



UNIVERSITÀ DEGLI STUDI DI FIRENZE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Tesi di Laurea Magistrale in Ingegneria Informatica

Anno Accademico 2024/2025

**SDN based Topology Management and Control for
6G Industrial Networks via NETCONF
Southbound Protocol**

**Sistema di Controllo e Gestione della Topologia
basato su SDN per Reti Industriali 6G tramite
NETCONF come protocollo Southbound**

Relatori

Francesco Chiti

Laura Pierucci

Candidato

Matteo Lotti

Contents

Acronyms	v
1 Introduction: From 5G to 6G in Industrial Networks	1
1.1 6G and the Industrial Internet of Things	3
1.2 SDN, NFV, and Reconfigurable Intelligent Surfaces (RISs) in 6G-IIoT environment	6
2 Reconfigurable Intelligent Surface (RIS)	9
2.1 Hardware Structure and Operating Principles	9
2.2 Prominent Applications in 6G Networks	10
2.3 Advantages and Key Obstacles	12
3 Software-Defined Networking (SDN): integration within 5/6G cellular networks	14
3.1 The Three Planes of SDN	15
3.2 Southbound Protocols	17
3.2.1 OpenFlow	17
3.2.2 NETCONF and YANG	19
3.3 Integration with NFV and Network Slicing	24
4 Reference Architecture and NETCONF-based Model	26
4.1 Reference Architecture	27
4.2 OpenFlow Limitations	32
4.3 NETCONF-based model	35
4.3.1 RIS configuration Workflow	36
4.3.2 YANG model	39

4.3.3	Notification pattern	44
4.4	Physical Control Frameworks for RIS Management	47
4.4.1	Wired RIS Control Framework	48
4.4.2	Wireless RIS Control Framework and Network Slicing	50
4.5	RIS Discovery Iterative Algorithm	53
4.5.1	Former Discovery Approach	53
4.5.2	Problem Formulation	54
4.5.3	Proposed NETCONF-Based Iterative Discovery Al- gorithm	55
4.5.4	Fallback Mechanism: Broadcast Get-Config Session	58
5	Experimental Platforms	60
5.1	Docker Emulation	61
5.1.1	System evaluation	62
5.2	Integration in RestartIn Discrete Event Simulator	63
5.2.1	Classes Diagram and Description	65
5.2.2	Simulation file description	67
5.3	Iterative Discovery Protocol Implementation	69
5.3.1	Protocol Architecture	69
5.3.2	Discovery Phases	70
5.3.3	Key Implementation Details	71
6	Experimental Results	73
6.1	Packet-exchange simulation: OF vs NETCONF	73
6.1.1	Simulation Environment Design	73
6.1.2	Simulation Results	76
6.2	RIS iterative discovery protocol	81

6.2.1	Discovery Performances Results	83
7	Conclusions	91
	Bibliography	94

List of Figures

1	6G Applications	3
2	RIS Hardware 3-layer Structure	10
3	RISs Applications	11
4	SDN 3-layer Architecture	15
5	OpenFlow (OF) Switch Model	18
6	NETCONF Model	21
7	Reference 3-layer Architecture	28
8	RIS abstraction into an Openflow Switch	30
9	NETCONF-based Architecture	37
10	IoT Device Detection	45
11	Slicing for Wireless RIS Control	52
12	Message Exchange during RIS discovery via NETCONF	57
13	UML Classes Diagram	68
14	Simulated Industrial Scenario (case involving 4 RIS and 30 UEs)	74
15	Simulation Results: 2 RISs with 3 beams	76
16	Simulation Results: 2 RISs with 5 beams	76
17	Simulation Results: 4 RISs with 3 beams	77
18	Simulation Results: 4 RISs with 5 beams	77

19	Simulation Results: NETCONF Overview	78
20	Simulation Results: OF Overview	78
21	Analysis of 95 th percentile of latency	80
22	Factory Layout for Scenario A (3 rd iteration)	82
23	Factory Layout for Scenario B (10 RISs, 5 beams)	84
24	RIS Iterative Discovery Results: Scenario A	86
25	RIS Iterative Discovery Results: Scenario B, RISs with 3 beams	88
26	RIS Iterative Discovery Results: Scenario B, RISs with 5 beams	89

List of Tables

1	Main parameters for the packet-exchange simulation	75
2	Numeric parameters used for the RIS iterative discovery protocol experiments.	83

Acronyms

3GPP 3rd Generation Partnership Project.

5G 5th Generation.

5GC 5G Core.

6G 6th Generation.

AGV Automated Guided Vehicle.

AI Artificial Intelligence.

AoA Angle of Arrival.

CAPEX CAPital EXpenditure.

CSI Communication Service Interface.

eMBB Enhanced Mobile BroadBand.

FPGA Field Programmable Gate Array.

gNB Next Generation Node B.

HARQ Hybrid Automatic Repeat reQuest.

I4.0 Industry 4.0.

I5.0 Industry 5.0.

IETF Internet Engineering Task Force.

IIoT Industrial Internet of Things.

IoT Internet of Things.

LLDP Link Layer Discovery Protocol.

LOS Line-of-Sight.

MAC Medium Access Control.

MIMO Multiple Input/Multiple Output.

ML Machine Learning.

mMTC massive Machine Type Communication.

NETCONF Network Configuration Protocol.

NF Network Function.

NFV Network Function Virtualization.

NS Network Slicing.

NTN Non-Terrestrial Networks.

OF OpenFlow.

OFDM Orthogonal Frequency Division Multiplexing.

OPEX Operational EXpenditure.

OWC Optical Wireless Communications.

PDCCH Physical Downlink Control Channel.

PHY PHYSical.

PUCCH Physical Uplink Control Channel.

PUSCH Physical Uplink Shared Channel.

PWE Programmable Wireless Environment.

QoS Quality of Service.

RAN Radio Access Network.

RF Radio Frequency.

RIS Reconfigurable Intelligent Surface.

RPC Remote Procedure Call.

RSSI Received Signal Strength Indicator.

SDN Software-Defined Networking.

SNR Signal-to-Noise Ratio.

TSN Time Sensitive Networking.

UE User Equipment.

URLLC Ultra-Reliable Low-Latency Communication.

vLoS Virtual Line of Sight.

VNF Virtual Network Function.

WPT Wireless Power Transfer.

Abstract

The advent of 6G networks brings strict requirements for Industrial Internet of Things (IIoT) environments, emphasizing the need for ultra-reliable, low-latency communication. Reconfigurable Intelligent Surfaces (RIS) have emerged as a pivotal technology to dynamically program the radio propagation environment. However, integrating these semi-passive surfaces into the Software-Defined Networking (SDN) paradigm introduces significant architectural challenges. This thesis exposes a critical hindrance in current RIS management research: the use of OpenFlow. Built for active packet-switching, OpenFlow is fundamentally incompatible with the passive, wave-reflecting nature of RIS hardware.

To address this limitation, this work proposes and validates a NETCONF-based architecture for RIS management. By treating the RIS as a programmable device governed by persistent configurations, NETCONF, paired with structured YANG data models, offers a highly efficient and state-driven alternative. The proposed architecture is evaluated through a dual-platform methodology. First, a Docker-based emulation demonstrates the functional feasibility of the NETCONF stack for manipulating RIS operational parameters via standard Remote Procedure Calls (RPCs). Second, a large-scale performance evaluation is conducted using the discrete-event simulator *restartin-sim*.

Simulation results reveal that while NETCONF introduces a control-plane overhead compared to OpenFlow, it eliminates the consistent processing delays at the RIS level caused by packet parsing. Furthermore, this work introduces and evaluates an Iterative Discovery Protocol for dynamic RIS-assisted environments, proving its linear scalability and robustness across varying network densities and obstructed layouts. Ultimately, this thesis concludes that migrating from a packet-switching OpenFlow approach to a state-based NETCONF control plane is not merely a performance optimization, but an architectural necessity to deploy semi-passive RIS hardware in next future 6G programmable radio environments.

Abstract

L'avvento delle reti 6G introduce requisiti stringenti di latenza e affidabilità per gli scenari Industrial Internet of Things (IIoT). In questo contesto, le Reconfigurable Intelligent Surfaces (RIS) possono affermarsi come tecnologia abilitante fondamentale per il controllo dinamico dell'ambiente di propagazione radio. Tuttavia, l'integrazione di questi dispositivi semi-passivi all'interno del paradigma Software-Defined Networking (SDN) introduce notevoli sfide a livello architetturale. Questa tesi evidenzia una criticità strutturale nella letteratura attuale sul controllo delle RIS: l'impiego del protocollo OpenFlow. Essendo concepito per il packet-switching attivo e per l'ispezione dei dati da parte di Switch ad-hoc, OpenFlow risulta intrinsecamente incompatibile con la natura passiva e riflettente dell'hardware RIS concepito fino ad ora. Per superare tale limitazione, questa tesi propone e valida un'architettura basata su NETCONF per il management delle RIS. Astruendo la RIS come un dispositivo di rete governato da configurazioni persistenti, il protocollo NETCONF, supportato da modelli dati YANG strutturati, sfrutta un paradigma di controllo state-driven particolarmente efficace. La validazione dell'architettura si articola in due fasi: in primo luogo, un ambiente emulato tramite container Docker dimostra la fattibilità funzionale dello stack NETCONF nella manipolazione delle configurazioni; in secondo luogo, è stata condotta una valutazione prestazionale su larga scala avvalendosi del simulatore a eventi discreti *restartin-sim*. I risultati delle simulazioni rivelano che, sebbene NETCONF introduca un overhead a livello di controllo rispetto a OpenFlow, esso azzeri i ritardi di elaborazione sul nodo RIS causati dalle inevitabili operazioni di packet parsing. Inoltre, viene introdotto e valutato un algoritmo di Iterative Discovery Protocol progettato per la scoperta delle RIS, dimostrandone la scalabilità lineare e la robustezza al variare della densità della rete e in topologie con ostacoli fisici. In conclusione, questa tesi dimostra come la transizione da un approccio packet-switching basato su OpenFlow a un control-plane governato da NETCONF non rappresenti una semplice ottimizzazione delle prestazioni, bensì una necessità architetturale per l'effettivo dispiegamento di hardware RIS semi-passivo negli ambienti radio programmabili 6G del prossimo futuro.

1 Introduction: From 5G to 6G in Industrial Networks

The deployment of 5th Generation (5G) mobile communication systems represented a central moment in telecommunications, shifting from a rigid, monolithic network model to a versatile, service-oriented architecture. This transition was supported by the 3rd Generation Partnership Project (3GPP) standardization of three primary usage scenarios: Enhanced Mobile BroadBand (eMBB), Ultra-Reliable Low-Latency Communication (URLLC), and massive Machine Type Communication (mMTC) [1]. Within the industrial sector, these capabilities helped the realization of Industry 4.0 (I4.0), enabling the deployment of sensor networks, automated guided vehicles or drones, and initial implementations of smart manufacturing. However, as the digital transformation of the industrial environment accelerates towards unprecedented complexity, the physical and architectural limitations of the current 5G standard are becoming increasingly evident. Advanced industrial applications impose stringent telecommunication requirements that exceed the theoretical bounds of 5G. These use cases demand deterministic communication with sub-millisecond to microsecond end-to-end latencies, and the capacity to support massive device densities without suffering from severe spectrum congestion and interference [2].

Consequently, the telecommunications industry and academic research communities have initiated great efforts toward defining and developing 6th Generation (6G) networks. The architectural vision for 6G is

not merely an incremental upgrade of 5G, but a fundamental paradigm shift in wireless communication technologies. A primary enabler for this shift is the aggressive expansion of the utilized radio frequency spectrum. To mitigate the severe bandwidth constraints of sub-6 GHz and lower millimeter-wave (mmWave) bands, 6G systems will exploit the Terahertz (THz) frequency bands and Optical Wireless Communications (OWC), including Visible Light Communication (VLC) [3]. The utilization of these ultra-high frequency bands is projected to deliver peak data rates exceeding 1 Terabit per second (Tbps) and accommodate staggering volumetric spectral efficiencies. Furthermore, 6G will mandate a transition from traditional cellular topologies to dynamic, cell-free massive Multiple Input/Multiple Output (MIMO) (m-MIMO) and 3D network architectures, deeply integrating terrestrial infrastructures with Non-Terrestrial Networks (NTN) to provide ultra-dense coverage [1].

In the context of the future industrial ecosystem, often referred to as Industry 5.0 (I5.0), these telecommunication advancements will enable a seamless convergence of the physical and digital domains. The stringent deterministic requirements of industrial automation will be addressed through the evolution of URLLC into enhanced URLLC (eURLLC). This involves profound redesigns at the PHYsical (PHY) and Medium Access Control (MAC) layers, including novel waveform designs, shorter Transmission Time Intervals (TTIs), and the native integration of Time Sensitive Networking (TSN) protocols directly over the wireless interface to guarantee strict bounds on packet delays [2]. Managing the immense complexity and highly dynamic channel conditions of such ultra-dense networks will fundamentally exceed the capabilities of traditional net-

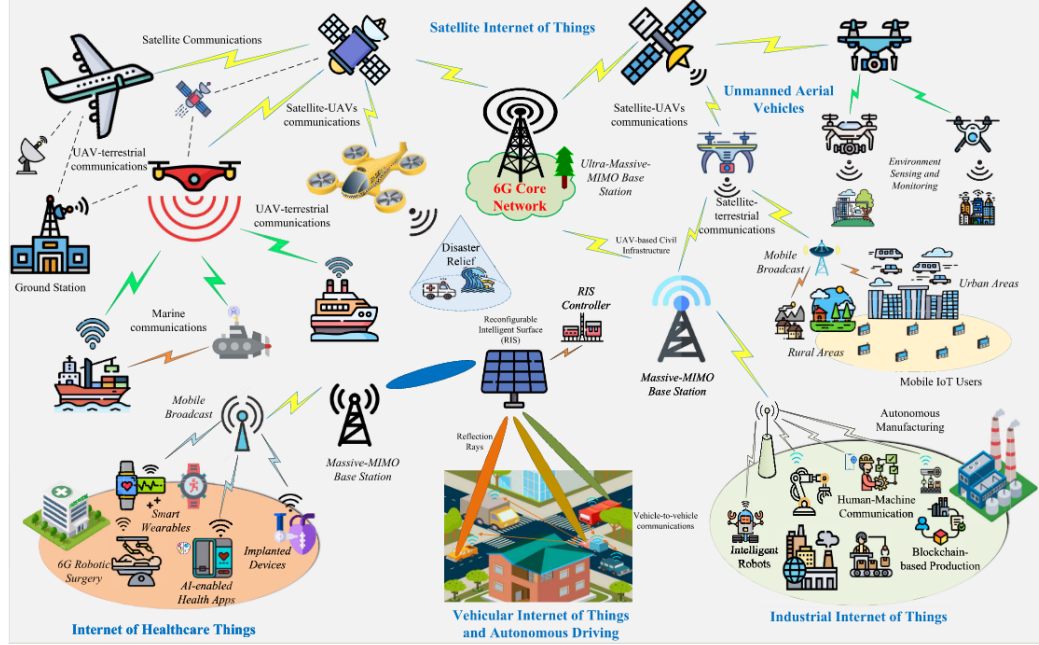


Figure 1: 6G Applications

work management. Consequently, 6G architectures will necessitate the pervasive integration of Artificial Intelligence (AI) and Machine Learning (ML) strictly as underlying operational orchestrators [4], tasked with autonomous spectrum allocation, predictive channel state estimation, and dynamic network slicing. Ultimately, the 6G telecommunications framework will provide the ultra-broadband and pervasive system of sensors required to sustain the next generation of industrial automation.

1.1 6G and the Industrial Internet of Things

The proliferation of the Industrial Internet of Things (IIoT) constitutes the critical backbone of modern automated manufacturing, relying heavily on the pervasive deployment of distributed sensor networks and industrial actuators. While the 5G standard introduced mMTC and URLLC

to support foundational IIoT use cases, the vast scale and demanding requirements of future industrial environments need a fundamental transformation of the underlying telecommunications infrastructure. 6G systems will transition these paradigms toward ultra mMTC (umMTC) and enhanced URLLC (eURLLC). From a networking architecture perspective, 6G may shift away from strictly terrestrial, two-dimensional cellular layouts to a deeply integrated, three-dimensional NTN [5]. This volumetric expansion, guarantees deterministic, highly reliable IIoT connectivity across geographically dispersed industrial facilities and remote logistics pipelines where terrestrial fiber or traditional base stations may be unfeasible [6]. The global telecommunications objective is to establish a seamless, ultra-reliable physical layer capable of supporting various Quality of Service (QoS) requirements with strict bounded latency.

A highly disruptive telecommunications paradigm introduced by 6G that directly revolutionizes the IIoT ecosystem is Integrated Sensing and Communication (ISAC). Historically, wireless communication systems and environmental sensing systems (such as industrial radar or LiDAR) operated on disparate frequency bands, using completely separate hardware infrastructures. However, as 6G explores the Terahertz (THz) and sub-THz frequency bands, characterized by extremely short wavelengths on the order of sub-millimeters, radio signals naturally exhibit high-resolution, radar-like propagation characteristics [7]. ISAC capitalizes on this phenomenon by unifying both functionalities onto a single radio waveform and a shared hardware platform. In a highly dynamic industrial environment, this implies that the very radio waves transmitting telemetric data to a User Equipment (UE) can simultaneously echo off the surrounding

machinery to compute centimeter-level localization, track moving assets, and construct real-time 3D environmental maps. This dual-functional physical layer dramatically reduces the necessity for standalone, energy-consuming sensor arrays, fundamentally transforming the 6G network from a passive data receiver into an active, pervasive sensing nervous system [7].

Furthermore, the exponential proliferation of IIoT nodes in 6G environments presents an insurmountable logistical challenge if operators must rely on conventional battery-powered devices. Consequently, telecommunications research for 6G-IIoT heavily prioritizes physical layer mechanisms that enable extreme energy efficiency and "Zero-Energy" networking [5]. To sustain an ultra-dense array of passive industrial sensors, 6G architectures are focusing on Wireless Power Transfer (WPT) and ambient backscatter communications. By allowing resource-constrained IIoT end-devices to harvest operational power directly from pervasive Radio Frequency (RF) signals, or to transmit data by simply modulating and reflecting ambient waves rather than actively generating their own energy-intensive carrier signals, 6G ensures a sustainable telecommunications infrastructure. These advanced physical layer innovations, coupled with intelligent structural orchestration, form the requisite foundation for the deeply interconnected industrial ecosystems of the future.

1.2 Software-Defined Networking (SDN), Network Function Virtualization (NFV), and Reconfigurable Intelligent Surfaces (RISs) in 6G-IIoT environment

Fulfilling the stringent performances demanded by 6G industrial networks requires radical innovations not only in the physical transmission medium but also within the global network management architecture. The unprecedented complexity, heterogeneity, and scale of 6G-IIoT ecosystems makes traditional network architectures entirely obsolete. This thesis is fundamentally driven by the necessity to explore and define advanced and highly adaptable control frameworks capable of orchestrating these complex environments. To achieve this, the research has its foundation on the convergence of Software-Defined Networking (SDN), Network Function Virtualization (NFV), and Reconfigurable Intelligent Surfaces (RISs).

To achieve the necessary structural agility and scalability, SDN and NFV serve as fundamental pillars for the modern telecommunications architecture. SDN conceptually decouples the network's control plane from its data forwarding plane, centralizing the intelligence into software-based controllers [8]. In a highly dynamic IIoT environment, this separation allows for dynamic and programmable traffic management, which is vital for assigning QoS priority levels to different signals [9]. Concurrently, NFV abstracts traditional hardware-based network appliances into agile software modules. Together, SDN and NFV enable end-to-end Network Slicing (NS), that is the creation of multiple, logically isolated virtual networks over a shared physical infrastructure. This paradigm

allows an industrial operator to guarantee distinct QoS parameters for differing IIoT workloads without deploying separate physical networks.

Parallel to these architectural changes in the upper layers, 6G introduces a profound paradigm shift in the physical layer through RISs. As 6G increasingly relies on high-frequency millimeter-wave (mmWave) and Terahertz (THz) bands to achieve ultra-high volumetric bandwidths, it faces severe propagation challenges. These high-frequency signals suffer from extreme path loss, low penetration, and high susceptibility to blockages, that are conditions that are highly prevalent in industrial factory floors filled with moving and dense metallic machinery. RIS technology addresses this critical bottleneck by strategically deploying arrays of sub-wavelength electromagnetic metamaterials that can actively alter the phase, amplitude, and polarization of incident radio waves [10, 11]. Instead of forcing the network to blindly adapt to an uncontrollable channel, RISs allow network engineers to programmatically adapt the propagation environment itself, effectively turning physical "blind spots" into reliable communication corridors [12].

The core motivation of this thesis lies in bridging the critical integration gap between the physical programmability offered by RISs and the logical programmability enabled by SDN/NFV. While RISs can secure transmission links and optimize signal coverage [13], the reflective elements and the necessity for real-time, context-aware reconfiguration in an IIoT setting require robust, standardized control protocols. By proposing a NETCONF-based control architecture, this work aims at integrating RIS components as manageable network devices within a global SDN industrial topology. This unified software-defined approach ultimately

ensures that both the logical routing paths and the physical electromagnetic environment can be jointly orchestrated to meet the reliable and ultra-low-latency demands of 6G smart manufacturing.

2 Reconfigurable Intelligent Surface (RIS)

Traditionally, the wireless propagation environment has been perceived as a highly probabilistic and uncontrollable medium. Wireless system engineers have historically relied on complex compensation techniques at the transceiver ends, such as massive MIMO, advanced modulation schemes, and error-correcting codes, to mitigate the severe degradations caused by fading, scattering, and path loss. However, the advent of Reconfigurable Intelligent Surfaces (RISs) introduces a profound paradigm shift: moving from adapting to the wireless channel to actively customizing and controlling it [14]. A RIS is a planar surface comprising a massive array of low-cost, passive scattering elements. By intelligently adjusting the electromagnetic properties of these individual elements, a RIS can dynamically alter the phase, amplitude, and polarization of incident Radio Frequency (RF) signals, effectively engineering a smart and programmable radio environment [11].

2.1 Hardware Structure and Operating Principles

The physical realization of a RIS is typically based on the concept of electromagnetic metamaterials. To achieve independent and real-time control over the incident electromagnetic waves, the hardware architecture of a RIS is generally structured into three distinct layers, as shown in figure 2.

The topmost layer consists of a two-dimensional array of metallic sub-wavelength patches printed on a dielectric substrate. These patches act as the scattering elements that directly interact with the incident RF

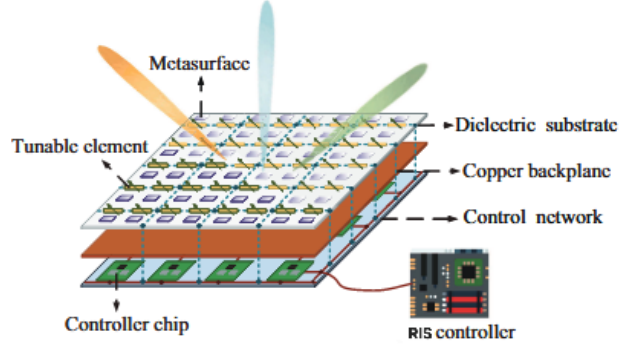


Figure 2: RIS Hardware 3-layer Structure

signals. The intermediate layer is a copper backplane, which serves to prevent signal leakage and electromagnetic interference behind the surface, maximizing the reflection efficiency. Finally, the bottom layer is the control circuit board, typically governed by an Field Programmable Gate Array (FPGA) or a dedicated microcontroller [10].

The real-time reconfigurability is achieved by embedding tunable electronic components, such as PIN diodes, varactors, or micro-electromechanical systems (MEMS), into the scattering elements [14]. For instance, switching the bias voltage of a PIN diode between "ON" and "OFF" states alters the equivalent capacitance and inductance of the element, thereby inducing a discrete phase shift to the reflected signal. The smart controller orchestrates the biasing across the entire array based on commands received from the global network controller, resulting in a desired directional reflection beam.

2.2 Prominent Applications in 6G Networks

The ability to dynamically sculpt the electromagnetic environment unlocks numerous transformative applications for 6G and IIoT ecosystems:

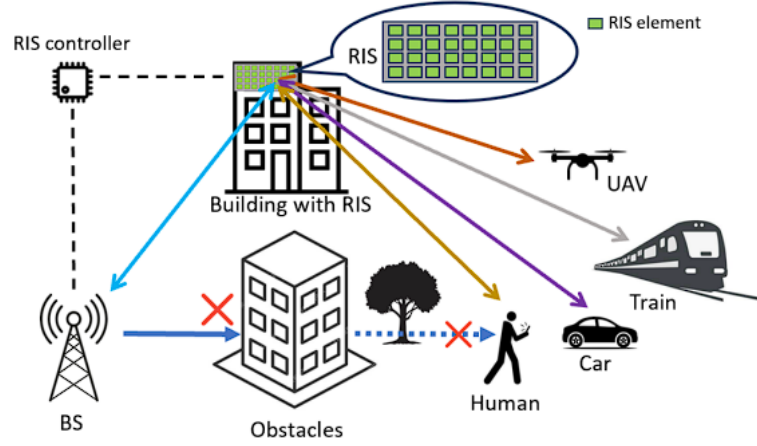


Figure 3: RISs Applications

- **Coverage Extension and Blockage Mitigation:** In high-frequency mmWave and Terahertz bands, signals suffer from severe penetration loss and are easily blocked by physical obstacles. A RIS can be strategically deployed on walls or ceilings to establish Virtual Line of Sight (vLoS) links, bypassing obstacles and illuminating signal blind spots.
- **Physical Layer Security (PLS):** RISs can enhance the inherent security of wireless transmissions. By carefully tuning the phase shifts, the RIS can constructively combine signals at the legitimate receiver's location while destructively combining them (creating a null signal) at the location of an eavesdropper [13].
- **Wireless Power Transfer (WPT):** For ultra-dense Internet of Things (IoT) networks with energy-constrained devices, RISs can focus ambient RF energy into sharp energy beams directed precisely at the target IoT nodes, significantly extending their operational lifespan [12].

2.3 Advantages and Key Obstacles

The integration of RISs into cellular networks offers distinct advantages over traditional active relays (like Amplify-and-Forward relays). Primarily, RISs operate as nearly passive devices; they only reflect incident signals without requiring power-hungry RF chains (e.g., amplifiers, analog-to-digital converters). This results in significantly lower energy consumption, reduced CAPital EXpenditure (CAPEX), and lower OPerational EXpenditure (OPEX) [14]. Furthermore, operating without amplification eliminates the issue of noise amplification and allows the RIS to inherently operate in full-duplex mode without self-interference.

Despite these unparalleled benefits, several critical obstacles must be addressed before widespread deployment [12, 11]:

- **Channel State Information (Communication Service Interface (CSI)) Acquisition:** Since the RIS elements are passive and lack active RF chains, they cannot transmit pilot signals or directly estimate the channel. Acquiring accurate, high-dimensional CSI for the whole Transmitter-RIS-Receiver link requires massive pilot overhead, which scales poorly with the number of RIS elements.
- **Hardware Constraints:** While theoretical models often assume continuous phase shifts, practical hardware (like PIN diodes) is limited to discrete, low-resolution phase shifts (e.g., 1-bit or 2-bit). This quantization leads to beam shift and performance degradation.
- **Control and Orchestration Overhead:** Managing thousands of individual elements across multiple RISs in a highly dynamic en-

vironment requires robust, low-latency control links. Integrating these physical entities into a standardized, higher-layer network management framework remains a significant open challenge.

3 Software-Defined Networking (SDN): integration within 5/6G cellular networks

Traditional computer networks have historically been built upon distributed control architectures utilizing proprietary, vendor-specific hardware. In these legacy environments, the control plane (the intelligence responsible for determining routing paths via protocols like OSPF or BGP) and the data plane (the specialized hardware responsible for physically forwarding the packets) are tightly coupled and housed within the exact same physical device, such as a router or a switch [15]. This highly distributed and rigid architecture requires network administrators to individually configure thousands of heterogeneous devices using low-level, vendor-specific Command Line Interfaces (CLIs). Consequently, traditional networks are notoriously inflexible, highly susceptible to human error during configuration, and fundamentally unable to dynamically adapt to the rapidly shifting traffic patterns and stringent requirements of modern applications.

Software-Defined Networking (SDN) emerged as a disruptive paradigm designed to resolve these intrinsic limitations by abstracting the network infrastructure. At its core, SDN proposes a strict architectural decoupling of the control logic from the underlying forwarding hardware [16]. By migrating the decision-making processes away from individual network devices and centralizing them into a software-based entity, SDN transforms the network from a static collection of autonomous boxes into a highly programmable system. This abstraction allows the underlying

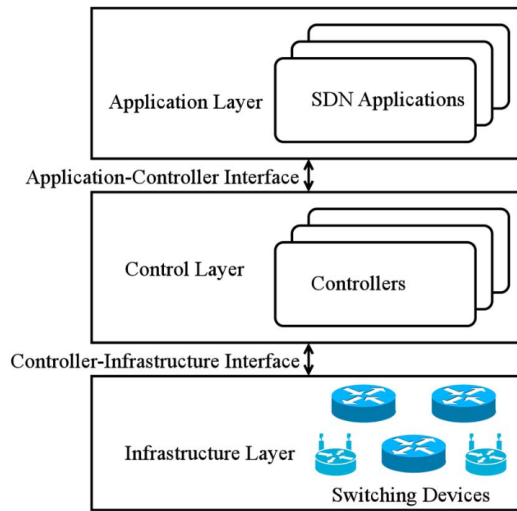


Figure 4: SDN 3-layer Architecture

infrastructure to be treated as a unified, logical switch, granting network operators unprecedented agility, centralized visibility, and the ability to automate complex network provisioning in real-time.

3.1 The Three Planes of SDN

To systematically achieve this paradigm shift, the SDN architecture is structurally divided into three distinct, highly interacting logical layers, shown in figure 4, connected via standardized Application Programming Interfaces (APIs) [16].

- **Data Plane (or Forwarding Plane):** Positioned at the bottom of the architecture, the Data Plane comprises the actual physical or virtual network infrastructure, including switches, routers, and wireless access points. In an SDN ecosystem, these devices are stripped of their complex routing algorithms and autonomous decision-making capabilities. Instead, they function as basic, highly efficient forward-

ing elements. Their sole operational purpose is to inspect incoming data packets and execute simple forwarding rules (e.g., forward out of a specific port, drop the packet, or modify the header) strictly dictated by the layer above. This simplification allows the hardware to be built using generic, cost-effective commercial silicon (bare-metal switches) rather than expensive, proprietary hardware [15].

- **Control Plane:** Operating as the centralized "brain" of the network, the Control Plane resides above the Data Plane and physically consists of one or more software-based SDN controllers (such as ONOS or OpenDaylight). The controller establishes a secure control channel with every device in the Data Plane to continuously gather telemetry and topology information. By doing so, it maintains a real-time, global view of the entire network's state. The controller utilizes this comprehensive intelligence to calculate optimal end-to-end routing paths, subsequently translating these high-level routing decisions into low-level forwarding rules that are pushed down into the flow tables of the Data Plane devices [15]. It acts as a Network Operating System (NOS), abstracting the complexity of the hardware from the applications.
- **Application Plane:** Situated at the apex of the architecture, the Application Plane houses the various business applications, network management tools, and specialized services (e.g., load balancers, intrusion detection systems, firewall policies, or industrial traffic engineering modules). Rather than interacting with physical hardware, these applications express their requirements as abstract network

policies. They communicate these intents to the SDN controller via "Northbound APIs" (typically RESTful APIs). The controller then compiles these abstract requirements and orchestrates the necessary changes across the Data Plane to fulfill the application's request [16].

3.2 Southbound Protocols

The critical point of the SDN architecture is the interface between the centralized Control Plane and the distributed Data Plane. The standardized communication protocols operating across this interface are termed *Southbound Protocols* [16]. These protocols are vital for network operation; they serve as the conduit for the controller to install, modify, and delete forwarding rules within the switches, while simultaneously allowing the switches to report asynchronous events, link failures, and port statistics back to the controller.

3.2.1 OpenFlow

Standardized by the Open Networking Foundation (ONF), OpenFlow (OF) is the pioneering and most widely deployed southbound protocol. OF completely redefines how a switch processes packets by replacing traditional routing tables with deeply programmable *Flow Tables* [17].

The operation of an OF-enabled switch relies heavily on a "Match-Action" pipeline. When a packet arrives at an ingress port, the switch parses the packet headers across multiple OSI layers (e.g., MAC addresses at Layer 2, IP addresses at Layer 3, and TCP/UDP ports at Layer 4) and

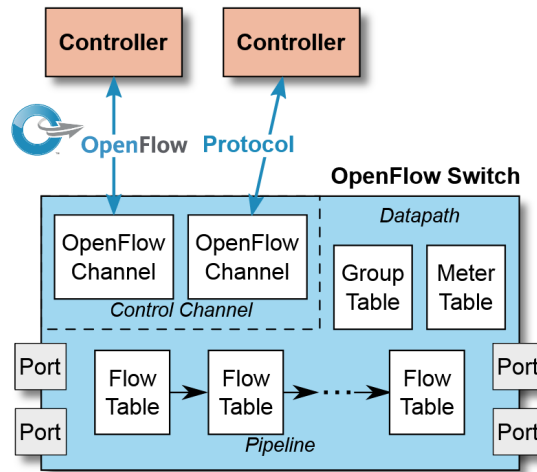


Figure 5: OpenFlow (OF) Switch Model

matches these fields against a series of prioritized flow entries installed by the controller.

- **Match Fields:** These dictate the specific criteria a packet must meet (e.g., all packets originating from a specific IIoT sensor's IP address).
- **Instructions/Actions:** If a packet matches an entry, the switch executes the associated actions, which may include forwarding the packet to a specific output queue, dropping it to mitigate a security threat, or modifying some fields of the packet itself [17].
- **Table Misses:** If an incoming packet does not match any existing flow entry, the switch encapsulates the packet header in an OF packet-in message and forwards it to the controller. The controller inspects the packet, determines the appropriate routing policy, and replies with a message to install a new rule in the switch, ensuring subsequent packets in that flow are processed directly by the hardware.

3.2.2 NETCONF and YANG

While OpenFlow is unparalleled in its ability to dynamically orchestrate real-time packet forwarding and traffic engineering, managing a complex network requires more than just routing data. Network operators must also handle the complete configuration lifecycle of the physical and virtual devices themselves, such as bringing physical interfaces up or down, configuring IP addresses, managing hardware queues, or, in the context of this thesis, structurally configuring the reflective elements of a RIS. To address this broader management scope, the Internet Engineering Task Force (Internet Engineering Task Force (IETF)) developed the Network Configuration Protocol (NETCONF) [18].

NETCONF supersedes legacy, inadequate management protocols like SNMP or CLI scraping by providing a highly robust, secure, and transactional mechanism to install, manipulate, and delete device configurations. Fundamentally, NETCONF operates on a strict **Client-Server paradigm**. In an SDN environment, the central SDN controller acts as the NETCONF client, while the managed network device acts as the NETCONF server. The server is responsible for maintaining the configuration data, validating incoming requests, and executing the configuration changes, while the client dictates the overall network policies and intents [19].

To systematically handle these interactions, the NETCONF architecture is conceptually partitioned into four distinct layers[18], outlined in figure 6:

1. **Secure Transport Layer:** NETCONF mandates a secure, connection-

oriented transport protocol. Secure Shell (SSH) is the universally adopted standard, ensuring that all configuration data and telemetry are encrypted, authenticated, and protected against tampering during transit.

2. **Messages Layer:** This layer defines the Remote Procedure Call (RPC) framing mechanism. The client encapsulates its requests within `<rpc>` elements, and the server responds with `<rpc-reply>` elements. This provides a reliable, asynchronous message-passing framework independent of the underlying transport.
3. **Operations Layer:** This layer defines the set of base protocol operations invoked via the RPC methods. It establishes the actual commands (e.g., retrieving data, modifying configurations) that the server can execute.
4. **Content Layer:** The uppermost layer represents the actual configuration and state data being manipulated. In modern implementations, this data is strictly modeled using YANG.

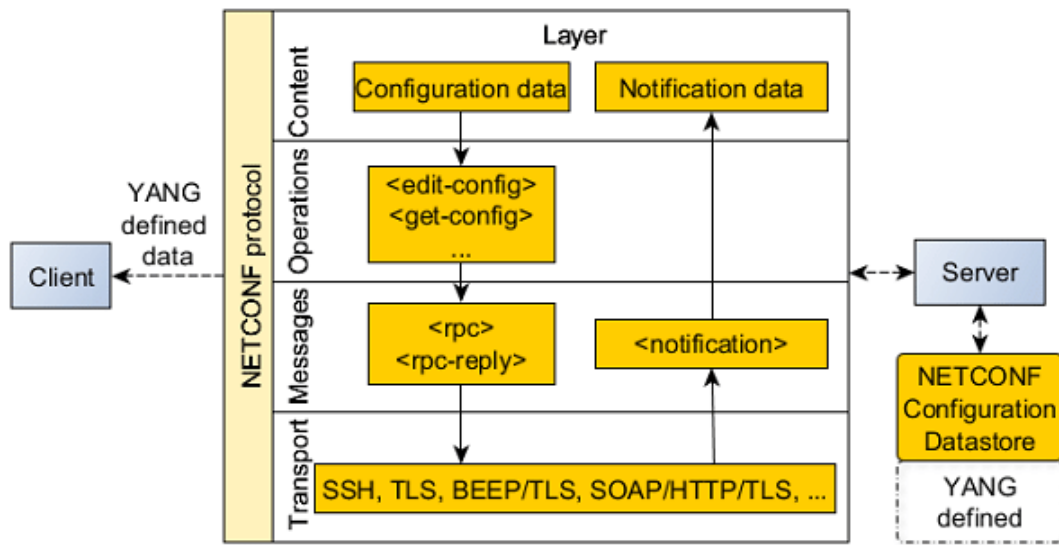


Figure 6: NETCONF Model

Building upon this layered architecture, NETCONF distinguishes itself through several advanced features and capabilities that make it exceptionally suited for SDN automation:

- **XML Encoding:** Both the protocol messages (RPCs) and the configuration data itself are encoded using the Extensible Markup Language (XML). This provides a strictly structured, hierarchical, and machine-readable format that is highly conducive to software automation, eliminating the fragility of parsing unstructured text output from legacy CLIs [19].
- **Configuration Datastores:** A defining feature of NETCONF is its clear separation of configuration data into distinct conceptual datastores [18].
 - **<running>:** Represents the complete, currently active configuration of the device.

- `<candidate>`: A scratchpad datastore where the client can make extensive, cumulative changes offline without affecting the device's immediate operation.
- `<startup>`: The configuration that will be loaded upon the device's next reboot.

This multi-datastore approach enables transactional configurations. Operators can modify the `<candidate>` datastore and then use a `<commit>` operation to push the entire configuration block to the `<running>` datastore as a single atomic transaction. If a structural error or conflict is detected, the transaction is rolled back entirely, preventing partial configurations that routinely lead to network outages.

- **Robust Base Operations:** NETCONF defines a powerful suite of standardized operations to manipulate these datastores, including [18]:

- `<get-config>`: Retrieves all or part of a specified configuration datastore (e.g., reading the current `<running>` setup).
- `<get>`: Retrieves both the configuration data and the read-only operational state data (e.g., current link status, packet counters, or RIS temperature).
- `<edit-config>`: Loads all or part of a specified configuration to a target datastore, allowing elements to be created, deleted, or replaced.
- `<copy-config>`: Completely replaces the contents of one datas-

tore with the contents of another (e.g., reverting a device to its `<startup>` state).

- `<lock>` and `<unlock>`: Allows a client to temporarily lock the configuration of a device, preventing simultaneous, conflicting edits from other management clients or human operators.

- **Event Notifications and Publish-Subscribe Paradigm:** Beyond simply polling devices for their status, NETCONF supports a highly efficient, asynchronous notification mechanism based on a publish-subscribe model [18]. A client can issue a `<create-subscription>` request to the server, actively subscribing to specific event streams. Once subscribed, the server will spontaneously push `<notification>` messages to the client the moment a predefined event occurs (such as a link going down, a hardware failure, or a new neighbor being discovered). This event-driven paradigm drastically reduces network overhead compared to continuous polling and ensures that the SDN controller can react to topological changes or faults in real-time.

- **YANG Data Modeling:** Finally, NETCONF relies intrinsically on YANG, a highly declarative data modeling language. YANG defines the exact hierarchical structure, data types, default values, and semantic constraints of the variables the device supports. Because the SDN controller can read the device's YANG models, it fundamentally "understands" the operational capabilities of the device beforehand. This allows the controller to perform rigorous syntax and semantic validation locally, ensuring that an invalid configuration is never

dispatched to the network device [19].

3.3 Integration with NFV and Network Slicing

While SDN revolutionizes how traffic is routed by abstracting the control plane, Network Function Virtualization (NFV) concurrently revolutionizes the deployment of the network services themselves. Historically, deploying a new network service, such as a firewall, a deep packet inspector, a load balancer, or core mobile network components like the 5G Core (5GC), required purchasing and physically installing expensive, proprietary hardware devices. NFV eradicates this hardware dependency by abstracting these middlebox functions into agile software modules known as (Virtual Network Functions (VNFs)) [20]. These VNFs are deployed as virtual machines or lightweight containers running on standard, Commercial Off-The-Shelf (COTS) servers located in centralized data centers or highly distributed edge computing nodes.

The architectural convergence of SDN and NFV is highly synergistic and represents the technological bedrock of modern 5G and 6G cellular networks. NFV provides the elastic compute infrastructure and the software-based network functions, while SDN provides the programmable connectivity required to dynamically steer traffic through these distributed VNFs, using a process known as Service Function Chaining (SFC) [15].

In the demanding context of industrial 6G environments, this SDN/NFV synergy is the primary enabler for Network Slicing (NS). NS allows telecommunications operators to logically partition a single, shared physical infrastructure into multiple, completely isolated end-to-end virtual net-

works (slices). Because both the routing (SDN) and the functions (NFV) are software-defined, each slice can be independently deployed, scaled, and managed to meet wildly diverging Quality of Service (QoS) requirements [20]. For example, an operator can simultaneously provision a high-bandwidth slice dedicated to streaming massive IoT (mMTC) telemetry data, alongside a rigorously isolated, ultra-low-latency slice dedicated to transmitting mission-critical control commands to an Automated Guided Vehicle (AGV) on the factory floor (URLLC). Each slice functions autonomously without interfering with the others, maximizing resource utilization while guaranteeing strict industrial performance metrics.

4 Reference Architecture and NETCONF-based Model

The transition toward the sixth generation of mobile networks (6G) signifies a fundamental paradigm shift in how we perceive and interact with the wireless channel. Traditionally, the propagation environment was treated as a stochastic, uncontrollable entity to which communication systems had to adapt through increasingly complex signal processing at the transceivers. However, the emergence of RIS has introduced the concept of the Programmable Wireless Environment (PWE), where the electromagnetic response of the physical world becomes a software-defined resource [21]. To transform this theoretical potential into a scalable industrial reality, especially within the context of the IIoT, a rigorous and standardized architectural framework is required. This chapter defines the reference architecture that bridges the gap between high-level SDN orchestration and the low-level physical reconfiguration of RISs.

Building upon the foundational concepts of SDN and NFV discussed in the previous chapter, we propose an integrated ecosystem where the RIS is not merely a passive reflector but a fully managed network element. In this vision, the control of the wireless environment is decoupled from the underlying hardware, allowing for centralized optimization and real-time adaptation to the stringent requirements of industrial applications, such as autonomous robotics and high-precision sensing [4]. While early explorations into SDN-based wireless control focused primarily on the data plane and flow-based forwarding, the complexity of managing large-scale

RIS deployments demands a more robust approach to the management plane. As highlighted in the broader 6G vision, the integration of heterogeneous devices requires a control framework that is both transaction-safe and highly interoperable [3].

The following sections detail the evolution of this architecture. We begin by revisiting the original system design established in the existing literature, providing a baseline for the integration of intelligent surfaces within SDN layers. We then critically analyze the limitations of legacy control approaches in handling the structural configuration and state monitoring of RIS devices. This critical review sets the stage for the core contribution of this work: a comprehensive NETCONF-based control architecture. By leveraging NETCONF and its underlying YANG data modeling, we define a framework capable of handling the entire lifecycle of the RIS, from discovery and capability exchange to precise beamforming configuration and asynchronous event notification, ensuring that the smart radio environment can be managed with the same rigor and reliability as traditional wired data center infrastructure.

4.1 Reference Architecture

The realization of a PWE in industrial contexts requires a structural evolution of the network architecture, moving toward a more holistic integration of the control logic and the physical layer. As proposed in [22], the reference architecture for 6G-based IIoT scenarios is built upon a modified three-layer SDN framework. This model extends the classic decoupling of control and data planes to encompass the unique require-

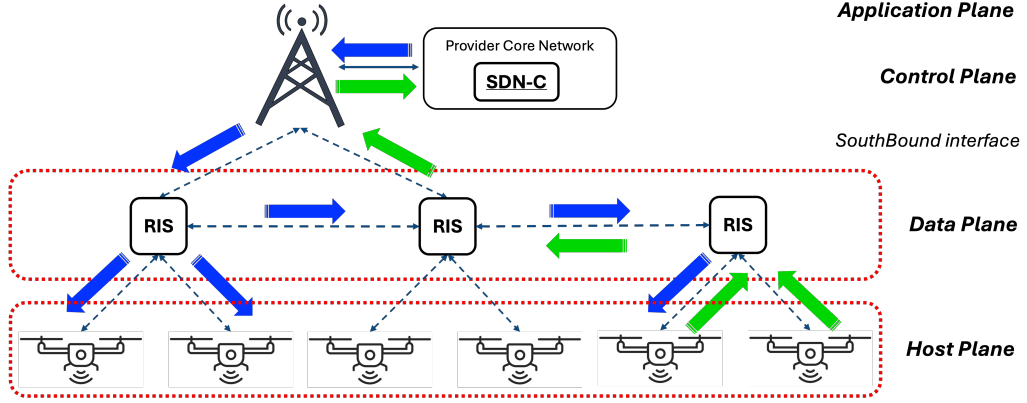


Figure 7: Reference 3-layer Architecture

ments of RIS management, ensuring that RISs are treated as fully integrated, programmable network nodes rather than isolated hardware components. This hierarchical organization consists of the *Application Plane*, the *Control Plane*, and the *Infrastructure Plane*, each interconnected through standardized interfaces to ensure interoperability and scalability within a dense industrial deployment. The whole architecture is depicted in figure 7.

At the foundation of this architecture lies the *Infrastructure Plane*, which represents the physical substrate of the 6G-enabled factory floor. This plane is characterized by a high degree of heterogeneity; therefore, it is divided in a *Host Plane*, hosting a vast array of IIoT end-devices (such as sensors, actuators, autonomous mobile robots, drones), and a *Data Plane*, comprising the RIS units themselves. In this layer, the RIS serves as a critical physical entity composed of a large number of low-cost, semi-passive reflecting elements. Each RIS is coupled with a local RIS Controller, a dedicated hardware component (often based on microcontrollers or microprocessors) that acts as the actuator for the surface.

The RIS controller is responsible for receiving low-level instructions and translating them into the specific electronic signals required to alter the impedance of the individual meta-atoms.

The communication bridge between this physical layer and the centralized intelligence is the Southbound Interface (SBI), which in this reference model is fundamentally built upon the OpenFlow (OF) protocol. Originally designed for managing packet-forwarding rules in Ethernet switches, OF is leveraged here as the primary signaling framework for the control of the smart radio environment. By utilizing an OF-based channel, the architecture ensures a standardized and secure path for the exchange of control messages, including feature discovery, configuration updates, and status monitoring. The choice of OF is strategic, as it allows the integration of RIS control into existing SDN ecosystems, taking advantage of mature software libraries and established security mechanisms. Through this interface, the controller can push down specific configuration messages that dictate how the infrastructure should behave, ensuring that the management of wireless propagation follows the same rigorous logic as traditional data plane forwarding.

Building upon this OF-based communication channel, the architecture introduces a groundbreaking abstraction: the RIS is modeled as a logical SDN switch, as shown in figure 8. Under this paradigm, the RIS is no longer viewed by the network as a complex electromagnetic device, but as a specialized "forwarding" node where the action performed is the reflection of energy toward a specific spatial direction. This abstraction allows the SDN Controller to treat the RIS as an OF-compliant entity, maintaining a "flow table" equivalent for the wireless channel. In this context, a

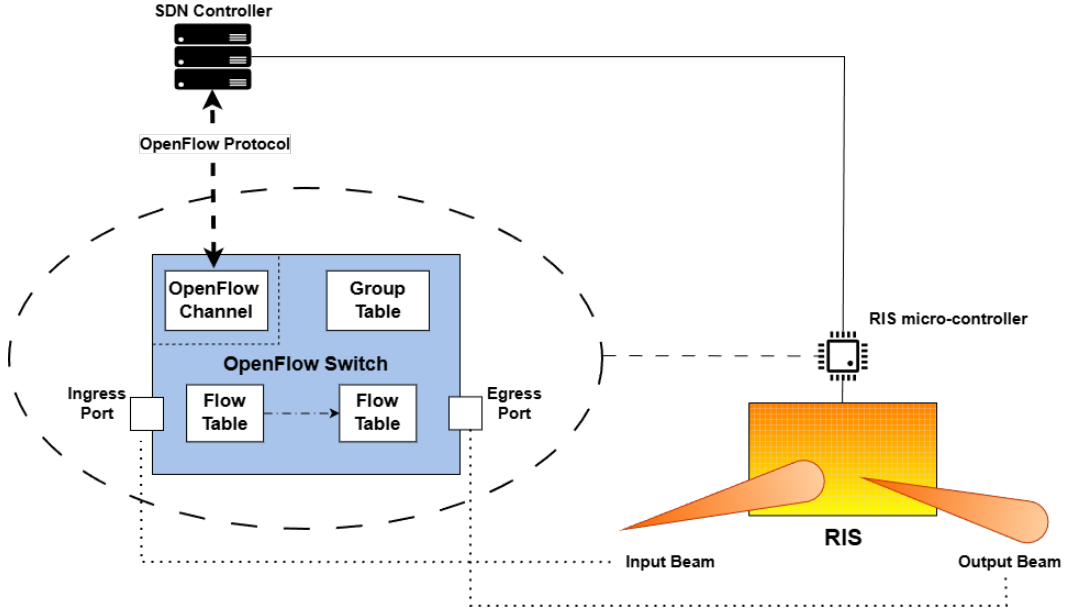


Figure 8: RIS abstraction into an Openflow Switch

"flow" represents a wireless link with specific spatial and phase parameters. Moreover, RIS beams are logically mapped as Switch ports, and the RIS microcontroller has the responsibility of solving this mapping problem in order to correctly enable the correct beams. Therefore, when the controller needs to establish a high-quality link for a mobile robot, it calculates the necessary RIS beams to be activated and wraps it into a structural configuration message that mimics an OF rule. This allows the network to apply graph-theory optimization and standard routing algorithms to the physical environment, effectively virtualizing the wireless channel and treating it as a software-defined resource.

The Control Plane constitutes the intelligence of the system, centered around a logically centralized SDN Controller. According to the functional decomposition described in [22], the controller is organized into several key modules that interact to provide seamless management. The

Service Manager provides the interface with the upper layers, while the Network Manager maintains a global and updated view of the topology and the operational state of all nodes. Crucially, the Resource Allocator and the RIS Management Module work in tandem to execute the optimization logic. The former identifies the best radio resources and paths, while the latter translates these high-level requirements into the specific phase-shift configurations required for each RIS logical switch. This functional architecture ensures that the controller can dynamically adapt to the mobility of IIoT devices or changes in the propagation environment by recalculating the necessary parameters and signaling them via the SBI in near real-time.

Finally, the Application Plane resides at the top of the hierarchy, hosting the business logic and industrial services that dictate the overall network behavior. Through the Northbound Interface (NBI) external applications can communicate high-level "intents" to the controller. For example, an industrial monitoring system might request a "High Reliability/Low Latency" slice for a critical control loop. The Application Plane translates this intent into specific performance requirements, which are then pushed to the Control Plane. The SDN Controller, having a global view of the infrastructure and the ability to control the physical environment via the abstracted RIS-switches, implements the necessary physical configurations to satisfy the request. This closed-loop functional interaction ensures that the network remains agile and capable of dynamically reconfiguring the wireless environment to meet the fluctuating demands of an automated industrial plant, effectively transforming the physical layer into a fully programmable, service-oriented asset.

4.2 OpenFlow Limitations

The integration of RIS within an SDN framework relies on a specific abstraction where the RISs is modeled as a logical SDN switch. In this architecture, the SDN controller interacts with the RIS microcontroller using standard OpenFlow (OF) rules, just as it would with a traditional Ethernet switch. The controller installs flow rules that follow the classic match-action pipeline: when a packet matches certain header criteria, the assigned action is to "forward" the traffic through a specific OF "port". The core of this mechanism lies in the local intelligence of the RIS microcontroller, which is responsible for maintaining a mapping between these logical OF ports and the actual physical RIS beams. When the microcontroller receives a command to forward traffic to "Port N ", it translates this logical instruction into the specific electromagnetic configuration (the phase-shift matrix) required to steer the signal in the corresponding spatial direction. While this approach allows for immediate compatibility with existing SDN controllers, it introduces several structural and operational drawbacks that limit the performance and reliability of 6G Industrial IoT networks. The primary limitations of this OF-based management can be summarized as follows:

- **Layer Mismatch and Signal vs. Packet Processing:** The most fundamental issue is that OF is designed for packet-level switching at Layer 2/3, whereas the RIS operates exclusively at the physical signal level (Layer 1). A RIS does not "see" or process digital packets; it manipulates electromagnetic waves in real-time. Forcing a physical-layer actuator to mimic a packet-switching logic creates a

significant semantic gap. The microcontroller must constantly perform intensive deep-packet inspection and de-serialization of complex OF headers just to trigger a physical reconfiguration. This digital overhead introduces a processing latency that is often incompatible with the microsecond-level requirements of beam-steering in high-frequency 6G bands.

- **Opaque State Management and Lack of Transparency:** Because the mapping between logical ports and physical beams is stored internally and locally by the RIS microcontroller, the SDN controller remains "blind" to the actual physical state of the wireless environment. The controller knows it is directing traffic to a specific "port", but it has no standardized way to verify what that port physically represents in terms of steering angle, focal point, or beamwidth. This lack of transparency prevents the centralized logic from performing accurate physical-layer optimizations or cross-verifying the environmental state against the actual QoS metrics.
- **Synchronization and Volatility Risks:** OF is a protocol designed to manage volatile flow tables in a switch's volatile memory (RAM). It lacks a native mechanism for persistent hardware configuration management. If the RIS microcontroller suffers a power cycle or a soft reset, the active flow rules are lost. Furthermore, if the internal mapping table, which is often managed via proprietary or out-of-band channels, becomes desynchronized from the controller's database, the system faces a "silent failure": the controller pushes rules to a port it believes is optimal, while the RIS applies a com-

pletely different or default beam, leading to a total breakdown of the reflective link.

- **Inflexible Data Modeling:** OF is a "wire protocol" that defines message formats but lacks a formal data modeling language. In an industrial deployment with heterogeneous RIS hardware (varying in element counts, bit-resolution for phase shifts, or frequency bands), OF provides no way for the controller to discover these specific capabilities. Without a modeling framework like **YANG**, the controller cannot validate if a requested configuration is physically possible for a specific device, forcing the use of non-standardized drivers and undermining the vendor-neutrality of the SDN ecosystem.
- **Absence of Transactional Integrity:** Managing a smart radio environment often requires the coordinated reconfiguration of multiple RISs to create an end-to-end reflective path. OF lacks transactional support, meaning it cannot guarantee that a set of updates is applied "atomically" (all-or-nothing). If a rule update fails at one microcontroller due to a transient link error, the controller has no standardized way to perform a "rollback" on the other surfaces. This can leave the physical environment in an inconsistent state, where signal reflections are misaligned, wasting power and causing interference.

These bottlenecks demonstrate that while the RIS-as-a-switch abstraction was a useful starting point, the use of a packet-centric protocol like OF for physical-layer management is inherently inefficient, at least for a semi-passive RIS. To achieve the level of reliability and precision required

for 6G, a shift toward a different protocol, for example NETCONF, might be necessary. This transition allows the network to treat the physical environment not as a "fake" switch, but as a formal, programmable, and persistent hardware device.

4.3 NETCONF-based model

To overcome the structural bottlenecks and the semantic mismatch inherent in the OF-based approach, this thesis proposes a transition toward a model-driven management framework based on the Network Configuration Protocol (NETCONF). The fundamental shift lies in moving away from the "RIS-as-a-switch" abstraction toward a system where the RIS is treated as a software-defined network element. In this new paradigm, the interaction between the controller and the RIS is no longer defined by transient forwarding rules, but by the formal management of the device's operational state. By leveraging a configuration-oriented protocol, we can align the control plane with the actual physical nature of the RIS, treating the electromagnetic environment as a persistent and programmable resource.

This architectural evolution redefines the roles within the SDN ecosystem. The SDN Controller acts as a NETCONF Client, while the RIS microcontroller is elevated to a NETCONF Server. This relationship allows for a much more granular and transparent control over the RIS. Instead of pushing opaque port numbers that must be locally translated by a hidden mapping table, the controller can now interact with a formal data model that explicitly describes the RIS's physical capabilities. This en-

sures that the intelligence of the network is not just "blindly" steering traffic, but is actively managing the physical substrate of the 6G environment with full visibility and transactional safety.

The proposed NETCONF-based model is structured around three key operational pillars that ensure the reliability and efficiency of the PWE. First, we define a comprehensive configuration workflow, which governs how the RIS transitions from an idle state to an active beamforming configuration through atomic transactions. Second, we utilize YANG as the data modeling language to provide a standardized schema for RIS parameters, ensuring interoperability across different hardware vendors. Finally, we implement an asynchronous Notification Pattern, allowing the RIS to proactively inform the controller of environmental changes or UE discovery or mobility. This whole framework addresses the volatility and abstraction overhead issues of previous solutions, providing the robust management plane required for mission-critical Industrial IoT applications.

4.3.1 RIS configuration Workflow

The transition from a passive hardware state to an active, beamforming-enabled state is managed through a structured Configuration Workflow. Unlike the best-effort rule installation seen in OF, the NETCONF workflow ensures that every change to the RIS electromagnetic response is validated, synchronized, and persistent. This process is centered around the interaction between the SDN Controller (acting as the NETCONF Client) and the RIS microcontroller (acting as the NETCONF Server), leveraging the protocol's native support for distinct datastores.

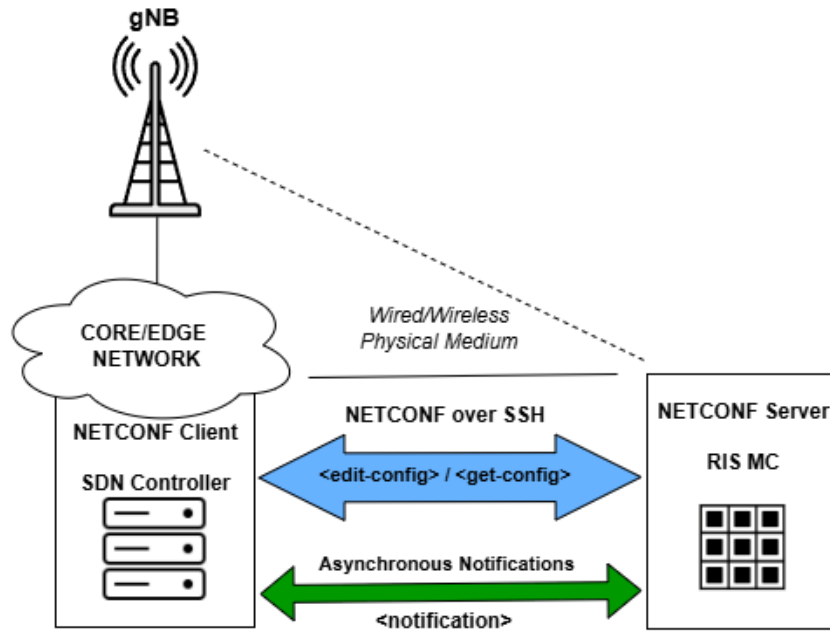


Figure 9: NETCONF-based Architecture

The standard operational workflow for enabling or modifying an RIS beam consists of the following phases:

- Session Establishment and Capability Exchange:** The process begins with the establishment of a secure transport layer connection (typically over SSH). Upon connection, both the controller and the RIS exchange `<hello>` messages containing their respective capabilities. During this phase, the RIS provides the URI of its YANG model, allowing the controller to understand the specific physical constraints of the RIS, such as the number of elements and supported phase-shift increments.
- Staging the Configuration (Edit-Config):** When a new beam configuration is required (e.g., to follow a mobile robot), the controller may not modify the active state directly. Instead, it could issue an

`<edit-config>` RPC targeting the `<candidate>` datastore. This message contains the high-level beam parameters or the specific phase-shift matrix. By writing to the candidate datastore, the controller can "stage" complex configurations involving multiple parameters without interrupting the current reflective state of the surface.

- **Validation and Consistency Check:** Before the configuration is applied, the NETCONF server on the RIS microcontroller performs an internal validation. It checks the incoming data against the constraints defined in the YANG model. If the controller attempts to set an invalid phase-shift value or a beam index that the hardware cannot support, the RIS microcontroller rejects the request with an `<rpc-error>`, ensuring the hardware never enters an undefined or dangerous state.
- **Atomic Commitment:** Once the configuration is staged and validated, the controller issues a `<commit>` command. This is the most critical step of the workflow: the NETCONF server atomically moves the data from the `<candidate>` datastore to the `<running>` datastore. At this precise moment, the RIS software logic triggers the local hardware drivers to update the bias voltages of the meta-atoms. This "all-or-nothing" transition ensures that the entire metasurface reconfigures simultaneously, preventing transient periods of signal misalignment.
- **Verification and State Retrieval:** After the commit, the controller can verify the operational status by issuing a `<get>` or `<get-config>` RPC. This allows the centralized logic to confirm that the physical

state of the RIS (e.g., the current beam power or orientation) matches the intended network intent, providing the full transparency that was missing in previous packet-based abstractions.

This workflow effectively transforms the RIS from a "black box" into a fully observable and manageable network asset. By decoupling the staging of the configuration from its execution, the network can pre-calculate and validate physical-layer changes, reducing the risk of connectivity drops during critical industrial operations. Furthermore, the use of a persistent `<startup>` datastore ensures that the RIS can autonomously restore its last valid configuration after a reboot, significantly increasing the resilience of the smart radio environment. It is also important to underline that, NETCONF is able to support also `<edit-config>` operation that modify the `<running>` datastore configuration directly, without decoupling the editing in a two-step *stage* and *commit* operation.

4.3.2 YANG model

To enable the model-driven management described in the previous sections, a formal and machine-readable definition of the RIS capabilities is required. This is achieved through YANG (Yet Another Next Generation), a data modeling language designed specifically for network configuration protocols like NETCONF. By utilizing a structured YANG model, the SDN Controller can perceive the RIS not as a generic set of unit cells, but as a sophisticated network element with well-defined properties, constraints, and operational states. The proposed YANG model for the RIS is organized into several functional blocks that capture both the static

administrative data and the dynamic physical-layer parameters:

- **Device Metadata and Spatial Context:** At the root of the model, the `ris` container identifies the specific unit via a unique `ris-id`. Furthermore, the model incorporates a `position` container defining the x, y, z coordinates of the surface within the industrial floor. This spatial awareness is fundamental for 6G path-planning algorithms, allowing the controller to calculate the geometric relationship between the RIS, the Next Generation Node B (gNB), and the mobile IIoT devices.
- **The Beam Management Sub-tree:** The core of the configuration lies in the `beam` list. Each entry in this list represents a pre-calculated spatial profile or "beam" that the surface can implement. Each beam is characterized by:
 - **id:** A unique identifier for the beam configuration.
 - **direction:** A descriptive string or vector defining the spatial orientation of the reflection.
 - **enabled:** A boolean leaf that acts as the primary switch to activate or deactivate a specific reflective pattern.
 - **status:** An operational leaf utilizing a custom `beam-status` enumeration (`IDLE`, `ACTIVE`, or `SCANNING`). This allows the controller to distinguish between a beam that is merely configured and one that is actively manipulating the electromagnetic environment.
- **Local Sensing and Neighbor Discovery:** Beyond simple reflection, the model includes a `neighbor-list` container. This represents

the RIS's capability to act as a localized sensor within the PWE. Through this list, the RIS can store data about other discovered RIS units in its proximity, including the `rsssi` (Received Signal Strength Indicator) and the estimated distance. This architectural choice transforms the RIS into an active participant in the network's topology discovery process, rather than a transparent hardware component.

A key advantage of this YANG-based approach is its extensibility through features. For instance, the model can include a `notifications` feature flag. When enabled, this allows the RIS to support asynchronous event-driven communication. While the configuration of beams and neighbors is handled through the synchronous `edit-config` RPCs discussed in the workflow section, the YANG model also defines the structure for event-based messages, such as the detection of a new IIoT device or the discovery of a neighboring node. By defining these parameters in a formal schema, the network achieves semantic interoperability. Any NETCONF-compliant controller can fetch this model from a new RIS upon connection, instantly understanding its physical constraints, such as how many beams it can support or its precise location, without requiring manual pre-configuration. This model-driven approach provides the high degree of automation and reliability necessary for the complex orchestration of 6G smart radio environments.

```

module ris {
    namespace "urn:ris";
    prefix ris;

    feature notifications {
        description "Enable NETCONF_
notifications_for_this_module";
    }

    typedef beam-status {
        type enumeration {
            enum IDLE {
                description "Beam is not active";
            }
            enum ACTIVE {
                description "Beam is actively_
transmitting/receiving";
            }
            enum SCANNING {
                description "Beam is in scanning_
mode_for_discovery";
            }
        }
    }

    container ris {
        leaf ris-id {
            type uint16;
        }

        container position {
            leaf x { type decimal64 { fraction-
digits 2; } }
            leaf y { type decimal64 { fraction-
digits 2; } }
            leaf z { type decimal64 { fraction-
digits 2; } }
        }

        list beam {
            key "id";
            leaf id {
                type uint8;
            }
            leaf direction {
                type string;
            }
            leaf enabled {
                type boolean;
                default false;
            }
            leaf status {
                type beam-status;
                default IDLE;
            }
        }

        // Neighbor Table for discovery protocol
        container neighbor-list {
            description "List of discovered_
neighbors_via_local_sensing";

            list neighbor {
                key "neighbor-id_beam";
                description "Entry in the neighbor_
table";

                leaf neighbor-id {
                    type uint16;
                    description "ID of the neighbor_
RIS";
                }
                leaf beam {
                    type uint8;
                    description "Local beam used to_
reach_this_neighbor";
                }
                leaf rssi {

```

```

    type int32;
    units "dBm";
    description "Received_Signal_Strength_Indicator";
}

leaf distance {
    type decimal64 {
        fraction-digits 2;
    }
    units "meters";
    description "Estimated_distance_to_neighbor";
}

// Notification for new IoT device detected
notification iot-device {
    if-feature notifications;
    description "Event_generated_when_a_new_IoT_device_is_detected.";
    leaf ris-id {
        type uint16;
    }
    leaf iot-id {
        type uint16;
    }
    leaf rssi {
        type int32;
    }
    leaf aoa {
        type decimal64 {
            fraction-digits 2;
        }
        // Notification for neighbor discovery
        notification neighbor-discovered {
            if-feature notifications;
            description "Event_generated_when_a_new_neighbor_RIS_is_discovered.";
            leaf ris-id {
                type uint16;
                description "ID_of_the_RIS_that_discovered_the_neighbor";
            }
            leaf neighbor-id {
                type uint16;
                description "ID_of_the_discovered_neighbor";
            }
            leaf beam {
                type uint8;
                description "Beam_used_for_discovery";
            }
            leaf rssi {
                type int32;
                description "Signal_strength_of_the_discovery";
            }
        }
    }
}

```

Listing 1: Complete RIS YANG model

4.3.3 Notification pattern

A fundamental requirement for the orchestration of a PWE is the ability of the network to react in real-time to environmental fluctuations, such as the appearance or mobility of IIoT devices or changes in the reflective topology. While the configuration workflow described previously handles synchronous, controller-initiated updates, the Notification Pattern introduces a vital asynchronous signaling mechanism. This framework is fundamentally based on a Publish-Subscribe paradigm.

In this model, the RIS microcontroller (acting as the NETCONF Server) functions as the Publisher, while the SDN Controller (acting as the NETCONF Client) assumes the role of the Subscriber. To initiate this asynchronous flow, the controller must first establish a subscription, typically via a `<create-subscription>` RPC specifying the event streams it wishes to monitor. Once the subscription is active, the RIS proactively "pushes" event-driven data to the controller as soon as they are detected by the hardware's local sensing capabilities. This shift from polling-based to event-driven communication is essential in order to reduce the SDN Controller overhead for RIS and IIoT devices discovery and tracking.

Based on the structure of the RIS data model described in section 4.3.2, the notification pattern is designed around two primary event types that enable a sophisticated, closed-loop control system:

- **IoT Device Detection (iot-device):** This notification is triggered whenever the RIS identifies a new wireless emitter within its sensing range. As shown in figure 10, the IoT device sends a pilot mes-

sage, analogous to a *beacon*. RIS receives it and sends to the Controller a notification that carries critical physical-layer telemetry, including the `iot-id`, the Received Signal Strength Indicator (RSSI), and the Angle of Arrival (AoA). By receiving these parameters in an asynchronous manner, the SDN controller can immediately calculate the precise phase-shift matrix required to steer a high-gain beam toward the mobile device. This avoids the high latency and signaling overhead associated with traditional, "blind" beam-sweeping procedures.

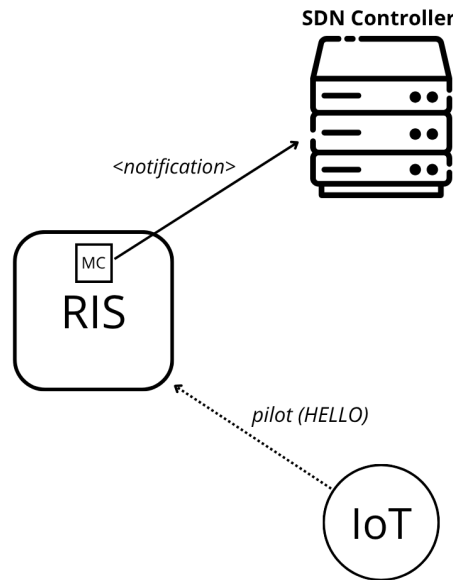


Figure 10: IoT Device Detection

- **Neighbor Discovery (neighbor-discovered):** In dynamic industrial environments where RISs may be mounted on mobile platforms or where signal paths are frequently obstructed, maintaining a real-time topology map is critical. This notification is generated when an

RIS identifies a neighboring unit, i.e. another RIS, through its local discovery protocol. It reports the `neighbor-id`, the signal strength, and the specific local beam index used during the discovery handshake. This allows the controller to dynamically update its global topology graph, facilitating the automated orchestration of multi-hop reflective paths without manual re-configuration.

The integration of this Publish-Subscribe framework into the management plane provides several deep technical advantages for 6G industrial networks:

- **Decoupling of Sensing and Action:** The Publish-Subscribe model allows the RIS to function as a distributed sensor node. The controller does not need to know *when* to check for a new device; it simply waits for the "push" from the surface. This ensures that the network remains reactive to physical changes even when the controller is busy managing other network segments.
- **Significant Reduction in Control Plane Overhead:** By eliminating the need for periodic polling, where the controller would repeatedly query the state of every RIS, the notification pattern ensures that the control channel is only utilized when a significant event occurs. This efficiency is relevant in dense deployments where several RIS might be active simultaneously.
- **Ultra-Low Latency Closed-Loop Control:** Because the signal-level data (such as AoA and RSSI) is pushed immediately upon detection, the controller can trigger a reconfiguration workflow in near

real-time. This provides the necessary agility to maintain ultra-reliable links for fast-moving autonomous robots that navigate through complex multipath environments.

- **Transactional and Persistence Support:** Notifications are integrated into the wider NETCONF ecosystem. If a notification indicates a hardware failure or a link drop, the controller can use the transactional integrity of the protocol to roll back the entire path to a known stable state, ensuring the network never remains in an inconsistent or broken configuration.

In conclusion, the transition to a notification-based management paradigm transforms the RIS from a passive reflector into an active, sensing participant in the network. This provides the centralized controller with the "situational awareness" required to manage the 6G physical environment with the same precision and reliability as traditional digital network layers.

4.4 Physical Control Frameworks for RIS Management

The effectiveness of the proposed NETCONF-based management architecture is inherently dependent on the physical control framework that interconnects the centralized SDN Controller and the distributed RISs deployed throughout the industrial environment. While the logical management layer defines the configuration semantics through YANG models and the transaction protocol through NETCONF, the physical control plane determines the reliability, latency, and scalability of such management transactions.

In a dense IIoT scenario, where RISs are deployed as programmable electromagnetic surfaces forming a PWE, the control connectivity cannot be considered an auxiliary aspect. Instead, it becomes a foundational design dimension that directly influences convergence time, discovery efficiency, and overall system stability.

Two primary architectural paradigms can be identified for the realization of the physical control plane:

- A **Wired Control Framework**, based on dedicated physical infrastructure.
- A **Wireless Control Framework**, enabled by 6G connectivity and enhanced through Network Slicing (NS).

These two models represent fundamentally different trade-offs between determinism and flexibility. The wired solution maximizes reliability and isolation, whereas the wireless approach introduces architectural agility and enables dynamic multi-hop reflective topologies. The following subsections analyze both paradigms in detail.

4.4.1 Wired RIS Control Framework

The wired control framework represents the most deterministic implementation for RIS management and is particularly suitable for static RIS permanently integrated into the factory infrastructure (e.g., walls, ceilings, structural columns).

In this architecture, each RIS microcontroller is connected to the centralized SDN Controller, or to an edge gateway, via dedicated physical

links such as optical fiber. The control traffic therefore traverses a physically isolated network segment, completely decoupled from the wireless data plane used by industrial devices.

The primary advantage of this model lies in its deterministic behavior:

- **Immunity to Radio Impairments:** Control messages are not affected by multipath fading, shadowing or analogous effects which are common in heavy industrial environments with metallic structures and high-power machinery.
- **Stable NETCONF Sessions:** Since NETCONF relies on TCP-based transactional integrity, packet loss or jitter may cause session resets or unintended <rollback> operations. A wired channel minimizes such risks.
- **Predictable Control Latency:** The absence of medium access contention ensures bounded delay for NETCONF RPCs.
- **Power Delivery via PoE:** Through Power over Ethernet, the RIS can receive both energy and management traffic over the same cable, simplifying deployment and ensuring continuous operation of active components.

However, the wired framework presents significant limitations:

- High costs in large-scale deployments.
- Limited scalability for high-density RISs.
- Incompatibility with mobile platforms

- Lack of adaptability to dynamic topologies.

In highly reconfigurable industrial scenarios, where RISs may be re-located or dynamically attached to moving assets, the rigidity of cabling infrastructure becomes a structural constraint.

4.4.2 Wireless RIS Control Framework and Network Slicing

The wireless control framework enables over-the-air management of the PWE, allowing the SDN Controller to reach RISs through the 6G Radio Access Network (RAN) infrastructure. In this architecture, NETCONF RPCs, configuration transactions, and telemetry notifications are encapsulated within IP packets transported over the cellular network.

This model introduces two key architectural properties:

1. **Mobility Support and Topological Flexibility:** Wireless control is the only viable option for RISs mounted on mobile entities. As these platforms move within the factory floor, the SDN Controller can continuously adapt the reflective configuration to maintain coverage and optimize channel conditions.
2. **Multi-Hop Reflective Control Expansion:** A distinctive advantage of wireless control in conjunction with programmable RISs is the possibility of establishing cascaded control paths. If a distant RIS does not have direct Line-of-Sight (LOS) to the RAN node, an intermediate surface can be configured to reflect a high-gain control beam toward it.

Formally, a control path can be represented as:

$$\text{Controller} \rightarrow r_i \rightarrow r_j \rightarrow r_k$$

where each intermediate RIS participates in extending the reachability graph of the management plane.

This mechanism effectively allows the network to dynamically construct its own control topology, aligning with the concept of programmable radio environments.

Network Slicing for Control Integrity The main challenge of wireless control lies in its dependency on the radio medium. Without isolation, management traffic could suffer from congestion caused by eMBB or massive mMTC flows.

To mitigate this risk, the architecture relies on a dedicated Management and Control (M&C) Slice instantiated by the network Orchestrator. This slice is logically isolated and mapped to URLLC-class resources to guarantee:

- **Low and Bounded Latency:** Ensuring real-time execution of beam reconfiguration.
- **High Reliability:** Reducing packet loss probability to maintain NETCONF session stability.
- **Radio Resource Isolation:** Preventing industrial traffic bursts from impacting control-plane delivery.

This solution could let the whole architecture manage two kind of uses, as shown in figure 11: a slice dedicated to *control* signals directed to RISs,

and another one dedicated to *data* directed from and to UEs. Therefore, RISs are treated differently by the logical separation in slices, as they are devices to be programmed by the Controller Network Function (NF) (i.e. SDN Controller) in the RIS Management and Control Slice, while they are configured reflective surfaces when inserted in the IIoT Data Slice.

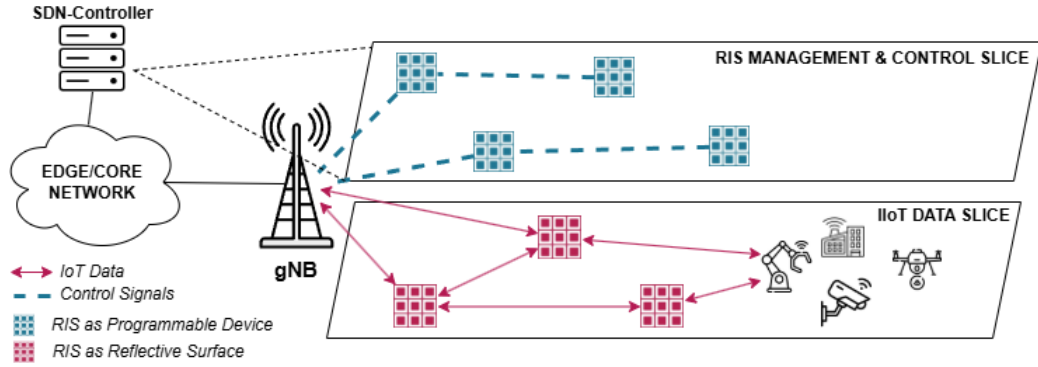


Figure 11: Slicing for Wireless RIS Control

Recursive Dependency Consideration Wireless control introduces a recursive dependency: the quality of the control link may depend on the configuration of the RISs themselves. Therefore, the slicing mechanism and the initial bootstrap configuration must ensure a minimal reliable control path at all times to avoid control deadlock conditions.

This recursive characteristic represents both the primary risk and the main innovation of the wireless framework. When properly engineered through 6G slicing and prioritized scheduling, the wireless control plane can achieve reliability levels approaching wired solutions, while enabling unparalleled deployment flexibility and multi-hop expansion.

In summary:

- The **wired framework** maximizes determinism and is ideal for static,

mission-critical RISs.

- The **wireless framework**, empowered by Network Slicing (NS), enables mobility, scalability, and dynamic topology construction.

4.5 RIS Discovery Iterative Algorithm

The deployment of distributed RISs within a complex and dynamic industrial environment necessitates a systematic discovery mechanism. Such a mechanism must be capable of identifying not only the available surfaces but also the feasible multi-hop reflective paths that link them. In a PWE, reachability is not a static property determined solely by physical proximity; rather, it is a dynamic state defined by the specific electromagnetic beam configurations of the participating surfaces. This section reviews the discovery mechanisms adopted in prior literature and introduces a novel, NETCONF-based iterative multi-hop discovery algorithm designed to reconstruct the network topology in real-time.

4.5.1 Former Discovery Approach

In traditional SDN-oriented 6G industrial architectures, such as the one proposed by Chiti et al. in [22], RIS discovery is largely treated as a static or semi-static process. This approach typically relies on a centralized optimization framework where the SDN controller is assumed to possess extensive *a priori* knowledge of the deployment. Specifically, the controller maintains:

- A fixed database of RIS positions and identifiers.

- A pre-computed radio map or environmental model.
- Centralized placement algorithms [22] that determine the optimal active subset $\mathcal{R}_{active} \subseteq \mathcal{R}$.

A key feature of this former approach is the utilization of protocols like the **Link Layer Discovery Protocol (LLDP)** to maintain topology awareness. In traditional networking, LLDP allows devices to advertise their identities and capabilities to neighbors. However, applying LLDP to RISs is problematic because it operates at the data link layer. Since an RIS does not natively process or generate digital frames, it natively cannot "speak" LLDP to its neighbors without a complex and power-hungry microcontroller. Furthermore, this approach does not adequately address the possibility of *multihop* patterns between RISs, as originally proposed for the wired network featured by Switch devices; *multihop* patterns could only be established letting RISs do a *flooding* activity, making the protocol non-deterministic.

4.5.2 Problem Formulation

We model the RIS network as a directed graph $G = (\mathcal{R}, \mathcal{E})$, where $\mathcal{R} = \{r_1, r_2, \dots, r_N\}$ is the set of deployed RIS units. An edge $(r_i, r_j) \in \mathcal{E}$ exists if and only if there exists a specific beam configuration on r_i that enables a signal to reach r_j with a signal quality above a defined threshold. The objective of our discovery phase is to reconstruct the reachable subgraph $G_{reachable}$ starting from the gNB, which serves as the root node and the primary source of the control signal.

4.5.3 Proposed NETCONF-Based Iterative Discovery Algorithm

The proposed algorithm follows a frontier-expansion paradigm, fundamentally adapting the Breadth-First Search (BFS) logic to the constraints of a programmable reflective environment. Unlike standard network discovery, reachability here is a function of both physical placement and specific beam configurations.

Initialization and Mutual Sensing Phase The first frontier, $\mathcal{F}_0$, consists of all RISs directly reachable from the gNB. This is initially determined through a sweep where the gNB transmits beacons; RISs detecting these beacons establish a NETCONF session with the controller to report their presence. However, to fully map the potential of the PWE, this phase is augmented by a Mutual Sensing/Scanning procedure among the surfaces. In this stage, RISs that are mutually visible, perform a localized handshake. During this mutual sensing, the surfaces exchange critical physical-layer parameters, including their unique `ris-id`, the specific beam index used for the sighting, and the RSSI. This allows the microcontroller of each unit to populate its local `neighbor-list` (as defined in the YANG model) with high-fidelity data. By measuring the signal strength and the specific angular direction of the sighting, the surfaces can estimate their relative distances and spatial orientations. This deep sensing is fundamental because it provides the SDN controller with a granular view of the "electromagnetic neighborhood". Instead of just knowing that two surfaces exist, the controller learns exactly how they can "talk" to each other. This information is stored locally in the RIS datastores, ready to be retrieved by the controller to build the global reachability graph.

Iterative Expansion At each iteration k , the controller expands the frontier by pulling the results of the sensing phase from the already discovered RISs. For each $r_i \in \mathcal{F}_k$, the controller performs the following steps:

1. **Configuration and State Retrieval:** The controller issues a `<get-config>` RPC to the RIS. Unlike a simple status check, this operation also retrieves the populated `neighbor-list` from the YANG model, which contains the IDs of all neighboring RISs detected during the previous sensing phase.
2. **Topology Mapping:** The controller processes the retrieved `neighbor-list` to identify r_j units that have not yet been included in the global graph. For each newly identified neighbor, the controller records the specific beam b and `rsssi` required to maintain that inter-RIS link.
3. **Frontier Update:** The newly identified RISs, those that are reachable from the current frontier r_i but were previously unknown, are added to the next frontier $\mathcal{F}_{k+1}$.
4. **Next-Hop Activation:** In the subsequent iteration $k + 1$, the controller establishes NETCONF sessions with the units in $\mathcal{F}_{k+1}$ (leveraging the wireless control slice and the multi-hop paths just discovered) to trigger their own local sensing and repeat the process.

The whole message exchange is depicted in figure 12, while algorithm 1 shows the full pseudocode. The process terminates when a `<get-config>` operation returns no new neighbors that are not already part of the discovered set ($\mathcal{F}_{k+1} = \emptyset$). This approach ensures that the controller "crawls"

through the reflective environment, using each newly discovered RIS as a management relay to go deeper and broader in the RISs discovery process.

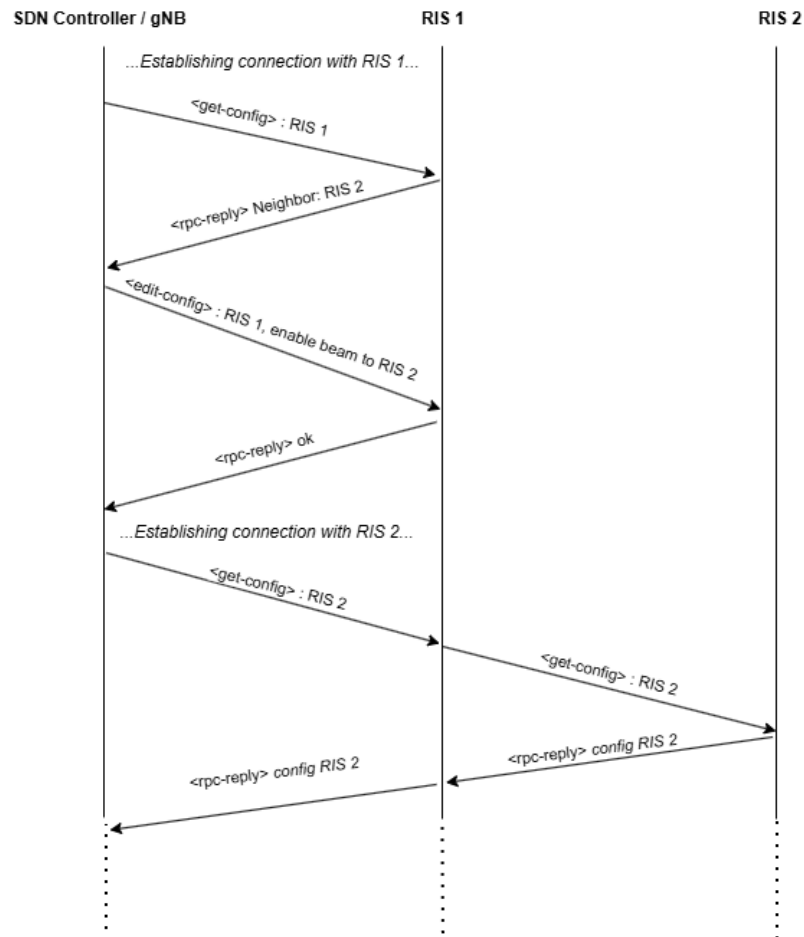


Figure 12: Message Exchange during RIS discovery via NETCONF

Input: Set of deployed RIS R , Controller C , gNB

Output: Discovered graph $G_{discovered}$

Initialize an empty set $Discovered \leftarrow \emptyset$

Initialize $Frontier \leftarrow \{r \in R \mid \text{Direct link with gNB exists}\}$

while $Frontier \neq \emptyset$ **do**

$NewFrontier \leftarrow \emptyset$

foreach r **in** $Frontier$ **do**

$neighbor_data \leftarrow \text{NETCONF_get}(r, "/ris/neighbor-list")$

foreach $entry$ **in** $neighbor_data$ **do**

$n_id \leftarrow entry.neighbor_id$

$n_beam \leftarrow entry.beam$

$n_rssi \leftarrow entry.rssi$

if $n_id \notin (Discovered \cup Frontier \cup NewFrontier)$ **then**

Add edge (r, n_id) to $G_{discovered}$ with (n_beam, n_rssi)

Add n_id to $NewFrontier$

$\text{NETCONF_connect}(n_id)$

end

end

Add r to $Discovered$

end

$Frontier \leftarrow NewFrontier$

end

return $G_{discovered}$

Algorithm 1: RIS Iterative discovery algorithm pseudocode

4.5.4 Fallback Mechanism: Broadcast Get-Config Session

A significant challenge in industrial environments is the occurrence of transient faults, physical obstructions, or protocol inconsistencies that may interrupt the sensing phase. If an RIS fails to send a notification due to a temporary link degradation within the control slice, the discov-

ery process for that branch could stall prematurely. To address this, the proposed algorithm should include a Fallback Mechanism. If the controller detects a period of inactivity or an unexpected termination of the expansion, it can initiate a Broadcast Get-Config Session. In this mode, the controller bypasses the sequential iterative logic and issues a global `<get>` RPC to all known identifiers in the network. This allows the controller to attempt to retrieve the state of every RISs, retrieving the local neighbor-list stored in the YANG model. By manually pulling the neighbor data that might have failed to be spread in the sensing and scanning phase, the controller can resolve inconsistencies and complete the reachability graph even in the presence of sporadic network faults.

5 Experimental Platforms

This section details the experimental setups and methodologies employed to validate the proposed management architecture and the iterative discovery algorithm. The evaluation is structured progressively, moving from a functional, containerized emulation to a comprehensive discrete-event simulation.

First, a Docker-based testbed is introduced to practically demonstrate the functional feasibility and effectiveness of NETCONF as the standard management protocol. This emulation deploys an SDN Controller instance acting as the NETCONF client and an RIS container acting as the server, allowing for the real-time validation of configuration exchanges and YANG model interactions.

Subsequently, the study scales the evaluation by integrating the NETCONF architectural framework into the RestartIn discrete-event simulator. Built upon the Python-based SimPy library, this simulator provides a robust environment to model network nodes, links, and queuing systems, enabling the accurate simulation of management packet dissemination and control plane latency.

Finally, this simulated environment serves as the foundation for implementing the proposed Iterative Discovery Protocol. By leveraging the simulator's capabilities, the protocol is subjected to rigorous testing to assess its scalability, convergence time, and overall effectiveness across varying densities of programmable radio environments.

5.1 Docker Emulation

The primary objective of this initial experimental phase is to validate the functional feasibility of the proposed NETCONF-based management architecture within a controlled, software-defined environment. By abstracting the physical hardware layer, this emulation focuses strictly on the control plane interactions, verifying that standard network management protocols can effectively configure, monitor, and manage RIS elements.

To achieve an isolated, reproducible, and scalable testbed, the experimental framework was implemented using Docker. The architecture relies on the deployment of distinct containerized entities that simulate the client-server relationship inherent in the NETCONF protocol. A dedicated Docker bridge network was instantiated to facilitate seamless, IP-based communication between the containers, mimicking a dedicated management network.

The server-side infrastructure, representing the RIS Master Controller, was built upon an Ubuntu-based container equipped with the Netopeer2 server and the Sysrepo configuration datastore. Netopeer2 serves as the NETCONF endpoint, handling secure SSH connections and processing remote procedure calls (RPCs), while Sysrepo acts as the central, YANG-driven datastore that maintains the running configuration and operational state of the emulated RIS.

A critical step in the server configuration involved the definition and deployment of a custom YANG data model tailored to the RIS paradigm. This model dictates the hierarchical structure of the metasurface pa-

rameters, defining actionable leaf nodes such as operational state, phase shift configurations, and active beam indices. The YANG module was formally installed into the server using the `sysrepoctl` utility, enabling Netopeer2 to rigorously validate all incoming NETCONF requests against the defined schema before applying them to the datastore.

Conversely, the client-side container, functioning as the SDN Controller, was provisioned with a Python environment utilizing the `ncclient` library. This setup allows the controller to programmatically establish secure NETCONF sessions over port 830 and automate the dispatch of XML-encoded management operations to the RIS server.

5.1.1 System evaluation

To evaluate the effectiveness and reliability of the deployed architecture, a series of automated management workflows were executed from the SDN Controller container. The Python-based client successfully established an SSH-secured NETCONF session with the Netopeer2 server, confirming fundamental connectivity.

The core of the validation involved the transmission of an `<edit-config>` RPC targeting the running datastore of the RIS. This payload was explicitly designed to modify the operational parameters defined within the custom YANG module. Upon receiving the request, the Netopeer2 server parsed the XML payload, successfully validated the proposed changes against the installed YANG schema, and applied the configuration to the Sysrepo datastore. The server subsequently replied with an `<ok/>` message, confirming the transactional integrity of the operation.

Following the configuration update, the controller issued a `<get-config>`

RPC to retrieve the current state of the datastore. The XML payload returned by the server perfectly matched the parameters previously pushed by the controller. These results are exactly as expected, definitively demonstrating that a standardized NETCONF/Sysrepo stack can seamlessly and deterministically manage RIS configurations. This successfully validates the foundational control plane mechanics required before transitioning the architecture into larger-scale discrete event simulations.

5.2 Integration in RestartIn Discrete Event Simulator

To evaluate the proposed management architecture on a larger scale, the system is integrated into *restartin-sim*, a discrete-event simulator originally developed under the RESTART project. Written in Python, this simulation framework is designed to model message exchanges and network dynamics within complex indoor environments, such as smart factories.

In its baseline configuration, the simulator implements the essential components of a cellular network through three primary classes:

- **Ue (User Equipment):** Represents the mobile or IoT devices in the network. Instances of this class are responsible for generating, processing, and transmitting periodic data packets.
- **gNodeB:** Acts as the central network node. It incorporates an internal scheduler responsible for managing and distributing available radio resources among the users, optimizing the QoS and ensuring efficient network utilization.

- **Channel:** Models an Orthogonal Frequency Division Multiplexing (OFDM) wireless channel. This class accurately simulates signal propagation, taking into account various physical factors and environmental geometry to reproduce realistic transmission conditions.

The communication protocol between the users and the gNB is modeled through a series of control and data packet exchanges. Typically, a UE initiates communication by sending a Physical Uplink Control Channel (PUCCH) packet to request network resources. Upon receiving this request, the gNB allocates the necessary resources and notifies the user via a Physical Downlink Control Channel (PDCCH) packet. Once resources are assigned, the user transmits the actual data payload using a Physical Uplink Shared Channel (PUSCH) packet, which is subsequently acknowledged by the gNB through Hybrid Automatic Repeat reQuest (HARQ) mechanisms.

Building upon this solid foundation, previous research successfully extended the simulator to introduce SDN principles and RIS. Specifically, this prior work enriched the simulation environment with:

- A RIS class, designed to simulate the physical properties and logical behavior of the reflecting panels within the environment.
- An SDN_Controller class, functioning as the centralized entity for network intelligence and routing decisions.
- A full implementation of the OF protocol. This included the simulation of standard control messages allowing the controller to dictate forwarding rules to the RIS nodes.

The current work leverages this pre-existing simulated topology. However, it replaces the OF-based logic with the newly proposed NETCONF and YANG-driven architecture. By integrating the NETCONF transactional model into this comprehensive discrete-event environment, the simulator provides the necessary framework to rigorously test and evaluate the advanced Iterative Discovery Protocol for 6G-IoT scenarios.

5.2.1 Classes Diagram and Description

The UML diagram (see Fig. 13) summarizes the set of classes introduced to model a RIS and to expose its configuration through a lightweight NETCONF-like server architecture. At a high level, the design separates

- The domain representation of a RIS
- The per-connection NETCONF session handling
- The controller that multiplexes multiple sessions
- Supporting components for configuration storage and metrics collection.

This separation keeps responsibilities well-scoped and makes it easier to extend the simulator with additional operations or monitoring features.

RIS domain model The `Ris` class represents the simulated device itself. It stores an identifier (`ris_id`) and basic geometric and operational parameters such as coordinates, norm (e.g., the normal vector represented as a list), and the number of beams. The method `is_device_in_beam()` encapsulates the core geometric check used by the simulator to determine

whether a given device position is covered by a specific beam configuration. In this way, `Ris` remains focused on the simulation semantics, independent of networking concerns.

NETCONF-facing abstraction per RIS The `NetconfRis` class acts as the interface between the NETCONF operations and a specific `Ris` instance. It aggregates a `Datastore` to persist and retrieve configuration-related structures (e.g., running/candidate/startup configurations), and it maintains session-related state such as `session_id`. In addition, it keeps beam-related configuration (`beam_config`) and a dedicated metrics object (`NetconfRisMetrics`) to record statistics about operations performed on that RIS. Conceptually, `NetconfRis` is the component where XML-oriented helpers and NETCONF operation logic are implemented, while still delegating pure device behavior to `Ris`.

Connection and session handling The `NetconfSession` class models one active connection to the server. It encapsulates the underlying socket, a unique `session_id`, and the identifier of the target RIS (`ris_id`) that the session interacts with. The methods `send()`, `receive()`, and `close()` provide the minimal primitives needed to exchange NETCONF messages and to manage the lifecycle of the connection. This class isolates networking details from both the controller and the RIS-specific logic.

Server controller and orchestration The `NetconfController` class is the main orchestration component. It binds to a host and port and maintains a collection of active sessions (`sessions`) as well as a set of

managed RIS instances (`ris_list`). Its responsibilities include session management (creation, tracking, teardown), dispatching NETCONF operations to the appropriate `NetconfRis`, and enabling higher-level behaviors such as path/beam configuration flows. Similarly to RIS-side metrics, the controller also owns `NetconfControllerMetrics` to keep global statistics at the server level.

Datastore and metrics support Finally, `Datastore` provides configuration helpers and separates configuration state into running, candidate, and startup dictionaries, mirroring common NETCONF datastore concepts. The two metrics classes, `NetconfRisMetrics` and `NetconfControllerMetrics`, expose a uniform structure (a `Metrics` dictionary) together with `record_metrics()` and getters methods, allowing both per-RIS and global measurements to be collected without polluting the core control logic.

5.2.2 Simulation file description

This section describes the `netconf_test.py` file, which implements a comprehensive simulation framework for testing Reconfigurable Intelligent Surface (RIS) management using the NETCONF protocol. Unlike traditional OF-based approaches, this implementation uses NETCONF for dynamic RIS beam configuration and control. A `UEDataMetrics` class is responsible for tracking per-UE data plane performance metrics throughout the simulation. It records packet transmission/reception counts, data volumes, latencies, and computes derived metrics like throughput and packet loss rate.

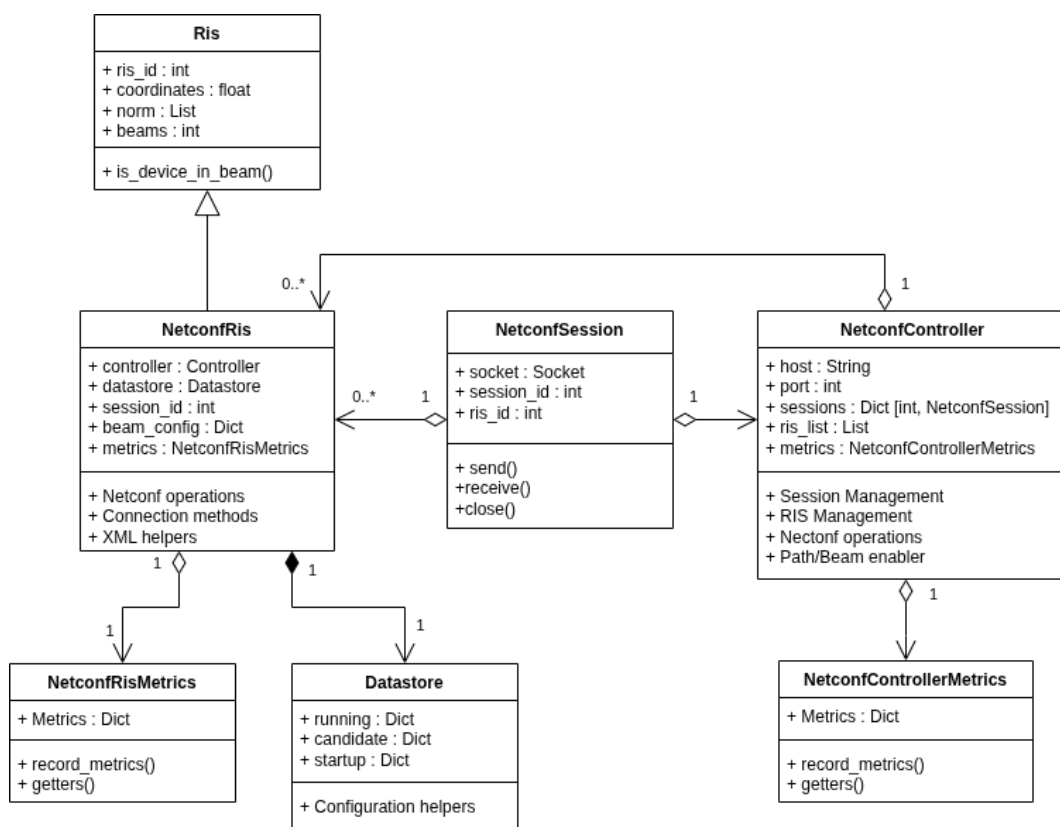


Figure 13: UML Classes Diagram

The `handle_ue_transmission_with_beam()` function manages NETCONF-based RIS configuration for all service types (PUCCH, PDCCH, PUSCH). It enforces the critical workflow:

1. Verify RIS state (LOW_POWER, RX, or ACTIVE)
2. Enable beam via NETCONF before transmission
3. Execute two-hop path (UE/gNB $\leftrightarrow$ RIS $\leftrightarrow$ gNB/UE)
4. Disable beam via NETCONF after transmission

This ensures RIS is configured *before* packet reception, preventing timing issues between control and data planes.

After the execution of the whole simulation, results data are summarized and saved in .csv files, in order to be plotted afterwards.

5.3 Iterative Discovery Protocol Implementation

The iterative discovery protocol represents the core mechanism through which the NETCONF-based RIS discovery system establishes connectivity and constructs a spanning tree topology. This implementation follows a breadth-first search (BFS) paradigm, as explained in section 4.5.3 combined with minimum spanning tree (MST) optimization to efficiently discover and connect distributed RIS nodes within factory environments.

5.3.1 Protocol Architecture

The discovery protocol is implemented in the `NetconfDiscoveryProtocol` class, which orchestrates the complete discovery lifecycle. The protocol maintains several critical data structures:

- `neighbor_tables`: A dictionary mapping RIS identifiers to their discovered neighboring RIS nodes, populated during the local sensing phase.
- `graph_paths`: A dictionary storing `PathInfo` objects for each discovered RIS, containing the parent node identifier, beam index, hop count, and signal quality metrics.
- `connected_nodes`: A set tracking which RIS nodes have been successfully integrated into the spanning tree.
- `factory_layout`: The environment model including obstacles and RF propagation constraints.

5.3.2 Discovery Phases

The iterative discovery process comprises three sequential phases:

Phase 1: Local Sensing Each RIS performs omnidirectional or beam-sweep sensing to detect neighboring nodes within communication range. The method `perform_all_local_sensing()` invokes beam steering patterns on each RIS to measure signal strength from potential neighbors. This phase generates the neighbor tables that form the foundation for subsequent discovery iterations.

Phase 2: Iterative BFS Discovery The core discovery mechanism executes iteratively until convergence. In each iteration:

1. Active nodes (initially only the controller) transmit probe messages to their known neighbors.

2. Each receiving RIS computes signal quality metrics and responds with its measurements.
3. The controller receives responses and evaluates which new nodes can be reliably reached.
4. Newly discovered nodes are added to the connected set and become active for the next iteration.

The iteration continues while new nodes are discovered, with a maximum iteration limit preventing infinite loops in disconnected topologies.

Phase 3: Spanning Tree Optimization Following discovery completion, the protocol optionally constructs a minimum spanning tree using either hop count or Signal-to-Noise Ratio (SNR) as the optimization metric. This phase selects the most efficient paths from the discovered topology, reducing network overhead while maintaining connectivity.

5.3.3 Key Implementation Details

Message Accounting The protocol distinguishes between three message types: GET operations retrieving RIS state, EDIT operations configuring RIS parameters, and REPLY messages confirming operations. Each operation type is individually counted to provide granular performance metrics.

Beam Management The `beam_angle` parameter defines the angular coverage of each RIS beam. During discovery, the protocol verifies line-of-

sight connectivity considering factory obstacles, filtering unfeasible connections before establishing links.

Hop Count Calculation Each RIS stores its hop distance from the controller, calculated as the minimum number of relay hops required for connectivity. This metric influences spanning tree construction and provides topology depth analysis.

Performance Metrics The `DiscoveryMetrics` class aggregates operational measurements including total discovery time, sensing time, average message latencies (GET/EDIT), discovery success rate, and failed connection attempts. These metrics enable comprehensive performance evaluation across different deployment scenarios.

6 Experimental Results

This section presents the experimental results obtained with the developed discrete-event simulator in an Industrial 5G-like environment. The evaluation is structured in two main parts:

1. The analysis of a packet-exchange simulation between multiple UEs and a gNB, where RIS-assisted control is managed and compared through two alternative southbound approaches, namely OF and NETCONF. The goal is to highlight differences in control overhead and end-to-end performance indicators (e.g. latency and throughput) under increasing system load and different RIS configurations.
2. The evaluation of the RIS iterative discovery protocol, which aims at discovering and connecting multiple RIS nodes to the controller by leveraging local sensing and a graph-based discovery procedure, and report the scalability trends of the protocol in different deployment scenarios.

6.1 Packet-exchange simulation: OF vs NETCONF

6.1.1 Simulation Environment Design

The simulated industrial scenario models a factory floor organised as a grid of machine units. Each machine is rendered as a cuboid, whose geometric centre is highlighted by a grey marker. Multiple UEs are then deployed with a uniform spatial distribution and are represented as red circles. Network coordination is provided by a single gNB, depicted as a blue triangle, which serves as the central base station. Finally, a set

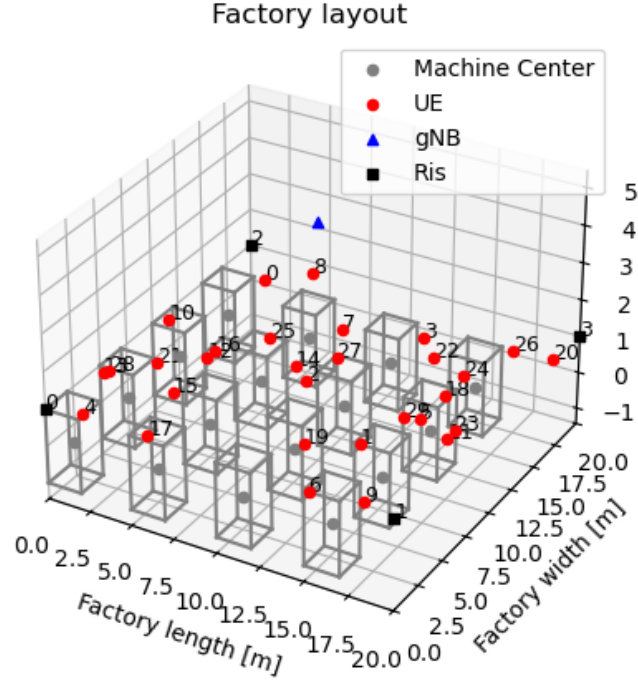


Figure 14: Simulated Industrial Scenario (case involving 4 RIS and 30 UEs)

of RIS elements is placed at selected positions around the factory layout; these are shown as black squares in Fig. 14. To compare OF and NETCONF in a controlled and repeatable way, a set of simulations was executed in which the number of UEs, the number of deployed RIS elements, and the number of available beams per RIS were varied systematically. For each configuration, the simulator collects aggregated metrics including control overhead, RIS processing time, latency statistics (also reported through percentiles) and throughput. The complete set of configurations and the corresponding measurements are exported in `results_master.csv`, which is used as the reference dataset for the plots and discussion in the next subsection.

Table 1: Main parameters for the packet-exchange simulation

Parameter	Values used in experiments
Scenario size (m^3)	$20 \times 20 \times 4$
Management protocol	OF, NETCONF
Number of UEs (ue_count)	15, 30, 45, 60, 75
Number of RIS (ris_count)	2, 4
Beams per RIS (beams)	3, 5
Simulation Time (s)	0.150

Experimental parameters Table 1 summarises the main parameters used to run the simulations. In particular, the experiments span two southbound protocols (OF and NETCONF), multiple UE population sizes, two RIS densities, and two beam configurations. This combination allows observing how both approaches behave as the number of controlled entities grows and as the RIS beamforming configuration changes.

6.1.2 Simulation Results

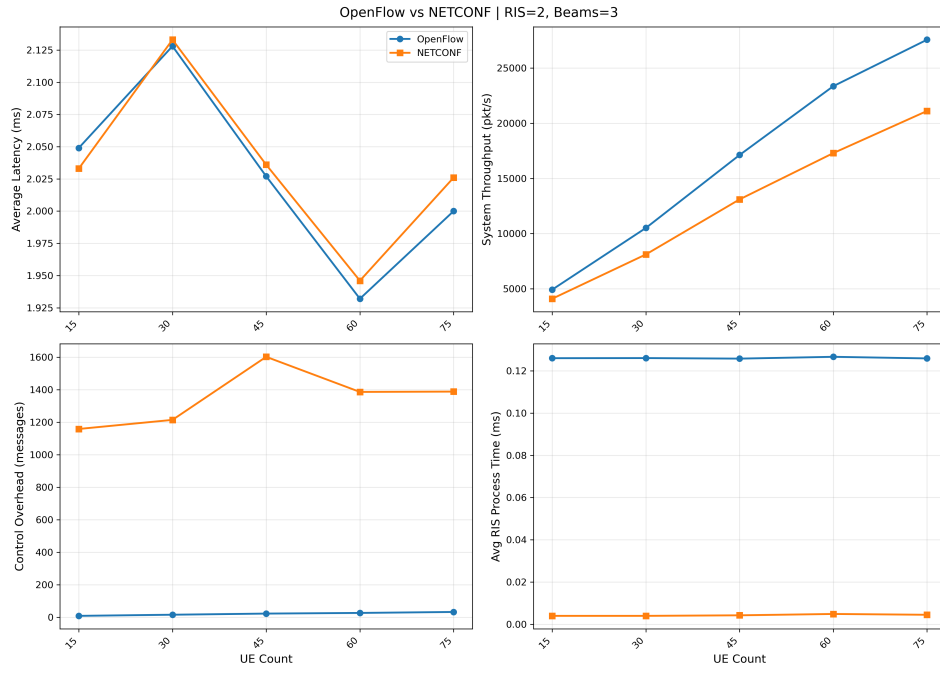


Figure 15: Simulation Results: 2 RISs with 3 beams

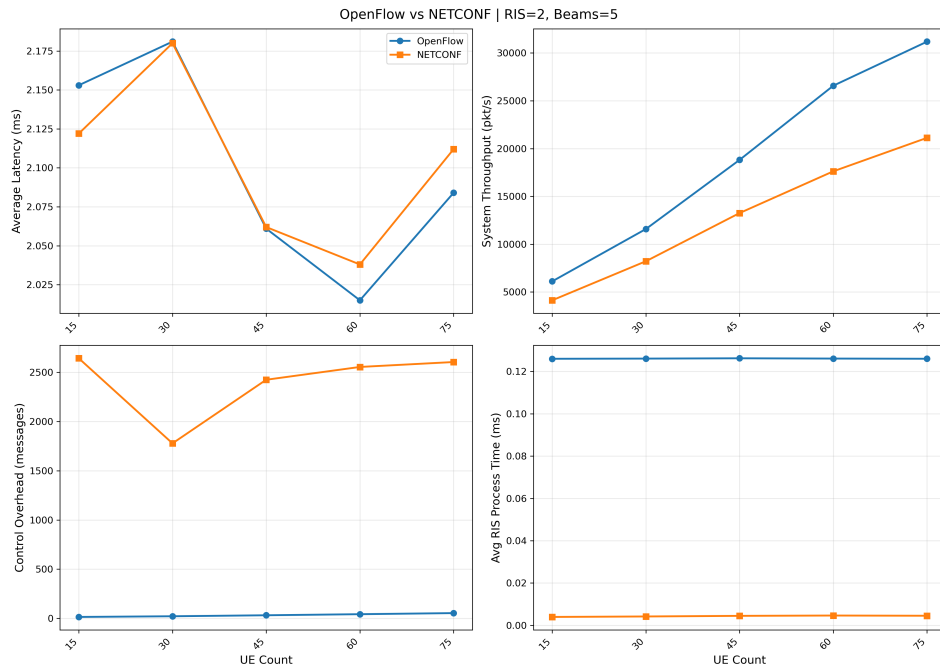


Figure 16: Simulation Results: 2 RISs with 5 beams

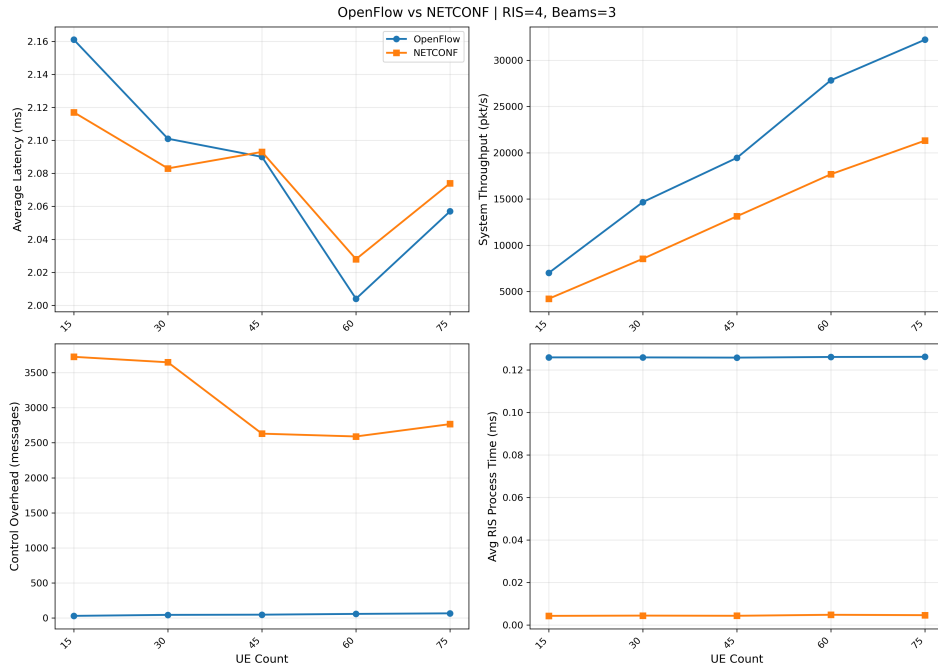


Figure 17: Simulation Results: 4 RISs with 3 beams

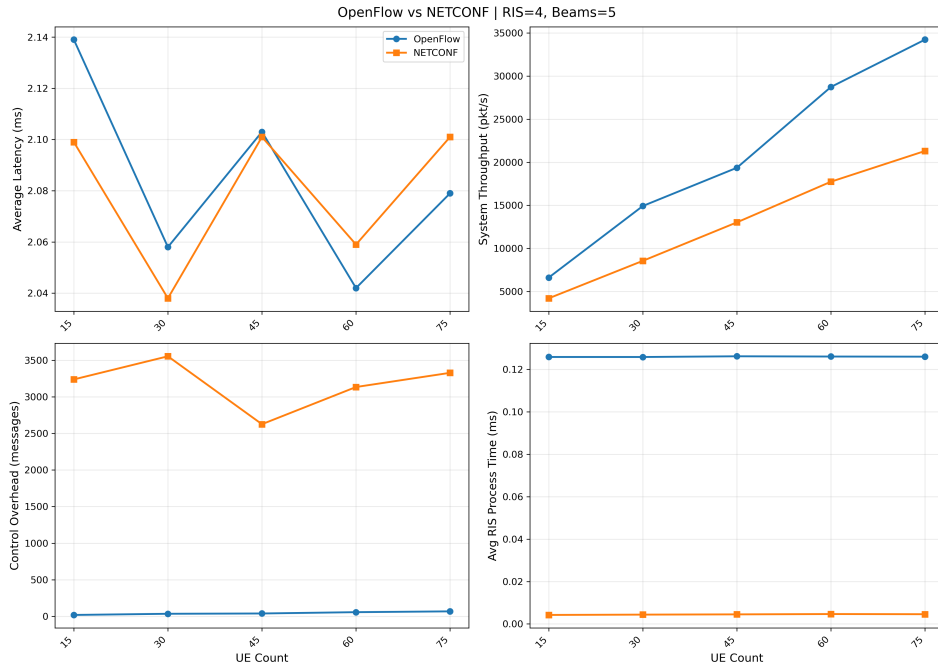


Figure 18: Simulation Results: 4 RISs with 5 beams

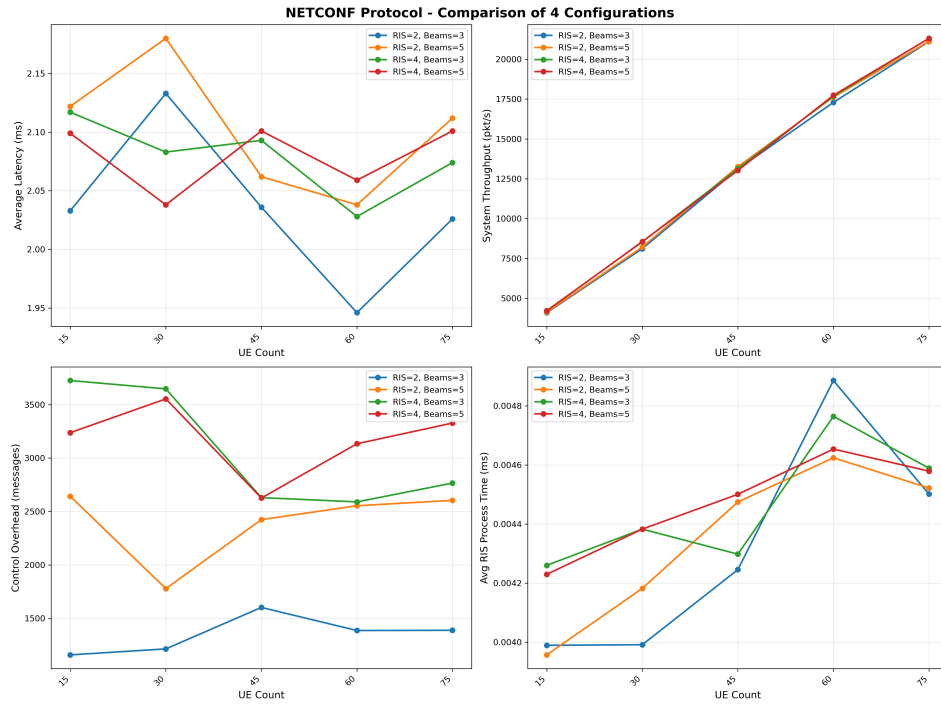


Figure 19: Simulation Results: NETCONF Overview

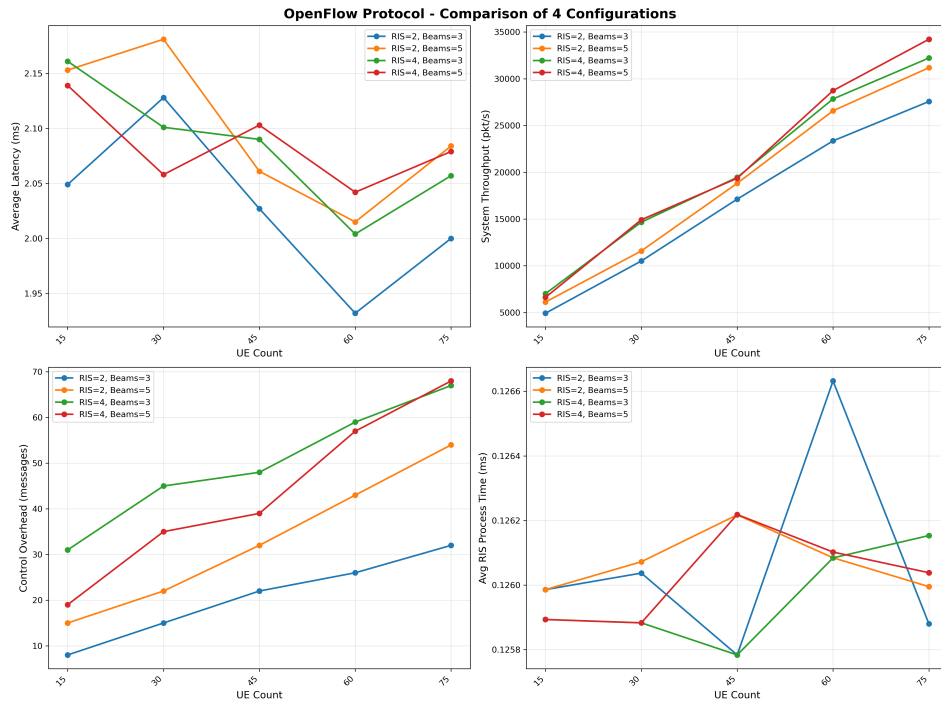


Figure 20: Simulation Results: OF Overview

The analysis of the data extracted from the simulations reveals interesting and complex dynamics regarding the comparison between the NETCONF and OF protocols. Taking into consideration figures 15, 16, 17, 18 and observing the average latency metric across all configurations, the results indicate that the two protocols demonstrate broadly similar performance. However, analyzing the *total throughput* trend, that increase linearly with the number of UEs involved, it emerges that OF manages to guarantee slightly superior performance, although the values measured for NETCONF remain within the same order of magnitude, at least for the considered numbers of UEs.

This performance discrepancy, and in particular the demonstration of non scalability in *throughput* observed for NETCONF, is attributable to the control overhead generated by the exchange of messages of the NETCONF protocol itself within the network. The negative impact of this overhead finds clear confirmation in the graphs relating to the 95th percentile (p95) latency, as shown in figure 21. These data unequivocally demonstrate that NETCONF presents a significantly higher latency in the *worst-case scenario*. It is precisely this considerable increase in latency peaks that consequently causes the overall decrease in *throughput* for NETCONF (and the relative advantage of OF).

Shifting the analysis instead to the *Processing Time* at the level of the single RIS, the situation is reversed: the OF-based simulation records significantly higher times. At a theoretical level, in fact, if a RIS operated via OF, it would be forced to operate according to a packet-based model. This requires the surface to perform the *parsing* of every single packet in transit, necessarily having to inspect and read the *header* fields to act

Complete P95 Latency Analysis: OpenFlow vs NETCONF

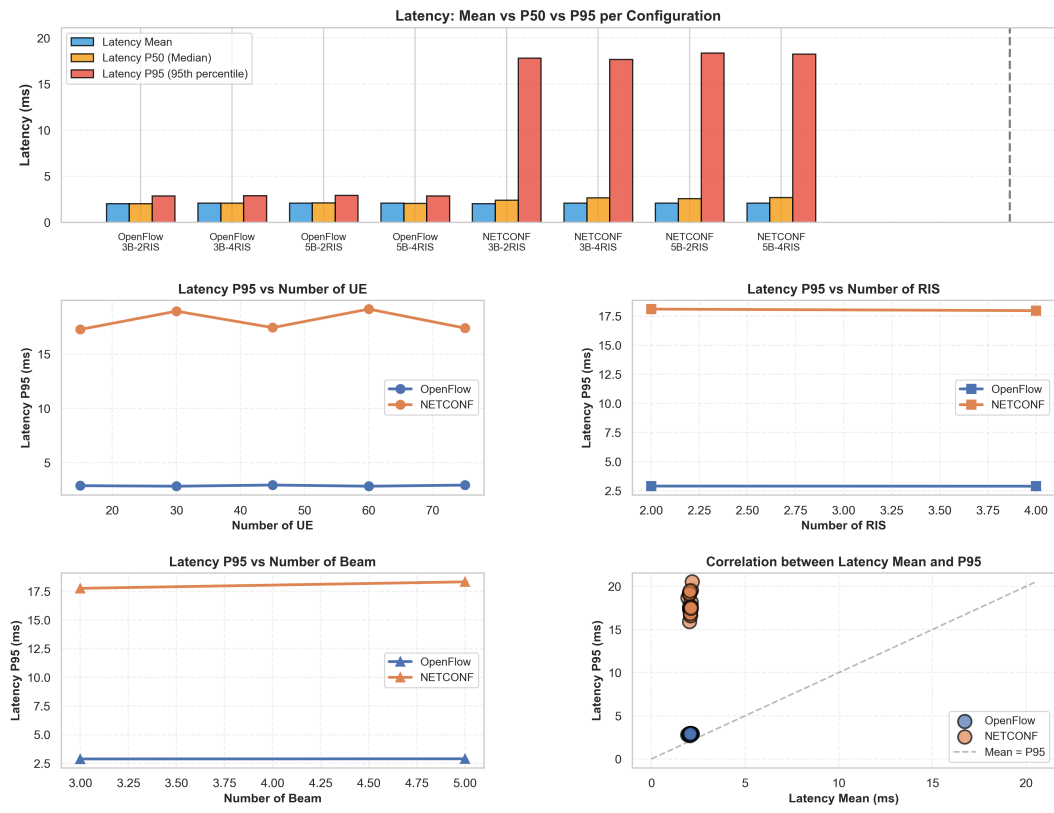


Figure 21: Analysis of 95th percentile of latency

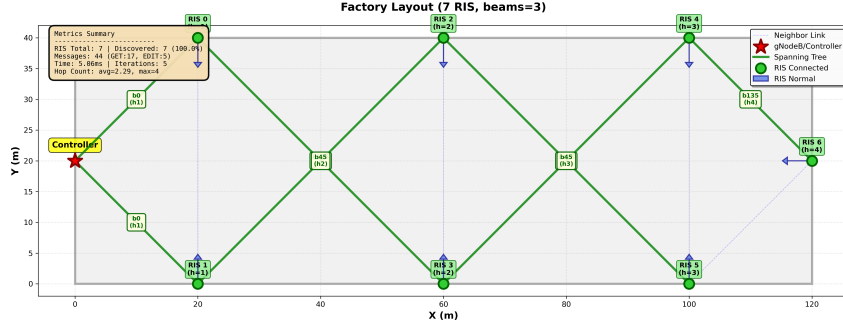
accordingly. These are inherently demanding operations that cause the high processing times recorded for OF.

6.2 RIS iterative discovery protocol

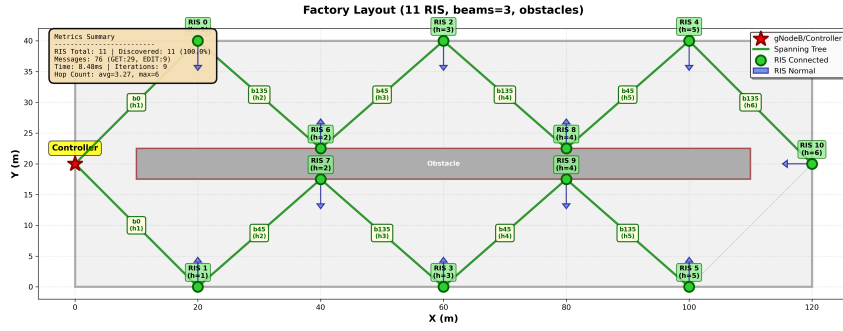
The second set of experiments focuses on the RIS iterative discovery protocol implemented in the simulator. The objective is to evaluate how the discovery procedure scales when the number of RIS nodes increases and when the spatial deployment becomes more challenging. The protocol is executed using real NETCONF interactions within the simulator, and it reports key indicators such as the total number of control messages, the overall discovery time, and the number of protocol iterations required to reach convergence. All the considered configurations are reported in Table 2.

Scenario A: manually constructed baseline In the first environment, the RIS nodes are placed according to a manually constructed, incrementally growing layout. The intent of this scenario is to act as a *baseline* for the protocol: the topology is deliberately designed to be simple, reproducible, and progressively expandable, so that the measured evolution of message cost and discovery time can be interpreted as the *expected* (i.e., idealised) scalability trend of the discovery procedure when the geometry does not introduce unnecessary irregularities. In other words, Scenario A is used to highlight the protocol’s *nominal* asymptotic behaviour under controlled growth, providing a clear reference curve for subsequent comparisons. As summarised in Table 2, Scenario A uses a fixed beam configuration (3 beams) and considers two variants: *no obsta-*

cles (reference case) and *with obstacles*, where an obstruction is added to emulate line-of-sight limitations and to quantify how the baseline trend is perturbed when connectivity constraints are introduced.



(a) Example Layout of Scenario A without obstacles



(b) Example Layout of Scenario A with obstacle

Figure 22: Factory Layout for Scenario A (3^{rd} iteration)

Scenario B: generalisation and stress-test In the second environment, the RIS nodes are distributed at uniform distances along the factory perimeter, so they are not placed manually anymore. This scenario is not meant to “optimise” the topology for the discovery protocol; rather, it is designed to *generalise* the evaluation and to verify whether the baseline trend observed in Scenario A still holds under a broader set of conditions. Concretely, Scenario B acts as a *stress-test*: it introduces layout diversity

Table 2: Numeric parameters used for the RIS iterative discovery protocol experiments.

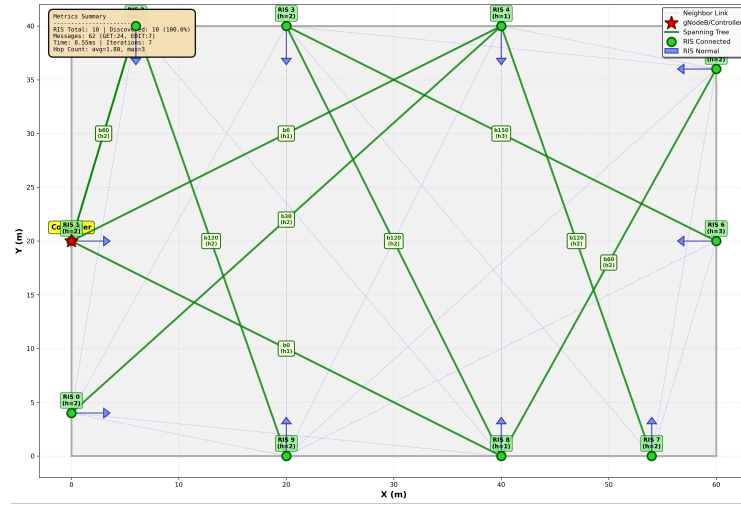
Scenario	Layout / variant	Beams	#RIS values
A	No obstacles	3	$\{3, 5, 7, \dots, 21\}$
A	With obstacles	3	$\{3, 7, 11, \dots, 39\}$
B	Perimeter open	3, 5	$\{3, 4, 5, \dots, 30\}$
B	Single obstacle	3, 5	$\{3, 4, 5, \dots, 30\}$
B	Dual obstacle	3, 5	$\{3, 4, 5, \dots, 30\}$

(open perimeter vs. one obstacle vs. two obstacles, shown in figure 23) and beam diversity (3 vs. 5 beams), while sweeping the number of RIS over a continuous range (Table 2). This allows assessing both robustness and sensitivity when moving from an ideal reference setup to more generic and heterogeneous deployments.

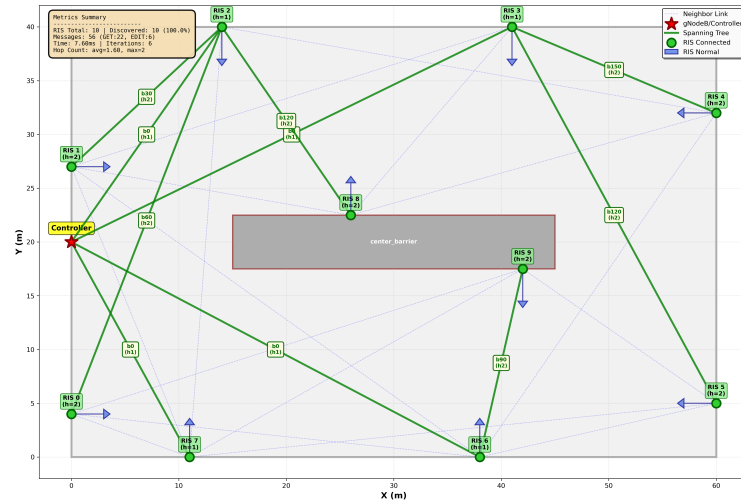
6.2.1 Discovery Performances Results

This section presents the quantitative evaluation of the RIS iterative discovery protocol across the distinct environmental setups previously defined. For each layout and configuration, the protocol’s efficiency and scalability are assessed using three primary metrics: the total number of control messages generated and processed by the SDN controller, the number of algorithmic iterations required to reach full convergence, and the overall execution (or completion) time of the discovery procedure.

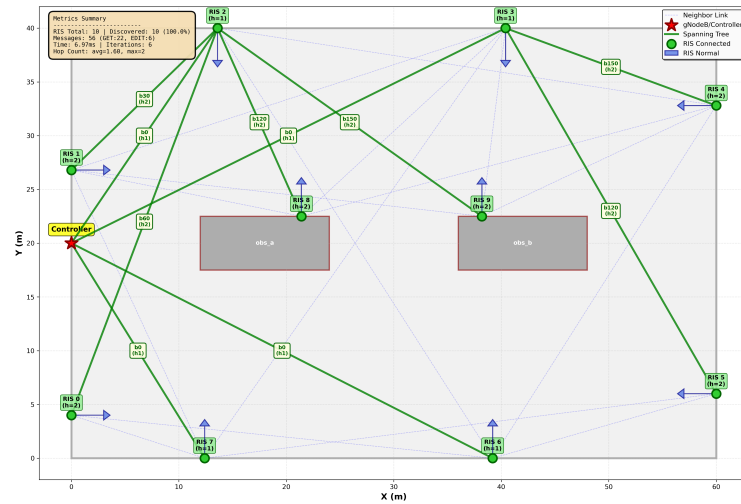
Scenario A: Baseline Performance The results obtained from Scenario A serve as the foundational baseline for the protocol’s behaviour.



(a) Scenario B: Open Perimeter



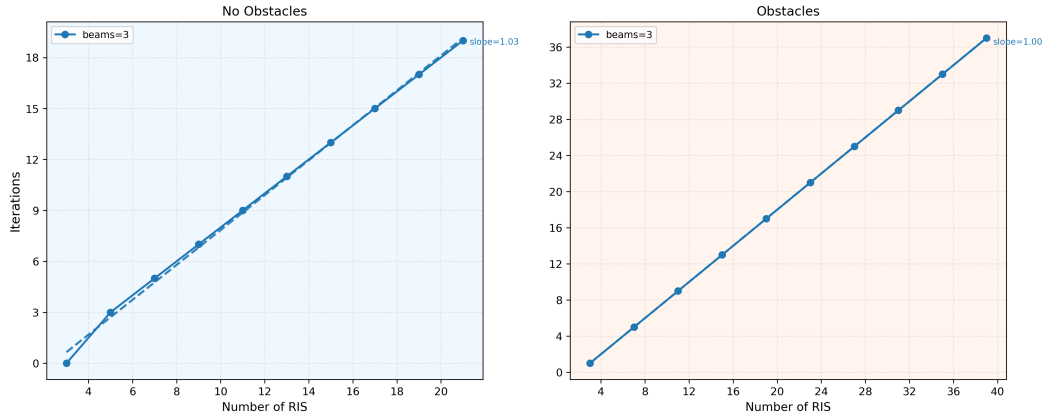
(b) Scenario B: Single Obstacle



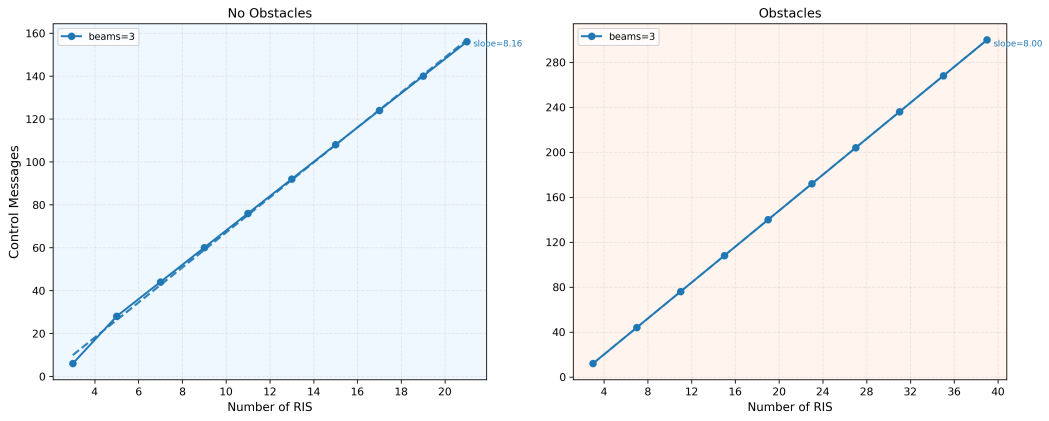
(c) Scenario B: Dual Obstacles

Because the RIS nodes are placed according to a manually constructed and highly regular layout, the performance metrics exhibit an extremely linear trend as the number of RISs increases. As shown in figure 24 both the number of control messages and the completion time scale linearly with number of RISs, demonstrating the inherent algorithmic stability of the discovery procedure. Furthermore, when comparing the reference case (no obstacles) with the constrained case (with obstacles), the results maintain a strikingly similar slope. This confirms that, under controlled and regular geometric growth, the proposed NETCONF-driven discovery behaves predictably and efficiently.

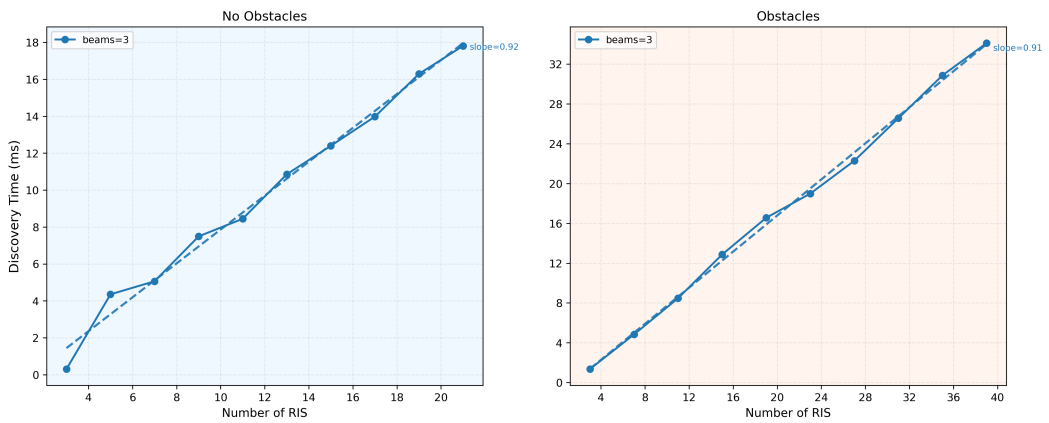
Scenario B: Generalised Deployment and Stress-Test Moving to the stress-test conditions of Scenario B, the RIS nodes are no longer manually placed but are instead uniformly distributed along the factory and obstacles perimeter. This scenario evaluates the protocol against three varying degrees of environmental complexity: an open perimeter (zero obstacles), a single obstacle, and a dual obstacle environment. As visible from figures 25 and 26, despite the introduced layout diversity and the generalisation of the RIS placement, the empirical results confirm that the protocol’s metrics still scale linearly with respect to the number of RIS nodes. However, unlike the perfectly smooth baseline of Scenario A, the performance curves in Scenario B present noticeable local fluctuations, characterized by specific peaks and lows. These variations are a direct consequence of the uniform distribution strategy: depending on the exact number of nodes, certain topological configurations result in particularly favourable (or redundant) RIS placements that makes user



(a) Number of Iterations



(b) Total Control Messages

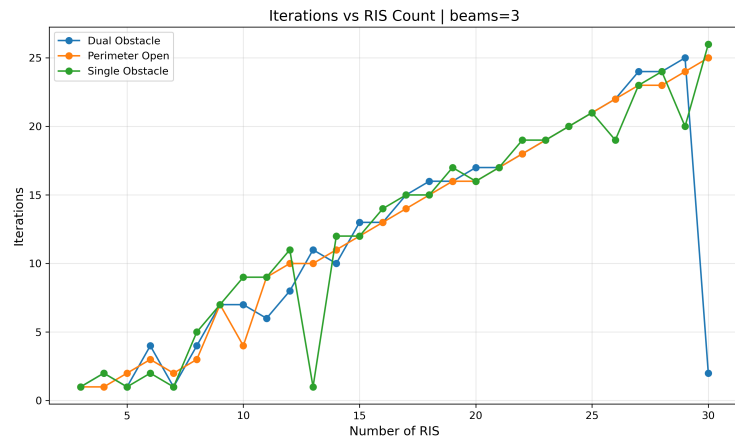


(c) Total Discovery Execution Time (ms)

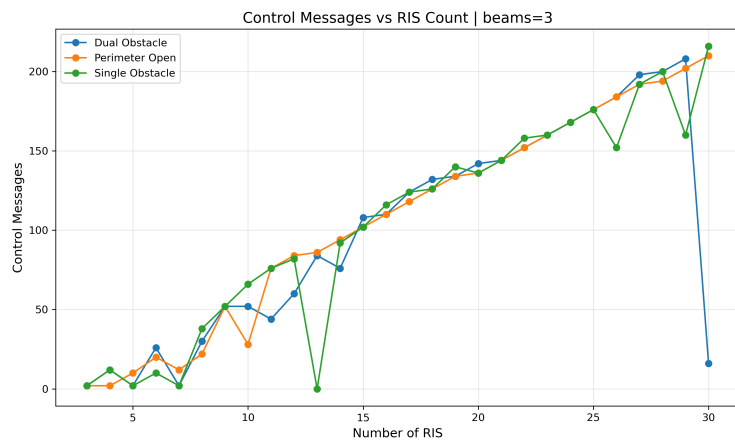
Figure 24: RIS Iterative Discovery Results: Scenario A

discovery slower and requires more iterations, as much more RISs are visible by each other, while others result in blind spots or not aligned RISs scenarios that temporarily decrease the required number of iterations, as the coverage is much lower. Nevertheless, the overarching linear trend firmly validates the robustness and scalability of the iterative discovery algorithm even in generalised, non-ideal industrial deployments. A final insight emerging from the presented results concerns the impact of the internal hardware complexity of the individual RIS. Specifically, the empirical data indicates that varying the number of available reflection beams does not significantly alter the overall performance of the discovery algorithm. The resulting metrics for control overhead, required iterations, and completion time remain remarkably consistent across these variations. This suggests that the procedure’s algorithmic scalability is primarily driven by the macroscopic network topology and the total number of deployed nodes, rather than being bottlenecked by the internal state-space dimension of a single RIS.

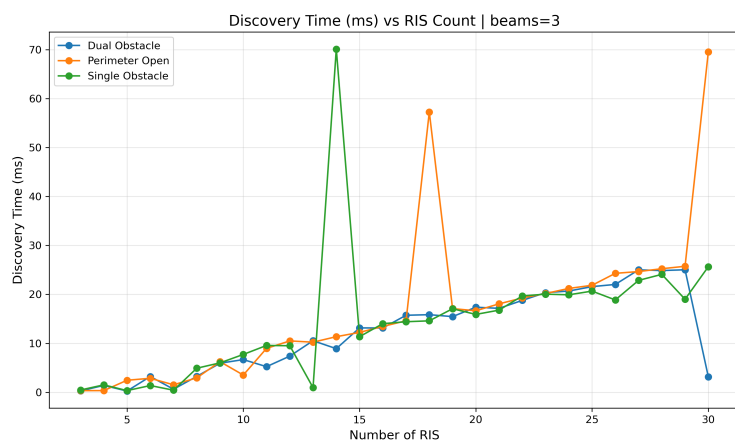
Comparison with Openflow/LLDP Discovery Protocols When comparing the proposed NETCONF Iterative Discovery Protocol with the traditional legacy approach based on OF and LLDP [22], two critical limitations of the latter emerge. First, OF and LLDP intrinsically assume a wired control-plane infrastructure. If this legacy approach is applied to a RIS environment, it implies that each RIS must be physically wired to the controller to exchange discovery messages. This need for wired connections contradicts the main goal of creating a wireless RIS multi-hop network; if a pervasive wired backhaul is already present for ev-



(a) Number of Iterations



(b) Total Control Messages

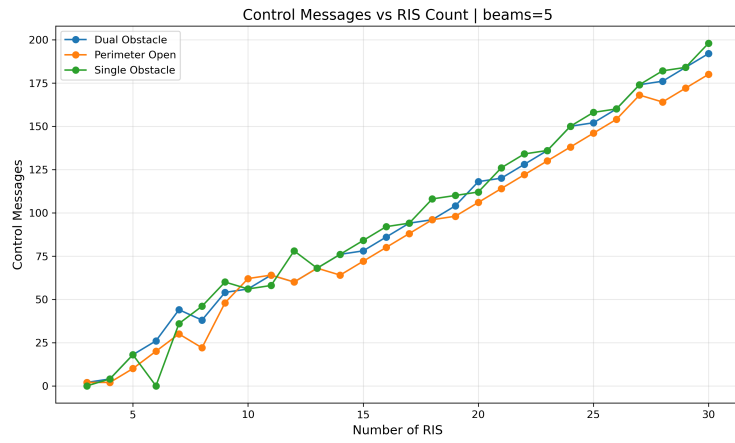


(c) Total Discovery Execution Time (ms)

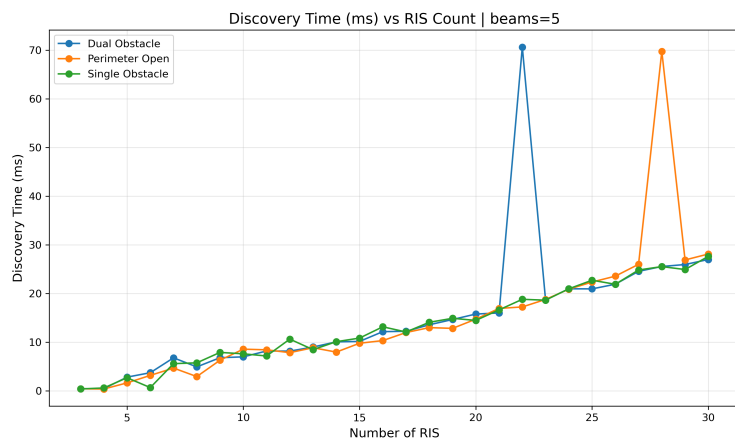
Figure 25: RIS Iterative Discovery Results: Scenario B, RISs with 3 beams



(a) Number of Iterations



(b) Total Control Messages



(c) Total Discovery Execution Time (ms)

Figure 26: RIS Iterative Discovery Results: Scenario B, RISs with 5 beams

ery RIS, establishing wireless multi-hop propagation patterns becomes architecturally redundant and unnecessary. Second, LLDP relies on a flooding-based discovery mechanism. This would require every newly discovered RIS to blindly broadcast LLDP packets across all its possible reflection beams to locate neighboring nodes. As the number of RISs and the amount of available beams per RIS increase, this flooding strategy triggers an explosion of control messages, leading to exponential complexity. In contrast, the proposed Iterative Discovery Protocol avoids both the need for wiring and the inefficiency of flooding, offering a scalable and targeted solution for 6G environments.

7 Conclusions

This thesis explored how to manage and control Reconfigurable Intelligent Surface (RIS) in 6G IIoT networks. Smart factories need highly scalable, reliable, and fast networks, making RISs an essential tool for controlling how radio waves travel. However, managing these semi-passive devices requires a completely new approach.

The main contribution of this work is replacing the traditional OpenFlow (OF) protocol with a NETCONF-based system to manage RISs. OF was built for active switches that read and route individual data packets. This makes it a poor fit for semi-passive RIS panels. Instead, NETCONF offers a better, state-driven alternative by treating the RIS as a physical node with long-lasting configurations.

We validated this new architecture using two methods. First, a Docker-based emulation showed that the NETCONF/Sysrepo stack works well, proving that standard remote procedure calls (RPCs) can safely update RIS settings. Second, we tested its performance on a larger scale using the *restartin-sim* simulator. The results showed that NETCONF creates more control-plane overhead than OF, because the controller actively manages beam steering (whereas OF just installs rules). However, NETCONF drastically reduces the time the RIS spends processing data. With NETCONF, the RIS is pre-configured; with OF, the RIS has to actively read packet headers and check Flow Tables, which takes much longer.

Additionally, we designed and tested an Iterative Discovery Protocol for RIS networks. Simulations showed that this protocol scales efficiently as the network grows, even when obstacles block the line of sight. Import-

tantly, the discovery process worked consistently well regardless of the RIS hardware complexity (such as the number of available beams).

Ultimately, while simulations suggest OF might offer slightly higher throughput in ideal conditions, looking only at data speed ignores real-world hardware limits. OF is meant for active devices like routers that inspect packets one by one. A RIS, however, is a semi-passive surface. It lacks the hardware and memory to read or route digital IP packets; it simply reflects and alters radio waves. Because of this, using a packet-switching protocol like OF on a passive mirror is physically impossible in the real world.

This reality proves that using NETCONF is a physical necessity, not just a way to improve performance. NETCONF focuses on the device's state: using YANG data models, the controller sets the long-term physical behavior of the RIS (like its reflection angles) instead of trying to route the data bouncing off it. This approach perfectly matches what semi-passive hardware can actually do, successfully bridging the gap between SDN management and real 6G radio environments.

Future Works Building on this thesis, there are several exciting paths for future research. The first goal is to move from software simulations to real hardware. For example, the NETCONF server could run on a Raspberry Pi acting as an edge controller for a real RIS prototype (developed in collaboration with the Electronic Engineering Department). This would test our NETCONF-based model against real-world physical limits and actual hardware delays.

Additionally, future work could explore using P4 [23] as a southbound

protocol. While NETCONF is great for setting device states, P4 allows for highly customized programming of how devices handle data. A setup where P4 manages active network nodes and dynamically updates rules so the RISs can execute custom actions as flows arrive could offer extremely fast and precise network adjustments. This would represent an innovative middle ground between the approaches of OF and NETCONF.

Bibliography

- [1] C.-X. Wang, X. You, X. Gao, X. Zhu, Z. Li, C. Zhang, H. Wang, Y. Huang, Y. Chen, H. Haas, J. S. Thompson, E. G. Larsson, M. D. Renzo, W. Tong, P. Zhu, X. Shen, H. V. Poor, and L. Hanzo, “On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds,” vol. 25, no. 2, pp. 905–974. [Online]. Available: <https://ieeexplore.ieee.org/document/10054381>
- [2] A. Dogra, R. K. Jha, and S. Jain, “A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies,” vol. 9, pp. 67 512–67 547. [Online]. Available: <https://ieeexplore.ieee.org/document/9224777>
- [3] C. D. Alwis, A. Kalla, Q.-V. Pham, P. Kumar, K. Dev, W.-J. Hwang, and M. Liyanage, “Survey on 6G Frontiers: Trends, Applications, Requirements, Technologies and Future Research,” vol. 2, pp. 836–886. [Online]. Available: <https://ieeexplore.ieee.org/document/9397776>
- [4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang. The Roadmap to 6G – AI Empowered Wireless Networks. [Online]. Available: <http://arxiv.org/abs/1904.11686>
- [5] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, “6G Internet of Things: A Comprehensive Survey,” vol. 9, no. 1, pp. 359–383. [Online]. Available: <https://ieeexplore.ieee.org/document/9509294>

- [6] H. Tataria, M. Shafi, A. F. Molisch, M. Dohler, H. Sjöland, and F. Tufvesson, “6G Wireless Systems: Vision, Requirements, Challenges, Insights, and Opportunities,” vol. 109, no. 7, pp. 1166–1199. [Online]. Available: <https://ieeexplore.ieee.org/document/9390169/>
- [7] A. Kaushik, R. Singh, M. Li, H. Luo, S. Dayarathna, R. Senanayake, X. An, R. A. Stirling-Gallacher, W. Shin, and M. Di Renzo, “Integrated Sensing and Communications for IoT: Synergies with Key 6G Technology Enablers,” vol. 7, no. 5, pp. 136–143.
- [8] A. H. Shamsan and A. R. Faridi, “SDN-Assisted IoT Architecture: A Review,” in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, Dec. 2018, pp. 1–7, iSSN: 2642-7354. [Online]. Available: <https://ieeexplore.ieee.org/document/8777339>
- [9] R. Yadav, R. Kamran, P. Jha, and A. Karandikar, “Applying SDN to Mobile Networks: A New Perspective for 6G Architecture,” 2024, published: arXiv preprint arXiv:2307.05924. [Online]. Available: <https://arxiv.org/abs/2307.05924>
- [10] W. Long, R. Chen, M. Moretti, W. Zhang, and J. Li, “A Promising Technology for 6G Wireless Networks: Intelligent Reflecting Surface,” vol. 6, no. 1, pp. 1–16.
- [11] P. Putranto, A. A. Hamza, S. Mabrouki, N. Armi, and I. Dayoub. Reconfigurable Intelligent Surfaces for 6G and Beyond: A

- Comprehensive Survey from Theory to Deployment. [Online]. Available: <http://arxiv.org/abs/2506.19526>
- [12] S. Basharat, S. A. Hassan, H. Pervaiz, A. Mahmood, Z. Ding, and M. Gidlund, “Reconfigurable Intelligent Surfaces: Potentials, Applications, and Challenges for 6G Wireless Networks,” vol. 28, no. 6, pp. 184–191. [Online]. Available: <https://ieeexplore.ieee.org/document/9530717>
- [13] F. Chiti, A. Degl’Innocenti, and L. Pierucci, “Secure Networking with Software-Defined Reconfigurable Intelligent Surfaces,” vol. 23, no. 5, p. 2726.
- [14] Y. Liu, X. Liu, X. Mu, T. Hou, J. Xu, M. Di Renzo, and N. Al-Dhahir, “Reconfigurable Intelligent Surfaces: Principles and Opportunities,” vol. 23, no. 3, pp. 1546–1577.
- [15] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A Survey on Software-Defined Networking,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.
- [16] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, “Software-Defined Networking (SDN): Layers and Architecture Terminology,” Internet Engineering Task Force, Request for Comments RFC 7426, Jan. 2015, num Pages: 35. [Online]. Available: <https://datatracker.ietf.org/doc/rfc7426>
- [17] *OpenFlow Switch Specification (Version 1.5.1)*. Open Networking Foundation, Mar. 2015.

- [18] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, “Network Configuration Protocol (NETCONF),” Internet Engineering Task Force, Request for Comments RFC 6241, Jun. 2011, num Pages: 113. [Online]. Available: <https://datatracker.ietf.org/doc/rfc6241>
- [19] J. Edelman, S. S. Lowe, and M. Oswalt, “Network Programmability and Automation.”
- [20] Y. Zhang, “Network Function Virtualization.”
- [21] C. Liaskos, L. Mamatras, A. Pourdamghani, A. Tsioliariidou, S. Ioannidis, A. Pitsillides, S. Schmid, and I. F. Akyildiz, “Software-Defined Reconfigurable Intelligent Surfaces: From Theory to End-to-End Implementation,” *Proceedings of the IEEE*, vol. 110, no. 9, pp. 1466–1493, 2022.
- [22] F. Chiti, M. Lotti, S. Picchioni, and L. Pierucci, “SDN-Oriented 6G Industrial IoT Architecture Design and Application to Optimal RIS Placement and Selection,” *Sensors*, vol. 26, no. 2, p. 411, Jan. 2026. [Online]. Available: <https://www.mdpi.com/1424-8220/26/2/411>
- [23] “P4~16~ Language Specification.” [Online]. Available: <https://p4.org/wp-content/uploads/sites/53/2024/10/P4-16-spec-v1.2.5.html>

Ringraziamenti

Desidero ringraziare in primo luogo il prof. Francesco Chiti, per la stima che mi ha dimostrato e per avermi dato l'opportunità di approfondire ed esplorare nuovi argomenti e nuove possibilità. Ci tengo poi a ringraziare anche la prof. Laura Pierucci e la dott. Sara Picchioni, per il sostegno e la collaborazione dati in questi mesi di lavoro insieme.

Ci tengo poi a ringraziare tutte le persone che in qualche modo mi hanno aiutato e accompagnato durante questo percorso. Non sono una persona di molte parole, come sapete; se vi ringrazio, lo faccio perché penso veramente che, in tempi e modi diversi, siete stati importanti per me.

Ringrazio quindi, prima di tutti, i miei genitori, perché non mi hanno fatto mai mancare nulla, mi hanno dato la possibilità di studiare, e il sostegno per farlo. Grazie per i sacrifici che avete fatto.

Ringrazio poi mia sorella Alice, per essere stata un punto di riferimento costante, per aver tracciato la strada e avermi accompagnato con lei, e per farmi fare tardi ad ogni singolo evento a cui dobbiamo andare insieme.

Ringrazio poi tutta la mia famiglia, gli zii e i cugini per essere sempre presenti e disponibili, e in particolar modo i miei nonni, che ho la fortuna di avere ancora tutti con me. Grazie per gli insegnamenti che mi date e l'affetto che continuate a dimostrarmi.

Voglio ringraziare poi i miei compagni di università, per aver collaborato e essere stati di supporto nell'affrontare questi due anni di magistrale. Avete alleggerito i momenti più pesanti e siete stati capaci di dare

una mano o un consiglio quando ne avevo bisogno. Grazie davvero.

Ringrazio poi i Fratellocci e le Sorellocce. Siete come una famiglia allargata per me, ci conosciamo da prima di nascere e ci siamo sempre nei nostri momenti importanti. Vi voglio bene.

Voglio ringraziare anche chi è con me dai tempi della scuola: Eleonora e Clelia, che anche se non ci vediamo spesso mi dimostrano sempre il loro supporto. E, soprattutto, Radu e Marina: grazie per tutti i momenti che passiamo insieme, per aggiornarci sempre su come vanno le nostre vite, per le nostre "giornate studio", per le chiacchierate, per le sessioni di gioco, per aver esplorato ogni tipologia di votazione su dove pranzare. Siete veri amici.

Ringrazio poi la mia squadra, la Scuola Basket Prato. Grazie perché durante gli anni di studio siete stati e siete le persone con cui per alcuni momenti della settimana ci si può sentire parte di qualcosa, con cui si condivide un impegno da portare fino in fondo. Grazie perché con voi ho imparato che si vince (poco) insieme e si perde (tanto) insieme. E grazie perché anche nei momenti più difficili per qualsiasi motivo, c'è sempre modo di alleggerire con una cena insieme. Forza SBP! In particolare, ci tengo a ringraziare Simone e Giulio, che sono sempre disponibili e pronti ad ascoltarmi nel momento del bisogno e a tirarmi su nei momenti no. Grazie ragazzi.

Voglio ringraziare poi i miei amici di sempre: Daniele, Leonardo, Elisa, Caterina, Marta, Giulia. Grazie per esserci sempre e da sempre. Grazie per i consigli che mi date, le chiacchierate serie e meno serie, la voglia che avete di passare del tempo insieme solo per il piacere di farlo. Grazie perché passare quella serata ogni settimana con voi è sempre un momento

speciale. Spero di passarne tante altre con voi, con l'augurio di rimanere sempre, indecisi come siamo, "da bosco e da riviera". Vi voglio tanto bene.

Infine, grazie a te, Silvia. Grazie per la stima che hai sempre nei miei confronti e per avermi trasmesso sempre la forza di contare su di me. Grazie perchè ci sei stata nei momenti più belli, e grazie perché hai avuto la forza e il coraggio di starmi accanto anche nei momenti più difficili. Grazie per il tuo supporto e per avermi permesso di vivere le mie cose con te.

Mollare, non mollare... Spaghetti, non spaghetti... Ti preoccupi troppo per ciò che era e ciò che sarà. C'è un detto: ieri è storia, domani è un mistero, ma oggi... è un dono. Per questo si chiama presente.

Maestro Oogway

L'ingrediente segreto è... niente!

Mr. Ping