Degree Project in Information and Communication Technology

First cycle, 15 credits

Architecture Analysis of Virtualized Protection, Automation, and Control Systems

KUNDAN VENGALA
VIGNESH RANGA

Architecture Analysis of Virtualized Protection, Automation, and Control Systems

KUNDAN VENGALA

VIGNESH RANGA

Bachelor's Programme in Information and Communication Technology
Date: May 2026

Supervisors: Eren Berk Kama, Vaclav Struhar
Examiner: Anders Västberg
School of Electrical Engineering and Computer Science
Host company: Confidential
Swedish title: Arkitekturanalys av virtualiserade skydds-, automations- och styrsystem

Abstract

Virtualized protection, automation, and control replaces dedicated protection relay hardware with protection software hosted on shared computing platforms. This thesis analyzes two virtualization technologies used in virtualized protection systems in digital substations: container-based virtualization and virtual machine based virtualization.

The thesis conducts a literature review of direct virtualized protection studies using IEEE Xplore as the primary academic source. The retained evidence is classified by virtualization technology and compared through four questions: what virtualization technology and software stack were used, what each study built and how the implementation was validated, and what deployment challenge remains. Three industrial approaches are analyzed separately: SIPROTEC V as a commercial virtualized protection platform, ABB centralized protection and control as an architecture transition concept, and SEAPATH as an open industrial platform layer for substation virtualization.

As a complement to the literature review, an experiment deployed containers and virtual machines on the same open substation virtualization platform. The goal was to compare whether both virtualization technologies could be deployed with automation, assigned separate static network addresses, and verified through basic communication. Both were successfully deployed, confirming platform-level deployment feasibility.

The findings show that virtualization technology choice alone is insufficient for virtualized protection deployment. A deployable platform must also address compute isolation, IEC 61850 communication architecture, time synchronization, redundancy, cybersecurity, lifecycle management, and engineering workflow. Containers are attractive for packaging, scalability, and service flexibility, while virtual machine based virtualization remains important for stronger workload isolation and strict execution boundaries. Virtualized protection deployments will likely use both virtualization technologies according to workload criticality and operational role.

Keywords: virtualized protection; vPAC; IEC 61850; digital substation; containers; virtual machines; hypervisors; virtualization technologies; real-time systems

Sammanfattning

Virtualiserade skydds-, automations- och styrsystem ersätter dedikerad relähårdvara med skyddsprogramvara som körs på delade beräkningsplattformar. Denna avhandling analyserar två virtualiseringsteknologier som används i virtualiserade skyddssystem i digitala ställverk: containerbaserad virtualisering och virtuell maskinbaserad virtualisering.

En litteraturgranskning av direkta studier om virtualiserat skydd genomförs med IEEE Xplore som primär akademisk källa. De inkluderade källorna klassificeras efter virtualiseringsteknologi och jämförs utifrån fyra frågor: vad som byggdes, vilken programvara som användes, vad som testades och vilka driftssättningsutmaningar som kvarstår. Tre industriella ansättser analyseras separat: SIPROTEC V som en kommersiell virtualiserad skyddsplattform, ABB:s centraliserade skydd och styrning som ett arkitekturövergångskoncept, samt SEAPATH som ett öppet industriellt plattformslager för ställverksvirtualisering.

Ett driftssättningsexperiment på en plattform baserad på SEAPATH jämför containerbaserad och virtuell maskinbaserad driftssättning på samma värd. Båda virtualiseringsteknologierna driftssättes och verifierades framgångsrikt genom grundläggande nätverkskommunikation, vilket bekräftar driftssättningsgenomförbarhet på plattformsnivå.

Resultaten visar att valet av virtualiseringsteknologi ensamt är otillräckligt för driftssättning av virtualiserat skydd. En driftssättningsbar plattform måste också hantera beräkningsisolering, IEC 61850-kommunikationsarkitektur, tidssynkronisering, redundans, cybersäkerhet, livscykelhantering och ingenjörsarbetsflöde. Containrar är attraktiva för paketering, skalbarhet och tjänsteflexibilitet, medan virtuell maskinbaserad virtualisering förblir viktig för starkare arbetsbelastningsisolering och strikta exekveringsgränser. Driftssättningar av virtualiserat skydd kommer sannolikt att använda båda virtualiseringsteknologierna baserat på arbetsbelastningens kritikalitet och operativa roll.

Nyckelord: virtualiserade skyddssystem; vPAC; IEC 61850; digitalt ställverk; containrar; virtuella maskiner; hypervisorer; virtualiseringsteknologier; realtidssystem

Acknowledgements

This thesis was carried out as part of a bachelor degree project at KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science. The work was carried out in collaboration with a confidential host company.

The authors would like to thank the host company for supporting the experimental part of this work and for providing access to a relevant technical environment. No confidential product details, internal project identifiers, or proprietary data appear in this thesis.

The authors also thank their supervisors at KTH for their guidance, feedback, and continued support throughout the project. The constructive criticism on scope, clarity, and evidence framing significantly improved the final result.

The authors declare that AI-assisted writing tools were used during the preparation of this thesis, primarily for language editing and structural revision suggestions. All technical content, analysis, conclusions, and citations were reviewed and verified by the authors. The responsibility for the accuracy and integrity of this work rests solely with the authors.

Contents

Abstract i

Sammanfattning ii

Acknowledgements iii

Contents iv

List of Figures viii

List of Tables ix

Abbreviations 1

Definitions 3

1 Introduction 5

1.1 Problem Statement 6

1.2 Purpose 7

1.3 Research Questions 7

1.4 Methodological Approach 7

1.5 Contributions 7

1.6 Delimitations 8

1.7 Structure of the Thesis 8

2 Technical Background 9

2.1 Technology Stack Used in vPAC 9

2.2 Protection Systems 11

2.3 IEC 61850 Communication 12

2.4 Real-Time Requirements 14
2.5 Virtualization Options 14
2.6 Platform and Technology Overview 15

3 Method **17**
3.1 Overview 17
3.2 Why This Method Was Used 18
3.3 Link to Research Questions 18
3.4 Systematic Thesis Roadmap 19
3.5 Literature Review Procedure 21
3.5.1 Search Source 21
3.5.2 Paper Review Questions 24
3.6 Classification 27
3.7 Analysis Method 27
3.8 Experiment 28
3.8.1 Purpose 28
3.8.2 Approach 28
3.8.3 Justification 28
3.9 Method Limitations 29
3.10 Validity Analysis 29

4 vPAC Deployment Architecture **30**
4.1 How the Studies Are Compared 30
4.2 Containers 40
4.3 Virtual Machines 41
4.4 Virtualization Technology Comparison 42

5 Industrial vPAC Platforms **46**
5.1 Why Industry Matters 46
5.2 SIPROTEC V 46
5.3 ABB Centralized Protection and Control 47

5.4 SEAPATH 47
5.5 Industrial Platform Comparison 48
5.6 Deployment Implications 50

6 Deployment Experiment 52

6.1 Experimental Setup 52
6.2 Scope of the Experiment 54
6.3 Docker and Podman 55
6.4 VM Baseline Procedure 56
6.5 Container Image Preparation 58
6.6 Container Deployment Procedure 58
6.7 Container Communication Results 61
6.8 VM Baseline Results 62
6.9 Interpretation of the Measurements 63
6.10 Compute Isolation 64
6.11 Networking 64
6.12 Privilege and Security Boundary 65
6.13 Time Synchronization 65
6.14 Redundancy and Reliability 66
6.15 Scalability and Lifecycle 66
6.16 Result 67

7 Discussion 68

7.1 What the Findings Mean 68
7.1.1 How are Containers and Virtual Machines used to deploy virtualized protection functions in the literature? 68
7.1.2 What deployment characteristics distinguish these virtualization technologies in terms of isolation, performance, scalability, and manageability? 69
7.1.3 How do industrial approaches such as SIPROTEC V, SEAPATH, and ABB centralized protection and control operationalize virtualization concepts for digital substation environments? 69

7.1.4 What engineering factors are most important for deploying virtualized protection functions in real-time environments? 70

7.2 Deployment Implications 70

7.3 Design Recommendations 72

7.4 Limitations 73

7.5 Discussion Summary 75

8 Conclusion **77**

8.1 Research Questions Answered 77

8.2 Key Contributions 78

8.3 Recommendations 79

8.4 Future Work 80

8.5 Concluding Remarks 80

References **82**

List of Figures

2.1 Technology stacks for vPAC compared in this thesis. 10

2.2 Traditional substation protection architecture. 12

2.3 IEC 61850 substation communication architecture. 13

3.1 Thesis argument structure and source roles. 20

3.2 Grouped timeline of retained evidence. Industrial sources serve as platform evidence, not research prototypes. 25

3.3 Source role classification. Direct vPAC studies form the architecture review table; industrial documents and background studies support the deployment discussion. 26

4.1 Distribution of virtualization technologies across the retained evidence sources. 36

4.2 Virtualization technology roles used to interpret the review table. 37

4.3 vPAC deployment layers: from field equipment and IEC 61850 communication through compute infrastructure to protection software and operations. 39

6.1 Experiment topology. 53

List of Tables

3.1 IEEE Xplore search strings used for the direct vPAC literature search. . . 22

3.2 Screening outcome and source roles. 23

4.1 Direct vPAC deployment evidence relevant to the transition from physical IEDs to software hosted protection platforms. 31

4.2 Justification for retaining the main vPAC evidence sources. 34

4.3 Requirements and deployment implications for vPAC platform design. . . 44

5.1 Comparison of industrial vPAC related platform evidence. 49

6.1 Scope of the SEAPATH container deployment experiment. 54

6.2 Reproducible procedure for the SEAPATH container deployment. 58

6.3 Container to container ICMP results from the SEAPATH experiment. . . 62

6.4 VM to VM ICMP results from the SEAPATH experiment. 63

7.1 Evaluation checklist for a future vPAC architecture. 73

7.2 Summary of limitations and their effect on the thesis conclusions. 74

Abbreviations

Abbreviation	Meaning
PAC	Protection, Automation, and Control
vPAC	Virtualized Protection, Automation, and Control
IED	Intelligent Electronic Device
vIED	Virtual Intelligent Electronic Device
VM	Virtual Machine
PTP	Precision Time Protocol
NTP	Network Time Protocol
NIC	Network Interface Card
GNSS	Global Navigation Satellite System
SV	Sampled Values
GOOSE	Generic Object Oriented Substation Event
MMS	Manufacturing Message Specification
MU	Merging Unit
PRP	Parallel Redundancy Protocol
HSR	High availability Seamless Redundancy
TSN	Time Sensitive Networking
RTDS	Real-Time Digital Simulator
SDN	Software Defined Networking
CNI	Container Network Interface
EPICS	Edge Protection and Intelligent Control System (containerized microservice platform from Villén Martínez and Prada Hurtado (2024))
CPC	Centralized Protection and Control
COTS	Commercial Off The Shelf
Ansible	Configuration automation tool used to deploy software to remote hosts
ICMP	Internet Control Message Protocol; the protocol used by ping to test network reachability
KVM	Kernel-based Virtual Machine; a Linux hypervisor integrated into the kernel
QEMU	Quick Emulator; processor emulator used alongside KVM to run guest virtual machines

libvirt	Virtualization management library and API used to manage KVM virtual machines

Definitions

This section defines the main technical terms used throughout the thesis. The definitions are intentionally practical, since the thesis focuses on virtualization technologies rather than protection algorithm design.

Intelligent Electronic Device (IED): A specialized embedded computer used in electrical substations to perform protection, monitoring, control, and automation functions.

Protection Function: A software or hardware mechanism that detects abnormal electrical conditions, such as short circuits, and initiates corrective actions such as opening circuit breakers.

Protection, Automation, and Control (PAC): The set of systems in a substation responsible for detecting faults, coordinating equipment operation, and supporting grid reliability.

Virtualized Protection, Automation, and Control (vPAC): An architecture in which protection and control functions run as software instances on shared industrial computing platforms rather than on one fixed hardware relay per function.

Digital Substation: A substation architecture where measurements, control signals, and protection messages are exchanged through Ethernet based digital communication rather than mainly through analog wiring.

IEC 61850: An international standard defining data models and communication services for substation automation, including Sampled Values, GOOSE, and MMS.

Sampled Values (SV): High frequency digital current and voltage measurement streams transmitted over the process bus for protection processing.

GOOSE messages: Fast event driven Ethernet messages used for protection and control events, including breaker trip commands.

Merging Unit: A device that converts analog CT/VT measurements into digital Sampled Values for transmission over the process bus.

Process Bus: The communication network that carries measurement and control traffic close to primary equipment, especially Sampled Values.

Station Bus: The communication network used for supervisory control, monitoring, automation, MMS traffic, and some GOOSE exchange within the substation.

Virtual Machine (VM): An isolated software computer managed by a hypervisor and normally running its own guest operating system.

Hypervisor: The software layer that creates, schedules, and manages virtual machines on shared hardware. A VM-based deployment and a hypervisor-based deployment refer to the same mechanism: workloads run as guest VMs under hypervisor control. VMs and hypervisors are therefore treated as one virtualization technology in this thesis.

Container: A lightweight isolated execution environment that packages an application and its dependencies while sharing the host operating system kernel.

Container Image: A read-only bundle of an application and its dependencies, used to instantiate a container.

VM Image: A pre-built disk file from which a virtual machine is started.

Real-Time System: A system where correctness depends on both the produced result and the time at which the result is produced.

Latency: The time between an input event and the corresponding system response.

Jitter: Variation in latency between repeated executions or message transfers.

Determinism: The property that system timing behavior remains predictable and bounded under specified operating conditions.

Temporal Isolation: The ability of one workload to preserve its timing behavior even when other workloads execute on the same platform.

CPU Pinning: A configuration method that binds specific software tasks or virtual workloads to selected processor cores to reduce scheduling interference.

Redundant Networking: A communication setup where multiple independent paths are used so that failure of one path does not interrupt critical operation.

Chapter 1

Introduction

Electrical power systems need protection, automation, and control (PAC) functions to operate safely. Protection functions detect faults such as short circuits. They then trigger actions such as opening a circuit breaker. These actions must happen within strict time limits because delays can affect equipment safety and grid reliability (Blackburn and Domin, 2014).

In traditional substations, PAC functions run inside dedicated Intelligent Electronic Devices (IEDs, physical hardware devices also called protection relays), each with its own processor, software, I/O wiring, and communication interface. This model is robust, but it creates many devices to install, configure, test, and maintain (IEEE Power System Relaying Committee, 2015).

Virtualized protection, automation, and control (vPAC) changes this model. Instead of placing each protection function in a dedicated hardware IED, vPAC runs protection functions as software on shared computing infrastructure. In this thesis, protection functions run as separate software instances, with multiple instances hosted on the same compute platform; the granularity is therefore one software instance per protection function, not one virtual machine per substation. IEC 61850 is the international standard for digital substation communication. It defines interoperable data models and protocol services that allow protection software on shared servers to exchange measurements, protection events, and control signals with field equipment from different manufacturers over standard Ethernet, and this is what makes vPAC possible, as Kabbara et al. (2022) describe. Without this standard, protection software on a shared computing platform would have no common communication layer with the substation field equipment it monitors and controls.

This thesis compares two virtualization technologies for hosting protection software: *Containers* and *Virtual Machines*. Containers package protection software as lightweight services that share the host operating system. Virtual Machines place protection software behind a dedicated virtualization layer: workloads run as guest operating systems managed by a hypervisor, which is the software layer that creates and controls the VMs. The literature review aims to answer what each virtualization technology provides for

isolation, resource control, timing behavior, and platform management.

A technology label alone does not prove system behavior. Calling a solution container-based or VM-based does not show whether CPU allocation, scheduler policy, network mode, I/O handling, memory limits, and guest operating system settings have been configured for protection use. These configurations determine latency, startup time, memory use, I/O behavior, and workload isolation (Abeni, 2024).

vPAC can reduce hardware dependency, make protection software easier to update, and simplify testing and reconfiguration (IEEE Power System Relaying Committee, 2015). However, preserving protection-grade behavior on shared hardware is the difficult part: the platform must control scheduling, isolation, communication, redundancy, and lifecycle management (MacDonald et al., 2025).

1.1 Problem Statement

No existing literature review provides a comprehensive, deployment-oriented comparison of how current vPAC research uses container and VM-based virtualization technologies alongside industrial platform evidence. Several papers show direct vPAC implementations. For example, Nguyen et al. (2025) demonstrate containerized vPAC at scale and Kabbara et al. (2024) validate virtualized protection controllers in a laboratory setup. These studies are read separately and do not provide a unified view of the deployment problem.

The result is that it is not always clear where the protection logic runs, what software stack is used, how IEC 61850 traffic is handled, or which constraint becomes limiting first when the system scales, such as CPU scheduling, network ingress, I/O latency, or validation scope.

Industrial platforms add another part of the picture. SIPROTEC V shows a commercial virtualized protection direction (Siemens, 2026). SEAPATH shows an open platform direction (LF Energy, 2026). ABB's centralized protection and control concept shows how protection architecture can move toward centralized software execution (ABB, 2019).

These academic and industrial views are often discussed separately. This thesis brings them into one deployment-oriented comparison of container-based vPAC, VM-based vPAC, and public industrial platform evidence (that is, architecture and capability descriptions from vendor datasheets and technical whitepapers that show how these technologies are packaged into deployable products).

1.2 Purpose

The purpose of this thesis is to compare how container-based and VM-based virtualization technologies support vPAC on shared computing platforms, and to identify what research prototypes and public industrial platforms each reveal about the requirements for reliable deployment.

1.3 Research Questions

1. How are Containers and Virtual Machines used to deploy virtualized protection functions in the literature?
2. What deployment characteristics distinguish these virtualization technologies in terms of isolation, performance, scalability, and manageability?
3. How do industrial approaches such as SIPROTEC V, SEAPATH, and ABB centralized protection and control operationalize virtualization concepts for digital substation environments?
4. What engineering factors are most important for deploying virtualized protection functions in real-time environments?

1.4 Methodological Approach

The method is based on a literature review that focuses on implementation detail: what virtualization technology was used to enable vPAC, what software stack was built on top of it, and what the validation revealed about real-world behavior. Industrial documents are reviewed separately because they show what a deployable platform requires beyond the virtualization technology alone. The experimental component compares container deployment and VM deployment on the same SEAPATH-based platform, assessing deployment feasibility and basic communication behavior for both virtualization technologies. The specific tools and configuration are described in the Method and Deployment Experiment chapters.

1.5 Contributions

- **Virtualization technology comparison:** A structured comparison of Containers and Virtual Machines as virtualization technologies for vPAC systems.
- **Architecture review table:** A structured table covering retained vPAC evidence sources, classified by virtualization technology, software stack, validation level, and

deployment implication.

- **Industrial platform analysis:** An analysis of SIPROTEC V, ABB centralized protection and control, and SEAPATH as industrial approaches to substation virtualization.
- **Platform requirement synthesis:** A synthesis of the key engineering factors (compute isolation, communication architecture, time synchronization, redundancy, cybersecurity, and lifecycle management) that a vPAC platform must address for reliable deployment, derived from gaps identified across the reviewed evidence.
- **Deployment experiment:** A direct comparison of Podman container and KVM-based VM deployment on the same SEAPATH-based platform, showing that both virtualization technologies can be deployed with automation, assigned separate static network addresses, and verified through basic communication.

1.6 Delimitations

The scope is limited to architecture and system-level deployment. The thesis does not evaluate algorithmic correctness, cybersecurity certification, utility approval, or full field deployment. The experiment checks container and VM deployment mechanics and basic communication; it does not benchmark a complete IEC 61850 protection system.

1.7 Structure of the Thesis

The thesis is organized as follows:

- **Chapter 2, Technical Background:** establishes the protection, IEC 61850 communication, real-time, and virtualization concepts the rest of the thesis builds on.
- **Chapter 3, Method:** explains the targeted literature review and the supporting deployment experiment.
- **Chapter 4, vPAC Deployment Architecture:** compares the direct vPAC studies by virtualization technology and synthesizes their deployment implications.
- **Chapter 5, Industrial vPAC Platforms:** discusses SIPROTEC V, SEAPATH, and ABB centralized protection and control as industrial platform directions.
- **Chapter 6, Deployment Experiment:** documents the container and VM deployment experiment on a SEAPATH-based platform.
- **Chapter 7, Discussion:** interprets the findings against the research questions and states the deployment implications and limitations.
- **Chapter 8, Conclusion:** answers the research questions and identifies future work.

Chapter 2

Technical Background

Moving protection functions from dedicated IED hardware to shared computing platforms introduces multiple interdependent engineering stack layers, where each stack layer independently affects protection behavior (MacDonald et al., 2025). These stack layers are:

- **Field equipment:** the primary substation equipment including circuit breakers, current transformers (CTs), and voltage transformers (VTs), which provides measurements and switching signals (Blackburn and Domin, 2014).
- **IEC 61850 communication layer:** carries those measurements and protection signals over digital networks.
- **Compute platform:** hosts the protection software on shared server hardware.
- **Virtualization layer:** separates workloads from each other on that shared hardware.

A fault or configuration gap in any of these stack layers can affect protection behavior regardless of what the other stack layers do. This chapter describes each stack layer, starting from traditional protection systems and IEC 61850 communication and then covering real-time requirements and virtualization technologies.

2.1 Technology Stack Used in vPAC

A vPAC system is a layered technology stack in which the protection application is only one layer; timing, isolation, communication, updates, and monitoring each add requirements at a separate level (MacDonald et al., 2025).

Figure 2.1 shows the two technology stacks compared in this thesis. The VM-based technology stack inserts a hypervisor and guest operating system between the protection application and the hardware; the container-based technology stack shares the host kernel directly.

The lowest layer is the industrial host hardware. It includes the processor, memory, storage, a dedicated time source (a GPS/GNSS receiver or PTP grandmaster clock that distributes microsecond-accurate reference timestamps across the substation network;

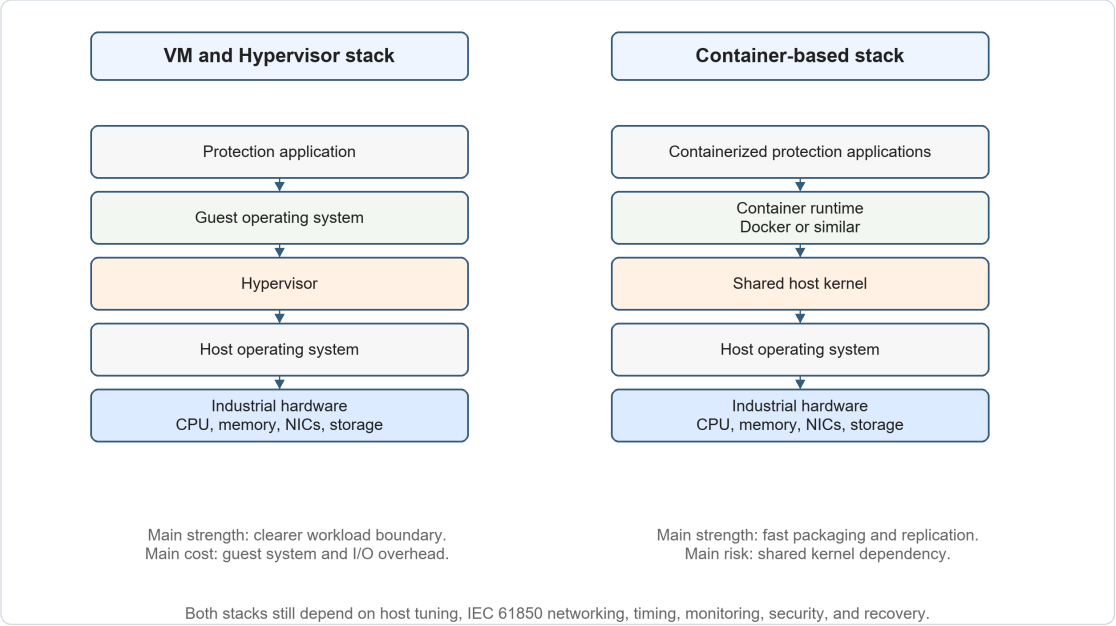

Figure 2.1: Technology stacks for vPAC compared in this thesis.

protection functions need these timestamps to align measurements from different field devices and to order fault events correctly), and network interface cards. Protection traffic, meaning Sampled Values (SV) streams that carry high-frequency measurement data from current and voltage transformers at up to 4000 frames per second, and GOOSE messages that carry fast event-driven protection signals such as trip commands, can be limited by network ingress, interface assignment, or physical Ethernet capacity before CPU capacity is exhausted, as Nguyen et al. (2025) demonstrate at scale.

Above the hardware is the host operating system, which in vPAC cannot be treated as a neutral detail. Its scheduling policy and resource control determine whether protection workloads remain predictable (Abeni et al., 2022). Scheduling policy matters because a general-purpose scheduler can preempt or delay the protection task in favour of other workloads, introducing jitter that may push response latency past the protection function's deadline (Abeni et al., 2022; Kabbara et al., 2024). Network configuration and driver behavior add further dependencies. Clock discipline is also a host OS concern: the operating system must continuously adjust its local clock to stay locked to an external reference such as a PTP grandmaster, and any drift or interruption in that synchronization causes timestamps on SV measurements and GOOSE events to become inaccurate, which can cause incorrect fault decisions (Kabbara et al., 2024).

In a Virtual Machine design, the hypervisor creates and controls isolated execution environments. A VM normally contains a guest operating system and the protection application. This gives a clearer boundary between workloads, but it also introduces scheduler, I/O, memory, and guest operating system overhead; a VM is therefore not automatically

deterministic and still requires careful platform tuning (Tran et al., 2016).

In a Container design, the container runtime starts packaged application instances that share the host kernel. A container image, a read-only bundle of the application and its dependencies used to instantiate a container, can make deployment repeatable, and a container instance can start quickly. The tradeoff is that the host kernel, network stack, and timing services become shared dependencies. For protection work, this means that container configuration must include resource limits, privileges, network mode, logging, update behavior, and recovery behavior.

The networking stack layer cuts across both the VM and container technology stacks. IEC 61850 traffic is time-critical and Layer 2 dependent: GOOSE (Generic Object Oriented Substation Event, defined in IEC 61850-8-1) and SV messages are carried as Ethernet multicast frames directly at Layer 2 of the network stack, using MAC addresses and not routed through IP (Kabbara et al., 2022; Rexwinkel et al., 2025). This means that a container or VM must have direct Ethernet-layer access to send and receive them; standard IP-based virtual networking is not sufficient. Bridge networking, host networking, macvlan (which gives each container its own MAC address on the physical Ethernet), VLAN assignment, multicast handling, and redundant paths can therefore each change whether protection traffic reaches the software instance at all (Tellado et al., 2025; Nguyen et al., 2025).

The upper stack layers are the protection application and the operational management stack layer. The application stack layer may include virtual IEDs (vIEDs), logical nodes, protection logic, gateway services, monitoring, and engineering services. The operational stack layer includes configuration, diagnostics, updates, rollback, logging, cybersecurity, traceability, and recovery. These stack layers explain why industrial vPAC should be studied as a platform and not only as a virtualization technology.

2.2 Protection Systems

Protection systems in electrical substations detect faults, isolate the affected zone within a defined time window, and trigger corrective actions such as opening a circuit breaker to prevent damage from spreading (Blackburn and Domin, 2014). In a traditional substation, each protection function runs in a dedicated IED assigned to a specific zone (a bay, a bus, or a transformer) and operates independently of other functions. IEDs that perform protection functions are commonly called protection relays, a term that reflects their historical role as electromechanical switching devices; modern IEDs have long been digital computers but the term persists (Blackburn and Domin, 2014). This section describes that traditional structure and how moving protection functions into software changes the engineering problem that IEDs, vPAC, and the broader PAC system must solve.

Figure 2.2 shows the baseline architecture used as the starting point for the thesis. It shows why traditional relay systems provide clear separation, but also why a move toward software-hosted protection changes the engineering problem.

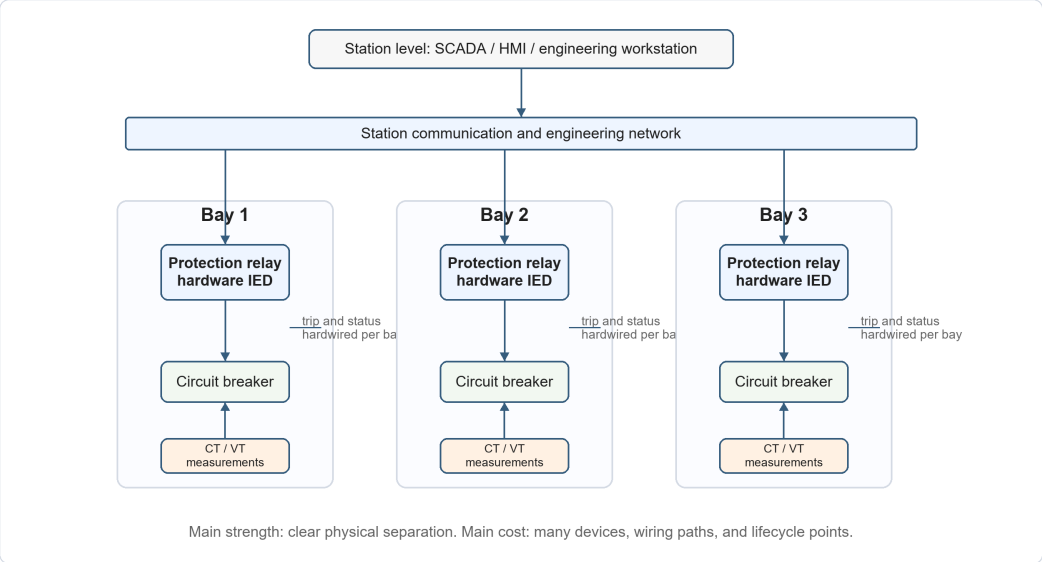

Figure 2.2: Traditional substation protection architecture.

Protection, automation, and control systems are critical components of modern electrical power networks. Blackburn and Domin (2014) describe their primary function as detecting abnormal operating conditions and initiating corrective actions to protect equipment and maintain grid stability. Traditional substations are organized into bays, where each bay is a functional protection zone covering one power circuit or feeder and served by its own dedicated relay. Each relay handles the measurements, protection logic, and trip output for that circuit independently. This provides strong functional separation and deterministic execution, but the IEEE centralized protection report notes that it also increases device count, wiring, testing effort, and lifecycle cost (IEEE Power System Relaying Committee, 2015).

2.3 IEC 61850 Communication

IEC 61850 is the international standard for digital substation communication. It defines data models and protocol services that allow IEDs, merging units, gateways, and control systems to exchange protection and control information over Ethernet. The standard separates the communication layer into a process bus, which carries high rate measurement and fast trip traffic close to primary equipment, and a station bus, which carries supervisory control, monitoring, and configuration traffic.

Figure 2.3 shows this two bus structure. The figure is important for the thesis because

vPAC moves where protection logic runs, but it does not remove the process bus and station bus requirements. A virtualized protection function must still receive Sampled Values, publish GOOSE messages, and operate within the timing constraints defined for the protection role.

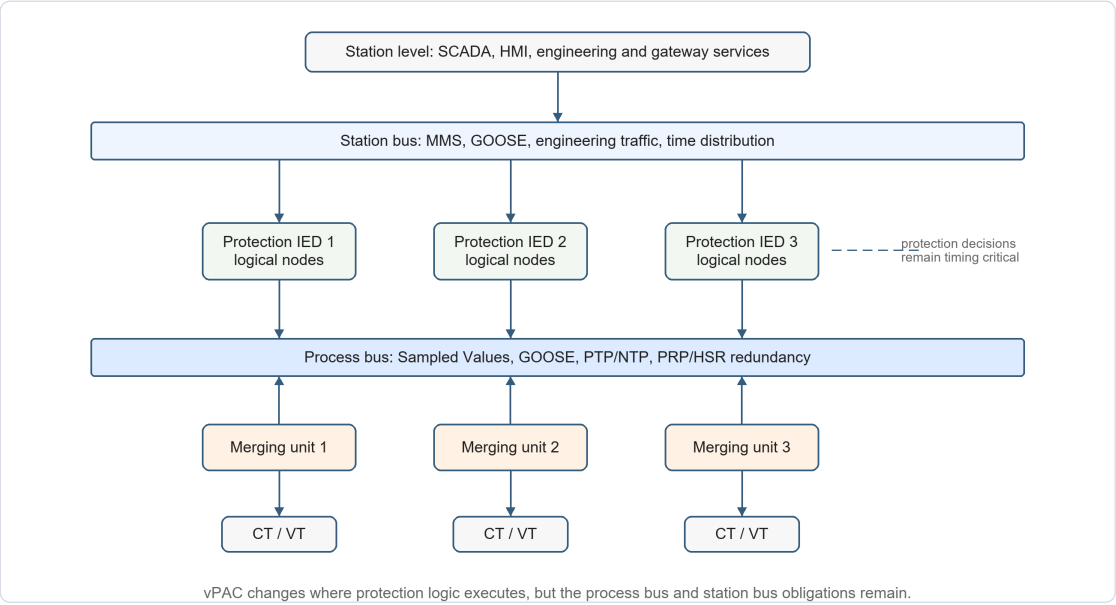

Figure 2.3: IEC 61850 substation communication architecture.

Kabbara et al. (2022) describe IEC 61850 as the main standardized communication framework for digital substation automation. For protection applications, the most important services are Sampled Values (SV), which carry high frequency measurement streams, and GOOSE messages, which carry fast event driven protection and control signals. MMS is used for monitoring, supervision, and engineering oriented communication.

The process bus carries time sensitive measurement and control traffic at high frame rates, and the station bus carries supervisory and automation traffic. This separation means that the volume and rate of SV and GOOSE traffic on the process bus becomes a first-order design constraint when protection functions are moved to a shared server, because the server's network interfaces must handle all incoming measurement traffic before any protection logic runs.

Time synchronization is a shared dependency across both buses and the protection application. IEC 61850 uses the Precision Time Protocol (PTP, IEEE 1588) to align clocks across the substation network to microsecond accuracy (Kabbara et al., 2024). SV streams rely on synchronized timestamps for measurement alignment, and GOOSE messages must be transferred within strict time bounds: IEC 61850-5 specifies a transfer time of 4 ms or less for bay-level protection functions, so that event ordering and fault decisions remain consistent (Rexwinkel et al., 2025).

High-availability substations also use redundancy protocols at the network level. PRP (Parallel Redundancy Protocol, IEC 62439-3) sends each Ethernet frame simultaneously on two independent network paths and delivers whichever copy arrives first, so that a single cable or switch failure causes no interruption to protection traffic (Rexwinkel et al., 2025). VLANs are used to logically separate SV, GOOSE, and management traffic on shared Ethernet infrastructure. A virtualized protection deployment must accommodate both mechanisms.

2.4 Real-Time Requirements

Timing is a hard constraint when protection functions move from dedicated relay hardware to shared compute platforms. Blackburn and Domin (2014) explain that if a fault is not cleared within the protection function's specified operating time window, thermal stress on primary equipment increases and the fault can propagate to adjacent feeders, widening the disturbance. In a digital substation, timing behavior depends on application code, operating system scheduling, CPU contention, virtualization overhead, SV traffic, GOOSE transfer time, and synchronization accuracy (Kabbara et al., 2024).

When protection functions move from dedicated relay hardware to shared compute platforms, temporal isolation becomes a central design requirement. Abeni et al. (2022) show that CPU partitioning and admission control become necessary to prevent one real-time workload from unpredictably delaying another.

2.5 Virtualization Options

Multiple virtualization approaches can host real-time workloads on shared hardware; Abeni (2024) list containers, VMs, microVMs, and related mechanisms. The architecture comparison in this thesis is reduced to two virtualization technologies: Containers and Virtual Machines.

VM-based virtualization runs applications inside guest operating systems or isolated execution domains managed by a hypervisor. This provides strong separation and compatibility with complete software stacks, but it introduces overhead through guest operating systems, virtualized I/O, and hypervisor scheduling. Tran et al. (2016) show that hypervisor scheduler configuration strongly affects whether such workloads remain real-time capable.

Container-based deployment packages applications as isolated processes that share the host kernel. Containers usually provide lower overhead and faster startup than VMs, but their real-time behavior depends strongly on host kernel configuration, CPU isolation,

runtime settings, networking mode, and orchestration (Queiroz et al., 2023).

2.6 Platform and Technology Overview

This section gathers the platforms, runtimes, and tools named throughout the thesis and classifies each one concretely, so that later chapters can refer to them without redefining them. The industrial platforms are analysed in Chapter 5; the one-line role given here states only what each one is.

SIPROTEC V (commercial platform): A commercial virtualized protection and control product from Siemens that hosts SIPROTEC 5 style bay protection functions as software on substation-grade computing hardware (Siemens, 2026).

SEAPATH (open platform): An open industrial platform layer from LF Energy that provides real-time Linux, KVM-based VM hosting, container support, orchestration, high availability, and lifecycle control for substation workloads (LF Energy, 2026).

ABB centralized protection and control (architecture concept): An architecture concept that moves protection from physically separated bay relays toward centralized software protection on industrial computing hardware (ABB, 2019).

Virtual Machine (isolation mechanism): An isolated execution environment that runs a guest operating system under a hypervisor, giving each workload its own kernel (Abeni, 2024).

Container (isolation mechanism): A packaged application instance that shares the host kernel through operating-system-level isolation (Queiroz et al., 2023).

Docker (container engine): The container engine most commonly named in the reviewed direct vPAC studies, using a background daemon to manage container lifecycle and OCI-format images (Nguyen et al., 2025).

Podman (container engine): A daemonless, OCI-compatible container engine that runs each container as a direct child process and is compatible with Docker images; it is the runtime used in the experiment of this thesis.

KVM (hypervisor): The Kernel-based Virtual Machine, a Linux hypervisor integrated into the kernel that provides hardware-assisted virtualization for guest VMs.

QEMU (emulator): The Quick Emulator, a processor and device emulator used together with KVM to run guest virtual machines.

libvirt (management library): A virtualization management library and API used to define, start, and manage KVM virtual machines.

Several tool names appear in the architecture review table of Chapter 4 and are defined here for reference:

vPTOC: A virtual time-overcurrent protection function, the software equivalent of the IEC 61850 PTOC protection logical node, run as a container in the reviewed containerized vIED setup (Nguyen et al., 2025).

vSAMU: A virtual sampled-value merging unit that publishes Sampled Values in software rather than from dedicated merging unit hardware (Nguyen et al., 2025).

libIEC61850: An open-source software library that implements IEC 61850 communication services such as Sampled Values, GOOSE, and MMS (Torres et al., 2024).

GTNETx2: A real-time digital simulator network interface card that exchanges IEC 61850 traffic between an RTDS and external protection software (Kabbara et al., 2024).

OpenDaylight: An open-source software-defined networking controller used to manage network paths in the reviewed virtualized controller testbed (Kabbara et al., 2024).

RSCAD: The modelling and simulation software used to build the power-system and fault scenarios run on an RTDS (Villén Martínez and Prada Hurtado, 2024).

Chapter 3

Method

3.1 Overview

This chapter describes the two-part method used to answer the research questions: a targeted literature review and a supporting deployment experiment. The review identifies how vPAC is implemented across the retained studies, what software and network design choices were made, and what deployment constraints each study reveals. The experiment contributes direct observations of container and VM deployment on a SEAPATH-based platform.

The search used IEEE Xplore with five keyword strings targeting direct vPAC implementations, screened candidate articles by title, abstract, and full text, and classified retained papers by virtualization technology, software stack, validation level, and deployment implication. Papers were retained only if they described a direct vPAC implementation or public industrial platform; generic virtualization and communication-only papers are used for background but are not in the architecture table. The experiment was scoped after the review established the relevant platform context and constraints (Queiroz et al., 2023; Struhár et al., 2020).

Each retained paper is read through the same four questions: what was built, what software was used, what was tested, and what problem remains before industrial deployment. These questions keep the review focused on vPAC deployment instead of general virtualization.

The experiment complements the review by comparing container-based and VM-based deployment directly on a SEAPATH-based platform, checking whether both virtualization technologies can be deployed with automation, assigned separate static network addresses, and verified through basic communication. The review provides the main evidence base for protection-grade conclusions; the experiment contributes direct deployment observations on the same class of platform.

3.2 Why This Method Was Used

The research problem requires understanding which virtualization technology is better suited for which deployment role, what each technology does well, what it does poorly, and what constitutes an appropriate baseline for an architecture decision in this domain. These questions cannot be answered by measuring one technology in isolation; they require a side-by-side comparison against the same platform and the same deployment criteria.

The literature review was designed to collect that comparison evidence across the retained studies, classifying each source by what was built, what software was used, what was validated, and what deployment challenge remains. This gives a consistent basis for identifying where containers outperform VMs, where VMs are preferable, and what neither technology addresses on its own. The experiment then applies the same comparative logic directly on a SEAPATH-based platform, checking whether both container-based and VM-based deployment can meet the same first-level requirement: separate addressability and basic communication on the same host. Because SEAPATH is a public reference platform for substation virtualization (LF Energy, 2026), this also confirms that the platform supports both deployment paths before any protection functionality is added.

3.3 Link to Research Questions

The parts of the method map directly to the research questions. The logic is sequential:

- First, the literature review identifies how vPAC is implemented in terms of virtualization technology, software stack, IEC 61850 role, and validation evidence.
- Second, the architecture table compares each retained study using the same questions: what was built, what software was used, what was tested, and what deployment issue remains.
- Third, the industrial review checks whether the same virtualization and platform concepts appear in deployable platforms, because public industrial sources show the engineering, lifecycle, and operational layers around the virtualization technology.
- Fourth, the experiment adds direct deployment evidence by comparing container and VM setup on the same SEAPATH platform, showing what configuration is needed before either can function as a separately addressable controller workload.

3.4 Systematic Thesis Roadmap

The method combines a literature review with a container and VM deployment experiment on a SEAPATH-based platform. The argument follows a clear order: the technical background is established before the literature is compared, and the experiment is interpreted only after the review has identified what industrial vPAC requires. Figure 3.1 shows how research questions, source roles, evidence extraction, experiment scope, and synthesis connect across the thesis.

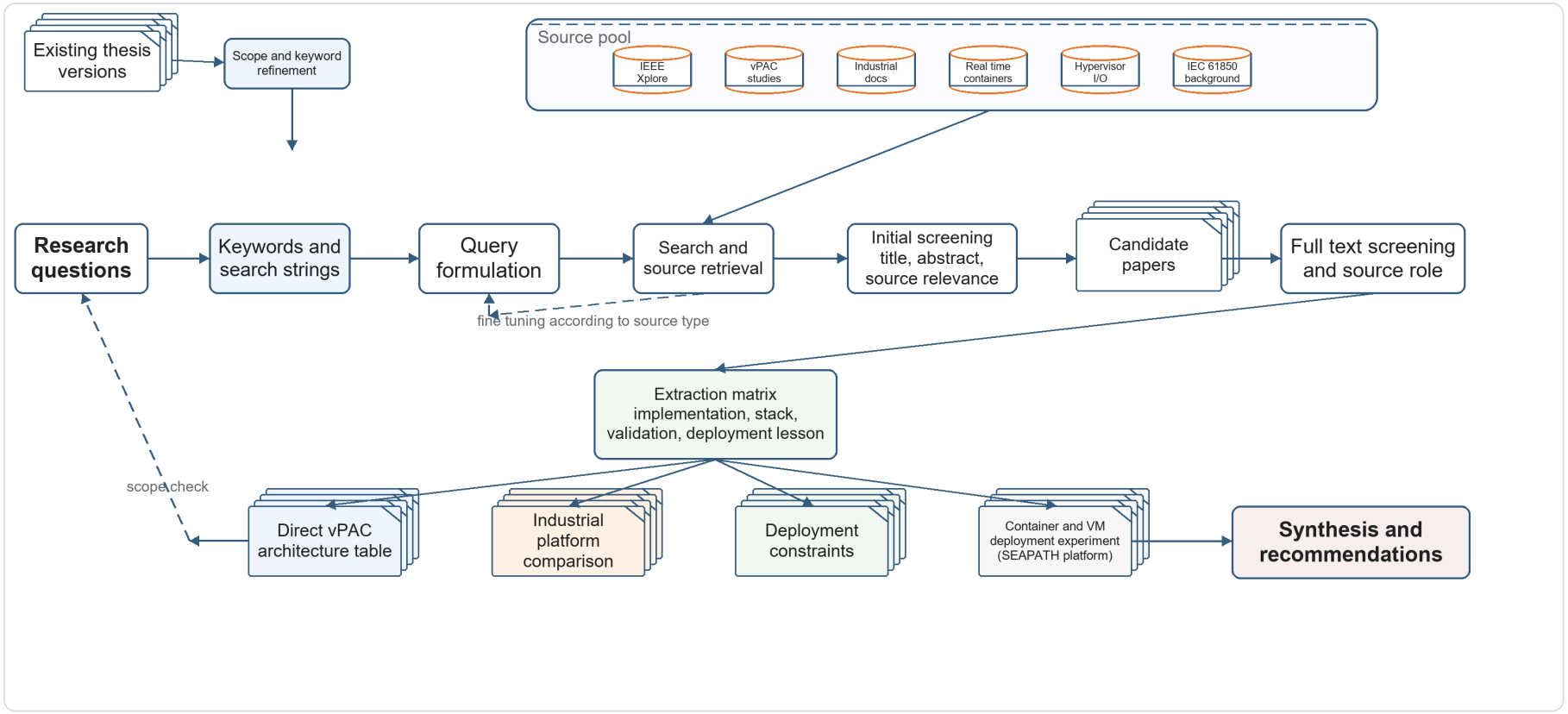

Figure 3.1: Thesis argument structure and source roles.

The roadmap also clarifies the logic behind source selection. Direct vPAC papers answer what was built and tested. Industrial sources show what deployable platforms need around the virtualization technology. Background studies explain constraints such as timing, networking, I/O behavior, and redundancy. The experiment checks one deployment feasibility point and is therefore interpreted after the review has established the broader platform requirements.

3.5 Literature Review Procedure

This section explains how sources were found, screened, and assigned roles in the thesis. The aim is to keep the review traceable while separating direct vPAC evidence from industrial platform evidence and enabling background studies.

3.5.1 Search Source

This subsection explains why IEEE Xplore was selected as the primary search source and how additional sources were used alongside it.

The academic search was limited to IEEE Xplore for direct vPAC study screening. This keeps the review traceable and fits the topic because IEEE Xplore contains many papers on protection systems, IEC 61850, digital substations, and industrial control. IEEE Xplore was preferred over broad aggregators such as Google Scholar and Scopus because it provides a curated, reproducible corpus for power-system protection and IEC 61850, which supports the targeted and traceable screening design used here (Queiroz et al., 2023). Broad aggregators return a larger share of non-peer-reviewed and loosely related hits that would conflict with this screening design, so they were not used as the primary source.

Additional background and enabling papers from other publication venues are used where they explain virtualization behavior, real-time constraints, cybersecurity, or industrial platform requirements. These papers were not found through the IEEE Xplore search strings; they provide explanatory context rather than direct vPAC architecture evidence.

Industrial documents are treated separately in Chapter 5. They are useful because vendors often describe platform architecture, tools, and deployment workflows. They do not always provide the same type of evidence as academic papers.

The search used five targeted keyword strings aimed at finding papers where protection was directly moved to a software-hosted platform. Title and abstract screening was followed by full text review; a source was retained for the architecture table only if it described a direct vPAC implementation or public industrial platform. The retention criteria are listed after the search strings so that the reason for including or excluding

each source is visible.

Table 3.1: IEEE Xplore search strings used for the direct vPAC literature search.

IEEE Xplore search string	**Why it was used**
`"virtualized protection" AND "IEC 61850"`	Find papers that explicitly connect virtualization with IEC 61850 communication in protection systems.
`"virtual IED" AND "IEC 61850"`	Find studies that replace physical IEDs with software-hosted IED instances.
`"software-defined" AND "IEC 61850" AND protection`	Find software defined protection and controller implementations.
`Docker AND "IEC 61850" AND protection`	Find container-based vPAC and Docker-based logical node implementations.
`"virtual protection relay"`	Find Virtual Machines or server hosted virtual relay studies.

The search produced many papers that were not direct vPAC studies. Most were about conventional IEC 61850 communication, protection algorithms, cybersecurity, or general virtualization. Those papers were useful for background, but they did not answer the architecture question.

After full text screening, the architecture table retained only direct vPAC studies and public industrial platform evidence; this set is treated as the evidence base for the architecture comparison, not as a complete survey of related papers.

Sources were retained for the architecture table when they met these conditions:

- The work describes a direct vPAC implementation or platform.
- The work gives enough software detail to classify it.
- The work explains what was tested or validated.

Generic virtualization papers were excluded from the architecture table; the keyword strings were designed to target direct vPAC implementations, so most returned results were already vPAC-specific or IEC 61850-specific, not generic virtualization. IEC 61850 communication-only papers were also excluded. They may still support background sections, but they are not direct vPAC solutions.

Note on the Docker keyword: the search string uses `Docker AND "IEC 61850" AND protection` because Docker is the most common runtime named in direct vPAC studies; the retained papers are classified as container-based regardless of whether they specifically use Docker or another container runtime such as Podman (a daemonless, OCI-compatible container engine compatible with Docker images) or a low-level OCI runtime.

This separation is important. Direct vPAC papers are used to compare how each virtualization technology is applied. Background and enabling papers are used to explain why those technology choices may succeed or fail. For example, papers on IEC 61850 communication, real-time container behavior, hypervisor I/O latency, and redundancy do not always build a vPAC platform, but they help explain the constraints that a vPAC platform must satisfy.

The screening outcome is summarized in Table 3.2. A hit count refers to the total number of results a search string returns in the database before any screening is applied; these raw counts are not treated as a result here because the review is targeted rather than fully systematic. The important outcome is the distinction between direct vPAC evidence, industrial platform evidence, and background evidence. Table 3.2 has three columns: *Source group* identifies the category; *Representative examples* names specific papers or documents in each group; and *How the group is used* explains the evidence role each category plays in the thesis. The direct academic vPAC studies form the core of the architecture comparison. The industrial and platform evidence shows what a deployable platform requires. The background studies explain the constraints that drive design choices in both categories.

Table 3.2: Screening outcome and source roles.

Source group	**Representative examples**	**How the group is used**
Direct academic vPAC studies	EPICS (Villen-Martinez), Torres, Tellado, Kabbara, Nguyen, Stinskiy	Included in the architecture review table (Chapter 4) because they describe direct software-hosted protection implementations or testbeds.
Industrial and platform evidence	SIPROTEC V (Siemens), ABB centralized protection and control, SEAPATH (LF Energy)	Analyzed in Chapter 5 to compare what research prototypes assume versus what a deployable platform requires.
Background and constraint studies	Real-time scheduling studies (Abeni), hypervisor I/O latency studies, IEC 61850 communication studies, redundancy protocol studies	Used throughout Chapters 2 and 4 to explain technical constraints such as scheduling, networking, time synchronization, I/O behavior, and redundancy. These papers do not build vPAC but explain the constraints a vPAC platform must satisfy.
Excluded from architecture table	Conventional protection algorithm papers, IEC 61850 communication-only studies, generic container or VM papers with no substation context	Not used as direct architecture evidence because they do not describe a vPAC deployment, but some provide background context.

The source groups have different evidence value. Academic papers usually provide more detail about method, test setup, and validation, but many of them remain laboratory prototypes. Industrial documents show which components and requirements a deployable platform must address, but they often omit low-level implementation details. For that reason, the thesis does not treat all sources as the same kind of proof. Academic sources are used mainly for implementation evidence, while industrial sources are used to identify platform requirements and deployment direction.

3.5.2 Paper Review Questions

Each direct vPAC paper was read using four questions:

- Is the solution based on Containers or on Virtual Machines?
- What software, networking configuration, and IEC 61850 communication software are reported?
- What type of validation is provided?
- What practical lesson does the study give for an industrial vPAC platform?

This matters because the word “Docker” is not enough. One Docker study may use simple bridge networking. Another may use macvlan, CPU allocation, and real SV traffic. Those systems have very different implications for protection use.

The same applies to Virtual Machines. A VM label does not explain scheduler configuration, I/O handling, redundancy, or synchronization. These configurations determine whether the architecture is only a prototype or a plausible substation platform.

The retained evidence is first summarized chronologically in Figure 3.2. The figure uses four color groups to reflect evidence phases: blue covers 2024 containerized protection microservices; green covers 2024 real-time vIED testbeds; orange covers 2025 virtual protection relays and edge logical nodes (IEC 61850 logical nodes, that is, protection and control functions defined by the standard, running on edge computing hardware close to primary substation equipment); and red covers the industrial platform evidence, which spans multiple years and is treated as platform evidence rather than research prototypes.

The screening and extraction logic is summarized in Figure 3.3. The figure shows how academic and industrial sources are connected to the two virtualization technologies used later in the thesis.

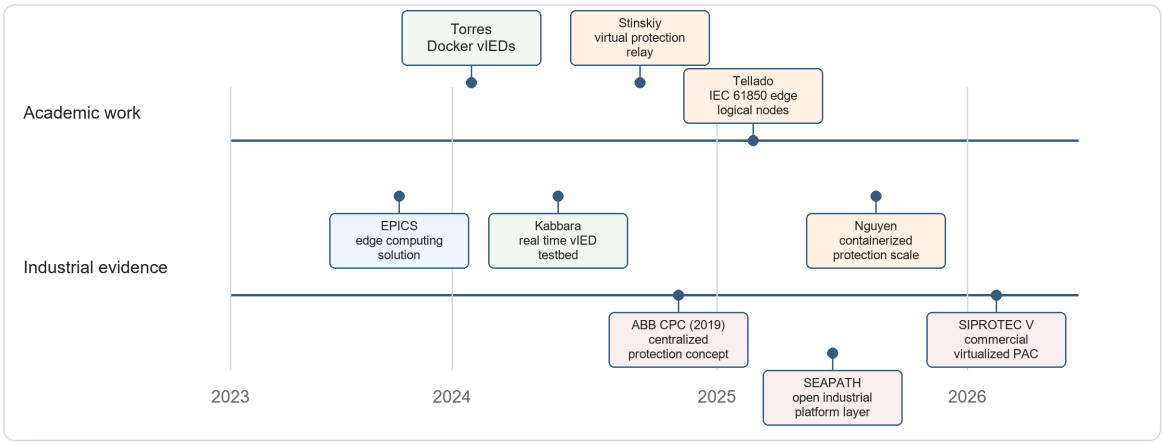

Figure 3.2: Grouped timeline of retained evidence. Industrial sources serve as platform evidence, not research prototypes.

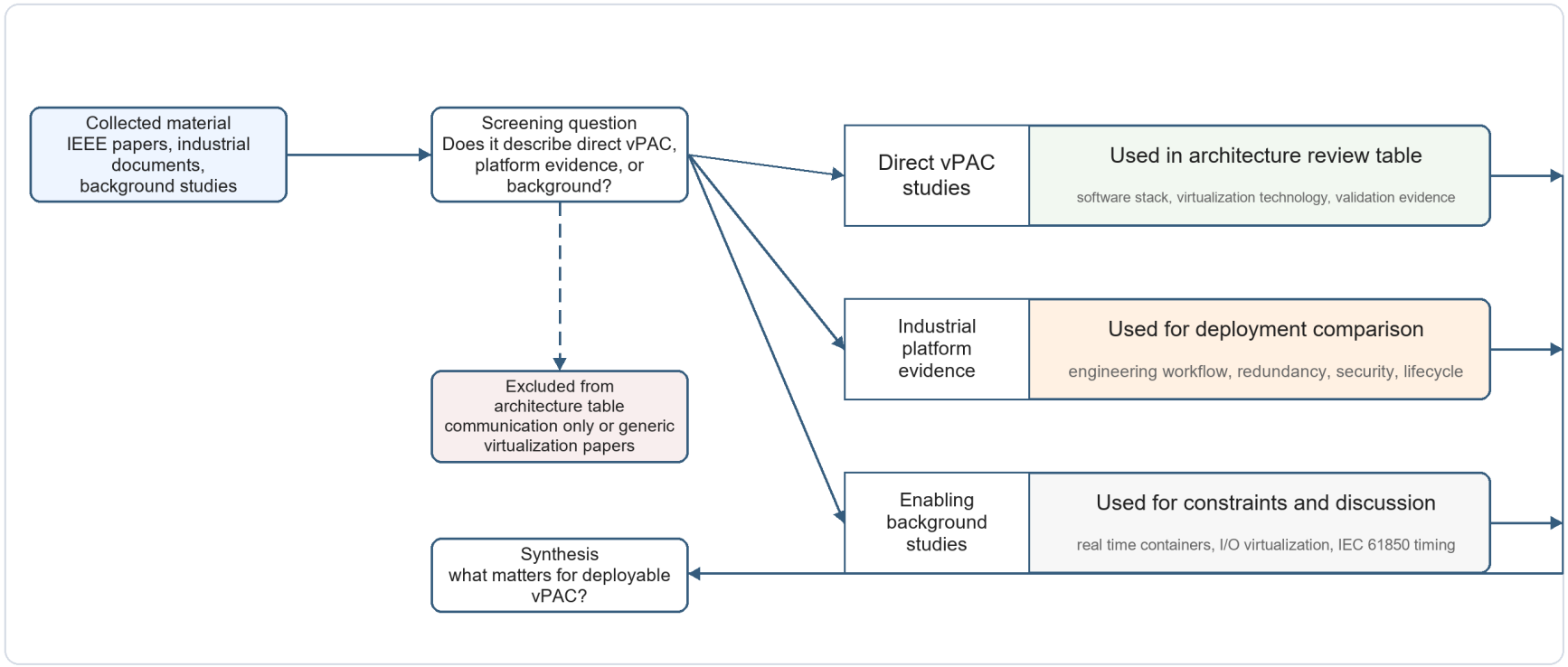

Figure 3.3: Source role classification. Direct vPAC studies form the architecture review table; industrial documents and background studies support the deployment discussion.

3.6 Classification

The retained direct vPAC studies were classified into two virtualization technologies: Containers and Virtual Machines. A small number of studies combine both technologies (for example, running containers nested inside VMs) and these are noted as hybrid cases in the analysis.

This classification is deliberately simple, meant to support the main comparison rather than create a large taxonomy.

The classification of Virtual Machines does not mean that all such platforms are identical. Hypervisor type, scheduler design, guest operating system, and I/O model can all change system behavior. They are classified together in this thesis because they share the main architecture role used in the comparison: they place the protection application inside a guest operating system with its own kernel, which means workloads do not share kernel state with each other or with the host (Abeni, 2024). This hardware-level isolation is the main distinction from containers, which share the host kernel directly.

Industrial platforms were analyzed separately when they combined several mechanisms or did not disclose the low-level runtime layer.

3.7 Analysis Method

The analysis uses three practical checks. These checks follow the review paper logic of moving from technology choice to measurable system behavior and then to remaining deployment constraints.

First, the study must show where the protection software runs: whether the solution uses Containers or Virtual Machines.

Second, the study must show how protection traffic moves through the system. This includes SV input, GOOSE output, MMS communication, time synchronization, and redundant networking when they are present.

Third, the study must show what was tested. The validation level matters: a basic container startup demonstration provides a weaker claim than an RTDS hardware-in-the-loop test (where a Real-Time Digital Simulator generates realistic IEC 61850 traffic, fault scenarios, and process data that interact directly with the protection software under test) that measures actual protection trip timing, or a commercial platform description that shows fielded deployment.

These checks keep the review focused on deployment. They also make Table 4.1 easier to read. Each row answers the same basic question: what was built, what virtualization

technology and software stack were used, and what lesson does it give for a real vPAC platform?

3.8 Experiment

3.8.1 Purpose

The experiment compares container-based and VM-based deployment on the same SEAPATH-based virtualized platform, assessing deployment feasibility and basic communication behavior for both virtualization technologies. The container side checks whether Debian-based containers managed by Podman (the daemonless, OCI-compatible container engine introduced in the search procedure note in Chapter 3) can be prepared offline, transferred via Ansible, and started as separately addressable controller instances. Each instance is then verified through ICMP communication. A KVM-based VM baseline is documented on the same host for direct comparison.

3.8.2 Approach

The deployment used Ansible to automate image transfer, local image loading, network creation, container recreation, and status reporting. Three Podman container instances were deployed on a dedicated bridge network and three KVM-based VM instances were deployed through KVM and libvirt on the same SEAPATH host. Basic ICMP tests were run between the container instances and between the VM instances so that both virtualization technologies could be compared at the same deployment level on the same platform (Abeni, 2024).

3.8.3 Justification

The experiment supports the architectural discussion by showing what must be configured before containers and VMs behave like separately addressable controller workloads on a virtualized substation platform, and by providing a direct side-by-side deployment comparison of the two virtualization technologies. Abeni (2024) motivate why such a comparison is useful for understanding overhead and packaging differences, while LF Energy (2026) shows why platform control, workload placement, and automation matter for deployable substation virtualization.

3.9 Method Limitations

The review is targeted rather than exhaustive: it covers direct vPAC implementations and public industrial platform evidence, using content screening rather than database hit counts as the inclusion criterion. This design matches the research questions, which ask about virtualization technologies rather than publication trends. Some industrial platform details are not publicly disclosed, and this is acknowledged in the analysis where it affects the conclusions. The experiment uses a general-purpose deployment approach on a public reference platform, so the procedure can be reproduced by other researchers with access to SEAPATH-class hardware.

The available literature gives only a partial view of each system. Some papers focus on software stack and container packaging; others focus on IEC 61850 network communication; few cover the full deployment path from hardware provisioning through protection output under realistic operating conditions. This means that the architecture comparison is necessarily based on incomplete evidence in some dimensions, and the table explicitly notes where information was not reported by the study's authors.

3.10 Validity Analysis

There are four main threats to validity.

First, the two-category grouping simplifies a complex technology landscape: many variants of containers and VMs exist, and the grouping does not capture all deployment differences. It is appropriate for the research questions but not for fine-grained technology comparisons.

Second, most direct vPAC studies are laboratory prototypes or simulations. The field is still developing, which limits how far the findings can be generalized to utility field deployment.

Third, some industrial platform details are not publicly disclosed. The analysis is based on publicly available documentation; implementation specifics that vendors have not released are acknowledged as gaps rather than treated as evidence.

Fourth, the experiment is scoped to deployment mechanics and basic ICMP communication on a single platform. It does not cover IEC 61850 protection traffic, redundancy, time synchronization, or failover. These are treated as open validation questions and identified as future work in Chapter 8.

These boundaries are part of the design, not errors. The thesis identifies deployment patterns and platform requirements; it does not claim that any virtualization technology is production-ready for a utility substation.

Chapter 4

vPAC Deployment Architecture

The retained evidence consists of 6 academic vPAC studies and 3 industrial platform documents, giving 9 retained sources in total. Each of these sources was read to extract the same set of information: which virtualization technology the study used, what software was built on top of it, how the system was validated, and what deployment constraint the study encountered or left open. The goal is not to evaluate the papers themselves, but to understand what each one reveals about implementing and validating protection software on shared hardware.

This chapter maps each of the 9 sources to one of the two virtualization technologies, notes hybrid cases where both are combined, and identifies the deployment constraints each source reveals.

Each study is evaluated by what it actually built, what software it used, what was tested, and what deployment problem it exposes. The discussion follows the software-defined PAC framing of Kabbara et al. (2022), but focuses on concrete deployment concepts rather than architectural diagrams.

4.1 How the Studies Are Compared

The reviewed papers use different names for similar concepts, including virtual IED, software-defined controller, and virtual protection relay, which are mapped to the two virtualization technologies defined in the Method chapter so that the sources can be compared consistently. The resulting classification is shown in Table 4.1, which includes only direct vPAC studies and public industrial platform sources.

Each row of Table 4.1 gives the implementation, the software stack, the validation level, and the deployment implication, so that the table functions as the bridge between source selection and synthesis rather than as a bibliography.

Table 4.1: Direct vPAC deployment evidence relevant to the transition from physical IEDs to software hosted protection platforms.

Study and category	What was built and what software was used	What the study teaches
Containers Nguyen et al. (2025)	The paper builds a centralized containerized vIED setup. Many software IEDs run on one server instead of on separate relay hardware. The server runs Debian Linux `5.10.197-1`. The container stack is Docker CE `25.0.3`, containerd `1.6.28`, and runc `1.1.12`. The test uses IEC 61850-9-2 Sampled Values at 4000 frames/s and GOOSE trip output. The protection logic is represented by vPTOC containers. The measurement side is represented by vSAMU containers. The largest test uses up to 400 vIED containers.	This study shows that container density is possible, but the network becomes the real limit. CPU is not the first bottleneck. The harder problem is getting high rate Sampled Values into the server through NICs, switches, and physical Ethernet channels. For an industrial platform, the process bus network must be sized before promising large scale vIED consolidation.
Containers Torres et al. (2024)	The paper builds a Docker-based IEC 61850 substation automation simulation. The virtual IEDs include overcurrent protection, trip conditioning, breaker logic, SV publishers, and GOOSE trip messages. The implementation uses Docker containers with Docker bridge networking. IEC 61850 behavior is implemented with libIEC61850. The setup also uses SCD configuration, SV, GOOSE, a Python IEC 61850 client, a primary process simulator, and a Wireshark container. The authors state that the setup is not a hard real time simulator.	This is a useful warning case. The protection function works at small scale. When the number of virtual IEDs increases, network traffic also increases. Trip delay increases as a result. The lesson is that default Docker networking is not enough for protection grade claims. Container networking must be selected and tested deliberately.
Containers Villén Martínez and Prada Hurtado (2024)	EPICS is a Docker microservice platform for edge protection and control. The platform separates the system into services for SV acquisition, shared data exchange, line differential protection, distance protection, disturbance recording, remote control, and GOOSE/SV handling. The host uses CentOS Linux on generic server hardware. The server has 20 CPU cores at 2 GHz. Docker runs the microservices. Critical microservices are implemented in C++. Critical services receive explicit CPU core allocation. Validation uses RTDS, GTNETx2-SV, GTNET-GSE, and RSCAD in hardware in the loop tests.	This paper gives a clear microservice pattern for vPAC. The useful idea is not only that protection can run in Docker. The more important idea is that acquisition, shared data, and protection algorithms can be separated into services. The risk is resource planning. The authors estimate at least 26 cores for a 10 bay setup. A real platform therefore needs CPU budgeting and service placement rules.

Continued on next page

Study and category	What was built and what software was used	What the study teaches
Containers Tellado et al. (2025)	The paper deploys IEC 61850 logical nodes as containers on an edge device. The containerized services include a protection logical node, a breaker logical node, a gateway, and an SV visualizer. The platform is an NVIDIA Jetson Nano. Docker is selected after discussing Docker, Podman, and LXD. Docker macvlan is used for process bus networks. Docker bridge networks are used for station bus and general traffic. Validation uses an OMICRON CMC256 SV generator. The tested SV rates are 4000 and 4800 packets/s. The paper also uses bmon and LSTM based network load forecasting.	This study shows that Docker network mode is a deployment design choice. Process bus traffic needs layer 2 access. That is why macvlan matters. Station bus or general traffic can use simpler bridge networks. The lesson is that vPAC cannot hide process bus design inside generic container defaults. Interface assignment, VLANs, and multicast behavior must be planned.
Virtual Machines Kabbara et al. (2024)	The paper builds a virtualized IEC 61850 controller testbed. The implementation is mixed: it uses VMs, Docker containers, SDN emulation, and RTDS process simulation. The setup uses four VMs. Two VMs host vIED logic and Docker containers. One VM hosts Mininet. One VM hosts OpenDaylight `0.8.4`. IEC 61850 SV, GOOSE, and MMS are generated from an RTDS/RSCAD IEEE 5 bus model through a GTNETx2 card. The tests vary the number of vIEDs, communication paths, SDN conditions, and topology.	This study shows that placement matters. A vIED is not only an application. It has a path through VMs, containers, switches, and IEC 61850 messages. Each extra hop can affect timing. The practical lesson is to treat VM boundaries, container boundaries, and network hops as part of the protection timing path.
Virtual Machines Rexwinkel et al. (2025)	The paper presents a virtual protection relay proof of concept for a realistic distribution substation. It discusses both relay hardware emulation through virtual machines and POSIX/Linux container-based PAC applications on COTS servers. Two VPR servers run in parallel for redundancy. The system uses IEC 61850-9-2LE SMV, GOOSE, PRP LAN A/B, VLANs, PTP/IEEE 1588, redundant GNSS clocks, Siemens 6MU85 merging units, GE MU320 merging units, and RTDS-generated process values. Validation includes fault scenarios, operating time comparison, interoperability checks, and abnormal condition testing plans.	This is one of the most practical transition studies. It shows that virtual protection is not just a technology problem. The platform also needs redundancy, time synchronization, merging unit interoperability, PRP networking, configuration tooling, and server tuning. The lesson is that vPAC must be specified as a complete platform. A virtualization technology alone is not enough.

Continued on next page

Study and category	What was built and what software was used	What the study teaches
Industrial Platform Siemens (2026)	SIPROTEC V is a commercial virtualized protection and control platform. It moves SIPROTEC 5 style bay protection functions from fixed relay hardware to software hosted virtual IEDs on substation grade computing hardware. Public material describes hardened Debian Linux with real time capabilities. It also describes a SIPROTEC V substation core system, software hosted protection per bay, up to 60 virtual IEDs, IEC 61850-9-2LE, IEC 61869-9, GOOSE, MMS, IEEE 1588, NTP, PRP, DIGSI 5, IEC 61850 System Configurator, Web UI, licensing, diagnostics, IEC 62443 safeguards, and CIS hardening. The exact low level VM/container runtime is not publicly disclosed.	This example shows what a commercial vPAC product must include. The virtualization technology is only one part. The product also needs engineering tools, security hardening, redundancy, diagnostics, update handling, licensing, and utility grade lifecycle support. The lesson is that industrial vPAC is a platform problem. It is not only an application packaging problem.

Classification note: Kabbara et al. (2024) and Rexwinkel et al. (2025) use both VMs and Docker in their setups. They are grouped here under Virtual Machines because that is the primary isolation layer in each architecture: the Docker containers in Kabbara are nested inside VMs, and the Stinskiy setup uses VMs as the main protection host. Siemens (2026) is analyzed as an industrial platform. Public SIPROTEC V material does not fully disclose the low-level virtualization technology; the product is evaluated here based on its platform-level properties, not its runtime.

No strictly VM-only academic vPAC solution was found in the retained literature. The VM-classified studies combine virtual machines with Docker or other container tools, with the virtual machine serving as the primary isolation layer. This is a characteristic of the published literature rather than a selection bias, since the search did not exclude VM-only implementations.

Table 4.2 gives the reason each retained evidence source belongs in the review, making the inclusion criteria explicit.

Table 4.2: Justification for retaining the main vPAC evidence sources.

Evidence source	**Reason for inclusion**
Nguyen et al. (2025)	Retained because it builds a large centralized container-based vIED setup with IEC 61850 Sampled Values and GOOSE output. It directly supports the discussion of scalability and network ingress limits.
Torres et al. (2024)	Retained because it builds Docker-based virtual IEDs with IEC 61850 behavior and shows how scaling affects traffic and trip delay. It is useful as a warning about default container networking.
Villén Martínez and Prada Hurtado (2024)	Retained because it shows a microservice style protection and control platform using Docker, explicit CPU allocation, and hardware-in-the-loop testing. It supports the discussion of service separation and resource planning.
Tellado et al. (2025)	Retained because it deploys IEC 61850 logical nodes as containers and compares network choices such as bridge and macvlan. It directly supports the claim that container networking is a deployment design decision.
Kabbara et al. (2024)	Retained because it validates virtualized IEC 61850 controllers in a testbed that combines VMs, Docker containers, SDN emulation, and RTDS process simulation. It shows that placement and network path both affect timing.

Evidence source	Reason for inclusion
Rexwinkel et al. (2025)	Retained because it presents a virtual protection relay proof of concept with redundancy, IEC 61850 traffic, merging units, PRP networking, and time synchronization. It is one of the clearest examples of vPAC as a full platform problem.
Siemens (2026)	Retained because it shows a commercial virtualized protection platform with engineering tools, diagnostics, security, lifecycle support, and IEC 61850 integration. It supports the industrial readiness discussion.
LF Energy (2026)	Retained because it represents an open industrial platform direction for hosting virtualized substation applications. It supports the discussion of workload separation, platform control, and lifecycle management.
ABB (2019)	Retained because it describes the architectural move from bay-level relay separation toward centralized protection and control. It supports the discussion of centralization as an architecture transition rather than a simple server consolidation step.

This retained set of 9 sources is small, so the thesis does not treat the counts as field-wide statistics. Figure 4.1 is used as a source role classification for the review table. It shows that the retained academic studies are mostly container-based. Some studies evaluate both containers and VMs within the same experimental setup; these are referred to here as mixed testbeds. The VM-based side of the evidence is represented by such mixed testbeds, virtual relay work, and industrial platform material, with the experiment contributing a direct deployment comparison on the same host.

The distribution in Figure 4.1 shows that the retained academic studies are predominantly container-based. Docker appears across the majority of the direct vPAC studies, while the VM-based side is represented by mixed testbeds and virtual relay work. This imbalance reflects the state of published evidence rather than a conclusion about which technology is more suitable for deployment; VM-based deployments require more platform infrastructure and are more difficult to evaluate in academic settings without industrial system access (Tran et al., 2016), which explains why fewer academic prototypes appear in that category. The counts in Figure 4.1 should not be interpreted as a market share or as a general survey of the field.

Figure 4.2 maps each virtualization technology to its primary contribution: Containers supply packaging flexibility and dense deployment; VM-based virtualization supplies stronger isolation boundaries; and industrial platform requirements supply the engineering, communication, and lifecycle layers that neither technology delivers on its own.

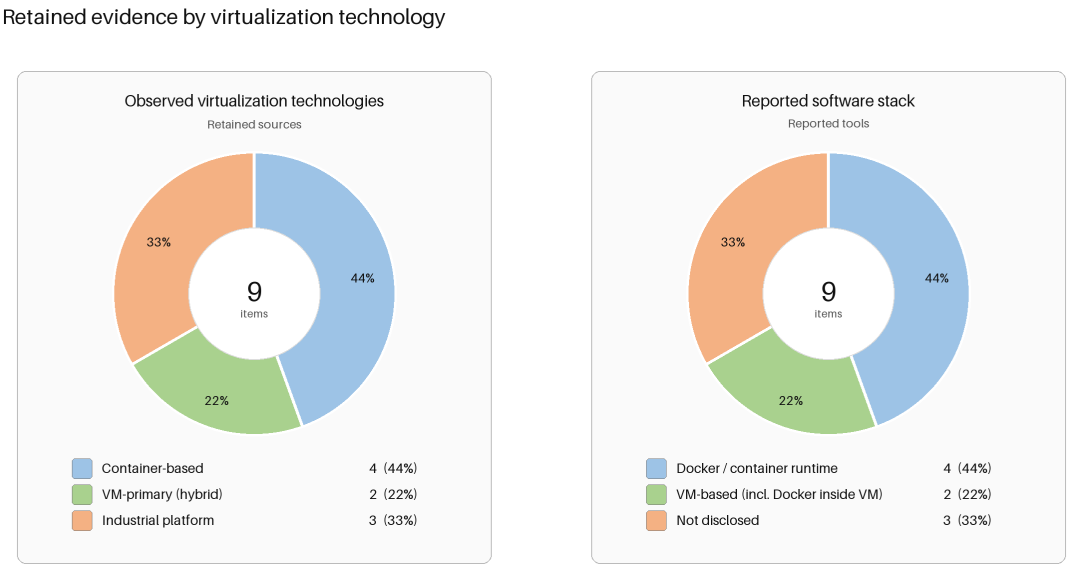

Figure 4.1: Distribution of virtualization technologies across the retained evidence sources.

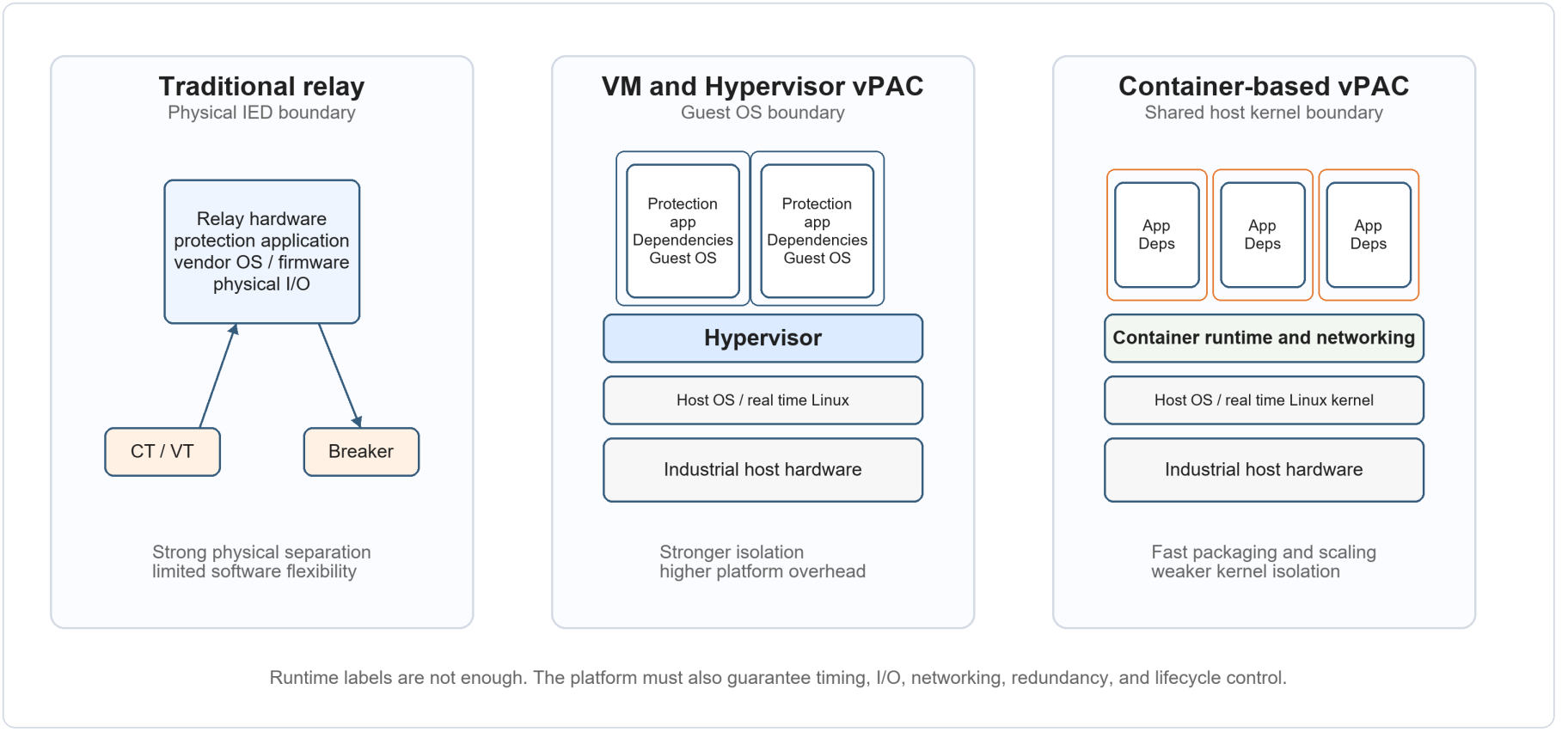

Figure 4.2: Virtualization technology roles used to interpret the review table.

Figure 4.3 places these findings into deployment layers, spanning technology choice, network design, CPU configuration, orchestration, and lifecycle management, to show why deployment readiness depends on the entire stack rather than on any single layer.

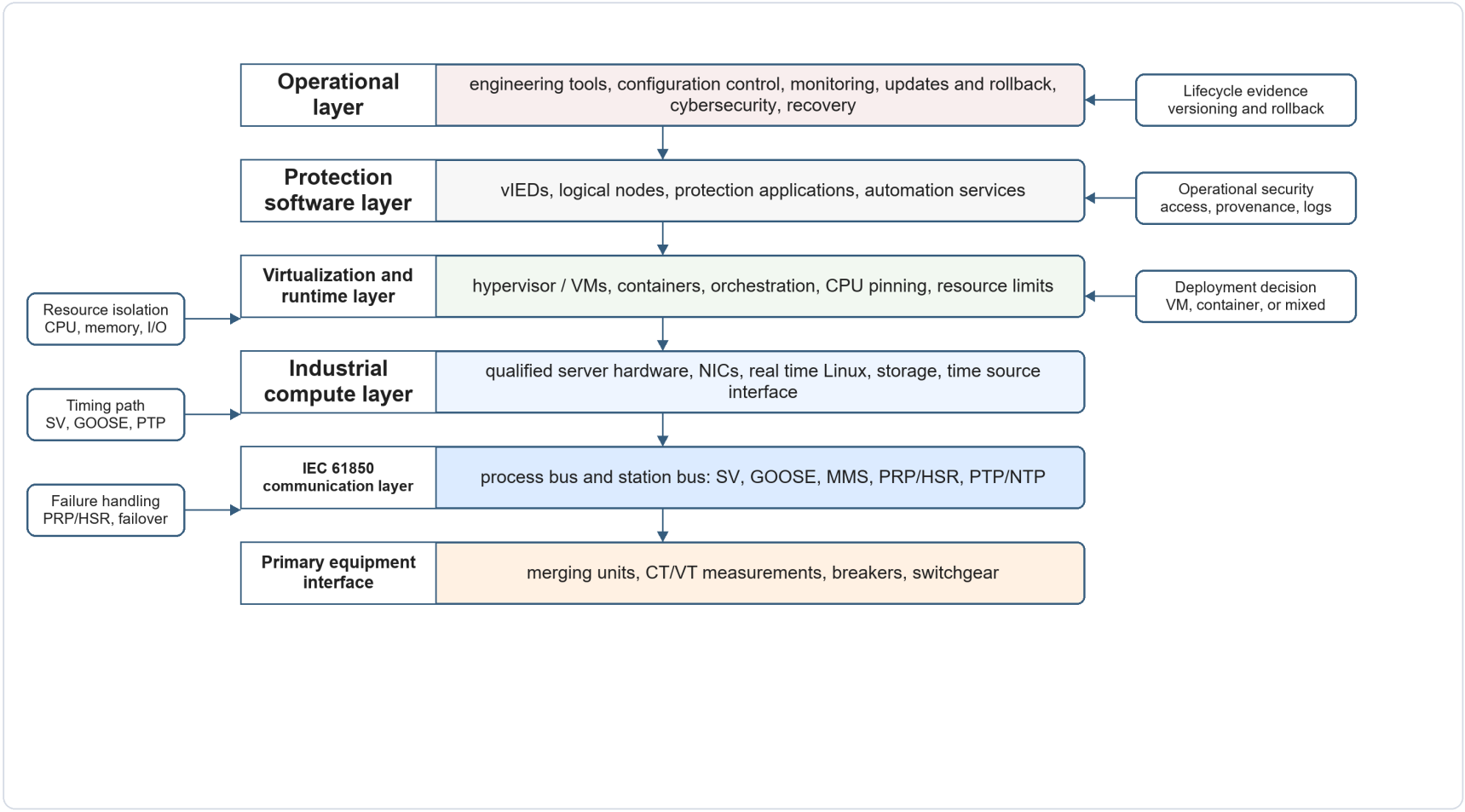

Figure 4.3: vPAC deployment layers: from field equipment and IEC 61850 communication through compute infrastructure to protection software and operations.

The lesson from Figure 4.3 is that neither virtualization technology operates in isolation. A container or VM that passes a scheduling test in a lab does not automatically satisfy the full deployment stack. Field timing arrives from the process bus; network timing constraints set the upper bound; the compute host must isolate and schedule correctly; the software protection instance must meet those bounds; and all of this must be supportable through operations and recovery. Each layer adds requirements that cannot be satisfied by technology selection alone. This is why later chapters treat SEAPATH and SIPROTEC V as platform evidence, not technology benchmarks.

4.2 Containers

Container-based architectures are the most prominent approach in the direct vPAC studies reviewed here, a pattern consistent with the broader real-time container literature (Queiroz et al., 2023). Protection functions are packaged as lightweight services, often called vIEDs, logical nodes, or microservice-structured protection functions (Kabbara et al., 2022), and Docker is the most common runtime in the reviewed studies (Nguyen et al., 2025). The primary strength in the reviewed evidence is packaging flexibility: a container image supports faster startup and many software instances on one host (Abeni, 2024), and makes a protection service easier to version, transfer, and redeploy (Queiroz et al., 2023). However, containers share the host kernel and therefore require careful host configuration before they can be trusted for protection work (Struhár et al., 2020).

Villén Martínez and Prada Hurtado (2024) illustrate one decomposition pattern through the EPICS platform: SV acquisition, shared data exchange, protection algorithm execution, and output handling are separated into distinct microservices, each managed independently. Critical microservices receive explicit CPU core allocation, meaning specific CPU cores are reserved exclusively for them so that scheduling interference from other workloads is prevented. The authors estimate at least 26 CPU cores for a ten-bay substation (ten protected circuits, each with its own protection function), which shows that containerized vPAC requires resource budgeting before deployment, not only after (Villén Martínez and Prada Hurtado, 2024).

The central bottleneck in the reviewed container studies is not raw computation but communication. Each container instance that hosts a protection function needs its own continuous stream of incoming SV frames: at the rates specified by IEC 61850, a single measurement source sends up to 4000 SV frames per second. When many container instances consolidate onto one server, all of these incoming streams arrive through the same physical NIC. Nguyen et al. (2025) test up to 400 vPAC container instances on a single server and find that this inbound SV load and the physical Ethernet channel capacity become the limiting factor before CPU capacity is exhausted. Torres et al.

(2024) confirm that scaling also increases GOOSE trip delay: as more virtual IEDs share a network, the time between fault detection and the breaker operation message grows as traffic competes for bandwidth.

As explained in Chapter 2, GOOSE and SV frames travel as Ethernet multicast at Layer 2 and cannot be received through a standard IP virtual network bridge. Tellado et al. (2025) show this directly: process bus access requires macvlan mode, which gives each container its own MAC address on the physical Ethernet so it can participate in Layer 2 multicast, while station bus and management traffic that uses IP can use standard bridge networking. Network mode is therefore a first design decision, not a configuration afterthought.

Containers help with packaging and scaling, but they do not automatically solve real-time networking or isolation problems: the host must be configured for CPU allocation, correct network mode, process bus access, and privilege control before container deployment can be treated as a vPAC deployment (Queiroz et al., 2023).

4.3 Virtual Machines

VM-based virtualization runs applications inside guest operating systems managed by a hypervisor. A hypervisor is the software layer that creates, schedules, and manages the VMs on shared hardware; the VM and the hypervisor are not separate deployment choices but one combined mechanism (Tran et al., 2016). The deployment question is what this mechanism provides for protection work: workload isolation, resource control, and I/O management.

This virtualization technology is attractive when protection functions require stronger isolation (each VM runs its own guest operating system, preventing workloads from sharing kernel state) or when legacy relay software depends on a complete operating system environment. A virtual protection relay can be hosted as a full software environment, while the hypervisor controls how compute, memory, and interfaces are allocated. Rexwinkel et al. (2025) describe this as one route for moving relay instances from fixed hardware to COTS server infrastructure.

Kabbara et al. (2024) show that placement affects timing in VM-based setups: their testbed runs virtual IED logic inside Docker containers that are themselves nested inside VMs, so each IEC 61850 message must cross VM boundaries, container boundaries, and network switches before reaching the protection function. Each boundary adds latency. A VM is therefore not only a packaging choice; it is part of the communication path, and the number and position of virtualization boundaries can determine whether protection timing requirements are met.

Rexwinkel et al. (2025) demonstrate what a VM-based virtual protection relay requires

beyond the virtualization technology itself. As introduced in Chapter 2, PRP sends each frame simultaneously on two independent physical paths so that one link failure causes no interruption; the Stinskiy implementation runs PRP on redundant LAN A and LAN B paths. The setup also requires VLAN configuration for traffic separation, PTP and IEEE 1588 time synchronization, redundant GNSS (satellite-based) clock sources, and validated interoperability with multiple merging unit types from different vendors. This level of platform specification is absent from most academic VM studies, showing that VM-based vPAC is understood at prototype level in research but requires substantial additional engineering for production use.

However, the tradeoff is overhead. Each VM typically requires its own guest operating system, virtualized device stack, and additional scheduler and I/O paths, which increases memory footprint and CPU demand compared to containers. VM startup is also slower because the guest OS must boot before the workload can run. Abeni (2024) document these overhead differences in a direct performance comparison: a KVM/QEMU VM running a Linux guest requires reserving approximately 399 MB of host memory, while a container started by a low-level runtime such as crun consumes approximately 18 MB, a difference of more than 20 times.

For that reason, VM-based virtualization is best understood as the conservative isolation option. It does not automatically support fast horizontal scaling, because creating a new VM instance requires booting a guest operating system rather than starting a container from an existing image (Abeni, 2024). It also requires validation of CPU scheduling policy (confirming that the scheduler respects the timing bounds of the protection function under concurrent workloads), I/O latency, guest timing, and process bus communication (Tran et al., 2016). VM-based virtualization is therefore the conservative choice for workloads that require clear OS-level separation and controlled resource allocation.

4.4 Virtualization Technology Comparison

Containers and VM-based virtualization show different performance and isolation characteristics that are relevant to vPAC. Abeni (2024) measure both in a direct comparison: containers (OS-level virtualization) introduce lower worst-case latency than KVM-based VMs running a standard Linux kernel, while VMs using Preempt-RT provide latency competitive with containers. On resource consumption, containers are substantially lighter. Neither technology removes the need for explicit platform configuration; both require scheduler tuning, appropriate network mode, and resource allocation before protection workloads behave predictably.

For vPAC, the practical distinction is isolation depth and operational density. Container-based deployment fits use cases where packaging, fast instance creation, and image-based

lifecycle management are priorities. VM-based virtualization fits use cases where each workload must be separated by a full guest operating system boundary, meaning workloads must not share any kernel state with other instances on the same host.

Container isolation is weaker because containers share the host kernel: a process inside one container can affect another container through shared kernel state, shared network stacks, or shared timing services if the host is not configured to prevent it. VM-based virtualization provides stronger isolation because each VM runs its own guest operating system, which removes most shared kernel dependencies (Abeni, 2024). The tradeoff is that VMs carry higher overhead and lower density. For protection use, this means that a container platform must compensate with host-level controls such as CPU allocation, cgroups, privileges, and network design.

Networking complexity cuts across both approaches: IEC 61850 SV, GOOSE, MMS, redundancy traffic, and time synchronization determine end-to-end behavior (Kabbara et al., 2022). Because GOOSE and SV are Layer 2 Ethernet multicast protocols, neither virtualization technology provides suitable process bus access by default. Containers must use macvlan or host networking as shown by Tellado et al. (2025). VM-based deployments can use device passthrough, which assigns a physical NIC directly to the guest VM and bypasses the hypervisor's virtual network stack, but this requires both hypervisor and hardware support and adds configuration complexity (Casini et al., 2021). Both approaches therefore require explicit network design decisions before process bus communication can work.

High container density also concentrates SV traffic: when many container instances run on one host, all their SV streams arrive through the same physical NIC and process bus path, which Nguyen et al. (2025) identify as the primary scaling constraint. VM-based deployments distribute workloads differently because each VM consumes more resources, meaning fewer instances per host, but each instance can have clearer I/O boundaries (Casini et al., 2021).

The comparison therefore does not produce a winner between the two virtualization technologies (Abeni, 2024). The reviewed studies show that a vPAC platform also needs the requirements summarized in Table 4.3: CPU control, predictable I/O, IEC 61850 networking, time synchronization, redundancy, security controls, engineering tools, and lifecycle management.

Table 4.3 turns the review into design implications by combining direct vPAC studies with enabling background evidence and industrial platform sources. It links each requirement to why it matters, how it affects Containers, how it affects Virtual Machines, and which evidence in this thesis supports the point.

Table 4.3: Requirements and deployment implications for vPAC platform design.

Requirement	Why it matters	Container implication	Virtual Machines implication	Evidence used in this thesis
Compute isolation	Protection functions need predictable CPU time, memory behavior, and reduced interference from other workloads.	Containers need explicit CPU pinning, cgroups, resource limits, scheduler policy, and host tuning. A container image alone does not provide isolation.	Virtual Machines provide stronger workload boundaries, but scheduler configuration, vCPU placement, guest behavior, and I/O path still need validation.	Abeni et al. (2022); Abeni (2024); Tran et al. (2016); Queiroz et al. (2023).
IEC 61850 networking	SV and GOOSE traffic are part of the protection path. Network behavior can become the limit even when CPU capacity is available.	Container network mode matters. Bridge, host networking, macvlan, interface assignment, VLANs, and multicast handling can change process bus behavior.	VM-based deployments must still handle I/O virtualization, NIC access, multicast behavior, network path length, and communication latency.	Nguyen et al. (2025); Torres et al. (2024); Tellado et al. (2025); Casini et al. (2021).
Time synchronization	Protection behavior may depend on time aligned measurements, event ordering, and clock accuracy.	Containers usually depend on host and network time services. Host clock discipline, privileges, and PTP or NTP integration must be controlled.	VM-based deployments must avoid unmanaged guest clock behavior and must define how synchronization crosses the virtualization layer.	Kabbara et al. (2024); Rexwinkel et al. (2025).
Redundancy and recovery	Centralization can reduce device count, but it can also concentrate dependency on shared servers and networks.	Container replication is useful, but restart or redeployment is not the same as protection-grade failover. Recovery behavior must be engineered.	VM-based workloads can be placed on redundant platforms, but failover, state handling, and recovery time still need validation.	IEEE Power System Relaying Committee (2015); Rexwinkel et al. (2025); ABB (2019).
Lifecycle management	A deployable platform must support configuration, updates, rollback, diagnostics, monitoring, and traceability over time.	Containers can help packaging and versioning, but image control, update policy, and configuration history must be part of the platform.	VM images can support controlled versions, but they still need update workflow, rollback process, monitoring, and maintenance procedures.	Siemens (2026); LF Energy (2026); MacDonald et al. (2025).
Cybersecurity	Virtualized protection introduces shared hosts, images, remote management, update paths, logs, and access control points.	Container images, privileges, registries, host access, and runtime settings must be controlled and audited.	VM boundaries can support separation, but management interfaces, guest images, patches, and logs still require security governance.	Kabbara et al. (2025); Siemens (2026).
Validation scope	Basic deployment does not prove protection readiness. The whole path from measurement input to protection output must be tested.	Container tests should include realistic traffic, host load, network mode, timing, redundancy, and recovery behavior before protection claims are made.	VM-based tests should include scheduler stress, I/O latency, guest timing, communication path, redundancy, and failover behavior.	Kabbara et al. (2024); Nguyen et al. (2025); Rexwinkel et al. (2025).

The networking row of Table 4.3 carries specific constraints on the VM side. A VM reaches the network through a virtualized I/O path, which adds overhead compared to direct host access, and the physical NIC is shared across the VMs placed on the same host, so contention can grow as VM density rises (Casini et al., 2021). Multicast traffic, which IEC 61850 GOOSE and Sampled Values depend on, must be handled by the hypervisor before it reaches the guest, and the guest boundary itself adds latency on the protection message path (Tran et al., 2016; Casini et al., 2021). These constraints mean that VM-based process bus access has to be designed with the same care as container network mode, rather than assumed to follow automatically from the stronger isolation that VMs provide.

The heavier per-instance footprint of VMs also bounds how many protection instances a single host can carry. SIPROTEC V supports up to 60 virtual IED instances per platform (Siemens, 2026), and the experiment in Chapter 6 measured approximately 627 to 638 MB of RAM per VM instance. Combining these figures gives a grounded estimate: hosting tens of VM-based protection instances on one platform would require on the order of tens of gigabytes of RAM for the guests alone, before host and hypervisor overhead, which is consistent with the lower density expected from VM-based deployment relative to containers.

Although compute resources, memory footprint, and startup time are the most visible hardware metrics, the reviewed evidence consistently places IEC 61850 communication at the centre of the deployment feasibility question. Nguyen et al. (2025) show that SV ingress from the process bus saturates the server's physical Ethernet channels before CPU capacity is exhausted: the network, not the processor, determines the upper limit of consolidation. Tellado et al. (2025) show that container network mode determines whether a container instance can receive Layer 2 multicast frames at all; a container configured with the wrong network driver cannot participate in process bus communication regardless of its compute allocation. Torres et al. (2024) confirm that GOOSE trip delay increases as more virtual IEDs share a network, so protection timing degrades through communication behavior rather than algorithm complexity. Kabbara et al. (2024) demonstrate that each VM boundary and container boundary on the message path adds latency, meaning the number and position of virtualization layers directly affects whether IEC 61850 timing requirements can be met. Rexwinkel et al. (2025) show that a virtual protection relay at production level requires PRP redundancy, PTP synchronization, and validated merging unit interoperability as non-negotiable communication requirements. Hardware provisioning sets the capacity envelope, but the IEC 61850 communication architecture determines whether a vPAC deployment is feasible in the first place.

If research prototypes show feasibility, the immediate question becomes: what must an industrial platform add before the architecture can operate in a real substation?

Chapter 5

Industrial vPAC Platforms

Academic prototypes show that protection functions can be moved into software-hosted environments. Industrial approaches show a different question: what has to surround that software before it can be operated as part of a substation platform. This chapter therefore reads SIPROTEC V, SEAPATH, and ABB centralized protection and control as platform evidence rather than as identical products.

SIPROTEC V is examined as a commercial virtualized protection platform (Siemens, 2026). SEAPATH is examined as an open industrial platform layer for substation virtualization (LF Energy, 2026). ABB centralized protection and control is included because it describes the architectural shift from separated bay relays toward centralized software based protection (ABB, 2019). The chapter asks what each source makes visible, what it does not disclose, and what deployment implication can safely be drawn for vPAC.

5.1 Why Industry Matters

Many academic studies focus on one part of the problem, such as scheduling latency, container overhead, IEC 61850 communication delay, or a specific protection prototype: Rösch et al. (2022) concentrate on communication delay modeling, and Kabbara et al. (2024) validate virtualized IEC 61850 controllers in a laboratory setup. Industrial platforms must solve the broader integration problem: they need engineering workflows, hardware qualification, cybersecurity, redundancy, communication integration, monitoring, updates, and long-term maintainability (MacDonald et al., 2025).

5.2 SIPROTEC V

SIPROTEC V is the most complete public example of a commercial vPAC product with full engineering workflow and lifecycle support. Siemens (2026) present SIPROTEC V as an industrial implementation of virtualized protection and control in which protection functionality is decoupled from fixed relay hardware and deployed on centralized computing infrastructure. Public Siemens material describes a hardened Debian Linux platform with real-time capabilities, a substation core system, software-hosted SIPROTEC 5 pro-

tection per bay, support for up to 60 virtual IEDs, IEC 61850 process bus and station bus integration, engineering through DIGSI 5 (Siemens' engineering software for configuring SIPROTEC protection devices), and security and lifecycle features.

The public material is useful because it shows the product level responsibilities around virtualized protection. It does not only describe a virtualization technology; it also describes engineering tools, licensing, diagnostics, time synchronization, redundancy, cybersecurity controls, network management, and update workflows. The low-level runtime implementation is not fully disclosed, so this thesis does not use SIPROTEC V as proof for one virtualization technology. It uses it as evidence that commercial vPAC must include operational controls around the hosted protection functions.

5.3 ABB Centralized Protection and Control

ABB centralized protection and control shows how moving from physical bay relay separation toward centralized software protection changes protection engineering requirements. ABB (2019) describe a shift from physically separated bay-level relays toward centralized protection and control units running software protection functions on industrial computing hardware. The document does not disclose a detailed container or VM runtime, so it is not evidence for Docker, Podman, KVM, or another specific execution environment.

Its value is architectural. The ABB concept frames the transition from physical separation to logical separation and emphasizes redundancy, process bus communication, merging units, and hybrid deployment options (configurations where some bays use centralized virtual protection while others retain local relay hardware during transition). This architectural direction is also consistent with the IEEE centralized protection report (IEEE Power System Relaying Committee, 2015). For this thesis, the deployment implication is that centralization cannot be treated as simple server consolidation. It must preserve the protection boundaries that were previously provided by separate relay hardware.

5.4 SEAPATH

SEAPATH addresses the platform infrastructure layer that vPAC deployments need below the protection application software. LF Energy (2026) present SEAPATH as an open industrial platform direction rather than a single packaged protection product. It is relevant because it focuses on the infrastructure layer needed to host virtualized substation applications: real-time Linux, virtualization support, platform separation, orchestration, high availability, and lifecycle control.

SEAPATH is important for this thesis because it makes explicit what the platform layer

must provide. A vPAC deployment needs somewhere to place workloads, apply resource rules, manage virtual networks, monitor state, and recover after failures. Compared with SIPROTEC V, which is closer to a packaged protection platform, SEAPATH is closer to enabling infrastructure for virtualized substation workloads; the experiment in Chapter 6 therefore uses a SEAPATH-based environment to test container deployment mechanics.

5.5 Industrial Platform Comparison

SIPROTEC V, SEAPATH, and ABB centralized protection and control do not provide the same kind of evidence. Table 5.1 therefore uses common comparison columns rather than a single broad lesson column. This makes the limits of each source visible.

Table 5.1: Comparison of industrial vPAC related platform evidence.

Approach	**Evidence type**	**Visible platform elements**	**Technology disclosure**	**Deployment implication for this thesis**	**Limit of the evidence**
SIPROTEC V	Commercial virtualized protection and control platform	Hardened Debian Linux platform, substation core system, IEC 61850 process bus and station bus integration, engineering through DIGSI 5, diagnostics, security, licensing, and lifecycle support	the specific low-level container or VM implementation is not publicly disclosed	A deployable vPAC product needs engineering workflow, diagnostics, cybersecurity controls, redundancy, and update management around the protection software	Public material does not provide enough low-level runtime detail to classify it as a pure container or VM architecture
ABB centralized protection and control	Centralized protection architecture concept	Centralized computing, logical separation, process bus connection, merging units, redundancy, fallback thinking, and hybrid deployment options	The specific container or VM technology is not disclosed	Centralization changes the protection architecture. It replaces physical relay separation with logical separation, so communication, redundancy, and recovery must be designed explicitly	The source is mainly architectural and does not provide a complete tested virtualization stack
SEAPATH	Open industrial platform layer for substation virtualization	real-time Linux direction, virtualization support (KVM-based VM hosting and container support), platform separation, orchestration, high availability, workload management, and lifecycle control	Public material describes platform capabilities. SEAPATH is a host infrastructure layer, not a single packaged protection product	vPAC needs a controlled platform layer for workload placement, network configuration, monitoring, updates, and recovery	SEAPATH is enabling infrastructure, so protection application behavior still has to be validated separately

The three industrial examples should not be read as the same type of evidence. SIPROTEC V represents product integration because it shows how virtualized protection can be packaged with engineering tools, diagnostics, security, and lifecycle support. ABB centralized protection and control represents architecture transition because it shows how protection can move from physical bay relay separation toward centralized and logically separated execution. SEAPATH represents platform infrastructure because it focuses on the host layer needed for workload separation, availability, monitoring, updates, and recovery. SEAPATH is based on KVM for VM hosting and also supports containers, providing the host-level services that both virtualization technologies depend on. It is particularly relevant to this thesis because Chapter 6 uses a SEAPATH-based platform for the deployment experiment, making these platform capabilities directly applicable to the experiment interpretation.

This distinction is important because industrial vPAC should be designed as a controlled operating environment, not as a general purpose virtualization host. A general purpose host can run software. A protection platform must also define where software is placed, how it is isolated, how IEC 61850 traffic reaches it, how changes are controlled, how failure is detected, and how service is restored after a fault.

These industrial examples point to the same deployment direction, but through different evidence paths. Technology selection is only one layer. A deployable vPAC platform also needs qualified hardware, tuned operating system behavior, IEC 61850 integration, redundancy, cybersecurity, engineering workflow, and lifecycle management (MacDonald et al., 2025).

5.6 Deployment Implications

The reviewed industrial sources support three deployment implications. SIPROTEC V makes engineering workflow and lifecycle control visible. SEAPATH makes the host platform and workload management layer visible. ABB centralized protection and control makes the move from physical relay separation to logical separation visible. Together, these sources show that vPAC must be evaluated as a platform with its virtualization layer, engineering, communication, security, and lifecycle layers, not only as a technology choice (MacDonald et al., 2025).

Isolation must be designed at several layers because each source exposes a different part of the separation problem. ABB's concept shows that physical relay separation is replaced by logical separation (ABB, 2019). SEAPATH shows that the platform layer must place and manage workloads (LF Energy, 2026). SIPROTEC V shows that a commercial platform also needs diagnostics, engineering tools, security controls, and lifecycle support (Siemens, 2026). For vPAC, this means that CPU allocation, network separation, I/O handling,

management access, and failure isolation must be treated as platform requirements.

Networking remains central because virtualized protection still depends on IEC 61850 traffic. SIPROTEC V public material describes process bus and station bus integration. ABB centralized protection and control emphasizes process bus communication and merging units. SEAPATH emphasizes platform networking, high availability, and workload separation. The deployment implication is that SV, GOOSE, MMS, PRP or HSR, and time synchronization cannot be added after the technology choice; they must be part of the architecture from the start (Rexwinkel et al., 2025).

Lifecycle management also becomes part of protection engineering. SIPROTEC V shows this through engineering, diagnostics, security, and update-related platform features (Siemens, 2026); SEAPATH supports the same conclusion from the infrastructure side through its focus on orchestration, monitoring, availability, and lifecycle control (LF Energy, 2026). A software-hosted protection function must be configured, updated, rolled back, monitored, and recovered without losing traceability (MacDonald et al., 2025).

Cybersecurity also changes when protection becomes software-hosted. A virtualized platform may introduce shared hosts, container images, remote management interfaces, update pipelines, access control rules, and configuration repositories. These parts must be protected and audited. For that reason, vPAC security is not only network security. It also includes image provenance, privilege control, patch management, logging, and configuration traceability (Kabbara et al., 2025).

Chapter 6

Deployment Experiment

This chapter reports the deployment experiment. The experiment compares container-based and VM-based deployment on the same SEAPATH-based platform: three Podman controller instances were prepared offline, transferred via Ansible, given separate static private addresses, and verified through ICMP, while three KVM-based VMs were deployed on the same host for direct comparison. The experiment did not run a protection algorithm; its scope was deployment feasibility and basic network connectivity for both virtualization technologies.

This chapter documents the only experimental evidence in the thesis. It names public platform and tools including SEAPATH, Debian, Podman, Ansible, KVM, and libvirt. The three container experiment workloads are called controller instance 1, controller instance 2, and controller instance 3. The three VM baseline instances are called VM instance 1, VM instance 2, and VM instance 3.

Figure 6.1 illustrates the experiment topology. A separate controller machine was used to prepare the container image and run Ansible playbooks. The image was transferred as an archive to the SEAPATH host, which ran both the container experiment and the VM baseline side by side. Three container instances and three VM instances were deployed on the same host for comparison. Exactly three instances were chosen because three instances exercise all six directed communication paths between instances and demonstrate multi-instance addressing without making site-specific scaling claims.

The results establish deployment feasibility: container creation, automated deployment, static private network addressing, and ICMP communication all succeeded. The experiment scope is platform-level deployment mechanics; IEC 61850 traffic and protection-grade timing are discussed in Chapter 7.

6.1 Experimental Setup

The experiment used a SEAPATH-based virtualized platform, documented as SEAPATH version 1.2.1. The documented host operating system was Debian GNU/Linux 13, also called Trixie. The host had 31 GB of total RAM. During the test, approximately 2.5 GB

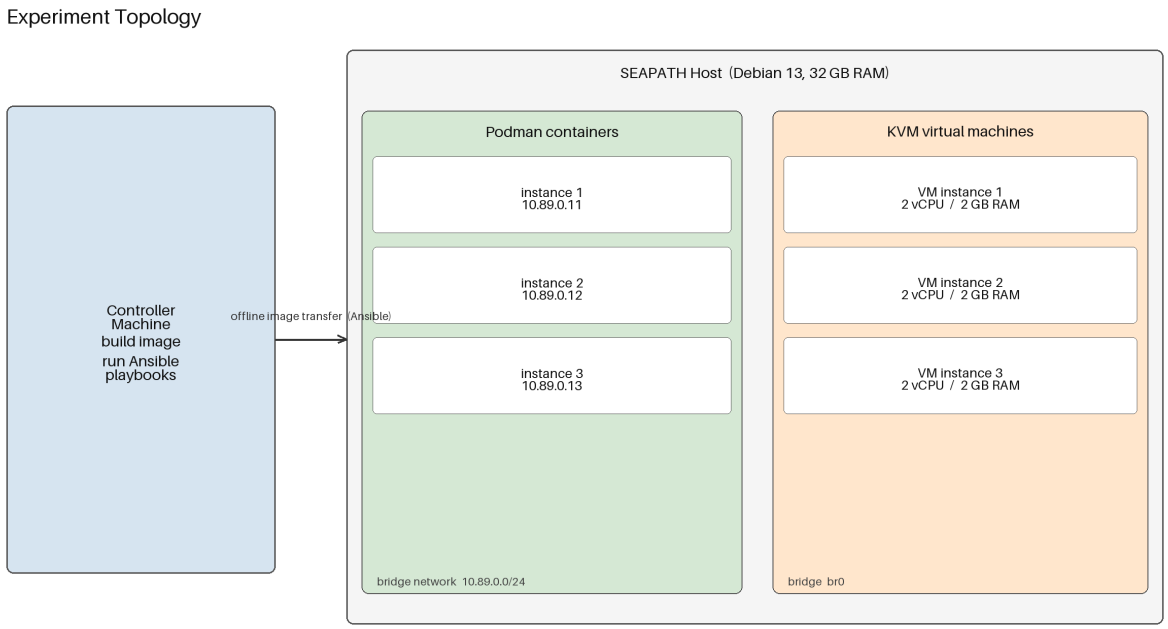

Figure 6.1: Experiment topology.

was in use with three VMs running, leaving about 26 GB free. Recorded CPU idle was 99.1 percent at that point, confirming that the test workload did not stress the host resources.

Before the experiment workloads were deployed, the SEAPATH server was installed from a purpose-built installer image and configured through the SEAPATH platform Ansible setup playbook. This platform-level setup established the br0 network bridge, which is the Open vSwitch (OVS) virtual switch interface on the SEAPATH host that connects VMs to the host network, along with KVM and libvirt hypervisor services, CPU isolation profiles, and the dedicated Ansible SSH user used for all subsequent remote deployment. The experiment workloads described in this chapter were added on top of this already-configured platform.

The VM baseline used KVM and libvirt with Debian 12 Bookworm guest images. Each VM was configured with 2 virtual CPUs and 2048 MB of RAM. The container experiment used Podman, with version 5.4.2 recorded in the experiment documentation.

A separate controller machine was used to prepare and transfer the container image and to run Ansible playbooks against the SEAPATH target. This separation matters because the target platform was not treated as an unrestricted development machine. Instead, the deployment was handled as a controlled platform workflow: prepare the image, transfer it, load it locally, create the network, recreate the instances, and verify the result.

The container image was based on Debian 12 Bookworm. The base image was ex-

tended with diagnostic tools needed for testing, including `iputils-ping`, `iperf3`, `netcat-openbsd`, DNS tools, `traceroute`, `iproute2`, `curl`, `wget`, `htop`, and `tcpdump`. The image was saved and exported as a tar archive. The documented archive size was about 210 MB, within the 200 to 250 MB range noted in the experiment files.

Each container instance ran a Debian-based environment with network diagnostic tools, representative of a software controller workload on a virtualized substation platform. The experiment verified that containerized controller instances could be deployed, individually addressed, and reached over a private network.

The work started on a separate controller machine, where a Debian 12 Bookworm container image was built by extending a base image with network diagnostic tools and exported as a tar archive. Ansible transferred the tar archive to the SEAPATH target via a copy task and then ran `podman load` to import the image into the local Podman image store. Ansible then created a dedicated Podman bridge network on subnet `10.89.0.0/24` using `podman network create`, removed any old instances, and started three new controller instances from the same image. Host networking was not used because it would have given all containers the same network address as the host (see Table 6.1). Each instance was assigned a static private address: `10.89.0.11`, `10.89.0.12`, and `10.89.0.13`. ICMP ping checks were then run between the instances to verify connectivity. For context, the same host was also running three Debian 12 KVM and libvirt VMs, and ping results were recorded from both the containers and the VM baseline. Resource figures including RSS (Resident Set Size) memory (the portion of physical RAM actively held by each process) and boot time are discussed in Section 6.10.

6.2 Scope of the Experiment

Table 6.1 summarizes the experiment scope. The measured ping results confirm basic connectivity; they are not protection-grade timing evidence.

Table 6.1: Scope of the SEAPATH container deployment experiment.

Aspect	**Scope in this experiment**
Platform	SEAPATH version 1.2.1 based virtualized platform using Debian GNU/Linux 13 on the host. The host had 31 GB total RAM and 99.1 percent CPU idle during testing.
VM baseline	Three Debian 12 VMs deployed through KVM and libvirt, each configured with 2 virtual CPUs and 2048 MB RAM. The VM baseline was used only as local context for the platform.
Container runtime	Podman was used for the container test. The experiment documentation recorded Podman version 5.4.2.

Aspect	Scope in this experiment
Automation	Ansible was used to transfer the image, load it into Podman, create the network, remove old instances, start new instances, and report their status.
Container image	Debian 12 Bookworm based diagnostic image with basic network and troubleshooting tools.
Container instances	Three controller instances were created from the same image and kept running for network tests.
Network model	Host networking was rejected because it gave all containers the host network identity. A dedicated Podman bridge network on subnet `10.89.0.0/24` was used instead, with static private addresses assigned to each controller instance.
Measured result	ICMP ping between container instances and between VM instances. Packet loss and round trip time were recorded from the experiment notes.

6.3 Docker and Podman

The reviewed academic studies use Docker as the container runtime throughout: Nguyen et al. (2025), Torres et al. (2024), Villén Martínez and Prada Hurtado (2024), and Tellado et al. (2025) all build their container setups on Docker CE or Docker Engine. The experiment in this thesis uses Podman instead, which requires a brief explanation since the two tools are not interchangeable in every context.

Docker uses a background daemon process that manages all container lifecycle operations. Podman is daemonless: each container is launched as a direct child process of the invoking user or of the system. Both tools build and run OCI-compliant container images, so an image built with one can be loaded and run by the other. The key operational difference is that Docker requires the daemon to be running and available, while Podman operates without a central service, which aligns better with system-level deployment on a controlled platform like SEAPATH.

The SEAPATH platform uses Podman as its supported container runtime. The experiment therefore used Podman on the SEAPATH host and used the `containers.podman` Ansible collection for all deployment automation. The container image itself was built using Podman on the controller machine, saved as a tar archive with `podman save`, transferred via Ansible, and loaded on the SEAPATH host with `podman load`. Because both tools use the OCI image format, this workflow produces images that are functionally equivalent to Docker-built images.

The choice of Podman over Docker in this experiment is a platform decision, not a performance or architecture claim. The networking and isolation behavior tested here

is not unique to Podman; the same deployment steps would apply with Docker on a platform that supports it.

6.4 VM Baseline Procedure

The first step checked the existing VM path. Three Debian 12 guest VMs were deployed on the SEAPATH platform through KVM and libvirt using cloud image templates and platform-level automation. The Ansible playbook described in Section 6.6 was not used for the VM deployment; Ansible was used only for the container steps. After the VMs were running, basic ICMP ping checks were performed between the VM instances.

A VM baseline was run to give the container results local context: both sets of instances ran on the same SEAPATH host, which makes the round trip time comparison meaningful as a platform-level observation. This baseline does not prove that VM-based protection is ready for operation. It only shows that the platform could run separate guest operating systems and that the guest instances could reach each other over the VM network.

The VM configuration was specified through an Ansible inventory file that centralised all guest parameters in one place. Rather than configuring each VM manually through `virsh` commands, the inventory defined the three instances as parameterized entries: virtual CPU count, memory allocation, disk image path, bridge interface, and MAC address were each set once and applied consistently across all three guests by the SEAPATH deployment playbook. Debian 12 cloud images block password login by default and require a cloud-init configuration step to inject login credentials before SSH access is available; the `wait_for_connection: false` flag reflects this, as the deployment playbook did not attempt SSH verification immediately after VM creation. The VMs received network addresses from the host bridge interface; actual addresses are omitted as they are site-specific. Listing 6.1 shows the inventory structure used.

The VM deployment playbook was invoked with both the SEAPATH host inventory and the VM inventory:

Listing 6.2: Command used to run the VM deployment playbook from the controller machine.

```
ansible-playbook \
    -i inventories/seapath_inventory.yaml \
    -i inventories/vm_inventory.yaml \
    deploy_vms_standalone.yaml
```

After the deployment playbook completed, the VM running state was verified on the SEAPATH host using `virsh list -all`, which reported all three guests in running state. Because Debian 12 cloud images disable password authentication by default, SSH access

Listing 6.1: VM deployment inventory for the SEAPATH experiment. Template path is abbreviated; VM network addresses are site-specific and omitted.

```
VMs:
  hosts:
    vm-instance-1:
      vm_template: "<ansible_dir>/templates/vm/guest.xml.j2"
      vm_disk: "files/guest.qcow2"
      standalone_main_disk: "debian-12-generic-amd64.qcow2"
      wait_for_connection: false  # cloud-init required before SSH is available
      vm_features: []
      nb_cpu: 2
      memory: 2048      # MB
      bridges:
        - name: "br0"
          mac_address: "52:54:00:e4:ff:01"   # unique per VM

    vm-instance-2:
      vm_template: "<ansible_dir>/templates/vm/guest.xml.j2"
      vm_disk: "files/guest.qcow2"
      standalone_main_disk: "debian-12-generic-amd64.qcow2"
      wait_for_connection: false
      vm_features: []
      nb_cpu: 2
      memory: 2048
      bridges:
        - name: "br0"
          mac_address: "52:54:00:e4:ff:02"

    vm-instance-3:
      vm_template: "<ansible_dir>/templates/vm/guest.xml.j2"
      vm_disk: "files/guest.qcow2"
      standalone_main_disk: "debian-12-generic-amd64.qcow2"
      wait_for_connection: false
      vm_features: []
      nb_cpu: 2
      memory: 2048
      bridges:
        - name: "br0"
          mac_address: "52:54:00:e4:ff:03"

  vars:
    ansible_user: ansible
```

to each VM required a separate cloud-init step. A cloud-init configuration ISO containing login credentials was created on the controller machine for each VM using `cloud-localds`, transferred to the SEAPATH host, and attached to the corresponding VM as a virtual CD-ROM device using `virsh attach-disk`. The VMs were restarted after ISO attachment so that the cloud-init agent applied the credential configuration on first boot.

6.5 Container Image Preparation

The second part prepared the container image. The target platform had no access to external container registries during deployment. The image was therefore prepared on the controller machine and moved to SEAPATH as an archive.

The procedure was:

1. Start from the Debian 12 Bookworm base image.
2. Install diagnostic tools such as ping, `iperf3`, DNS tools, `iproute2`, `curl`, `wget`, `htop`, and `tcpdump`.
3. Save the prepared container as a reusable image.
4. Save the image as a tar archive.
5. Transfer the archive to the SEAPATH platform through Ansible.
6. Load the archive into the local Podman image store on SEAPATH.

This image preparation step matters for vPAC deployment because software-hosted protection systems cannot depend on informal manual setup. Images, versions, transfer paths, and local loading must be repeatable and traceable.

6.6 Container Deployment Procedure

The final container deployment was automated with Ansible. Table 6.2 shows the reproducible procedure using labels that do not reveal internal system details.

Table 6.2: Reproducible procedure for the SEAPATH container deployment.

Step	Action	Purpose and interpretation
1	Copy the Debian-based image archive to the SEAPATH target.	This avoids depending on live registry pulls during target deployment.

Step	Action	Purpose and interpretation
2	Run `podman load` to import the image archive into the local Podman image store.	This makes the same prepared image available for all controller instances without requiring registry access.
3	Create a dedicated Podman bridge network.	This gives the controller instances a separate private network instead of sharing the host network identity.
4	Remove old controller instances if present.	This makes reruns repeatable and avoids stale state from earlier tests.
5	Start three controller instances from the prepared image.	This checks whether multiple containers can be created from the same image on the target platform.
6	Assign static private addresses.	Static addresses allow each controller instance to be reached consistently after recreation.
7	Add the required network capabilities.	`NET_RAW` is the Linux capability that allows processes to open raw sockets; ICMP ping requires it by default. `NET_ADMIN` was also added in the documented playbook for network diagnostic operations.
8	Report status and run ICMP checks.	The checks verify that instances are running and can exchange basic IP traffic. They do not test IEC 61850 behavior.

The Ansible playbook that implemented these steps handled the full deployment sequence in a single invocation against the SEAPATH host. Each task corresponds to one step in Table 6.2: the copy task moved the image archive from the controller machine, `podman_load` imported it into the local image store, `podman_network` created the private bridge subnet, a removal task cleared any stale instances from previous runs, and the main deployment task started all three controller instances from the same image with static addresses and the required Linux capabilities. Because the SEAPATH host could not reach external container registries during the experiment, `pull: never` was set explicitly to prevent Podman from attempting a registry pull and failing. Container names use generic labels in Listing 6.3; the subnet `10.89.0.0/24` and instance addresses `10.89.0.11`–`10.89.0.13` are as used in the experiment. The playbook labels `container1`, `container2`, and `container3` correspond to controller instance 1, controller instance 2, and controller instance 3 in this thesis.

Listing 6.3: Ansible playbook used to deploy the three Debian-based controller instances on the SEAPATH host. Container names and controller machine path use generic labels.

```
- name: Deploy Debian controller instances on SEAPATH
```

```
hosts: standalone_machine
become: true          # containers live in root's Podman store
vars:
  container_image_src: "<controller_path>/debian-tools-v2.tar"
  container_image_name: "localhost/debian-tools:v2"
  network_name: "controller-network"
  network_subnet: "10.89.0.0/24"
  containers:
    - name: container1
      ip: "10.89.0.11"
    - name: container2
      ip: "10.89.0.12"
    - name: container3
      ip: "10.89.0.13"

tasks:
  - name: Copy container image to SEAPATH
    ansible.builtin.copy:
      src: "{{ container_image_src }}"
      dest: /tmp/debian-tools-v2.tar
      mode: '0644'

  - name: Load container image into Podman
    containers.podman.podman_load:
      input: /tmp/debian-tools-v2.tar

  - name: Create controller network
    containers.podman.podman_network:
      name: "{{ network_name }}"
      subnet: "{{ network_subnet }}"
      state: present

  - name: Stop and remove existing containers
    containers.podman.podman_container:
      name: "{{ item.name }}"
      state: absent
    loop: "{{ containers }}"

  - name: Deploy containers with static IPs
    containers.podman.podman_container:
      name: "{{ item.name }}"
      image: "{{ container_image_name }}"
      state: started
      network: "{{ network_name }}:ip={{ item.ip }}"
      command: sleep infinity
      recreate: true
      pull: never          # image transferred offline; registry not available
```

```
        cap_add:
          - NET_ADMIN        # required for network diagnostic operations
          - NET_RAW          # required for ICMP ping (raw socket)
      loop: "{{ containers }}"

    - name: Collect container info
      containers.podman.podman_container_info:
        name: "{{ item.name }}"
      loop: "{{ containers }}"
      register: container_info

    - name: Print container status and IPs
      debug:
        msg: "{{ item.item.name }} -- IP: {{ item.item.ip }}
              -- Status: {{ item.containers[0].State.Status }}"
      loop: "{{ container_info.results }}"
```

The playbook was invoked from the controller machine with the following command:

Listing 6.4: Command used to run the container deployment playbook from the controller machine.

```
ansible-playbook \
    -i inventories/seapath_inventory.yaml \
    playbooks/deploy_containers_debian.yaml
```

6.7 Container Communication Results

After deployment, three controller instances were running on the dedicated Podman bridge network. ICMP ping tests of 100 packets were run between all three controller instances, covering all six directed paths.

Before interpreting the results, the scope of this test must be clear. The experiment validates IP-layer reachability only. ICMP ping measures basic IP reachability and round trip time at Layer 3. GOOSE and Sampled Values are Layer 2 Ethernet multicast protocols: they use MAC-addressed multicast frames rather than IP, apply strict priority handling, and carry protection signals with sub-4 ms transfer time requirements. These Layer 2 protocols were not tested in this experiment. The test here confirms that the container network supports basic IP communication between separately addressed instances; it does not show that the setup is ready for protection traffic or IEC 61850 process bus behavior.

Inter-instance ICMP was used because it is the simplest check that each instance is separately addressable and reachable on the platform. Traffic from external field equipment

to the instances is the more operationally relevant test, but it depends on process bus access and IEC 61850 traffic that are outside the scope of this deployment check; it is therefore left to future work, consistent with the framing of this experiment as a first connectivity check rather than final validation.

The documented results are shown in Table 6.3.

Table 6.3: Container to container ICMP results from the SEAP-ATH experiment.

Path	**Packets**	**Packet loss**	**Average RTT**	**RTT range**
Controller instance 1 to controller instance 2	100 sent, 100 received	0%	0.031 ms	0.019 to 0.052 ms
Controller instance 1 to controller instance 3	100 sent, 100 received	0%	0.029 ms	0.019 to 0.044 ms
Controller instance 2 to controller instance 1	100 sent, 100 received	0%	0.031 ms	0.021 to 0.050 ms
Controller instance 2 to controller instance 3	100 sent, 100 received	0%	0.030 ms	0.020 to 0.048 ms
Controller instance 3 to controller instance 1	100 sent, 100 received	0%	0.028 ms	0.018 to 0.043 ms
Controller instance 3 to controller instance 2	100 sent, 100 received	0%	0.030 ms	0.019 to 0.046 ms

These values are far below 10 ms, the transfer time threshold commonly referenced for GOOSE protection functions in IEC 61850 compliant substations (Kabbara et al., 2022). The observed average round trip times were about 0.03 ms, which is sub-millisecond local communication inside the tested container network. The result supports basic deployment feasibility: all three instances were separately addressable and communicated without packet loss across 100 samples.

6.8 VM Baseline Results

The VM baseline used the same ICMP approach described for the container results in the preceding section. Three Debian 12 guest VMs were running, and ping tests of 100 packets were performed between the VM instances. These are basic IP reachability measurements, not IEC 61850 process bus tests.

The documented results are shown in Table 6.4.

Table 6.4: VM to VM ICMP results from the SEAPATH experiment.

Path	Packets	Packet loss	Average RTT	RTT range
VM instance 1 to VM instance 2	100 sent, 100 received	0%	0.194 ms	0.130 to 0.259 ms
VM instance 1 to VM instance 3	100 sent, 100 received	0%	0.193 ms	0.114 to 0.272 ms
VM instance 2 to VM instance 1	100 sent, 100 received	0%	0.212 ms	0.121 to 0.398 ms
VM instance 2 to VM instance 3	100 sent, 100 received	0%	0.224 ms	0.108 to 0.453 ms
VM instance 3 to VM instance 1	100 sent, 100 received	0%	0.260 ms	0.104 to 0.555 ms
VM instance 3 to VM instance 2	100 sent, 100 received	0%	0.229 ms	0.105 to 0.468 ms

The VM baseline showed sub-millisecond ICMP round trip times and 0 percent packet loss across all six directed paths.

The container samples averaged about 0.03 ms while the VM samples ranged from about 0.19 ms to 0.26 ms on average, consistent with the expectation that containers carry lower overhead by sharing the host kernel while VMs include a guest operating system boundary (Abeni, 2024).

The comparison is still limited. The VM and container tests were diagnostic ping checks, not identical protection workloads. They did not use IEC 61850 traffic, controlled background load, synchronized clocks, repeated long duration samples, or statistical timing analysis. The numbers therefore support a local deployment observation, not a general claim that containers are always faster or more suitable than VMs for protection.

6.9 Interpretation of the Measurements

The measured values answer a narrow question: could the created instances communicate at a basic level after deployment. For both VMs and containers the answer was yes: all documented ping samples showed 0 percent packet loss and all average RTT values were below 1 ms.

The values do not answer the protection question by themselves. IEC 61850 SV traffic has high packet rates and strict process bus requirements. GOOSE traffic depends on event behavior, multicast delivery, priority handling, and substation network configuration. Nguyen et al. (2025) show that communication bottlenecks can appear before compute resources are exhausted, and Tellado et al. (2025) show that container network

mode affects process bus access. For that reason, ping results should be treated as a first connectivity check, not as final vPAC timing validation.

6.10 Compute Isolation

The experiment also shows the tradeoff between container lightness and VM isolation. Containers share the SEAPATH host kernel, while VMs run with separate guest operating systems. The recorded RSS memory footprint in the experiment documentation was approximately 50 MB per container instance, compared to approximately 627 to 638 MB per VM, approximately 12.5 times more RAM per VM instance, consistent in order of magnitude with the 29-fold difference documented by Abeni (2024) in a controlled comparison. Container RSS figures were obtained using `podman stats -no-stream`, which reports the resident process memory held by each container on the shared host kernel. VM RSS figures were obtained from `virsh dommemstat`, which reports the resident set size of the QEMU-KVM process representing each guest; these values reflect the hypervisor-level RAM committed to each guest domain. Container instances became available in under one second after the Ansible playbook started them, while the VM guests required roughly 30 to 60 seconds to boot completely. Container deployment is therefore lighter and faster, but container-based protection workloads require explicit host-level controls before production use (Queiroz et al., 2023).

For Virtual Machines, the guest boundary gives a clearer starting point for workload separation, but scheduler configuration and I/O handling still need validation (Tran et al., 2016). For Containers, the platform would need explicit CPU allocation, cgroups, resource limits, scheduling policy, host tuning, and careful privilege control before protection-grade timing claims could be made (Queiroz et al., 2023).

6.11 Networking

Networking is the most direct lesson from the experiment. As noted in Table 6.1, host networking was not used because it gave all containers the same network address as the host, making the controller instances indistinguishable and unsuitable for a design that expects separately addressable workloads.

The dedicated Podman bridge network solved the basic addressing problem by giving each controller instance its own static private address on the `10.89.0.0/24` subnet. Controller instances received addresses `10.89.0.11`, `10.89.0.12`, and `10.89.0.13`. Each container instance received a static IP address via the Podman bridge. The virtual Ethernet interface of each container also has a MAC address, but the experiment did not use macvlan networking, so the containers did not have their own MAC address on the

physical Ethernet segment. This is relevant to IEC 61850 GOOSE and Sampled Values, which depend on Layer 2 MAC addressing and therefore require macvlan or equivalent process bus access that was not configured here. The network model is therefore treated as part of the deployment design rather than as a minor setup detail. The result supports deployment feasibility, but it remains below the communication level required for protection. A protection deployment would still need to test IEC 61850 SV, GOOSE, multicast behavior, VLAN separation, traffic priority, redundant paths, and behavior under load.

The VM instances connected to the SEAPATH host br0 bridge and received addresses from the same network segment as the host itself; actual VM addresses are omitted as they are site-specific. The container instances were on a separate private Podman bridge subnet. The two workload types therefore operated on distinct network paths despite running on the same physical host: VMs through the br0 bridge and containers through the dedicated Podman bridge. This separation means that VM-to-VM and container-to-container traffic took different paths through the SEAPATH networking stack, which is reflected in the different RTT values reported in Tables 6.4 and 6.3.

6.12 Privilege and Security Boundary

The experiment required additional container capabilities for network diagnostics. `NET_RAW` is the Linux capability that allows processes to open raw sockets; ICMP ping requires it by default. `NET_ADMIN` was also added in the documented playbook for network diagnostic operations. This is a useful engineering detail because containers drop many capabilities by default for security.

For vPAC, added capabilities must be justified carefully. A capability may be needed for diagnostic tools, raw socket behavior, packet capture, or network configuration, but it can also weaken containment. This connects the experiment to the cybersecurity concerns discussed by Kabbara et al. (2025): a container-based protection platform must control image provenance, privileges, management access, patching, logging, and configuration traceability.

6.13 Time Synchronization

Time synchronization is a critical platform requirement for protection use. Protection functions that depend on sampled values require clock accuracy within a few microseconds; Rexwinkel et al. (2025) show that virtual relay setups require redundant time sources and PTP-based synchronization. A vPAC platform must therefore define how PTP or NTP is distributed across the virtualization boundary, how clock disturbance is detected, and how recovery is handled. These requirements apply equally to container-based and VM-

based deployments and should be addressed in platform-level validation before production claims are made.

6.14 Redundancy and Reliability

Successful container deployment does not prove dependability. Production protection platforms require failure detection, redundant communication paths, controlled restart behavior, state handling, and validated recovery time (IEEE Power System Relaying Committee, 2015). Containers can be recreated quickly, but restart speed is not equivalent to protection-grade failover; the platform must define what happens when a container, host, network path, or management service fails.

On the VM side, reliability rests on different mechanisms. VM-based workloads can be placed on redundant platforms and restarted on a healthy host after failure, and the guest operating system boundary keeps a failing workload from corrupting kernel state shared with other instances, which supports cleaner fault containment (Tran et al., 2016). However, VM failover still requires validation of state handling and recovery time, and the same failover requirements established for centralized protection apply to VM hosts as much as to container hosts (IEEE Power System Relaying Committee, 2015).

6.15 Scalability and Lifecycle

Three container instances deployed from one prepared image through a single Ansible playbook confirm that container-based packaging simplifies repeatable deployment. The same build, transfer, load, and start procedure applies regardless of instance count: the same Ansible playbook that created three instances would create ten or thirty without per-instance manual steps. This is the lifecycle advantage described by Queiroz et al. (2023): image-based deployment enables version control, rollback, and traceability, provided the host platform also enforces scheduling, resource limits, and network configuration.

VM scaling is more constrained. The heavier per-instance footprint of a VM, observed in this experiment as approximately 627 to 638 MB of RAM per instance, limits how many guests a single host can carry compared with containers (Abeni, 2024). A VM also boots a full guest operating system before its workload runs, so instance creation is slower than starting a container from an existing image. The guest boundary that limits density is the same boundary that provides stronger isolation, so the scaling cost is the direct counterpart of the reliability benefit (Tran et al., 2016).

6.16 Result

The experiment supports a clear but limited conclusion. On the tested SEAPATH-based platform, Debian-based Podman containers could be prepared on a controller machine, transferred as an image archive, loaded locally, deployed through Ansible, assigned separate static private network addresses, and used for successful ICMP communication: the documented container ping samples showed 0 percent packet loss and average RTT values between 0.028 ms and 0.031 ms across all six directed paths. The KVM-based VM baseline also succeeded, with 0 percent packet loss and average RTT values between 0.193 ms and 0.260 ms across all six directed paths. Container RTT was approximately 6 to 8 times lower than VM RTT in this local diagnostic setup, consistent with the overhead difference documented by Abeni (2024), but the experiment is not strong enough to rank the virtualization technologies for protection use. Production use in critical protection functions requires additional platform validation: IEC 61850 SV and GOOSE testing, timing under load, synchronization accuracy, redundancy behavior, and cybersecurity controls (Kabbara et al., 2025).

Chapter 7

Discussion

7.1 What the Findings Mean

The literature review, the industrial platform analysis, and the SEAPATH experiment all point to the same conclusion: moving from physical IEDs to vPAC is not only a technology change: it is a full platform change that redistributes the responsibilities a physical relay previously handled alone (MacDonald et al., 2025).

In a traditional substation, a dedicated relay provides computing hardware, I/O interfaces, communication behavior, and operational boundary as one unit; virtualization splits these responsibilities across several layers, making the protection algorithm only one part of a larger system (IEEE Power System Relaying Committee, 2015). The design question therefore shifts from algorithmic correctness to platform completeness: the system must preserve timing, isolation, communication, redundancy, and recovery, and not only execute the algorithm. The following subsections answer each research question in turn, drawing on the evidence above.

7.1.1 How are Containers and Virtual Machines used to deploy virtualized protection functions in the literature?

The reviewed studies show that both virtualization technologies can support parts of this transition, but neither covers the full platform requirements on its own. As shown by Kabbara et al. (2022), a protection function can be packaged as a software service that is started, copied, updated, and tested more easily than a fixed relay device; Docker-based virtual IEDs, microservice-structured protection functions, and logical node containers therefore appear consistently across the reviewed studies. VM-based virtualization appears where stronger separation is needed: it provides workload boundaries and guest operating system isolation, which is relevant when protection functions carry higher criticality than monitoring or engineering services (Tran et al., 2016).

7.1.2 What deployment characteristics distinguish these virtualization technologies in terms of isolation, performance, scalability, and manageability?

The container studies reveal the primary deployment constraint: the bottleneck is not protection calculation but communication, as Nguyen et al. (2025) show through Sampled Value ingress and physical Ethernet channel limits at scale. vPAC is sometimes framed as a compute consolidation problem, but Nguyen et al. (2025) show that network ingress limits consolidation before compute capacity is exhausted: the NICs, switches, VLAN design, multicast behavior, time synchronization, and traffic prioritization become part of the protection platform. Torres et al. (2024) add that GOOSE trip delay increases as more virtual IEDs share a network, so communication degradation at scale is not only about Ethernet channel saturation but also about growing delay between fault detection and breaker operation.

Tellado et al. (2025) further show that Docker bridge networking, macvlan, CPU allocation, and host kernel configuration each lead to different system behavior; the technology label alone does not indicate protection suitability (Queiroz et al., 2023).

VM-based virtualization contributes a different characteristic. Its value is not density but separation: it provides stronger workload boundaries and guest operating system isolation (Tran et al., 2016). However, a VM does not automatically provide deterministic behavior; CPU scheduling and hypervisor configuration still require explicit validation (Tran et al., 2016), and I/O virtualization also needs explicit design (Casini et al., 2021). Kabbara et al. (2024) illustrate a subtler consequence of this: when VM boundaries and container boundaries are stacked on the message path, each boundary adds latency, so the number and position of virtualization layers in the communication path becomes a design variable that can determine whether IEC 61850 timing requirements are met at all.

7.1.3 How do industrial approaches such as SIPROTEC V, SEAPATH, and ABB centralized protection and control operationalize virtualization concepts for digital substation environments?

Public platform descriptions show that industrial vPAC is operationalized as a platform rather than as a technology choice: they emphasize engineering tools, lifecycle management, redundancy, security, and communication integration around the virtualization technology rather than the technology alone (MacDonald et al., 2025). This supports the interpretation that the evidence does not favour selecting one virtualization technology

over the other universally; rather, it favours workload-driven placement, where functions near the protection trip path use stronger isolation and supporting services use container packaging (Abeni, 2024).

7.1.4 What engineering factors are most important for deploying virtualized protection functions in real-time environments?

The SEAPATH experiment demonstrates the same conclusion at a smaller scale: Podman containers and KVM-based VMs were deployed on the same SEAPATH platform, with average container ICMP RTT between 0.028 ms and 0.031 ms and VM RTT between 0.193 ms and 0.260 ms across all six directed paths, both at 0 percent packet loss. This distinguishes feasibility, meaning both virtualization technologies can be deployed on the platform, from readiness, meaning the platform can be trusted in a protection environment under realistic operating conditions. Virtualization technology choice is necessary but not sufficient for vPAC readiness (Abeni, 2024); a protection platform must be designed from measurement source to protection output, accounting for network path, time synchronization, redundancy, engineering workflow, and validation scope.

7.2 Deployment Implications

vPAC deployment should start from workload criticality rather than technology preference: functions near the protection trip path require the strongest isolation and resource guarantees, while monitoring, engineering, and visualization services can accept more flexible deployment (Abeni et al., 2022). Highly critical functions need predictable access to CPU time, memory, and network interfaces with bounded I/O behavior (Abeni, 2024), which makes VM-based virtualization attractive because it provides clearer workload separation and stronger control boundaries (Tran et al., 2016). Containers are attractive where packaging, replication, and lifecycle flexibility matter most (Abeni, 2024); they fit microservice-structured platforms where acquisition, protection logic, logging, visualization, and gateway services are separated, making each component easier to update and test independently.

However, container flexibility can hide shared dependencies (Struhár et al., 2020). Containers share the host kernel and may also share network stacks, timing services, and resource control mechanisms; without explicit CPU pinning, scheduling policy, network mode, interface assignment, privileges, logging, and recovery rules, containerization becomes packaging rather than protection architecture (Queiroz et al., 2023).

IEC 61850 communication must be treated as part of the platform architecture, not as in-

frastructure that can be configured after the virtualization technology is chosen. In vPAC, SV streams carry the measurements used by protection functions, GOOSE messages may carry trip and interlocking signals, and time synchronization governs measurement interpretation (Kabbara et al., 2022). Default container networking is therefore unsuitable for process bus traffic: a process bus path requires Layer 2 access, multicast handling, VLAN separation, traffic prioritization, and redundancy, and Tellado et al. (2025) show that Docker networking mode is a deployment design decision, not an implementation detail.

Scalability must also be defined carefully for vPAC. In vPAC, scalability is not only the number of protection instances that fit on a server; it also encompasses the number of SV streams that can be received, the GOOSE traffic that can be handled, and the configurations that can be managed (Nguyen et al., 2025). Nguyen et al. (2025) show that network ingress becomes the limiting factor before compute capacity is exhausted, so scalability claims based only on CPU measurements are incomplete.

Validation must cover the whole protection path rather than individual components in isolation. The reviewed studies measure different things: trip delay, container count, RTDS hardware-in-the-loop performance, or vendor documentation, which makes direct comparison difficult. For industrial deployment, validation must include measurement input, protection processing, communication output, synchronization, redundancy behavior, and failure handling, including stress cases such as network congestion, host overload, clock disturbance, and failover.

Industrial platforms need an engineering workflow, not only a deployment environment (MacDonald et al., 2025). Protection engineers must configure, test, understand, and maintain the system; Siemens (2026) show this through SIPROTEC V's engineering and lifecycle tooling, confirming that utility-grade vPAC is a platform problem rather than a technology selection.

A virtualized substation platform may host protection, monitoring, data collection, and engineering services on shared hardware; these workloads must not receive the same treatment (Abeni et al., 2022). Virtualization makes protection software-defined, which transfers update, rollback, version control, vulnerability handling, and configuration traceability from IT procedures to protection engineering responsibilities (MacDonald et al., 2025). The choice of virtualization technology must therefore follow workload criticality, timing requirements, communication path, isolation needs, and operational workflow, not convenience or implementation familiarity (Abeni et al., 2022).

7.3 Design Recommendations

Seven design recommendations are derived from the review. They are tied directly to the industrial evidence: SIPROTEC V supports the need for engineering tools, diagnostics, security, and lifecycle workflow (Siemens, 2026); SEAPATH supports the need for a controlled platform layer for workload placement, monitoring, updates, and recovery (LF Energy, 2026); ABB centralized protection and control shows that vPAC is an architecture transition from physical relay separation to logical platform-based separation (ABB, 2019). These recommendations are not a replacement for utility validation or vendor qualification; they identify what a project team should verify before treating a virtualized protection architecture as credible.

1. **Criticality-first placement:** Functions near the trip path should receive the strongest isolation and clearest resource guarantees (Abeni et al., 2022); supporting functions such as monitoring, logging, and engineering services can accept more flexible deployment.

2. **Communication as architecture:** SV, GOOSE, MMS, multicast behavior, time synchronization, VLANs, and redundant paths must be designed together with the virtualization technology (Kabbara et al., 2022). A vPAC design that addresses only CPU resources is incomplete.

3. **Container use:** Use Containers where packaging, replication, and lifecycle flexibility matter most (Abeni, 2024), provided the host platform also specifies CPU allocation, network mode, privileges, logging, update behavior, and recovery behavior (Queiroz et al., 2023).

4. **Virtual Machine use:** Use Virtual Machines where stronger separation, clearer guest boundaries, or conservative platform behavior is required (Tran et al., 2016). Scheduler and I/O configuration still need explicit validation even with a hypervisor present.

5. **Full-chain validation:** Validate measurement input, protection processing, communication output, timing behavior, redundancy, and recovery from faults (Kabbara et al., 2024). Basic container or VM communication is only the first validation level.

6. **Engineering workflow:** Configuration, version control, rollback, monitoring, alarms, cybersecurity, and documentation must be part of the architecture review (Siemens, 2026), because operators must maintain the system after deployment.

7. **Operational procedures:** Updates, rollback, version control, vulnerability handling, and configuration traceability are protection engineering responsibilities, not IT tasks (MacDonald et al., 2025). Mixed-criticality platforms need explicit rules about which functions can be updated independently and which require coordinated validation.

Table 7.1 converts these recommendations into an evaluation checklist. It is not a certification method; it identifies the design questions a credible vPAC architecture must answer before deeper validation begins.

Table 7.1: Evaluation checklist for a future vPAC architecture.

Evaluation area	**Question the design should answer**
Workload criticality	Which functions are close to the trip path, and which functions are support services?
Technology choice	Why is each function placed in a Container, a Virtual Machine, or a combination of both?
Resource control	How are CPU, memory, I/O, and scheduling behavior controlled for critical workloads?
IEC 61850 communication	How are SV, GOOSE, MMS, multicast behavior, VLANs, and interfaces handled?
Time synchronization	How is time distributed, monitored, and protected from disturbance?
Redundancy and recovery	What happens when a host, network path, container, VM, or service fails?
Security and lifecycle	How are access, images, updates, rollback, logs, and configuration history managed?
Validation scope	Which tests prove basic feasibility, and which tests prove protection-grade readiness?

A credible vPAC design must specify why each workload is placed where it is, how the communication path is protected, and how the platform will be tested and maintained; technology selection alone does not answer these questions (Abeni, 2024).

7.4 Limitations

The first limitation is the size and maturity of the literature base. vPAC is still an emerging field. Many studies are prototypes, laboratory demonstrations, or platform descriptions. This limits the strength of the conclusions. The thesis can identify patterns and risks, but it cannot claim that one architecture has been proven superior in field deployment.

The second limitation is that public industrial information is incomplete. Vendor platforms often disclose system goals, architecture principles, engineering tools, and communication support. They rarely disclose the full low-level runtime design. That means the thesis can analyze industrial direction, but not every implementation detail.

The third limitation is the search scope. The academic search was limited to IEEE Xplore for traceability. This made the method easier to explain, but it also means that some

relevant papers from other databases may not be included in the method chapter's search process.

The fourth limitation is that the architecture table focuses on direct vPAC studies. This was a deliberate choice. It made the table more actionable, but it also excluded some enabling research. For example, papers on general real-time scheduling, deterministic networking, or I/O virtualization can be highly relevant even when they do not implement vPAC directly.

The fifth limitation is the experimental scope. The experiment validates basic container creation and communication. It does not validate IEC 61850 SV ingestion. It does not validate GOOSE transfer time. It does not test clock synchronization, redundancy, failover, or protection algorithm correctness.

The experimental result should therefore not be overinterpreted. It supports the claim that container deployment is technically possible on the evaluated platform, but not the claim that the setup is ready for protection use.

The sixth limitation is that the thesis does not evaluate cybersecurity certification or safety approval. These are important for industrial adoption. A platform may be technically feasible and still be difficult to approve for utility operation. Certification, compliance, and organizational procedures can affect deployment as strongly as technical performance.

The seventh limitation is that the thesis treats Virtual Machines as one virtualization technology. This is useful for the thesis argument because it allows a clean two-way comparison with Containers, but there are important differences between hypervisor types, VM schedulers, guest operating systems, and I/O models that a deeper study could separate.

Table 7.2 summarizes how these limitations affect the conclusions and how later work could address them.

Table 7.2: Summary of limitations and their effect on the thesis conclusions.

Limitation	**Effect on conclusions**	**How later work can address it**
Emerging literature base	The thesis can identify patterns, but it cannot prove that one virtualization technology is superior in field deployment.	Add more field studies, pilot deployments, and repeated tests across different platforms.
Incomplete industrial detail	Public vendor documents show platform direction but not every low-level runtime detail.	Use vendor interviews, access to technical manuals, or controlled industry collaboration.

Limitation	Effect on conclusions	How later work can address it
Search scope	Limiting the academic search to IEEE Xplore improves traceability but may omit relevant work from other databases.	Repeat the search in Scopus, Web of Science, ACM Digital Library, and vendor technical libraries.
Direct evidence focus	The architecture table becomes clearer, but some enabling studies are kept outside the main comparison.	Add a separate enabling research table for scheduling, networking, security, and I/O.
Experiment scope	The experiment supports only basic container deployment and communication, not protection readiness.	Add IEC 61850 SV and GOOSE tests, timing measurements, redundancy tests, and fault cases.
Approval and security scope	Technical feasibility may not be enough for utility acceptance.	Study certification, safety process, cybersecurity approval, and operational procedures.
Technology grouping	Grouping Virtual Machines simplifies the comparison.	Separate hypervisor type, scheduler, guest operating system, and I/O model in a deeper study.

7.5 Discussion Summary

vPAC should be understood as a controlled platform transition rather than a simple replacement of relays with software.

Containers are promising for packaging, scalability, and lifecycle flexibility, but their main challenges are protection-grade isolation, deterministic networking, and reliability (Queiroz et al., 2023). VM-based virtualization remains important because it provides stronger separation and clearer isolation boundaries, at the cost of higher resource overhead and operational complexity (Abeni, 2024); both still require careful scheduling and I/O design (Tran et al., 2016).

The architecture must be full stack: compute isolation, communication design, synchronization, redundancy, cybersecurity, engineering tools, monitoring, and lifecycle control are all required (MacDonald et al., 2025). Research prototypes show feasibility; the industrial examples show what readiness requires. The experiment confirms that basic container and VM deployment on SEAPATH is feasible, with 0 percent packet loss and about 0.03 ms average container ICMP RTT and about 0.22 ms average VM RTT, but it also makes clear how much remains outside a basic connectivity test.

Across the reviewed evidence, hardware provisioning sets the capacity envelope but IEC 61850 communication architecture determines whether a vPAC deployment is feasi-

ble in the first place. SV ingress saturates network interfaces before CPUs, GOOSE delay grows with scale, and VM or container boundaries on the message path can push latency above acceptable transfer times. This means that a vPAC system designed without a communication architecture is not merely incomplete; it is unverifiable as a protection platform (Kabbara et al., 2022).

vPAC architectures will likely combine both virtualization technologies, with critical protection functions using stronger isolation and supporting services using container flexibility (Abeni, 2024). The correct design depends on workload criticality, timing requirements, communication paths, and operational expectations.

Chapter 8

Conclusion

In this thesis we analyzed two virtualization technologies, containers and virtual machines, as approaches to hosting protection, automation, and control software in digital substations. We conducted a literature review of direct vPAC studies, analyzed three industrial platform directions, and performed a deployment experiment on a SEAPATH-based platform. We found that both virtualization technologies can be deployed on the platform, but that virtualization technology selection alone is insufficient: a production-ready vPAC platform must address compute isolation, communication architecture, time synchronization, redundancy, lifecycle management, and cybersecurity. The following sections answer the research questions explicitly.

8.1 Research Questions Answered

For Research Question 1, the literature shows that vPAC functions are deployed using two main virtualization technologies. Containers package protection functions as lightweight services that share the host kernel. Virtual Machines place protection applications behind stronger virtualization boundaries.

The container studies show a clear direction toward software packaging, service decomposition, and higher instance density. Docker is the most common runtime in the direct studies implementing IEC 61850 communication (Nguyen et al., 2025). Some studies treat each virtual IED or logical node as a container. Others use microservice-structured architectures for acquisition, protection logic, data exchange, and output handling (Kabbara et al., 2022).

VM-based virtualization is used when stronger separation or a more complete software environment is needed (Tran et al., 2016). It fits cases where a virtual protection relay behaves more like a complete relay system, and cases where guest operating system separation, I/O control, or platform isolation matters more than fast scaling (Abeni, 2024).

For Research Question 2, the main differences concern isolation, resource overhead, scalability, communication behavior, and manageability. Virtual Machines provide stronger

boundaries but carry significantly higher memory overhead per instance: Abeni (2024) measure that a KVM/QEMU Linux VM reserves over 20 times more host memory than a comparable low-level container runtime instance; containers provide faster deployment and easier replication but depend heavily on host configuration (Queiroz et al., 2023). The distinction that matters most for vPAC is not technical overhead alone but the kind of risk each technology creates: containers carry risk through shared kernel resources and default networking choices (Struhár et al., 2020), while VM-based virtualization carries risk through scheduler configuration, I/O virtualization, and heavier platform management (Tran et al., 2016).

For Research Question 3, industrial approaches show complementary directions. SIPROTEC V represents a packaged commercial virtualized protection platform. ABB centralized protection and control represents an architectural transition toward centralized software based protection. SEAPATH represents platform-oriented infrastructure for hosting virtualized substation applications.

These approaches show that industry is not only choosing a virtualization technology: industrial vPAC requires hardware qualification, engineering tools, IEC 61850 integration, redundancy, cybersecurity, monitoring, update control, and lifecycle management. Research prototypes demonstrate one function in a controlled setup; deployable platforms must sustain all of these requirements in parallel (MacDonald et al., 2025).

For Research Question 4, the most important engineering factors are compute isolation, communication design, time synchronization, redundancy, cybersecurity, lifecycle management, and validation under realistic IEC 61850 traffic.

Among these factors, communication design is the most frequently cited constraint: Nguyen et al. (2025) show that SV ingress and physical Ethernet channels limit consolidation before CPU capacity is exhausted, and Tellado et al. (2025) show that container network mode determines process bus behavior. A platform cannot therefore be evaluated only by CPU capacity or container count.

8.2 Key Contributions

- **Virtualization technology comparison:** A structured comparison of Containers and Virtual Machines as virtualization technologies for vPAC systems, identifying what each technology provides and what platform requirements remain regardless of technology choice.
- **Architecture review table:** A structured table covering retained vPAC evidence sources, classified by virtualization technology, software stack, validation level, and deployment implication.

- **Industrial platform analysis:** An analysis of SIPROTEC V, SEAPATH, and ABB centralized protection and control as industrial approaches to substation virtualization, identifying the platform requirements that research prototypes omit.
- **Engineering constraint synthesis:** A synthesis of the compute isolation, communication, synchronization, redundancy, and lifecycle requirements that determine whether virtualized protection can be deployed reliably (MacDonald et al., 2025).
- **Deployment experiment:** A container and VM deployment comparison on a SEAPATH-based platform, showing 0 percent packet loss and average ICMP RTT between 0.028 ms and 0.031 ms for containers and 0.193 ms to 0.260 ms for VMs across all six directed paths per technology.

The primary contribution is the deployment-oriented reading of the literature: we do not only ask whether a paper uses Docker, VMs, or a hypervisor, but what was built, what software was used, what was tested, and what deployment constraint remains. The architecture table is therefore useful as a design tool: it separates basic feasibility from platform readiness and shows where each study contributes evidence.

8.3 Recommendations

Future vPAC deployments should select virtualization options based on workload criticality rather than only scalability (Abeni et al., 2022). Highly critical protection functions should use strong isolation, such as VM-based virtualization, unless a container platform can demonstrate equivalent timing and failure isolation behavior (Tran et al., 2016). Containers are useful for modular services, support functions, test environments, and possibly protection services when the host platform is engineered for real-time behavior (Queiroz et al., 2023).

IEC 61850 communication should be treated as a core platform requirement, not as infrastructure that can be added after the technology choice (Kabbara et al., 2022). A vPAC design must include SV traffic handling, GOOSE transfer behavior, MMS supervision, multicast behavior, VLAN/interface design, and redundant paths, as the evidence in this thesis consistently shows that networking constraints appear before compute limits are reached (Nguyen et al., 2025; Tellado et al., 2025).

Time synchronization and redundancy should be designed early (Rexwinkel et al., 2025). Protection functions that depend on sampled measurements need reliable timing, and a central server replacing many physical relays concentrates dependency: server redundancy, network redundancy, failover behavior, and recovery procedures must all be specified before deployment (IEEE Power System Relaying Committee, 2015).

Lifecycle management should be treated as protection engineering, not as IT operations

(MacDonald et al., 2025). Version control, update procedures, rollback, diagnostics, logging, and configuration traceability are required because a software-hosted protection function that cannot be safely updated or traced is not deployable regardless of its runtime performance (Siemens, 2026).

A successful vPAC platform must therefore integrate virtualization isolation, network design, engineering tools, monitoring, update control, and recovery behavior (MacDonald et al., 2025).

8.4 Future Work

Future work should validate containerized protection using realistic IEC 61850 traffic, including SV and GOOSE. The next experimental step should test the full protection path from measurement input to protection output.

Future studies should also investigate deterministic container networking, including macvlan, host networking, TSN-aware designs, kernel bypass approaches, and CNI behavior where orchestration is used.

Time synchronization, redundancy, and failover need deeper testing. A platform must show how it behaves under PTP/NTP disturbance, clock loss, server failure, traffic redirection, and recovery.

Finally, future work should compare mixed deployments. A realistic platform may use Virtual Machines for critical functions and Containers for supporting services. Certification, safety approval, and utility specific constraints also need study because they may decide whether a technically feasible platform can actually be deployed.

8.5 Concluding Remarks

Virtualized protection systems are feasible, but feasibility is not enough. The evidence shows that vPAC becomes credible only when technology choice, IEC 61850 communication, timing behavior, redundancy, engineering tools, and lifecycle control are designed together.

Containers are attractive because of flexibility and scalability, but they require platform-level adaptation before they can support protection-grade workloads. Virtual Machines remain important because they provide stronger separation and a more conservative execution boundary. The realistic direction is an engineered stack that uses both virtualization technologies according to workload criticality and operational role.

Hardware provisioning determines capacity, but IEC 61850 communication architecture

determines feasibility: SV ingress can saturate network interfaces before CPUs are loaded, GOOSE delay grows as virtual IED count increases, and virtualization boundaries on the message path can push latency above what protection functions tolerate (Nguyen et al., 2025; Torres et al., 2024; Kabbara et al., 2024). A vPAC platform that does not address SV handling, GOOSE network mode, and process bus access at the design stage cannot be evaluated as a protection system.

The conclusion is direct: vPAC should be evaluated by asking whether the complete platform can preserve protection behavior under realistic operating conditions.

References

ABB (2019). *Centralized Protection and Control*. ABB. White paper 2NGA000256 REV A EN 12.2019.

Abeni, L. (2024). Virtualized real-time workloads in containers and virtual machines. *Journal of Systems Architecture*, 154:103238.

Abeni, L., Biondi, A., and Bini, E. (2022). Partitioning real-time workloads on multi-core virtual machines. *Journal of Systems Architecture*, 131:102733.

Blackburn, J. L. and Domin, T. J. (2014). *Protective Relaying: Principles and Applications*. CRC Press, 4 edition.

Casini, D., Biondi, A., Cicero, G., and Buttazzo, G. (2021). Latency analysis of i/o virtualization techniques in hypervisor-based real-time systems. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*.

IEEE Power System Relaying Committee (2015). Centralized substation protection and control. Technical report, IEEE Power & Energy Society. Working Group K15 report.

Kabbara, N., Belaid, M. O. N., Gibescu, M., Camargo, L. R., Cantenot, J., Coste, T., Audebert, V., and Morais, H. (2022). Towards software-defined protection, automation, and control in power systems: Concepts, state of the art, and future challenges. *Energies*, 15(24):9362.

Kabbara, N., Cibin, N., Morais, H., Stefanov, A., and Gibescu, M. (2025). A cybersecurity assessment for hybrid virtualized-physical digital substations. *Sustainable Energy, Grids and Networks*, 43:101795.

Kabbara, N., Mwangi, A., Stefanov, A., and Gibescu, M. (2024). A real-time implementation and testing of virtualized controllers for software-defined iec 61850 digital substations. *IEEE Open Journal of Industry Applications*, 5:300–310.

LF Energy (2026). *SEAPATH: Open Digital Substation Platform*. LF Energy. Project documentation, accessed 2026-05-06.

MacDonald, D. J., Kanabar, M., and Gudimetla, R. (2025). Software and hardware considerations for virtualised protection and control. In *IET Conference Proceedings*.

Nguyen, V. H., Abil, H., Herrera, D. R., Thomas, R., and Coste, T. (2025). Containerized iec 61850-based protection in smart grid – lab demonstration and performance assessment for large scale deployment. *Electric Power Systems Research*, 249:111999.

Queiroz, R., Cruz, T., Mendes, J., Sousa, P., and Simões, P. (2023). Container-based virtualization for real-time industrial systems: A systematic review. *ACM Computing Surveys*, 56(3):Article 59.

Rexwinkel, N., Khalilpour, F., Silva, J., Stinskiy, A., Talwar, S., Stollfuss, M., and Oliveira, E. (2025). Virtualization of protection systems in distribution substations: Design, implementation and performance testing. In *Proceedings of the IEEE Conference on Power and Renewable Energy*.

Rösch, D., Schäfer, K., and Nicolai, S. (2022). Delay modeling for virtualization-based co-simulation of iec 61850 substations. *Journal of Cleaner Production*, 368:133179.

Siemens (2026). *SIPROTEC V: Virtualized Protection and Control for Substations*. Siemens. Product brochure, accessed 2026-05-06.

Struhár, V., Behnam, M., Ashjaei, M., and Papadopoulos, A. V. (2020). Real-time containers: A survey. In *Proceedings of the Workshop on Fog Computing and the Internet of Things*.

Tellado, B., Yurrebaso, X., López, J. A., Sanchez, E., and Vicario, B. (2025). Virtualization topologies of ieds with the iec 61850 protocol and their application in edge devices. In *Proceedings of DPSP Europe*.

Torres, E., Escobar, N., Eguia, P., Abarrategi, O., Larruskain, D. M., Valverde, V., and Buigues, G. (2024). A practical implementation of virtualized protection system with iec 61850 under docker. In *International Conference on Renewable Energies and Power Quality (ICREPQ)*.

Tran, G. P. C., Chen, Y.-A., Kang, D.-I., Walters, J. P., and Crago, S. P. (2016). Hypervisor performance analysis for real-time workloads. In *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*.

Villén Martínez, M. T. and Prada Hurtado, A. A. (2024). Description of an edge computing solution to be used in digital substations. In *Proceedings of the IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. EPICS edge protection and control platform (ELEWIT).

TRITA-EECS-EX-2026:424
Stockholm, Sweden 2026

www.kth.se