, there has been the first commercial effort to introduce SDN into power grid operation [12]. The situation with P4 is even worse as there are no industrial P4 devices for OT networks. CPN based on eBPF technology, on the other hand, can be deployed on any Linux-based device and therefore is not limited to devices made for data center environments.

NFV as a concept will be used mainly in IOL where virtualized μNF will be prepared in form of eBPF source codes and the layer will be responsible for deploying these functions via the southbound interface into COMML located on the CPN. There are three basic types of virtualization which will be used for various purposes throughout the project.

3.3.1 Virtualization types

NFV relies on various types of virtualization to deploy and manage Virtual Network Functions (VNF). These virtualization types differ in how they abstract hardware resources and provide isolation between network functions.

Full virtualization (VM)

Full or Virtual Machine (VM) virtualization enables strong isolation between functions as every VM creates a complete representation of a physical machine including kernel, drivers, processes, etc. This type of virtualization is suitable for network functions needing high security and stability such as firewalls and routers. A disadvantage is a higher demand on resources, as creating a virtual machine poses a significant overhead. This is caused by the fact that the whole operating system has to be replicated for every virtual machine and cannot be shared as shown in Figure 3.6. In full virtualization, hypervisor is responsible for allocation of physical resources such as Central Processing Unit (CPU) and Random-Access Memory (RAM) to virtual machines.

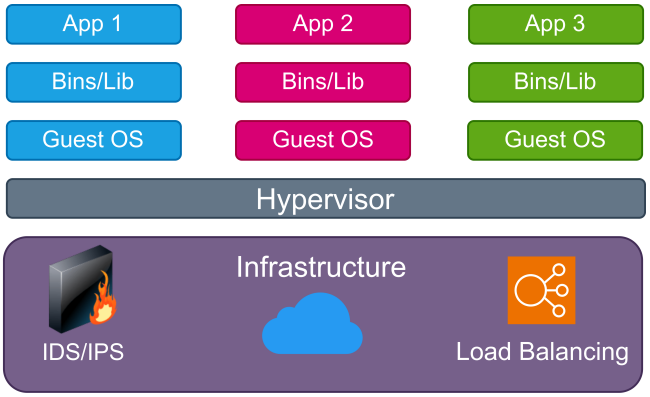

Figure 3.6: Full virtualization (VM)

Examples of full virtualization are VMware ESXi, KVM, or Microsoft Hyper-V, which can all create virtual machines on a single physical device. As these tools support internal networking, a "black box device" composed of several VMs can be created with one VM having functionality of Intrusion Detection Prevention System (IDPS), another of a router, etc. In this manner, a programmable processing pipeline is created but with a significant overhead and limited flexibility.

This type of virtualization will be mostly used for CSL which supports more complex services developed by various teams. The full virtualization in this scenario will prevent any potential compatibility problems between the services such as conflicting libraries and packages.

Container-based virtualization

Container-based virtualization is a lightweight type of virtualization where only network functions and their dependencies are running in a virtualized separated environment. All the underlying layers including the operating system are shared among all the containers as shown in Figure 3.7.

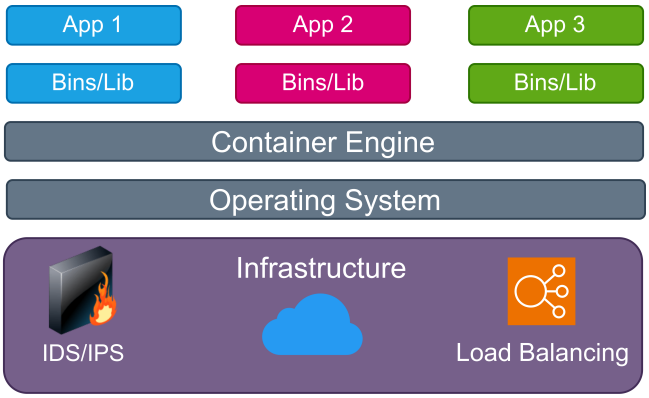

Figure 3.7: Container-based virtualization

Examples of container-based virtualization tools are Docker and Kubernetes. This type of virtualization is suitable for μNFs which need high performance and an option to be quickly restarted if the service modification is required. This type of virtualization might be less suitable for more complex services in CSL.

Bare metal

Bare metal virtualization involves running VNFs directly on physical hardware without an intermediate hypervisor or host operating system. This achieves the best performance, but at the cost of reduced flexibility and increased configuration complexity. For these reasons, this type of virtualization is at the moment not considered in the COCOON project.

3.4 Network function chaining

The key concept for the COMML is network function chaining which will be used for creating the data plane processing pipeline on CPN. This will form a sequence of how datagrams will be processed by the CPN. An example is shown in Figure 3.8. Note that network function chaining is different from service function chaining employed on CSL. As every CSL service might be broken up into several μNFs, these functions will need their own chaining system.

μNFs implemented in eBPF will be inserted into the pipeline in programmable order depending on the orchestration performed by IOL. In the provided example, datagrams will first be processed by the AD μNF which will extract specified header fields and update the content of the associated eBPF table which can be accessed by the high-level CSL AD service. The datagram will then continue to the FDII μNF which will perform similar processing. In the example, this service is saving data into two separate eBPF tables. This might be due to the volume of data, or efficiency of tables (for example hash vs array). Finally, the datagram is sent to the forwarding μNF which sends the datagram to the appropriate outgoing interface.

Note that a high-level services in the CSL such as AD, might need to use more than one μNF - eBPF function. This will depend on the service complexity, or the need to be able to replace part of the function without any downtime.

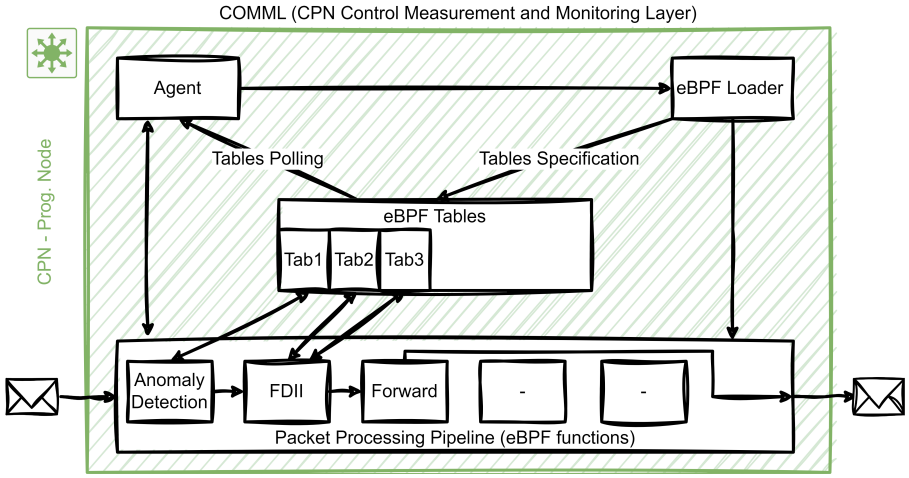

Figure 3.8: Exemplar of COMML network function chaining

There are two methods how network function chaining can be implemented within eBPF architecture: tail calls and control plane agent.

3.4.1 Network function chaining via eBPF tail calls

A tail call is an eBPF mechanism for breaking up network functions into multiple logical parts. It works on the same principle as the "goto" statement of any programming language. With a tail call, an eBPF function can jump to another eBPF function. In this way, up to 32 eBPF programs can be chained.

Tail calls are implemented by defining an eBPF map of the array type for storing references to other eBPF programs which can then be called by the *bpf_ tail_ call* helper call [16]. Important rule of the tail calls is that the called program can never call back the original one (cannot create a loop).

3.4.2 Network function chaining via a control plane agent

An alternative mechanism for network function chaining is implementation within the control plane of the CPN. This would involve an agent responsible for handling datagrams forwarding between μNFs. An advantage of this approach is that practically an unlimited number of μNFs in the processing pipeline can be programmed. The main disadvantage, however, is a more difficult implementation and potentially worse performance than with eBPF tail calls.

3.5 COMML properties

3.5.1 eBPF features

eBPF has several unique features which makes this technology more suitable in OT networks than previous network programmability paradigms. This section explains why eBPF has been selected as the main CPN technology.

Resource-constrained devices

eBPF can run on any device with recent Linux kernel version, the first supported version is 3.18 released in 2014 [11], but newer versions support more features. This enables deployment on resource-constrained devices such as SBC and does not require expensive data center switches like SDN and P4 technologies. This will significantly reduce CPN cost and allow deployment of multiple CPN nodes with scalable performance. eBPF could technically also be deployed on end devices such as future IEDs, RTUs and MUs, where it could monitor their status, including status of running processes.

Safety of the code

Network programmability in OT domain has so far been looked upon at least suspiciously mostly because of the uncertainty of the code reliability. eBPF has, unlike the previous paradigms, an advantage in built-in safety and a security check as the created code has to always pass the eBPF verifier process which checks the following [17]:

1. Program termination checks that the code will always lead to a termination, i.e. creates and iterates a direct acyclic graph representing all states of the program.
2. Execution simulation checks if memory and registers states are always valid.
3. Calls verification checks if the helper function calls are allowed.

This verification makes the eBPF especially suitable for CPN implementation, which will require safety. This would not be achievable with previous programmable technologies.

Implementation flexibility

eBPF started as a technology intended for kernel implementation, but it can nowadays be deployed in various use cases within the system including: userspace, in the kernel with or without Express Data Path (XDP) offload [18], which improves performance, and also implemented completely in Network Interface Cards (NIC), thus achieving line rate speeds [19]. Such a variety makes it versatile for OT networks with heterogeneous devices in different life cycle stages and different requirements on functionality and performance.

The userspace version will be used for initial development and testing of the CPN, as it is compatible with any device including virtualized switches such as OvS. This will simplify the development as the power grid OT infrastructure topology can be emulated in the Mininet network emulator [20].

The userspace implementation can also be combined with Data Plane Development Kit (DPDK) [21], which passes received traffic directly to the userspace thus avoiding kernel completely and achieving great performance. This implementation is preferred for the final CPN deployment for performance reasons and high compatibility with current NICs. All the implementation types are shown in Figure 3.5 and marked with the bee symbol.

Runtime configurability

eBPF allows changing the code at runtime without a need to reboot the system. This is particularly important for high availability systems and for systems such as CPN where network functions might be dynamically loaded or unloaded.

Programming flexibility

eBPF is written in C language and it has a more flexible instruction set than P4 language. This enables use of more complex programming language constructions including loops (with a limited number of executions) which can support more advanced traffic processing in PDP such as encryption. This is another reason why eBPF was selected for the CPN implementation as it will more easily support advanced networking functions.

3.5.2 CPN packet-level primitives for COMML

This section shows examples of eBPF packet level primitives and code snippets of implementation of some of them.

Packet level primitives

A packet level primitive is a basic building block of μNFs (their decomposition). Examples of packet level primitives are:

- Packet timestamping.
- In-network aggregation.
- Flowstat table updates.
- Probabilistic data structure update.
- Read/write from/to interface.
- Packet header parsing.
- Packet dropping/throttling.
- Traffic reroute.
- Traffic mirroring.

Examples of packet header parsing and data structure update primitives implemented in eBPF are shown in the following subsections.

Header parsing

Listing 3.1 shows an example of parsing of two Ethernet header fields, source MAC address and protocol. These header fields are stored into a custom *count_entry* data structure. Note that the function *prog* defines the eBPF code of the μNF and it will be executed when a new message is received.

```
1
2 uint64_t prog(struct packet *pkt)
3 {
4     struct count_entry *item;
5     item->source = pkt->eth.h_source
6     item->protocol = pkt->eth.h_proto
7 }
```

Listing 3.1: Parsing Ethernet header fields in eBPF

Data structures - eBPF maps

μNFs which require saving state or exchanging data with IOL need to utilize eBPF maps. These are eBPF internal data structures and the only mechanism for keeping information between execution of eBPF μNFs. Listing 3.2 shows how an eBPF map *protocol_logger* is created and updated with data from the packet. Note that eBPF maps are created outside the main *prog* function.

```
1
2  struct bpf_map_def SEC("maps") protocol_logger = {
3      .type = BPF_MAP_TYPE_HASH,
4      .key_size = 6,
5      .value_size = 2,
6      .max_entries = 256,
7  };
8
9  uint64_t prog(struct packet *pkt)
10 {
11     bpf_map_update_elem(&protocol_logger, pkt->eth.h_source,
12        pkt->eth.h_proto, 0);
13 }
```

Listing 3.2: eBPF map definition and update

3.5.3 eBPF limitations

Services of CSL will be utilizing μNFs implemented as eBPF programs. These functions will have to follow restrictions posed by eBPF including:

- Maximum number of instructions: every eBPF program can have a maximum of 1 000 000 instructions [22]. This is a significant increase as previous versions of Linux kernels supported only 4096 instructions. This limit applies to the compiled eBPF bytecode and the number of instructions of the function written in C might be slightly lower. Services supported by CPN are not expected to reach anywhere near this limit.

- Bounded loops: maximum number of loop cycles is 8 388 608 in the most recent kernel versions [23]. This is an improvement with respect to the previous versions where loops were not supported at all. The loop cycles are performed by a kernel helper function call and do not count into the maximum number of instructions limit. This is an important feature for implementation of COCOON services as they are expected to use loops.

- Programs chaining: maximum number of chained (tailed) eBPF programs is 32 [16]. This can be used for creating a complex functionality of connected services. In case of COCOON, this number should be sufficient for support of CSL services decomposition.

- Event oriented functionality: all the PDP paradigms including eBPF are event oriented. This means that the network function is only executed when a certain event happens. These events are defined within the kernel and can include a new packet reception or the start of a new OS process. This makes CPN less suitable for functions such as active network scanning which requires sending traffic proactively. Passive network scanning, on the other hand, is an ideal use case for event oriented PDP.

4 COMML measurement and monitoring properties

This section describes two types of network measurement, active and passive, and presents networking and EPES protocol measurement requirements.

4.1 Network measurement and monitoring

The goal of network measurement and monitoring is to assess the behavior of the infrastructure based on operational traffic dynamics. Quantitative measures of such temporal performance properties can then be used by services in CSL to provide the necessary input to control and adaptation algorithms, which ultimately facilitate a managed and optimized operation of the networked environment.

Network measurement and monitoring need to provide continuous reporting of network status so the network performance can be accessed. This is challenging to provide in traditional network architectures, where the measurement and monitoring are not taken into consideration in the design phase and, instead, are being added *ad-hoc*. Programmable network architectures such as CPN, on the other hand, can provide efficient network measurement and monitoring independently on the network topology. Moreover, they are flexible in what parameters are being collected. This is especially important for OT networks, where monitoring of network performance is not the only criterium, but where monitoring of OT protocols and values transmitted, field measurements or setpoints, within the messages is required as well.

Network monitoring can be fundamentally categorized into active and passive measurement [24] as shown in Figure 4.1. This figure shows measurement done in the architecture of traditional networks. In the COCOON scenario, passive measurement can be done directly on the CPN without any actions required from end devices. A further clarification about active and passive network measurement follows in the next subsections.

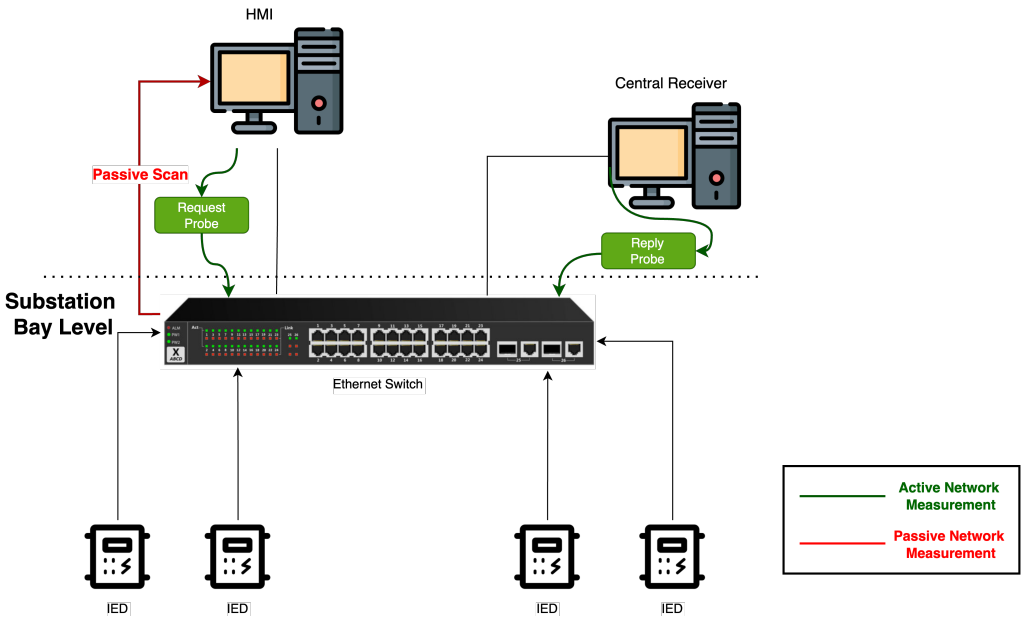

Figure 4.1: Active and passive network measurement in the traditional network architecture

4.1.1 Active network measurements

Active measurements generate and inject traffic into the network in order to get various performance metrics of the network. This type of measurement can be potentially problematic for OT networks for several reasons. First, active network measurement can trigger false positive alarms with existing security appliances as it presents an unknown type of traffic. Secondly, this traffic can saturate the limited network bandwidth and, finally, it can bring down low performance end devices such as IEDs.

Moreover, the main goal of active network measurement is to get network performance parameters and not metrics related to the power grid operation. The typical performance values include:

- Propagation and transmission times.
- Bandwidth.
- Maximum jitter.
- One-way / round-trip delay.
- One-way packet loss.
- IP packet delay variation.

This type of measurement, therefore, does not analyze existing traffic in the network and cannot provide information about used protocols and datagrams' payloads. Active network measurement is performed between two points in the network, as shown in Figure 4.1, and in traditional networks it can be done with the following tools:

- ICMP protocol.
- iPerf.
- hping3.

The use of these tools in OT networks could negatively affect the availability of other protocols and its usage will be carefully considered in the project. Moreover, the event-based operation of PDP is not suitable for implementation of active network measurement functionality on CPN.

4.1.2 Passive network measurements

Passive network measurement is based on observing already present traffic without generating any new traffic. This type of measurement is more suitable for OT networks, as it does not interfere with existing traffic and cannot be detected by security appliances. It focuses on packet and byte counters, timing information related to departure and arrival of datagrams, etc. These metrics can also be aggregated per interface or data flows.

Passive network measurement can be done on any networking or end device, but an end device might not be able to process traffic destined for other devices. Traditional network devices typically collect per port statistics such as number of received packets, bytes, errors, etc. In order to get detailed information about datagrams, port mirroring has to be configured. Such a solution is not very scalable as the mirroring port can only process a traffic amount corresponding to the port speed. Furthermore, it requires an additional system for traffic analysis

which is done offline, thus, the original traffic cannot be controlled in any way. Depending on the amount of traffic, such a system can present significant costs as large storage space and high performance CPU might be required. Typical tools for passive network measurement include:

- SNMP.
- Cisco IOS Netflow.
- Packet monitoring (via traffic capture).

Passive network measurement will be done by COMML for monitoring all the data from EPES protocols. Using CPN for passive measurement will ensure that the network traffic will not be influenced in any way and that the process will be invisible for the grid devices.

4.2 Networking protocol measurement requirements

This section presents header fields of general networking protocols, whereas EPES protocols required in COCOON are explained in Section 4.3. The purpose is to provide a general overview of fields and values which might be captured by a PDP and which will be used in the higher layers of the CPN. Only those protocols used in the project are described.

The "CSL Services" column in the subsequent tables marks services from the CSL which will be using the data: either AD or FDII. Protocols for which there are no application requirements at this stage do not have this column, but the tables are still presented, as it is expected they will be used within the project.

4.2.1 Metadata

The COMML layer of the CPN will have to maintain several metadata fields which are not included in datagram header fields. These are shown in Table 4.1. These fields will be appended to every datagram which is being processed by the COMML and those metadata will be removed before transmission.

Table 4.1: Metadata fields

Field	Length (bits)	Description
Incoming port	8	On the networking device
Timestamp	Variable	When received on the networking device
Length	Variable	Length of the entire datagram

4.2.2 Ethernet

Ethernet is a layer 2 protocol responsible for frames delivery on the local level, for example within a substation. It is the minimum layer which always has to be present for datagrams encapsulation. The higher layer protocols such as IP and TCP are optional. GOOSE and SV protocols, which are of primary focus in the CPN, are encapsulated only in the Ethernet frame and, therefore, do not contain higher layer headers. The Ethernet protocol frame structure is shown in Figure 4.2.

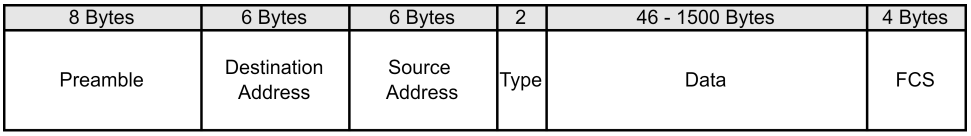

Figure 4.2: Ethernet protocol header structure

Ethernet header fields which can be processed by PDP are shown in Table 4.2. The most important fields which will be critical for the AD service are destination and source MAC addresses and the Ethertype field. The addresses are represented in format of 6 groups of two hexadecimal digits separated by symbols : or -. The Ethertype field indicates what protocol is encapsulated in the data field - for example IP or GOOSE.

Table 4.2: Ethernet protocol header

Field	Length (bits)	Description	CSL Services
Destination address	48	00:00:00:AA:AA:AA format	AD
Source address	48	00:00:00:AA:AA:AA format	AD
802.1Q (optional)	4	VLAN ID	
Ethertype	16	Protocol type	AD
FCS	32	Frame Check Sequence	

* AD = Anomaly Detection

4.2.3 IPv4

Industrial networks are typically using IPv4 over IPv6 as the usage of the older protocol simplifies configuration and management while improves performance on low resources devices. As these networks are relatively small and isolated, disadvantages of IPv4 such as limited number of addresses do not pose a problem. Moreover, several redundancy technologies including Virtual Router Redundancy Protocol (VRRP) might not support simultaneous operation of IPv4 and IPv6 [25].

IPv4 is responsible for delivery over a global network and must be used for protocols which need to leave the local network, for example the IEC104 which exchanges data between a substation (LAN) and the SCADA operation center via a WAN. In the COCOON project, IPv4 will be used for IEC104 and Modbus TCP/IP protocols. IPv4 packet structure is shown in Figure 4.3.

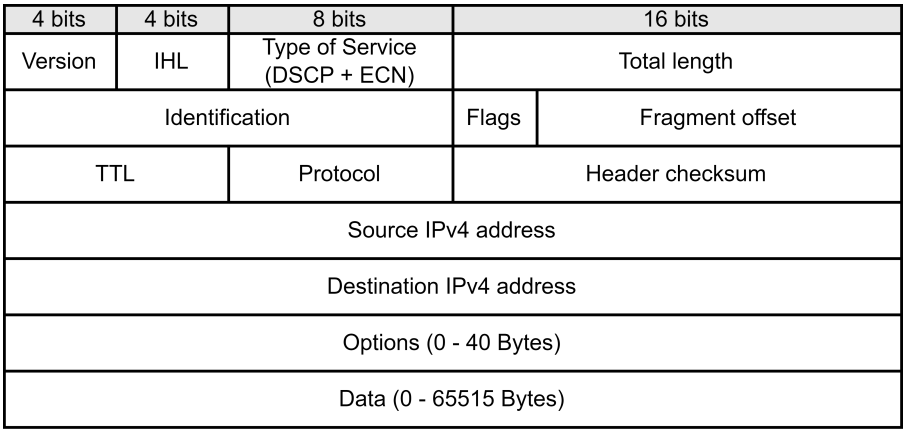

Figure 4.3: IPv4 header structure

IPv4 header fields which can be processed by PDP are shown in Table 4.3. The most important fields are source and destination IPv4 addresses. These are represented in doted decimal format.

Table 4.3: IPv4 header fields

Field	Length (bits)	Description
Version	4	IPv4 or IPv6
Header Length	4	Only the header
DSCP	6	Differentiated Services Code Point
ECN	2	Explicit Congestion Notification
Total length	16	Length of the entire datagram
Identification	16	ID if fragmented
Flags	3	For fragmentation
Fragment offset	13	Position of the fragment
Time to live	8	Decremented with every hop
Protocol	8	Higher layer (TCP or UDP)
Header checksum	16	Header error verification
Source address	32	127.0.0.1 format
Destination address	32	127.0.0.1 format

4.2.4 TCP

TCP is the most common protocol on the transport layer, responsible for distinguishing between target applications running on the device. An alternative to TCP is UDP which is made for time-sensitive applications and it uses minimal header for low overhead and best effort delivery without establishing a connection. These properties are not preferred in EPES scenarios and, therefore, this protocol will not be considered within the COCOON project.

Unlike UDP, TCP is connection oriented which means that a connection is first established between two applications via a so-called three-way handshake. Keep alive messages are then periodically exchanged before the connection is terminated. TCP has advanced features such as congestion control (allows maximum utilization of the link capacity) or re-transmission of lost datagrams.

TCP is used by IEC104 and Modbus TCP/IP protocols, used within the energy community and PV power plant respectively. TCP has a more complex segment header structure when compared to UDP as shown in Figure 4.4.

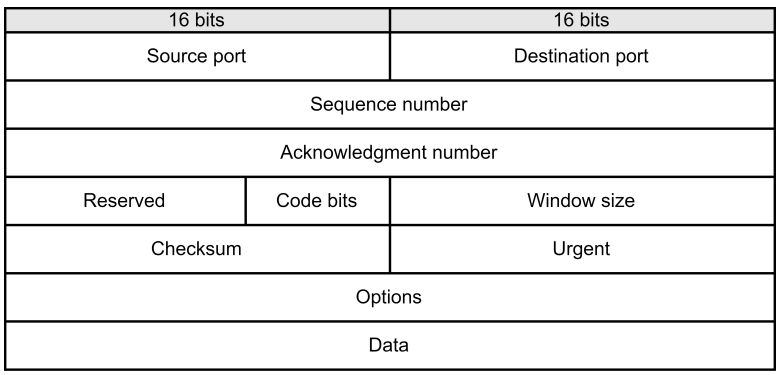

Figure 4.4: TCP header structure

TCP header fields which can be processed by PDP are shown in Table 4.4. The most important fields are source and destination port numbers which are used for delivery to the correct application running on the device.

Table 4.4: TCP header fields

Field	Length (bits)	Description
Source port	16	0 - 65535 range
Destination port	16	0 - 65535 range
Sequence number	32	Incremented with every segment
ACK	32	Acknowledgment number
Reserved	8	Data offset + reserved
Flags	8	For connection management
Window size	16	How many bytes before ACK required
Checksum	16	Error check, header only
Urgent pointer	16	Index to urgent data byte
Options	Variable	Optional parameters

4.3 EPES protocol measurement requirements

This section describes the structure of communication protocols used in EPES networks. Only protocols used in the project are described. These protocols use general communication protocols described in the previous section. As the communication monitored by CPN is bidirectional, data values of protocols can contain either sensor readings statuses or control setpoints. This is shown in data fields in appropriate tables.

4.3.1 IEC 61850 GOOSE Protocol

The GOOSE protocol is used for event-based communication within a digital substation boundary. For this reason, it is encapsulated only within the Ethernet header as shown in Figure 4.5. GOOSE frames, therefore, do not use any higher layer such as IP or TCP.

The entire data of the GOOSE message form a so-called GOOSE Protocol Data Unit (GPDU) which is inserted into the data payload of the Ethernet frame. GPDU then contains its own header and Application Protocol Data Unit (APDU) which contains the most important fields including data itself.

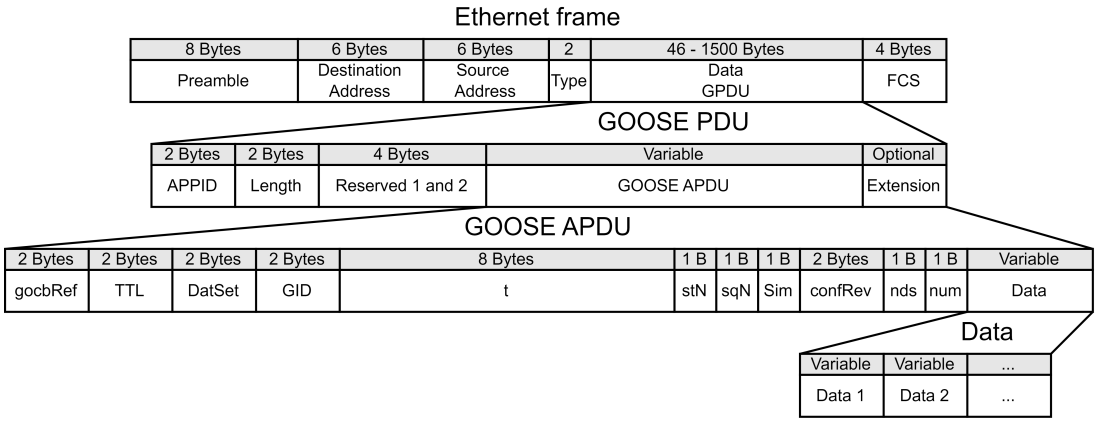

Figure 4.5: GOOSE protocol structure

GOOSE protocol header fields which can be processed by PDP are shown in Table 4.5. The most important ones which will be used by the AD service are the dataset name, status and sequence numbers as well as the number of values inserted in the message. Status and sequence numbers are especially important as they are the primary targets of False Data Injection (FDI) attacks and can be also used in Denial of Service (DoS) attacks.

Table 4.5: GOOSE protocol message structure

Field	Length (bits)	Description	CSL Services
APPID	16	Application ID	
Length	16	Message length	
Reserved 1	16	Reserved field	
Reserved 2	16	Reserved field	
GOOSE APDU (variable)		Application Protocol Data Unit	
gocbRef	16	GOOSE Control Block Reference	
timeAllowedtoLive	16	Max time before message discarded	
DatSet	Variable?	Dataset name	AD
goID	Variable?	GOOSE ID	
t	64	Last status change timestamp	
stNum	8	Status number	AD
sqNum	8	Sequence number	AD
simulation	8	Simulated IED	
confRev	16	Configuration revision	
ndsCom	8	Needs commissioning	
numDatSetEntries	8	Number of elements within the dataset	AD
allData (variable)		Can contain multiple values	
Data: floating-point	40	Status / setpoint	

* AD = Anomaly Detection

4.3.2 IEC 61850 Sampled Values (SV) Protocol

The structure of the SV protocol is very similar to the GOOSE protocol as it is also encapsulated only in the Ethernet header and does not contain headers of higher layers as shown in Figure 4.6. The main difference between the two is in the format of values, which in case of the SV protocol are encoded into a single sequence of data. SV APDU structure can contain up to 8 Application Service Data Unit (ASDU) which are identified by the unique svID [26].

The SV protocol header fields which can be processed by PDP are shown in Table 4.6. The most important fields used by the AD service are application ID and length of the entire SV message without headers of other protocols.

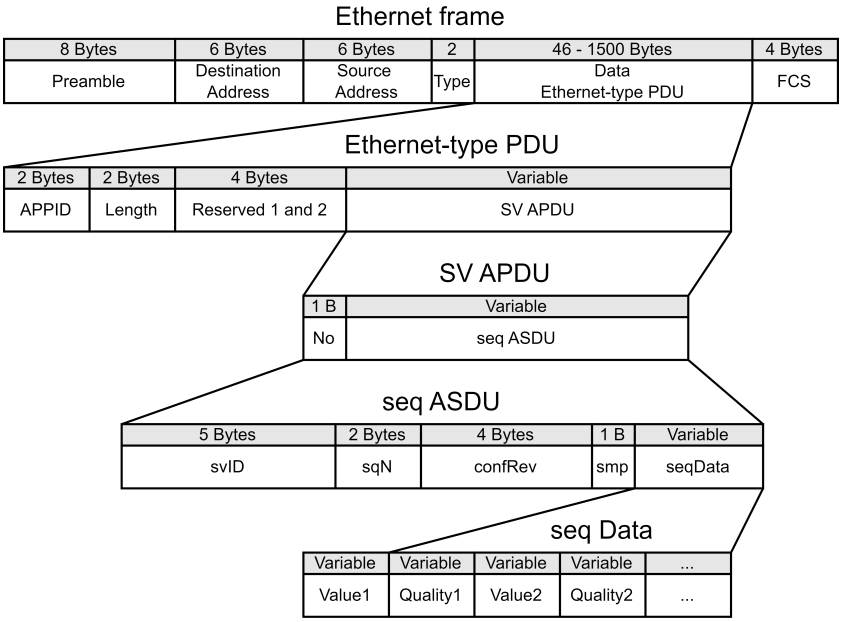

Figure 4.6: SV protocol structure

Table 4.6: SV protocol message structure

Field	Length (bits)	Description	CSL Services
APPID	16	Application ID	AD
Length	16	Message length	AD
Reserved 1	16	Reserved field	
Reserved 2	16	Reserved field	
SV APDU (variable)		SV Application Protocol Data Unit	
noASDU	8	Number of ASDUs	
seqASDU (variable)		Application Service Data Unit	
svID	40	Sampled Values Identifier	
smpCnt	16	Index of the SV message	
confRev	32	Configuration revision	
smpSynch	8	Synchronization	
seqData	512 (variable)	Sequence of measured values	

* AD = Anomaly Detection

4.3.3 IEC 60870-5-104 Protocol

IEC104 is used for communication between substations and control center and the messages use TCP and IP for global delivery and reliability. There are several message types with slightly different structures. Figure 4.7 shows the I type message as an example, as this is the most common type used for values reporting. The message is encapsulated within Ethernet, IP and TCP headers. Reported values from sensors are stored in the so-called Information Object Address (IOA) structures. Every message can contain multiple of these objects.

IEC104 communication relies on TCP to establish the connection, keep it alive, and terminate it when the transmission is done. This means that there are TCP messages sent in parallel with the IEC104 communication itself.

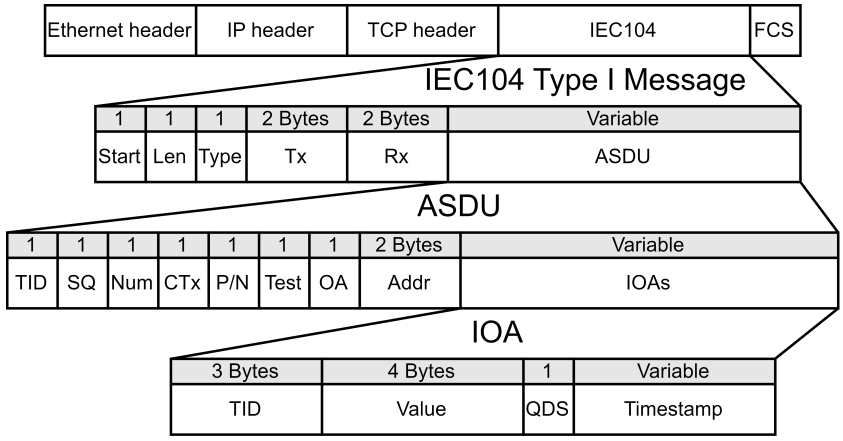

Figure 4.7: IEC104 protocol structure

IEC104 protocol header fields which can be processed by PDP are shown in Table 4.7. The table shows I and S message types. The I type is used for values reporting and the S type for acknowledgment of those messages.

Table 4.7: IEC104 protocol message structure

Field	Length (bits)	Description	CSL Services
I Message Type			
Start	8	Start of the message	
ApduLen	8	Length of the APDU	
Type	8	APCI type	
Tx	16	Send sequence number	
Rx	16	Receive sequence number	
ASDU (variable)			
TypeId	8	ASDU type ID	
SQ	8	Sequence	
NumIx	8	Number of information objects	
CauseTx	8	Cause of transmission	
Positive/Negative	8	Confirmation of activation	
Test	8	For testing messages	
OA	8	Originator address	
Addr	16	Common address of ASDU	FDII
IOAs (0-n, variable)			
IOA	24	Information Object Address	
Value (optional)	32	Status / setpoint	FDII
QDS (optional)	8	Quality descriptor	
Timestamp (optional)	56	Can be in various formats	
S Message Type			
Start	8	Start of the message	
ApduLen	8	Length of the APDU	
Type	8	APCI type	
Rx	16	Receive sequence number	

* FDII = False Data Injection Identification

The most important fields which will be used for the FDII service are the address of ASDU and data values inserted in IOAs.

4.3.4 Modbus TCP/IP

Modbus TCP/IP acts in a similar role as IEC104 since it is used within PV power plants to communicate inverters with the Power Plant Controller (PPC). It is also based on a client-server type of communication. For this reason, it uses the same encapsulation within Ethernet, IP and TCP headers as shown in Figure 4.8. The main difference between IEC104 and Modbus TCP/IP is a more simplistic data encapsulation [5].

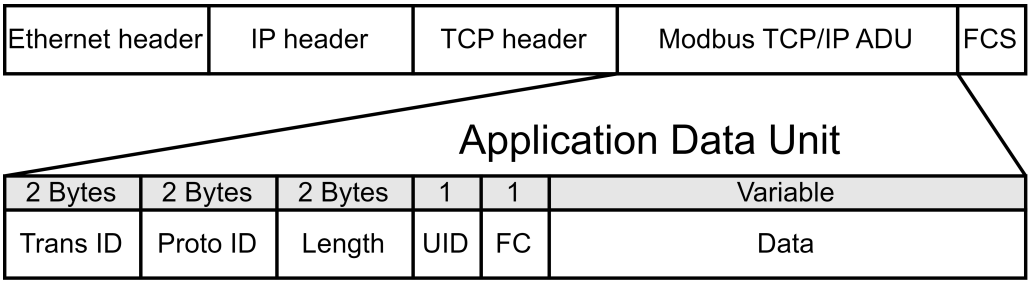

Figure 4.8: Modbus TCP/IP structure

Modbus TCP/IP header fields which can be processed by PDP are shown in Table 4.8. Data are inserted directly and not within an IOA like in the IEC104 protocol. FDII service should be using similar header fields like in case of IEC104, especially data, but the concrete values will be determined in the subsequent work.

Table 4.8: Modbus TCP/IP message structure

Field	Length (bits)	Description
Transaction Identifier	16	For transaction pairing
Protocol Identifier	16	Reserved, 0 for Modbus
Length	16	Byte count of remaining fields
Unit Identifier	8	Remote server ID
Function code	8	Requested action
Data	Variable	Status / setpoint

5 CPN algorithmics for packet-level primitives composition

This section describes how services of the CSL are translated into COMML μNFs and their packet-level primitives. The synthesis of packet-level primitives and the algorithmics underpinning this synthesis act as the basis for enforcing control in the EPES network through application-level services on the CPN CSL. Hence, examples of CSL services enabling control via attack mitigation are provided in subsequent sections. These examples are, however, at the early research stage as they are mostly part of the task T1.5: *Threat mitigation strategies*, which has started only recently. Their algorithmic forms as well as targeted mitigation areas can therefore be modified in the future.

A set of low-level primitives comprises the fundamental blocks of μNF for cyber CSL services and devise a composable PDP that can be configured to incorporate any subset of those components required to support a specific functionality. Packet-level primitives are potentially of time-critical nature (e.g., per-packet operations at line rate), with high reusability and sharing potential between different higher level detection and mitigation modules (e.g., timestamping and flow statistics are both needed for traffic characterization and volumetric attack detection). Providing such functionality as part of the PDP architecture (as opposed to, e.g., virtualized network function components) will improve throughput by eliminating processing redundancy and will reduce packet latency by minimizing context and Input / Output (I/O) switching.

Figure 5.1 shows such a composition for packet-level primitives. Services of the CSL are decomposed to μNFs which are formed from one or more packet-level primitives. Those μNFs are then pre-compiled by the IOL, utilizing Linux header files and libraries. Upon a northbound API request from the corresponding service, a μNF is installed to the processing pipeline by the μNF loader. If the μNFs require storage space for saving states or any other data, they can interact with eBPF maps created during the installation process of the μNF. A received message first goes through a metadata prepend function which saves information such as received port number. Then the message is stored in the ring buffer before entering the eBPF processing pipeline and the first μNF.

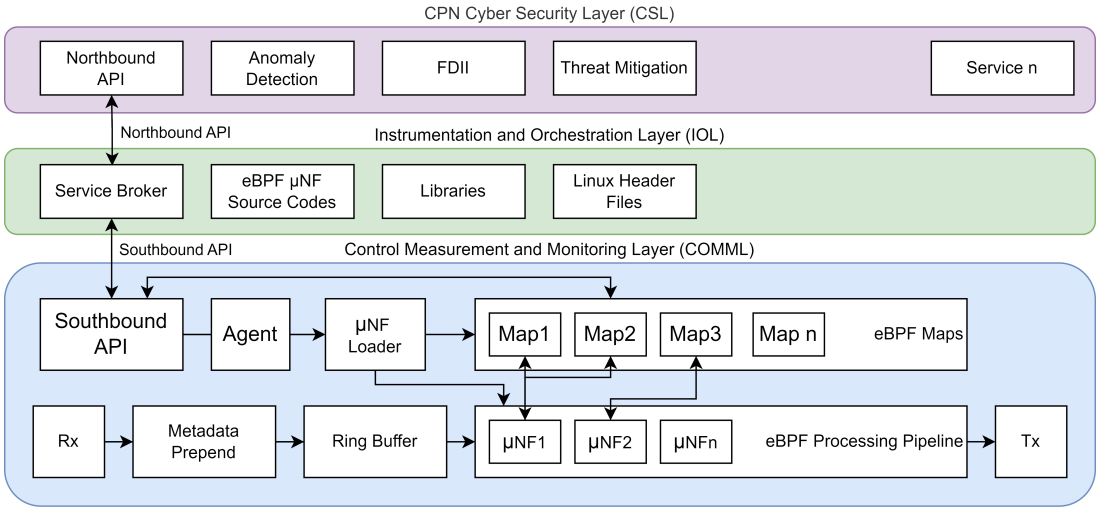

Figure 5.1: Composition of packet-level primitives in the CPN for enabling CSL services

5.1 Control requirements in COMML

Support for control-based high-level CSL services will require adequate control packet-level primitives. These will be defined based on concrete requirements of such services, but might include the following ones:

- Write to interface.
- Packet dropping.
- Packet throttling.
- Traffic reroute.
- Traffic mirroring.
- Header field modification.

These control packet-level primitives might be combined to form a control-oriented μNF. For example, a threat mitigation CSL service might need a μNF to modify a header field of the Ethernet protocol to separate traffic into a different VLAN and forward it to a specific port. Such a μNF will be composed from header field modification and traffic reroute primitives. First, the μNF will be programmed and its source code pre-compiled and stored in IOL. Then, when the CSL service will be started, it will use the northbound API to request service broker to install the μNF into the COMML pipeline. The service broker then utilizes the southbound API to load the function. More examples of CPN algorithmics used in COCOON are shown in the following sections.

5.2 Machine Learning (ML) for measurement and control

ML allows computers to learn from data and improve their performance over time without being explicitly programmed. Network measurement and control are essential components of the modern EPES. They are critical to the communication and the network security of the power grid. With more and more high-performance devices being deployed in the power grid, ML-based strategies are becoming more realistic and popular in the network measurement and control of the power grid. ML technologies offer easy and quick means of analyzing acceptable and appropriate decisions for the smooth operation of the smart grid [27].

The introduction of SCADA and Phasor Measurement Units (PMU) has exposed networks to a range of risks and vulnerabilities associated with open communication technologies, such as the Internet which causes the main worry for energy operators and stakeholders. In security and stability investigations, numerous statistical models and signal-processing methods have been put forth over time. Although the conventional methods have shown adequate performance, they have proven to be costly, time-consuming, and computationally inefficient as they struggle to meet the complex modern power system's rising analytical needs.

In recent times, ML techniques have been vastly used in modelling and monitoring complex applications. Numerous ML technologies such as Long Short Term Memory (LSTM) Reinforcement Learning (RL), Convolutional Neural Networks (CNN), Support Vector Machine (SVM), and Genetic Algorithm (GA) etc. have been proposed in various capacities involving power system security and stability assessments [28] [29]. Cyber protection is the core objective within the COCOON project through the cross-layering operations across the CPN architecture.

In the context of COCOON, as addressed by Figure 5.2, a ML-based detection service can be used to detect attacks in three particular steps: data preprocessing, training process, and detection process. Data provided to the service are collected by COMML measurement and monitoring functionality organized and instrumented as μNF.

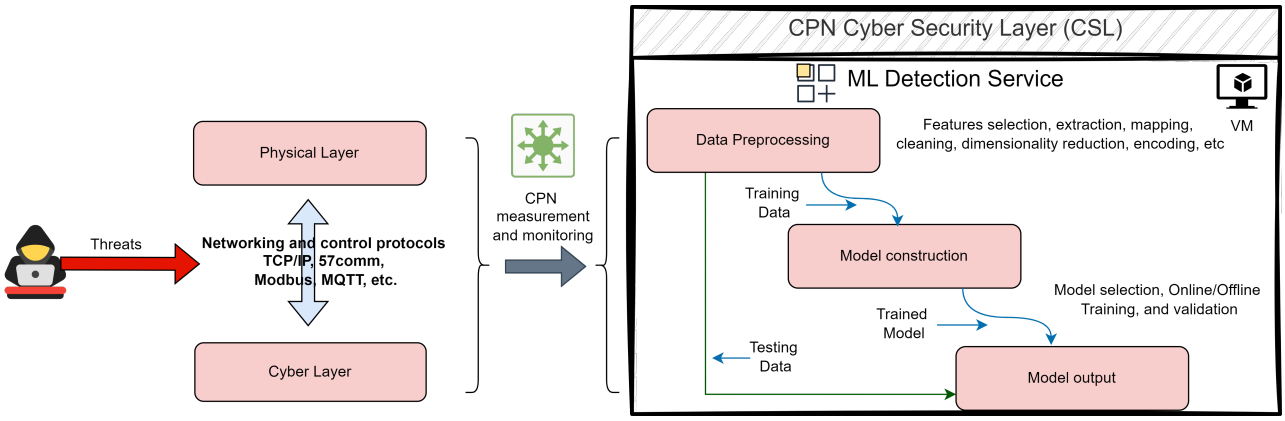

Figure 5.2: Exemplar of ML modelling of the cyber attack detection process in COCOON

Deep Learning (DL)-based detection models have been applied in IDS solutions tailored for smart grid scenarios. The potential of DL algorithms like CNN, LSTM, and Recurrent Neural Networks (RNN) are capable of capturing the attack patterns from the network traces in the smart grid. AD in the digital substation is critical to the cyber security of the modern power grid. To detect anomalous network traffic within a digital substation, GOOSE messages have been identified as key indicators [30] and will be fully utilized within the AD CSL service of the CPN. The stNum and sqNum which represent the status number and sequence number of the GOOSE message are mostly used as input for the DL-based AD model in the substation. Hence, such formulations can effectively track and profilie anomalous patterns and further identify attack scenarios within such setups.

RL focuses on training software to make decisions to achieve the optimums by trial and error. Cyber attack mitigation demands an algorithm which can continuously improve and adapt the mitigation strategy based on new data and observed threats. Moreover, RL provides proactive and dynamic defense mechanisms, helping to safeguard critical infrastructure against evolving cyber risks efficiently. This will be the main objective of the task T1.5: *Threat mitigation strategies.*

5.3 Deep Reinforcement Learning (DRL)

RL is an agent interacting with the environment, and learning an optimal policy. DRL, other than traditional RL methods, uses deep neural networks to approximate any of the following components: Q-function $\hat{q}(s, a; \theta)$, policy function $\pi(a|s; \theta)$, and the reward function [31]. The Q-function, also known as the action-value function, returns the optimal reward since the agent starts at state s and executes action a; the policy function indicates the agent at state s to execute action a; the reward function will provide a feedback signal that indicates the performance of the state-action pair. The complexity of the modern power grid causes the dimension of the state space to surge. The curse of dimensionality introduces the need for a faster way to calculate these functions in RL algorithms. Deep neural networks provide a robust approach to approximate the functions, bypassing the limitations of traditional RL approaches and making this concept suitable for control-based CSL services of the CPN such as for threat mitigation.

5.3.1 DRL Application in Power Grids

Prevention of stealthy attacks is difficult, but all attacks and anomalies have quantifiable effects and symptoms that enable an experienced analyst to deduce new signs for misuse-based classifiers to detect. It has long been anticipated that anomaly-based intrusion detectors will overcome this problem by effectively utilizing statistical techniques. However, misused signature-based IDS are prone to this problem [32].

In the domain of the power grid, the environment is changing rapidly, and features that indicate the status of the substation are immense. For example, the network traffic like throughput, latency, and Round Trip Time (RTT) of each link, the payload of the packet, and even the voltage or power can be part of the state space. Hence, the learning process of RL will be significantly inefficient when using traditional RL approaches. With the help of the Deep Neural Network (DNN) and COMML measurement and monitoring functionality, massive input data can be processed, and the most relevant information for decision-making will be identified.

In the previous work done at UGLA, the use of the State-Action-Reward-State-Action (SARSA) algorithm was proposed at the network flow level to confront Distributed Denial of Service (DDoS) scenarios. The RL agents interact with the online services and explore the optimal policy to drop the malicious traffic packets hence mitigating the DDoS attacks. This work, however, in the current form, would be infeasible to replicate in EPES environments due to the traffic criticality. Nevertheless, adoption of the SARSA algorithm properties in COMML will be explored.

In summary, the application of RL in mitigating cyber threats in power grid is still in its early stages, with limited related work available. Nonetheless, RL holds promise as an effective solution for managing the dynamic and varied statuses of power grids and digital substations and it's suitability as a CSL service will be explored in task T1.5 of the COCOON project.

6 Conclusions

This deliverable presented the key concepts associated with the operations of communication networks and EPES communication protocols. Section 2 presented the ISO/OSI and TCP/IP models which defines the fundamental underlying operations of the computer networks, by describing working of every EPES protocol used in the project, and by mapping these protocols to the models. This will benefit the partners of the COCOON project towards an equal understanding of the terminologies and common knowledge of the CPN and communication protocols to make collaborative work more efficient. Moreover, the provided generic material can benefit the EPES operators, stakeholders, and the wider community.

The main focus of the deliverable, Section 3, however, was on the COMML of the CPN. The deliverable presented the architecture and basic properties. Based on online and offline discussions with the project partners, architectural requirements were gathered which facilitated the architectural choices based on existing PDP paradigms. A three layer architecture based on the SDN paradigm was selected for the CPN and eBPF was chosen as the implementation technology for the COMML. Concepts of NFV and network function chaining, which will support COMML development, were also highlighted together with detailed COMML properties based on the eBPF features and limitations. While eBPF is the most flexible PDP up to date, it still has its boundaries in terms of size of the code and supported functions. These limitations are important for achieving safety, security and performance levels required for high demanding production environments including OT networks. As eBPF is now widely used in major data centers where it delivers unprecedented performance, reliability and flexibility, it provides the best choice for the CPN requirements in power grid networks. At the same time, it has an architecture which can be generalized and used in any OT environment.

Next, measurement and monitoring properties of COMML were summarized in Section 4. This section thoroughly presented the structure of protocol headers of generic and EPES communication protocols, which will be used in the project and these are the most common in current EPES network implementations. Hence, it included their inner structures and a clear overview of all the header fields and sizes. The presented material was used by partners developing CSL services, specifically TUD for AD, AUTH and USE for FDII, and allowed them to specify, which header fields are expected to be needed for effective operation of the CSL services. This will be important for future development of the CPN architecture and in decomposition of high-level CSL services to μNFs managed by the IOL and implemented by the COMML. As the decomposition to μNFs can be done in many ways, this material and information gathered will support future development in regards to optimal functionality and use of effective tools including NFV, network function chaining and eBPF features.

Finally, CPN algorithmics for packet-level primitives composition were described in Section 5. The description included control requirements in COMML and examples of control-based threat mitigation CSL services which will allows the CPN to manipulate the network traffic.

In conclusion, this deliverable laid out comprehensive foundations for the lowest CPN layer, COMML and its operations. These foundations will be utilized for the implementation of COMML as well as for the development of IOL and CSL, respectively, which will be presented in the upcoming deliverable D4.2: *COCOON System Architecture* that will serve as the blueprint for the CPN development and practical implementation to be tested in the COCOON pilots.

Bibliography

[1] ISO. *ISO/IEC 7498-1:1994 Standard*. Online (accessed 2024-06-29): `https://www.iso.org/standard/20269.html`. 2024.

[2] *IEC 61850:2024 SER Series*. International Standard. Online (accessed 2024-07-03): `https://webstore.iec.ch/publication/6028`. IEC, 2024.

[3] *IEC 61850-9-2:2011+AMD1:2020 CSV Consolidated version*. International Standard. Online (accessed 2024-07-03): `https://webstore.iec.ch/publication/66549`. IEC, 2020.

[4] *IEC 60870-5-104:2006+AMD1:2016 CSV Consolidated version*. International Standard. Online (accessed 2024-07-03): `https://webstore.iec.ch/publication/25035`. IEC, 2024.

[5] Acromag. *INTRODUCTION TO MODBUS TCP/IP*. Technical Reference 8500-765-A05C000. Online (accessed 2024-06-30): `https://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf`. Acromag, Inc., 2005.

[6] M. Silveira. *IEC 61850 9-2 Sampled Values, Wireshark, and the Cloudy effect*. Online (accessed 2024-07-05): `https://www.linkedin.com/pulse/iec-61850-9-2-sampled-values-wireshark-cloudy-effect-silveira/`. 2020.

[7] X. Long. "Primitives For match-action In theory and practice". Online (accessed 2024-06-29): `https://ecommons.cornell.edu/server/api/core/bitstreams/746743cc-0aa9-48ee-8a69-7aa08042cbb4/content`. PhD thesis. Cornell University, 2021.

[8] P. Govindan Kannan and M. C. Chan. "On programmable networking evolution". In: *CSI Transactions on ICT* 8 (2020), pp. 69–76.

[9] S. Jain et al. "B4: Experience with a globally-deployed software defined WAN". In: *ACM SIGCOMM Computer Communication Review* 43.4 (2013), pp. 3–14.

[10] L. Foundation. *Open vSwitch*. Online (accessed 2024-06-29): `https://www.openvswitch.org/`. 2024.

[11] KernelNewbies. *Linux_3.3*. Online (accessed 2024-06-29): `https://kernelnewbies.org/Linux\_3.3`. 2017.

[12] S. E. Laboratories. *SEL-5056 Software-Defined Network Flow Controller*. Online (accessed 2024-05-17): `https://selinc.com/products/5056/`. 2024.

[13] ONF. *P416 Language Specification version 1.2.3*. Online (accessed 2024-05-17): `https://staging.p4.org/p4-spec/docs/P4-16-v-1.2.3.html`. 2022.

[14] P. Salva-Garcia et al. "Xdp-based smartnic hardware performance acceleration for next-generation networks". In: *Journal of Network and Systems Management* 30.4 (2022), p. 75.

[15] P4lang. *BEHAVIORAL MODEL (bmv2)*. Online (accessed 2024-06-24): `https://github.com/p4lang/behavioral-model`. 2024.

[16] eBPF Docs. *Tail calls*. Online (accessed 2024-06-28): `https://ebpf-docs.dylanreimerink.nl/linux/concepts/tail-calls/`. 2024.

[17] S. Miano et al. "Creating complex network services with ebpf: Experience and lessons learned". In: *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. IEEE. 2018, pp. 1–8.

[18] J. D. Brouer. *XDP - eXpress Data Path*. Online (accessed 2024-05-23): `https://prototype-kernel.readthedocs.io/en/latest/networking/XDP/`. 2016.

[19] Netronome. *Agilio CX SmartNICs*. Online (accessed 2024-03-11): `https://www.netronome.com/products/agilio-cx/`. 2024.

[20] Mininet Project Contributors. *Mininet An Instant Virtual Network on your Laptop (or other PC)*. Online (accessed 2024-06-26): `https://mininet.org/`. 2022.

[21] DPDK Project. *DPDK*. Online (accessed 2024-05-23): `https://www.dpdk.org/`. 2024.

[22] Cilium Authors. *'BPF Architecture,* Online (accessed 2024-06-21): `https://docs.cilium.io/en/stable/bpf/architecture/`. 2024.

[23] yunwei37. *eBPF Advanced: Overview of New Kernel Features*. Online (accessed 2024-06-04): `https://medium.com/@yunwei356/ebpf-advanced-overview-of-new-kernel-features-in-2022-a90c6a294a78`. 2023.

[24] D. P. Pezaros. "Network Traffic Measurement for the Next Generation Internet". Online (accessed 2024-06-29): `https://eprints.lancs.ac.uk/id/eprint/12698/`. PhD thesis. Lancaster University, 2005.

[25] Cisco. *Substation Automation Implementation Guide v. 3.1*. Online (accessed 2024-07-05): `https://www.cisco.com/c/dam/en/us/td/docs/solutions/Verticals/Utilities/SA/3-1/IG/SA-3-1-IG.pdf`. Cisco. 2023.

[26] Typhoon HIL Documentation. *IEC 61850 Sampled Values protocol*. Online (accessed 2024-06-29): `https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/iec_61850_sampled_values_protocol.html`. 2024.

[27] K Umapathy et al. *Machine Learning Applications for the Smart Grid*. Springer, 2023, pp. 251–270.

[28] O. A. Alimi, K. Ouahada, and A. M. Abu-Mahfouz. *A review of machine learning approaches to power system security and stability*. IEEE, 2020, pp. 113512–113531.

[29] E. Bader and H. H. O. Nasereddin. "Using Genetic Algorithm in Network Security". In: *International Journal of Research and Reviews in Applied Sciences* 5 (Nov. 2010), pp. 148–154.

[30] H. Shen, Z. Xiao, and Y. Liu. *Research of SV and GOOSE Message Transmission with Common Port in Highly Reliable Substations*. IEEE. 2024, pp. 2262–2267.

[31] Y. Li. *Deep reinforcement learning: An overview*. 2017.

[32] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. *Network anomaly detection: methods, systems and tools*. 1. IEEE, 2013, pp. 303–336.