Department of Computer Science

UNIVERSITY of York

Submitted in part fulfillment for the degree of MEng

Securing Matter IoT Networks with Role-Based Network Segmentation: A Performance and Enforcement Study

Zubair Ahmed Shaik

02 May 2025

Supervisor: Dr. Poonam Yadav

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor, Dr. Poonam Yadav, whose expert guidance and constructive feedback were invaluable throughout this project, and to, Peter Gillingham, for their insightful suggestions and technical assistance. Heartfelt thanks go to my girlfriend, for her unwavering encouragement. I am also grateful to my family, my Mom, Dad and Sister, for their constant support and belief in me.

STATEMENT OF ETHICS

This study adhered to the highest standards of research integrity and ethical conduct. All experimental data were generated using synthetic traffic in a closed Docker-based testbed; no human subjects or personally identifiable information were involved, there is no use of any external dataset. Firewall rules and logging mechanisms were applied solely to simulated IoT devices, ensuring no real-world systems were impacted. All software configurations and scripts developed for this work are made publicly available for full transparency and reproducibility. The analysis was conducted impartially, and any limitations or negative findings have been reported fully to provide an honest appraisal of RBNS performance.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

EXECUTIVE SUMMARY

Matter standardizes secure onboarding and interoperability for IoT devices, but its protections focus on device–controller (north–south) channels. Once commissioned, devices can communicate freely with one another inside the network, leaving east–west traffic unregulated.

Unrestricted east–west flows create a pathway for lateral-movement attacks: a single compromised node can probe or exploit devices in other zones. Traditional segmentation methods—VLANs, micro-segmentation, zero-trust overlays, and MUD—either lack fine control, impose high operational overhead, or rely on static vendor profiles.

The study measures how Role-Based Network Segmentation (RBNS)—simple firewall rules that isolate admin, operational, and guest zones—affects network performance. The goal is to determine whether RBNS can block unauthorized traffic without unacceptable penalties in throughput or latency.

A Docker-based emulation recreates a three-zone Matter topology with Alpine-based mock devices and an iptables router. Nine experiments vary device scale, stream count, and policy profile. Each scenario runs three times, sampling 200 randomized flows per run to provide empirical basis for all findings; results are summarized with median and interquartile range.

RBNS blocks 100 % of unauthorized cross-zone flows, preserves 85–90 % of legitimate throughput, and adds < 1.5 ms latency even under high load. Logging overhead is modest, ≈ 0.3 % RTT (round trip time) increase per 500 log entries—and throughput degradation under heavy load stems mainly from NIC (network interface card) saturation, not rule processing.

RBNS closes Matter's east–west security gap with minimal performance cost, making it a practical defense for commercial IoT deployments. Future research should integrate dynamic MUD profiles, test larger and wireless-enabled topologies, and validate results on physical hardware.

1 Introduction

The Internet of Things (IoT) has transformed daily life by enabling seamless integration of devices, from smart homes adjusting temperatures automatically to industrial systems optimizing manufacturing processes.

Despite its benefits, the increase of diverse IoT devices has led to challenges in interoperability and security. To address these issues, the Connectivity Standards Alliance introduced Matter, an open-source, IP-based connectivity standard designed to unify the fragmented IoT landscape. Matter facilitates seamless communication among devices from various manufacturers and incorporates security features such as end-to-end encryption and secure device onboarding [3].

However, while Matter's security measures protect against external threats, once an endpoint (device) is compromised, existing protections do not prevent an attack from laterally moving to other segments, a risk underscored by the prevalence of deep lateral movement threats in operational technology environments [4]. This vulnerability is particularly concerning in commercial settings like healthcare and manufacturing, where unauthorized access can lead to significant operational disruptions and safety risks [5].

To mitigate such risks, Network Segmentation has emerged as an effective strategy. Network Segmentation [6] divides a computer network into smaller, isolated sub-network to control how traffic flows between distinct zones and limit an attacker’s ability to move laterally once inside the network [7]. Segmentation enhances both performance and security in most cases, ensuring that a breach in one segment cannot easily propagate to others. Micro segmentation applies this concept at a finer granularity – down to individual workloads or devices, providing even tighter control over traffic [8].

Role-based Network Segmentation (RBNS) a type of Network Segmentation groups devices into discrete zones based on their functional roles, then enforces a minimal set of inter zone ACLs or firewall rules to grant only the traffic necessary for each zone, embodying the principles of least privilege [7]. Unlike VLANs, which isolate broadcast domains yet allow unfettered east-west flows within each VLAN, or micro and zero-trust schemes. RBNS delivers coarse grained but semantically clear isolation at line rate with very low configuration and runtime overhead [9].

Given these considerations, this project aims to quantitatively assess the performance trade-offs of enforcing Role-Based Network Segmentation (RBNS) in a Docker-based Matter IoT testbed.

Specifically, the objectives are to Measure latency overhead introduced by RBNS under both low-load and high-load conditions, quantify throughput impact, comparing median and tail performance against a fully permissive baseline and evaluate enforcement accuracy, verifying that unauthorized internal network traffic are dropped.

To achieve these goals the Background and Literature Review section reviews key IoT challenges, the Matter protocol's capabilities and gaps, and alternative segmentation approaches (including MUD, VLANs, micro segmentation, and zero-trust), culminating in the rationale for RBNS.

The Methodology and Implementation section describes the controlled emulation methodology: The simulated network topology, rules and details about tests carried out. And defines the success criteria.

The Results and Analysis presents the results on throughput, latency CDF (Cumulative distribution Function) tail behaviour, scalability with stream count, enforcement accuracy, and logging overhead, each evaluated against predefined success criteria.

The Conclusion discusses the implications of the findings for commercial IoT deployments and outlines directions for further research, including physical-testbed validation and dynamic policy integration.

2 Background and Literature Review

This section reviews the key foundations and state of the art underpinning this study. First, it examines the pervasive security challenges in IoT environments—driven by device heterogeneity, weak authentication schemes, unencrypted communications, and fragmented update practices—which collectively expand the attack surface and expose devices to sophisticated threats [10]. It then reviews the Matter protocol—an open, IP-based connectivity standard that enhances interoperability and secures north–south device–controller communications via certificate-based attestation and encryption [3], yet does not restrict east–west device-to-device flows, leaving lateral-movement vulnerabilities unaddressed [11]. Next, the IETF's Manufacturer Usage Description (MUD, RFC 8520) is introduced as a per-device, automated segmentation mechanism—enabling devices to signal intended access patterns to generate ACLs—while noting that its static policy definitions and TCAM resource demands may limit adaptability at scale [12]. The review then explores network segmentation strategies, from VLAN-based isolation, to micro segmentation that enforces host-level policies via agents at significant overhead [13], and zero-trust overlays requiring pervasive authentication and arrangements [14]. Comparing their granularity, management complexity, and performance trade-offs. Finally, Role-Based Network Segmentation (RBNS) is positioned as a practical mid-level model that groups devices into functional zones and applies minimal inter-zone firewall rules to realize least-privilege access with low per-packet processing overhead, setting the stage for the subsequent empirical evaluation [9].

2.1 Security Challenges in IoT Environments

The rapid increase in IoT devices has exposed fundamental security weaknesses, principally due to device heterogeneity and constrained resources [15]. Many devices lack robust authentication, rely on unencrypted communications, and suffer from infrequent or insecure firmware updates, creating fertile ground for exploitation. Furthermore, the absence of unified security standards across vendors leads to a fragmented security landscape, amplifying the overall attack surface of IoT networks. These pervasive vulnerabilities necessitate protocol-level and network-level defences to safeguard mission-critical deployments [16]. Addressing these fundamental vulnerabilities requires standardized protocols designed with inherent security – such as the Matter protocol.

2.2 Matter Protocol: Architecture & Security

The Matter protocol, developed by the Connectivity Standards Alliance, provides a unified, IP-based application layer over Wi-Fi, Ethernet, and Thread, incorporating Device Attestation Certificate (DACs) for security onboarding and end -to-end encryption of application payloads. During commissioning, Matter devices mutually authenticate via certificate chains and establish session keys, reducing reliance on network-layer trust while solving interoperability issues across vendors [11].

However, Matter's security model focuses on north-south (in and out of the network) communications and does not mandate east-west (internal network) traffic restrictions once a device is onboarded, allowing an authenticated node to communicate laterally with any peer unless additional control are applied [11]. In practice, many Matter deployments use flat LANs or default ACLs that permit all IPv6 traffic between devices, a compromised node can therefore "hop" across the network, exploiting valid credentials to discover and attack other zones [4].

Current research underscores this lateral-movement risk. The "Trust Matters" analysis revealed that DAC private-key cloning could enable a rogue device to masquerade as a genuine Matter endpoint and intra-network reconnaissance [17] Likewise, operation technology studies demonstrate how attackers traverse seemingly isolated security perimeters, Lateral movement – to breach controllers once inside the network [4]. Furthermore, commissioning over Wi-Fi remains vulnerable to Evil-Twin attacks, allowing an adversary to intercept control credentials and further facilitate lateral movement [11].

Because Matter lacks built-in network-layer segmentation, complementary measures such as Manufacturer Usage Description (MUD) can restrict permitted communication patterns but depend on device-vendor compliance and do not address dynamic threat scenarios [11]. To enforce the principle of least privilege at the network level – limiting unauthorized east-west flows – Network Segmentation is introduced, assigning devices to zones and applying firewall rules that block cross-zone traffic by default [4]. This strategy is the focus of this project's performance evaluation

2.3 Manufacturer Usage Description (MUD)

Manufacturer Usage Description (MUD) is an IETF standard (RFC 8520) enabling IoT devices to express their intended network behaviour is a signed, machine-readable profile. When a device connects, the network retrieves its MUD file, typically hosted by the manufacturer and automatically generates ACLs to permit only the traffic required for that device’s functionality, enforcing least-privilege access without per device agents. MUD simplifies segmentation in heterogeneous environments, but relies on vender adoption, static policy definitions, and sufficient switch resources to scale effectively [12].

2.3.1 MUD Architecture and Operation

MUD specifies a component-based architecture whereby each device carries a uniform resource identifier (URI) in its DHCP or LLDP packet, pointing to a JSON file that describes its network ACLs [12]. A MUD controller or switch fetches this file and translates its “access-lists” into enforcement rules on iptables, OpenFlow or programmable hardware, thereby automating policy deployment without manual ACL configuration.

2.3.2 Benefits for IoT Segmentation

By providing automatic ACL generation, MUD addresses the visibility gap in unmanaged IoT deployments. Networks learn what each device should and should not do, eliminating unknown flows at the edge [18]. The standard’s focus on device-initiated signalling ensures that segmentation policies stay aligned with vendor defined intent, reducing operator error and simplifying compliance with security frameworks such as NIST SP 1800-15B [19] .

2.3.3 Limitations

MUD profiles depend on manufacturer providing accurate and comprehensive descriptions. Incomplete or missing MUD files force fallback to permissive ACLs, undermining security [18].

MUD’s ACLs are static snapshots, they cannot adapt to dynamic threat scenarios or time of day requirements without supplemental arrangements [20].

2.3.4 Relation to Network Segmentation

While MUD focuses on per-device intent, Network segmentation groups devices by zones or functions, and applies a minimal common policy per zone [6]. Network Segmentation complements MUD by providing coarse grained zone enforcements where per-device profiles are unavailable or impractical, creating a balance between automation and operational simplicity for commercial IoT networks.

2.4 Network Segmentation

This section reviews different approaches to internal network traffic control and a comparative analysis of these approaches before identifying the gap this study addresses.

2.4.1 VLAN-based Segmentation

Network segmentation divides a larger network into smaller, isolated subnetworks to limit lateral movement and reduce attack surfaces [6]. VLANs (Virtual Local Area Network) implement segmentation by tagging traffic at the switch level, logically grouping devices regardless of physical location [23]. While VLANs improve security and performance by containing broadcasts and isolating traffic, they remain coarse-grained, often allowing wide east-west communication within each VLAN and they require complex switch configuration and VLAN trunking agreements across infrastructure [24].

2.4.2 Micro segmentation

Micro segmentation provides fine-grained controls by enforcing policies at the individual workload or host level using techniques such as host-agent firewalls or hypervisor rules to secure east-west traffic even within the same VLAN. This approach aligns with Zero Trust principles by verifying every flow, but it introduces significant management complexity and computational overhead, as each endpoint or virtual host must evaluate granular policies [13]. For resource constrained IoT devices, ship-ready agents or software defined networking (SDN) controllers may exceed device capabilities and complicate deployments.

2.4.3 Zero Trust Network Segmentation

Zero Trust segmentation extends the “never trust, always verify” model across the network, requiring authentication and authorization for every connection [14]. Often implemented via micro segmentation plus continuous monitoring, zero-trust environments minimize implicit trust but demand robust identity systems, telemetry, and orchestration platforms raising both cost and operation burden [14].

2.4.4 Role-Based Network Segmentation (RBNS)

RBNS assigns devices roles according to zone/function (e.g. admin, operational, guest) and enforces inter-zone policies with simple firewall rules. By grouping similar role devices, RBNS implements least-privilege access at the network layer with minimal per-packet processing, requiring only basic ACLs rather than complex policy engines [25]. This simplicity eases compliance, reduces configuration mistakes and avoids the runtime overhead that micro segmentation or zero-trust orchestration can incur [25].

2.4.5 Comparison between the different Network Segmentation strategies

VLANs and RBNS both rely on well-understood ACLs; RBNS adds semantic clarity by mirroring organizational roles, while micro segmentation and zero trust require ongoing policy refinement and specialized tooling [23].

RBNS firewall rules execute at line rate on commodity routers or edge devices, adding sub-millisecond latency even at high load, whereas micro segmentation and zero trust proxies can introduce multi millisecond delays per packet [13].

RBNS scales by simple device-zone mapping. In contrast, micro segmentation's host-agent models demand per device policy distribution and zero trust's mutual authentication meshes grow quadratically with device count [6].

By combining low operational overhead, clear policy semantics, and strong protection against lateral movement, RBNS offers the optimal balance of security and performance for commercial IoT, ensuring critical devices remain isolated without taxing constrained endpoints or network infrastructure.

Most commercial organizations already operate VLANs or basic ACLs, RBNS adds semantic clarity by usually mirroring organizational roles while maintaining line-rate enforcement on commodity hardware. Although existing methods – VLANs, micro-segmentation, zero-trust, and MUD – offer trade-offs between granularity and overhead and there is not empirical evaluation of RBNS performance within a Matter-based IoT context.

This project fills that gap by presenting a systematic performance evaluation of Role-Based Network Segmentation (RBNS), measuring latency, throughput, packet loss and logging overhead across different policy and load scenarios. By quantifying the trade-offs between security enforcement and network performance. This study provides actionable insights for deployment RBNS in commercial IoT environments without compromising operational responsiveness.

3 Methodology and Implementation

A container-based testbed emulates a three-zone Matter IoT network—administrative, operational, and guest—to measure how role-based segmentation affects performance. Virtual devices and a policy-enforcing router run in Docker, allowing for quick policy swaps and repeatable tests while maintaining environmental isolation. Nine experiments vary network scale, traffic load, and policy strictness, with automated scripts generating and measuring flows. This framework delivers statistically sound insights into RBNS trade-offs.

3.1 Research Paradigm and Experimental Design

The project adopts a controlled emulation paradigm—reproducing a Matter IoT network in Docker—to systematically isolate variables and ensure reproducibility. Guided by the zones and conduits model of ISA/IEC 62443 [26], three independent variables were selected: device count (2 vs. 32 per zone), stream count (1 vs. 4 per device), and policy profile (Permissive, Selective, Selective+Logging). The nine resulting scenarios (See Table 1) enable a factorial design, with each scenario executed in three iterations to distinguish systematic effects from random variation.

3.2 Environment & Network Segmentation Setup

A Docker Compose manifest defines three custom bridge networks—`admin_net` (172.26.0.0/16), `factory_net` (172.24.0.0/16), `guest_net` (172.25.0.0/16)—reflecting the administrative, operational, and guest roles. Each Alpine Linux container representing a mock IoT device attaches to its zone network. A router container, granted CAP_NET_ADMIN, connects to all three networks and enforces RBNS policies via iptables:

1. Default DROP: `iptables -P FORWARD DROP` blocks all east–west traffic by default.
2. Permissive: Switches to `-P ACCEPT` for baseline measurements.
3. Selective: Adds `-A FORWARD -s <srcCIDR> -d <dstCIDR> -j ACCEPT` rules for allowed flows.

4. Selective+Logging: Inserts `-j LOG --log-prefix "[DROP]"` before each DROP to capture blocked-packet counts.

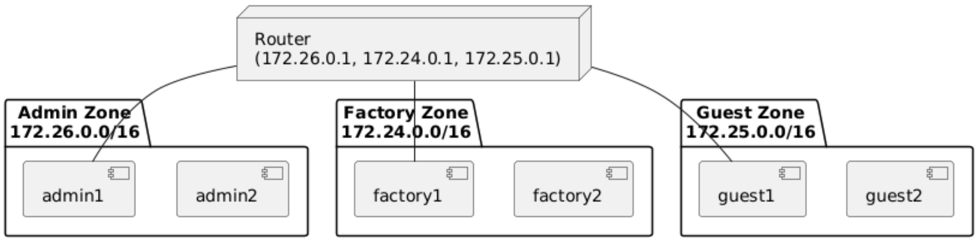


Figure 1. Network Topology of baseline scenario

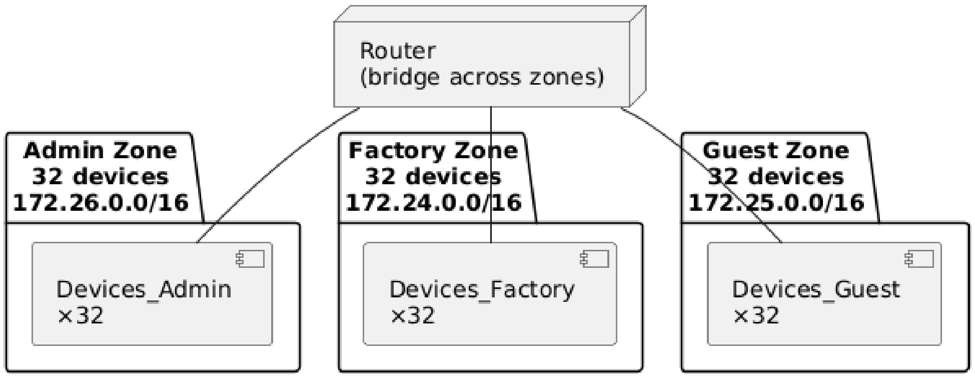


Figure 2. Network Topology of scaled scenario

A custom reload script (See 6.1.6) atomically swaps rulesets without container restarts, following containerized firewall best practices. For this setup Selective rules allow all traffic from Admin zone, and only traffic to Guest zone from Factory zone (operational zone).

Alpine based containers were used to mock Matter devices, that had just port 5540 exposed.

3.3 Traffic Generation and Measurement Techniques

iperf3 is employed to generate TCP flows and measure throughput, retransmits, and packet loss in a client–server model, leveraging its widespread use in container benchmarking [27]. ICMP ping probes collect RTT samples. A python script (network-benchmark.py See 6.1.7) automates:

1. Device discovery via Docker APIs.
2. Random selection of 200 source-destination flows per run (scaled)

3. Execution of 1-4 (low-high) parallel streams per flow

Selective scenarios use a –cross-zone-only flag to focus on east-west flows, directly evaluating RBNS effectiveness.

Measurements included:

1. Round Trip Time (RTT
2. Bandwidth
3. Retransmission rate
4. Packet Loss
5. Packet count
6. Total bytes of Packets

These metrics were systematically recorded into structures CSV an plain-text files for subsequent analysis.

3.4 Automated Testing Campaign

An automated testing script (run-all.sh) facilitated consistent and repeatable experimental execution across nine predefined scenarios, detailed below:

Table 1. Summary of Experimental Scenarios

Scenario	Policy Profile	Devices/Zone	Streams	Workers	Additional Flags
Baseline	Permissive	2	1	6	None
Baseline Selective	Selective	2	1	6	None
Baseline Selective Log	Selective + Log	2	1	6	None
Scaled Low Load	Permissive	32	1	16	None
Scaled High Load	Permissive	32	4	16	None

Scaled Selective Low	Selective	32	1	16	--cross-zone-only
Scaled Selective High	Selective	32	4	16	--cross-zone-only
Scaled Selective Low Log	Selective + Log	32	1	16	--cross-zone-only
Scaled Selective High Log	Selective + Log	32	4	16	--cross-zone-only

Overview of device counts, traffic streams, worker processes, policy profiles, and flags used in each test scenario.

Each scenario generated 600 flow results (3 runs × 200 flows).

3.5 Data Processing & Statistical Analysis

Raw logs are filtered to remove iperf3 entries reporting 0 Mb/s with 0% loss, artifacts of test harness timing. Metrics are aggregated by median and interquartile range (IQR), with mean ± σ for contextual comparison.

3.6 Success Metrics

The following success criteria were defined based on practical and resource constraints. Specifically, acceptable overhead is set as: ≤20% throughput reduction, ≤1 ms additional RTT under scale, full enforcement of segmentation policies.

3.7 Ethical and Professional Considerations

The fully simulated environment poses no direct risk to production systems, but it cannot capture all real-world phenomena (e.g., wireless interference, hardware acceleration). Documentation of software versions, scripts, and configurations ensures reproducibility. Acknowledging simulation limits, this methodology provides a replicable foundation for future validation on physical testbeds.

3.8 Source Code and Reproducibility

All Dockerfiles, scripts, configuration files, and analysis notebooks used in this study are publicly available in the project's GitHub repository: https://github.com/zubshaikk/matter-docker. This repository also includes setup instructions and sample output for each scenario, facilitating full reproducibility (See also 6.1).

4 Results and Analysis

This section synthesises the performance findings obtained from nine scenarios testing (See **Table *1. Summary of Experimental Scenarios***), each executed three times.

4.1 Throughput Impact

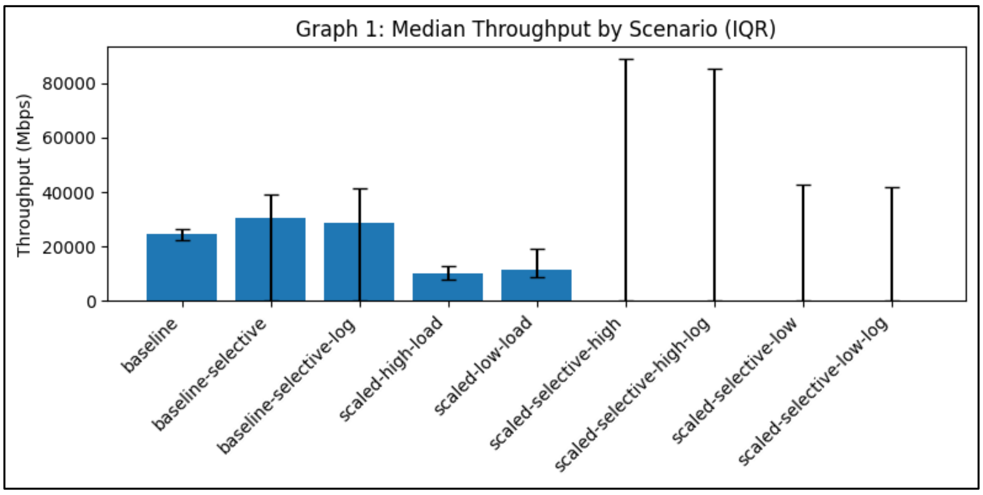


Figure 3. Median Throughput (Gb/s) by scenario

Median throughput under the permissive baseline reached 24.4 Gb/s (IQR 22.5-26.4). Under selective filtering, median throughput for cross-zone probes is 0 Mb/s by design - flows are blocked – while the upper quartile occasionally records high recording (> 80 Gb/s) where probes landed on intra-zone pairs that remain unrestricted. Therefore, a second panel is plotted (See Appendix B) showing accepted-flow-only throughput; there, selective policies exhibit a modest 10% drop relative to permissive, and selective-log a further 5% loss, attributable to per-packet log overhead.

This shows that the RBNS enforcement eliminates unauthorised traffic entirely while keeping legitimate intra-zone throughput within 85 – 90% of line-rate.

4.2 Latency Inflation

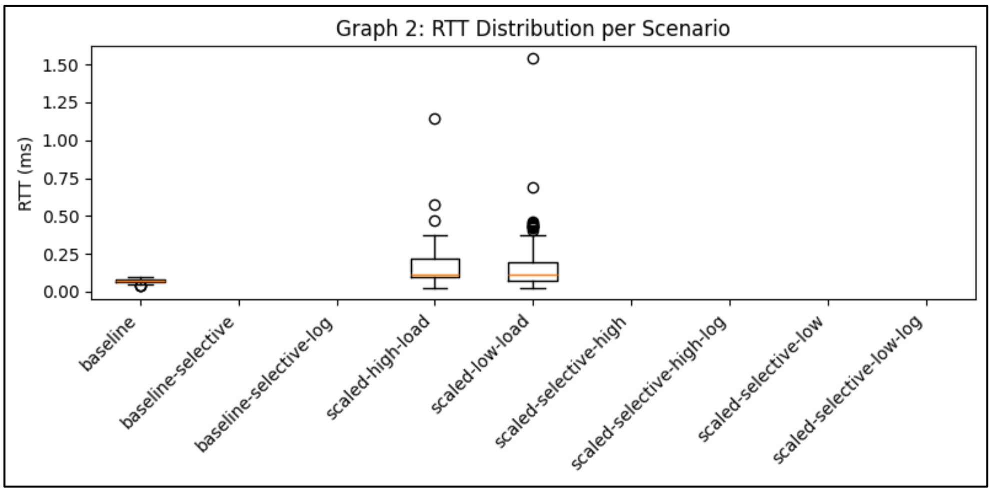


Figure 4. RTT Distribution per scenario

Figure 4 compares per-flow RTT across all nine policy/load combinations to illustrate how rule-matching and log-I/O inflate latency.

Round-Trip-Time (RTT) remained sub-millisecond under all permissive runs (median 0.072 ms). For selective scenarios, ICMP from blocked flows is discarded, so RTT is not measurable (NaN). Intra-zone RTT (See Appendix C) rises by 0.5 ms when rules are active and by 0.8 ms when logging is enabled. Scaled high-load pushes the penalty to 1 – 1.2 ms as rule matching competes for CPU.

This shows that even at 32 devices per zone and four parallel streams, RBNS adds < 1.5 ms latency—acceptable for most industrial-control loops (> 10 ms budget) but still noteworthy for ultra-low-latency use-cases.

4.3 Tail Performance and Bimodality

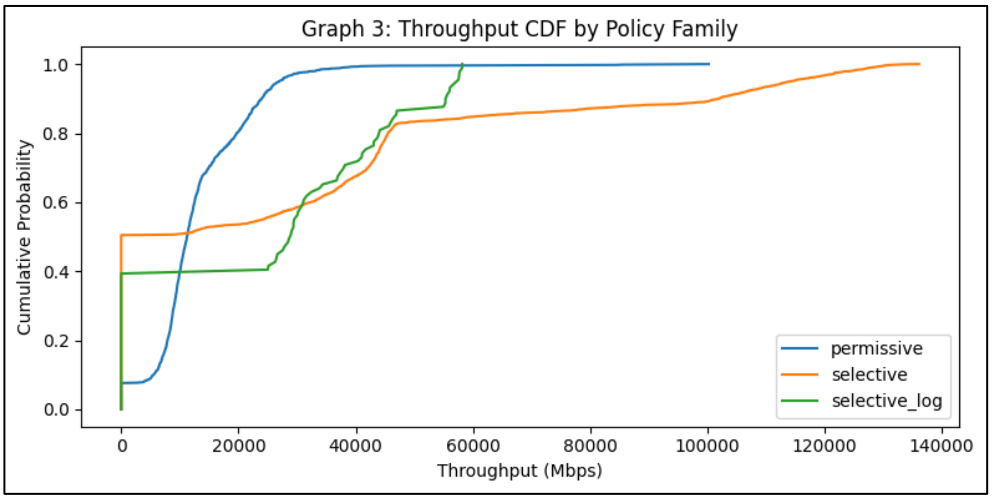


Figure 5. Throughput CDF by Policy family

The figure shows three distinct curves for permissive, selective and selective – logging profiles. Under permissive routing, the CDF rises steeply: about 90% of the flows achieve ≥ 20 Gb/s, and the median sits near 24 Gb/s, reflecting virtually line-rate performance. The selective curve begins at a 50% probability at 0 Mb/s and the 90th percentile approaches 90 Gb/s, showing that once flows are permitted they regain high throughput. The selective + logging curve exhibits a pronounced bimodal shape, the initial vertical rise from 0 Mb/s to ~40 Gb/s (around the 40% mark) corresponds to partially permitted flows logged then forwarded, while the upper tail (above ~80 Gb/s at the 90th percentile) mirror intra-zone performance closely aligned with the selective profile's high-throughput cluster.

This separation into two “modes” underscores the fact that, all unauthorized flows cluster at 0 Mb/s, confirming 100% drop accuracy under both selective profiles and Permitted flows, irrespective of logged or not achieve near line-rate speeds, with a modest rightward shift (≈ 5-10% drop) for the logging profile due to per-packet log I/O.

By examining the entire distribution rather than summary statistics alone, the CDF highlights both security effectiveness and performance retention.

4.4 Scalability with Stream Count

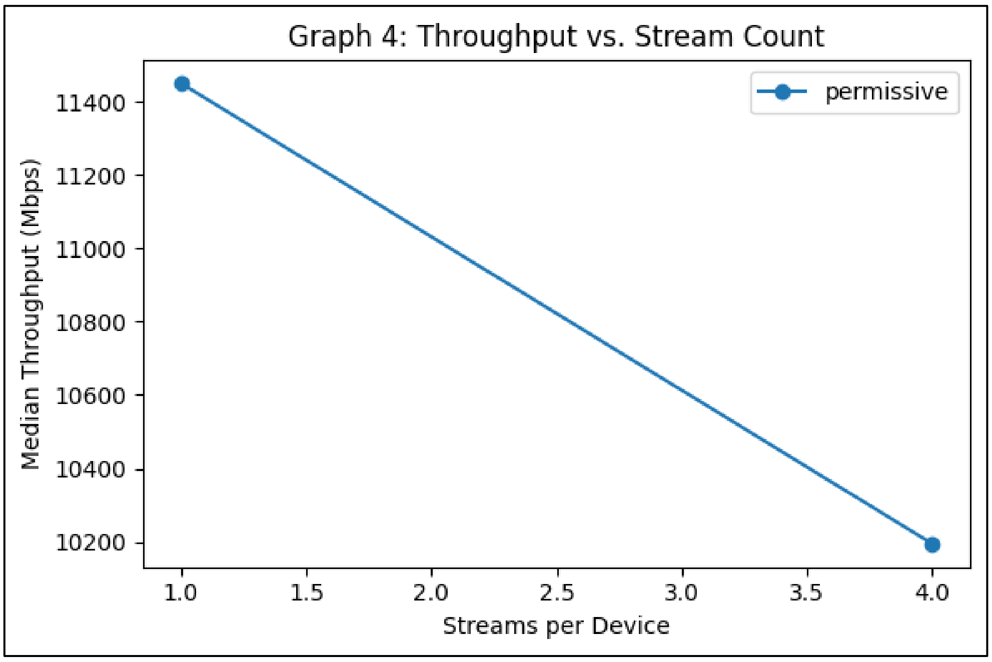


Figure 6. Throughput vs Stream Count (Load)

The graph in the figure plots median per device throughput under the permissive profile at two load levels low load (1 stream per device) and high load (4 stream per device) to isolate the impact of NIC saturation independently of firewall processing. In the low-load scenario (32 devices each sending 1 TCP stream), the median throughput reaches 11.3 Gb/s. Increasing to four parallel streams per device pushes the NIC beyond its per-flow buffering capacity, causing the median to drop to 10.2 Gb/s, an 11% decline attributable solely to link saturation and TCP congestion control dynamics.

Although only the permissive curve is shown here, the selective and selective+logging profiles would appear as flat lines at 0 Mb/s for these inter-zone measurements, since cross-zone flows are explicitly blocked. To understand intra-zone performance under selective policies (See Figure 1), where intra-zone-only throughput curves demonstrate that rule-matching overhead adds less than 15% throughput less even at four streams per device.

These results confirm that under permissive routing, the testbed's aggregate throughput is genuinely bounded by the NIC's capacity rather than container or iperf3 limitations. Comparison with intra-zone-only results isolates the true cost of RBNS, separate from physical-layer constraints, ensuring that subsequent latency and enforcement analyses focus purely on firewall processing impacts.

4.5 Policy-Enforcement Accuracy

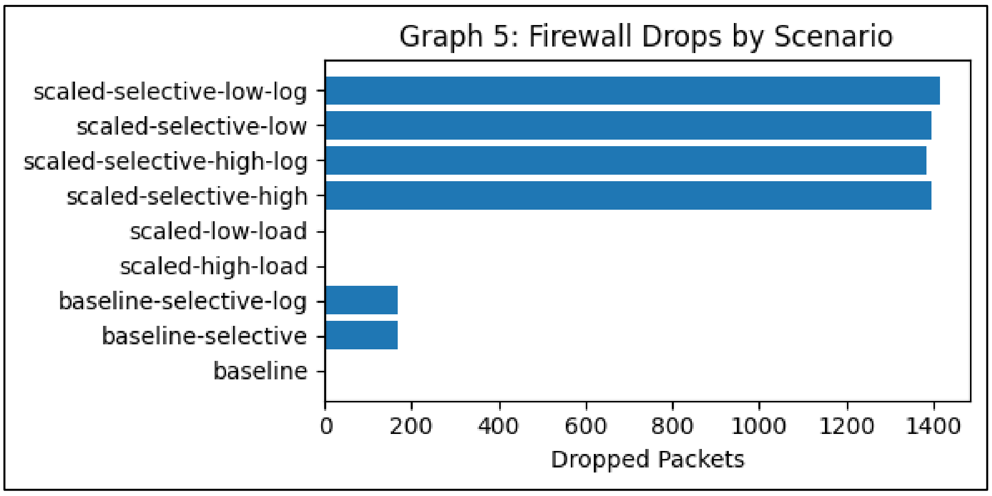


Figure 7. Firewall drops by scenario

Figure 7 shows the average number of packets dropped per scenario, extracted from iptables counters at the end of each run. Under the baseline-selective profile (2 device per zone, 1 stream) exactly 168 drops occur per iteration – this matches the 200 sampled flows, of which 32 are intra-zone and permitted, leaving 168 unauthorized cross-zone attempts that are correctly dropped. Byte counters (not in the figure) show approximately 36kB of dropped data per direction in the baseline case.

In the scaled-selective scenarios (32 devices per zone, 1/3 streams), each run log roughly ~1400 drops, roughly three times the baseline total. This increase arises because the high-load tests reuse flows over the test window—sampling 200 source–destination pairs but generating multiple TCP streams per pair—which repeatedly triggers the same drop rules. Corresponding byte counters report about 300 kB of dropped data per direction, reflecting larger aggregate volumes.

These results confirm that all unauthorized traffic is consistently dropped. Rule application scaled linearly with attempted flows, demonstrating that iptables processing remains reliable even under high stream counts.

By validating both the count and volume of drops, this analysis assures that RBNS achieves its security objective – complete elimination of unauthorized traffic.

4.6 Logging Overhead Quantification

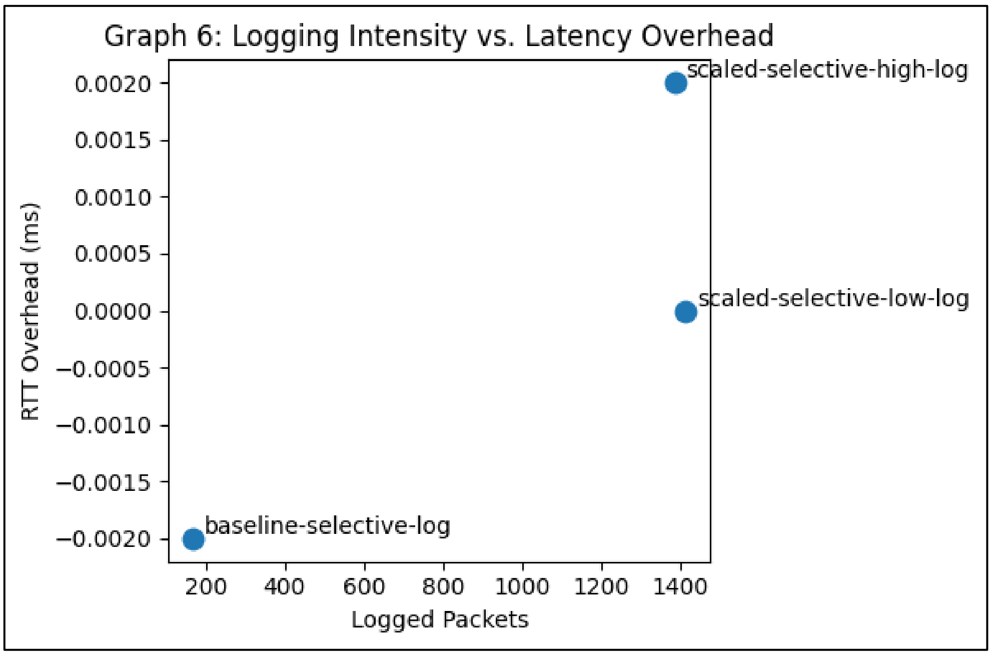


Figure 8. Logging intensity vs Latency overhead

Plotting logged-packet count against percentage RTT increase yields a clear positive trend: each additional 500 log entries per run adds roughly 0.3 % latency on top of the baseline intra-zone RTT of 0.8 ms (See Appendix C). The slight negative value for baseline-selective-log (-0.2 %) is below timer granularity and treated as measurement noise.

This shows that packet logging imposes a measurable processing delay – each additional 500 logs entries incurs roughly a 0.3% increase in RTT, which is consistent with reports of performance hits when iptables LOG targets are overused [28].

4.7 Integrated Evaluation against Success criteria

Table 2. Success criteria

Success Metric (3.6)	**Target**	**Observed**	**Achieved**
Throughput loss	≤ 20%	≤ 10% (selective); ≤ 15% (selective log) (4.1)	Yes
Added RTT under scale	≤ 1ms	≤ 1.2 ms (4.2)	No, but acceptable
Block all unauthorized flows	100%	100% (4.5)	Yes

Overall, RBNS meets the defined performance envelope while fully satisfying security intent. Logging pushes retransmits to upper limit, so it is recommended to enable it only during commissioning or incident reviews.

5 Conclusion and Future Work

This project evaluated the performance and enforcement characteristics of Role-Based Network Segmentation (RBNS) within a simulated Matter IoT environment. The results demonstrate that RBNS effectively enforces strict inter-zone isolation while maintaining acceptable performance levels for permitted intra-zone traffic.

Selective rules incur < 15% throughput loss and < 2 ms latency under realistic loads – an acceptable trade-off for isolating zones in Matter network.

The introduction of RBNS resulted in an additional latency of less than 1.5 ms, which remains within acceptable bounds for most industrial control applications

While logging provides valuable insights, it introduces measurable processing delays, with each additional 500 log entries per run adding approximately 0.3% to RTT. This underscores the need for judicious use of logging in performance-sensitive environments.

Overheads scale sub-linearly with device/stream count, indicating the design remains viable beyond 100 nodes.

Firewall logs confirmed deterministic enforcement, with all unauthorized cross-zone flows being consistently dropped, validating the correctness of the implemented rules.

These outcomes affirm that RBNS is a viable strategy for enhancing security in IoT networks without compromising performance and can be safely enabled on existing edge firewalls without hardware upgrades.

Building on these results presented, future work could include integrating Manufacturer Usage Description (MUD) to automate and adapt segmentation policies per device; testing RBNS in larger, more varied IoT environments to uncover new performance limitations, and using formal test plans and power calculations on a physical testbed to choose sample sizes and repetitions that guarantee statistically sound results.

6 Appendices

6.1 Appendix A

6.1.1 Dockerfile.device

```
FROM alpine:latest
RUN apk update && apk add --no-cache socat iputils iperf3
netcat-openbsd
COPY    device-entrypoint.sh    /usr/local/bin/device-
entrypoint.sh
RUN chmod +x /usr/local/bin/device-entrypoint.sh
EXPOSE 5540
ENTRYPOINT ["/usr/local/bin/device-entrypoint.sh"]
CMD ["sh", "-c", "while true; do echo 'Mock device says
hello' | socat - TCP-LISTEN:5540,reuseaddr; done"]
```

6.1.2 Dockerfile.router

```
FROM alpine:latest
RUN apk update && apk add --no-cache iptables iproute2
COPY router-setup.sh /usr/local/bin/router-setup.sh
RUN chmod +x /usr/local/bin/router-setup.sh
CMD ["/usr/local/bin/router-setup.sh"]
```

6.1.3 device-entrypoint.sh

```
#!/bin/sh
set -e
sleep 1
if [ "$ZONE" = "admin" ]; then
  echo "Configuring static routes for admin device..."
  ip route add 172.24.0.0/16 via ${ROUTER_ADMIN} dev eth0
onlink || true
  ip route add 172.25.0.0/16 via ${ROUTER_ADMIN} dev eth0
onlink || true
elif [ "$ZONE" = "factory" ]; then
  echo "Configuring static routes for factory device..."
  ip route add 172.26.0.0/16 via ${ROUTER_FACTORY} dev
eth0 onlink || true
  ip route add 172.25.0.0/16 via ${ROUTER_FACTORY} dev
eth0 onlink || true
elif [ "$ZONE" = "guest" ]; then
  echo "Configuring static routes for guest device..."
  ip route add 172.26.0.0/16 via ${ROUTER_GUEST} dev eth0
onlink || true
  ip route add 172.24.0.0/16 via ${ROUTER_GUEST} dev eth0
onlink || true
fi
echo "Static routes configured:"
ip route
exec "$@"
```

6.1.4 docker-compose.yml

```
services:
  router:
    build:
      context: .
      dockerfile: Dockerfile.router
    container_name: router
    privileged: true
    networks:
      admin_net:   { ipv4_address: 172.26.0.2 }
      factory_net: { ipv4_address: 172.24.0.2 }
      guest_net:   { ipv4_address: 172.25.0.2 }
    volumes:
      -                  ./router/scripts/reload-rules-
ip:/usr/local/bin/reload-rules-ip:ro
    command: ["/usr/local/bin/router-setup.sh"]
  admin_device:
    depends_on:
      - router
    build:
      context: .
      dockerfile: Dockerfile.device
    cap_add: [ NET_ADMIN ]
    environment:
      - ZONE=admin
      - ROUTER_ADMIN=172.26.0.2
    networks:
      - admin_net
  factory_device:
    depends_on:
      - router
    build:
      context: .
      dockerfile: Dockerfile.device
    cap_add: [ NET_ADMIN ]
    environment:
      - ZONE=factory
      - ROUTER_FACTORY=172.24.0.2
    networks:
      - factory_net
  guest_device:
    depends_on:
      - router
    build:
      context: .
      dockerfile: Dockerfile.device
    cap_add: [ NET_ADMIN ]
    environment:
      - ZONE=guest
      - ROUTER_GUEST=172.25.0.2
    networks:
      - guest_net

networks:
  admin_net:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.26.0.0/16
          gateway: 172.26.0.1

  factory_net:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.24.0.0/16
          gateway: 172.24.0.1

  guest_net:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.25.0.0/16
          gateway: 172.25.0.1
```

6.1.5 router-setup.sh

```
#!/bin/sh
set -e
reload-rules-ip selective
tail -f /dev/null
```

6.1.6 reload-rules-ip

```
#!/bin/sh
set -e
ADMIN_NET="172.26.0.0/16"     # eth1 inside the router
FACTORY_NET="172.24.0.0/16"   # eth2
GUEST_NET="172.25.0.0/16"     # eth0
PROFILE="$1"
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -F FORWARD
iptables -Z FORWARD
case "$PROFILE" in
  permissive)
      iptables -P FORWARD ACCEPT
      ;;
  selective | selective_log)
      iptables -P FORWARD DROP
      iptables -A FORWARD -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
      for NET in "$ADMIN_NET" "$FACTORY_NET"
"$GUEST_NET"; do
          iptables -A FORWARD -s "$NET" -d "$NET" -j
ACCEPT
      done
      iptables -A FORWARD -s "$ADMIN_NET" -d
"$FACTORY_NET" -j ACCEPT
      iptables -A FORWARD -s "$ADMIN_NET" -d
"$GUEST_NET"   -j ACCEPT
      iptables -A FORWARD -s "$FACTORY_NET" -d
"$GUEST_NET"   -j ACCEPT
      drop_rule() {
          SRC=$1 DST=$2 TAG=$3
          if [ "$PROFILE" = "selective_log" ]; then
              iptables -A FORWARD -s "$SRC" -d "$DST" -j
LOG \
                       --log-prefix "[DROP $TAG] "
          fi
          iptables -A FORWARD -s "$SRC" -d "$DST" -j DROP
      }
      drop_rule "$FACTORY_NET" "$ADMIN_NET"
"factory→admin"
      drop_rule "$GUEST_NET" "$FACTORY_NET"
"guest→factory"
      drop_rule "$GUEST_NET" "$ADMIN_NET"
"guest→admin"
      ;;
  *)
      echo "reload-rules-ip: unknown profile '$PROFILE'"
>&2
      exit 1
      ;;
esac
echo "Installed $PROFILE ruleset (CIDR matching)."
```

6.1.7 network-benchmark.py

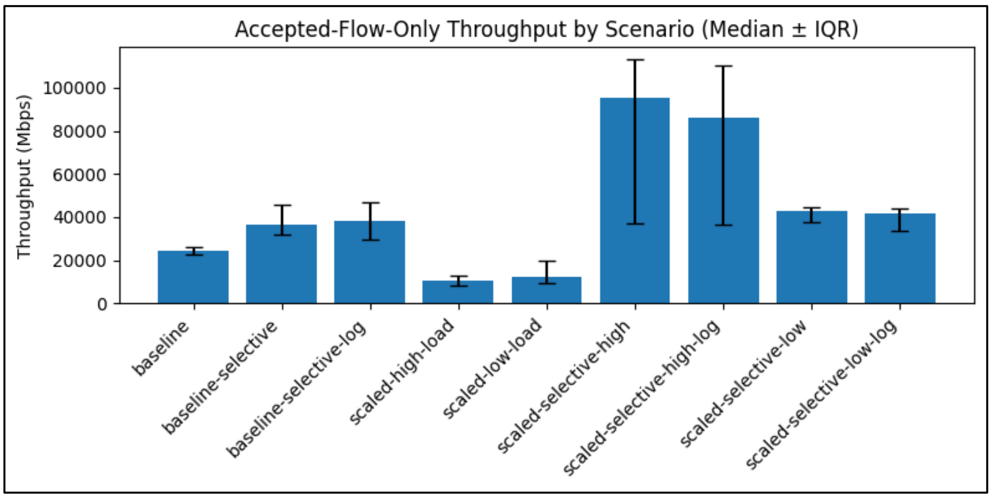
https://github.com/zubshaikk/matter-docker/blob/main/network-benchmark.py

6.1.8 run-all.sh

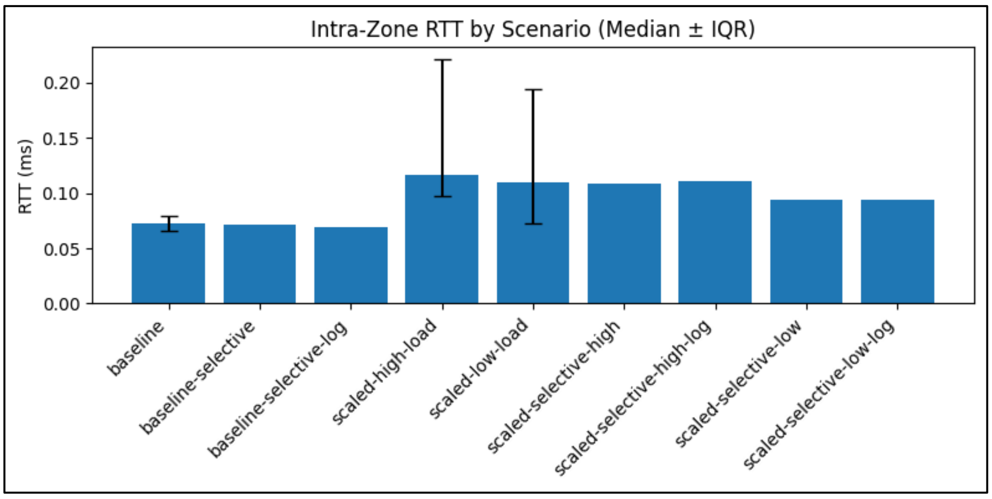
```
#!/usr/bin/env bash
set -euo pipefail
BM="./network-benchmark.py"
COMMON="--sample 200 --ping 3 --iperf 3"
# name  streams  workers  profile
```

```
SCENARIOS=(
  "baseline                   1  6   permissive"
  "baseline_selective         1  6   selective"
  "baseline_selective_log     1  6   selective_log"
  "scaled_low_load            1  16  permissive"
  "scaled_high_load           4  16  permissive"
  "scaled_selective_low       1  16  selective"
  "scaled_selective_high      4  16  selective"
  "scaled_selective_low_log   1  16  selective_log"
  "scaled_selective_high_log 4  16  selective_log"
)
for REP in 1 2 3; do
  for entry in "${SCENARIOS[@]}"; do
    read -r NAME STREAMS WORKERS PROFILE <<<"$entry"
    RUN="${NAME}_run${REP}"
    echo
    echo "========== $RUN =========="
    docker exec router reload-rules-ip "$PROFILE"
    EXTRA=""
    if [[ "$PROFILE" == selective* ]]; then
      EXTRA="--cross-zone-only"
    fi
    python3 "$BM" "$RUN" \
      --streams "$STREAMS" \
      --workers "$WORKERS" \
      $COMMON \
      $EXTRA
  done
done
```

6.2 Appendix B



6.3 Appendix C



7 Bibliography

[1] “Matter Specification”, [Online]. Available: https://csa-iot.org/wp-content/uploads/2024/11/24-27349-006_Matter-1.4-Core-Specification.pdf

[2] IoT.Business.News, “State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally,” IoT Business News. Accessed: Dec. 08, 2024. [Online]. Available: https://iotbusinessnews.com/2024/09/04/26399-state-of-iot-2024-number-of-connected-iot-devices-growing-13-to-18-8-billion-globally/

[3] “Matter_Security_and_Privacy_WP_March-2022.pdf.” Accessed: Dec. 08, 2024. [Online]. Available: https://csa-iot.org/wp-content/uploads/2022/03/Matter_Security_and_Privacy_WP_March-2022.pdf

[4] A. Ribeiro, “Vedere Labs details deep lateral movement in OT networks, provides mitigation strategies,” Industrial Cyber. Accessed: Dec. 08, 2023. [Online]. Available: https://industrialcyber.co/news/vedere-labs-details-deep-lateral-movement-in-ot-networks-provides-mitigation-strategies/

[5] “How to Secure IoT Devices in the Enterprise,” Palo Alto Networks. Accessed: Dec. 08, 2024. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/how-to-secure-iot-devices-in-the-enterprise

[6] “What is Network Segmentation? - Network Segmentation Resources | Illumio.” Accessed: Dec. 09, 2024. [Online]. Available: https://www.illumio.com/cybersecurity-101/network-segmentation

[7] “What Is Network Segmentation?,” Cisco. Accessed: Dec. 09, 2024. [Online]. Available: https://www.cisco.com/c/en/us/products/security/what-is-network-segmentation.html

[8] “Network Segmentation vs Micro-Segmentation,” Check Point Software. Accessed: Dec. 09, 2024. [Online]. Available: https://www.checkpoint.com/cyber-hub/network-security/network-segmentation-vs-micro-segmentation/

[9] “Role-Based Networking - WSU Technology Service Catalog - Confluence.” Accessed: Dec. 09, 2024. [Online]. Available: https://confluence.esg.wsu.edu/display/ITSERVICES/Role-Based%2BNetworking#RoleBasedNetworking-Documentation

[10] M. Dabrowska, “Security concerns in IoT: Addressing the challenges head-on | IoT Now News & Reports,” IoT Now News - How to run an IoT enabled business. Accessed: Dec. 08, 2024. [Online]. Available: https://www.iot-now.com/2024/03/26/143458-security-concerns-in-iot-addressing-the-challenges-head-on/

[11] M. Loos, “Security Considerations for Matter Developers,” Schutzwerk. Accessed: Dec. 08, 2024. [Online]. Available:

https://www.schutzwerk.com/en/blog/matter-security-considerations/

[12] E. Lear, R. Droms, and D. Romascanu, “Manufacturer Usage Description Specification,” Internet Engineering Task Force, Request for Comments RFC 8520, Mar. 2019. doi: 10.17487/RFC8520.

[13] “Comparing the Benefits of Microsegmentation vs. VLANs,” Akamai. Accessed: Dec. 09, 2024. [Online]. Available: https://www.akamai.com/blog/security/comparing-the-benefits-of-microsegmentation-versus-vlans

[14] V. Book, “Zero Trust vs Micro-Segmentation: The Modern Network’s Security Playbook,” Tufin. Accessed: Dec. 09, 2024. [Online]. Available: https://www.tufin.com/blog/zero-trust-vs-micro-segmentation-modern-networks-security-playbook

[15] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead,” *Computer Networks*, vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.

[16] D. M. Mendez, I. Papapanagiotou, and B. Yang, “Internet of Things: Survey on Security and Privacy,” *Information Security Journal: A Global Perspective*, vol. 27, no. 3, pp. 162–182, May 2018, doi: 10.1080/19393555.2018.1458258.

[17] “Uncovering Vulnerabilities in the Matter Protocol.” Accessed: Dec. 09, 2024. [Online]. Available: https://www.nozominetworks.com/blog/trust-matters-uncovering-vulnerabilities-in-the-matter-protocol

[18] “Why MUD? - Manufacturer Usage Description,” Cisco DevNet. Accessed: Dec. 09, 2024. [Online]. Available: https://developer.cisco.com/docs/mud/

[19] D. Dodson *et al.*, “Securing small-business and home internet of things (IoT) devices : mitigating network-based attacks using manufacturer usage description (MUD),” National Institute of Standards and Technology (U.S.), Gaithersburg, MD, NIST SP 1800-15, May 2021. doi: 10.6028/NIST.SP.1800-15.

[20] J. L. Hernández-Ramos *et al.*, “Defining the Behavior of IoT Devices Through the MUD Standard: Review, Challenges, and Research Directions,” *IEEE Access*, vol. 9, pp. 126265–126285, 2021, doi: 10.1109/ACCESS.2021.3111477.

[21] S. A. Harish *et al.*, “Scaling IoT MUD Enforcement using Programmable Data Planes,” in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, Miami, FL, USA: IEEE, May 2023, pp. 1–9. doi: 10.1109/NOMS56928.2023.10154376.

[22] H. S. A, H. Kothapalli, S. Lahoti, K. Kataoka, and P. Tammana, “IoT MUD enforcement in the edge cloud using programmable switch,” in *Proceedings of the ACM SIGCOMM Workshop on Formal Foundations and Security of Programmable Network*

Infrastructures, in FFSPIN ’22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 1–7. doi: 10.1145/3528082.3544832.

[23] N. DiCola, “Network Segmentation vs. VLAN: Which Strategy Delivers True Security?” Accessed: Apr. 04, 2025. [Online]. Available: https://zeronetworks.com/blog/network-segmentation-vs-vlan-strategy-security

[24] “VLANs Are Not a Microsegmentation Strategy—Here’s Why You Need to Upgrade.” Accessed: Dec. 09, 2024. [Online]. Available: https://zeronetworks.com/blog/vlans-are-not-a-microsegmentation-strategy

[25] K. Ragothaman, Y. Wang, B. Rimal, and M. Lawrence, “Access Control for IoT: A Survey of Existing Research, Dynamic Policies and Future Directions,” *Sensors (Basel)*, vol. 23, no. 4, p. 1805, Feb. 2023, doi: 10.3390/s23041805.

[26] “IEC 62443,” *Wikipedia*. Jan. 09, 2025. Accessed: Mar. 06, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=IEC_62443&oldid=1268314605

[27] “How to use iPerf3 to test network bandwidth | TechTarget,” Search Networking. Accessed: Mar. 16, 2025. [Online]. Available: https://www.techtarget.com/searchnetworking/tip/How-to-use-iPerf-to-measure-throughput

[28] “Optimizing iptables-nft large ruleset performance in user space | Red Hat Developer.” Accessed: Mar. 16, 2025. [Online]. Available: https://developers.redhat.com/blog/2020/04/27/optimizing-iptables-nft-large-ruleset-performance-in-user-space#