Network Monitoring: OpenTelemetry vs SNMP

CS331 - Computer Networks Project

Group 23

Arin Mehta - 23110038
Bhavya Parmar - 23110059
Vedant Chichmalkar - 22110282

GitHub Repository

Motivation

- Modern networks require visibility into both infrastructure health and application performance.
- SNMP has been the traditional standard for device-level monitoring.
- Rise of distributed microservices demands application-aware observability.
- Objective: Compare SNMP and OpenTelemetry in a controlled, hybrid setup.

Objectives

- Implement SNMP and OpenTelemetry pipelines in a simulated Mininet environment.
- Integrate both into a unified Prometheus + Grafana monitoring stack.
- Evaluate differences in resource usage, data granularity, and latency detection.
- Determine if they can work together.

Background - SNMP

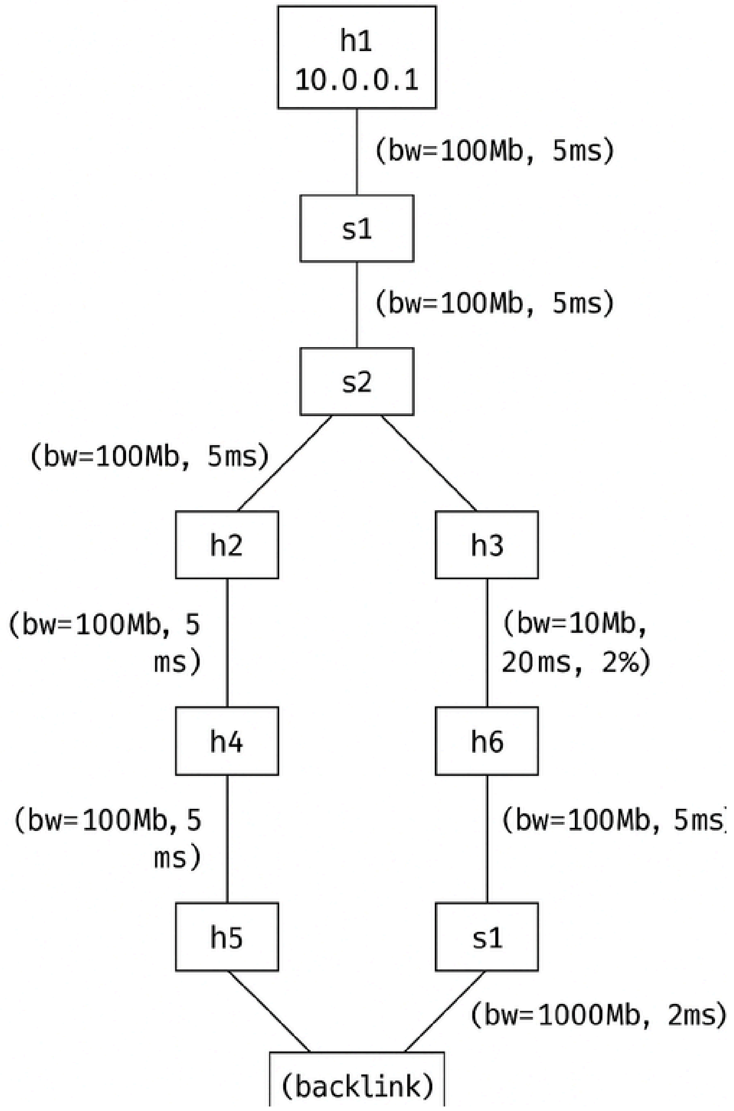
- Standard protocol for device monitoring (since 1988).
- Pull-based model: Manager polls agents via OIDs in MIBs.
- Strengths:
    - Lightweight and standardized.
    - Ideal for routers, switches, and legacy infrastructure.
- Limitation:
    - Limited visibility beyond device metrics.

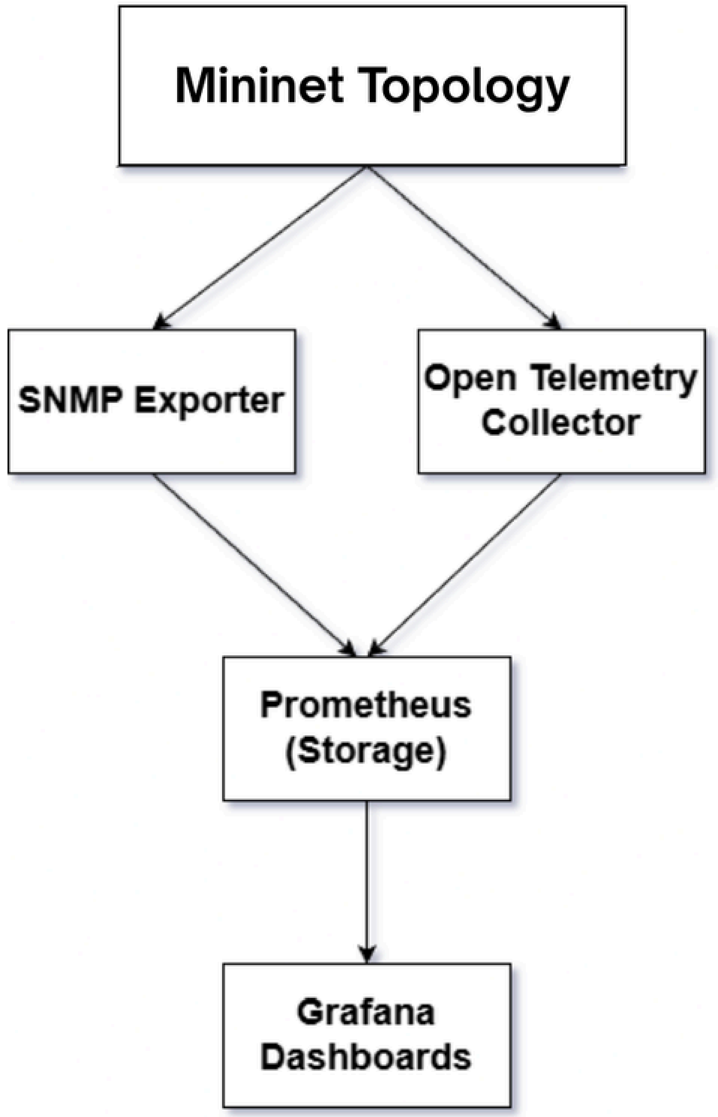
Background - OpenTelemetry (OTel)

- Modern push-based observability framework by CNCF.
- Collects metrics, traces, and logs in real-time.
- Strengths:
  - High-granularity, customizable metrics.
  - Natively supports distributed tracing.
- Limitation:
  - Higher resource overhead and complexity.

System Architecture

- Hybrid setup combining:
  - Mininet for network simulation.
  - Docker Compose for observability stack.
- Centralized stack:
  - Prometheus (time-series DB)
  - Grafana (visualization)
  - SNMP Exporter
  - OTel Collector
  - SNMP Simulator
- Ensures reproducibility and cross-platform integration.





Experimental Environment

- OS: Windows 11 (WSL2 Ubuntu 22.04)
- Hardware: i7-12700K, 16GB RAM
- Software: Docker 24.0.6, Mininet 2.3.0
- Network: 6 hosts, 2 switches with varied link bandwidth/delay.

Implementation - SNMP

- Modified topology.py to install and configure snmpd on 6 hosts.
- Custom configuration binds to all interfaces and uses public community string.
- Key metrics collected via Prometheus SNMP Exporter:
    - sysUpTime, ifInOctets, ifOutOctets, ifInErrors.
- Focused on IF-MIB and SYSTEM groups for link-level monitoring.

Implementation - OpenTelemetry (OTel)

- Python agent (network_monitor.py) measures:
  - Real-world latency (ping 8.8.8.8, 1.1.1.1).
  - Packet loss and jitter.
- Uses OpenTelemetry SDK to record:
  - network.latency (histogram)
  - network.packet_loss (counter).
- Data exported to OTel Collector, processed, and exposed to Prometheus.

Integration and Visualization

- Prometheus scrapes both:
  - SNMP Exporter → Port 9116
  - OTel Collector → Port 8889
- Grafana Dashboards:
  - SNMP Dashboard – Device health & bandwidth.
  - OTel Dashboard – Latency & packet loss.
- Comparison Dashboard – Correlates both in real-time.

Integration and Visualization

- Prometheus scrapes both:
  - SNMP Exporter → Port 9116
  - OTel Collector → Port 8889
- Grafana Dashboards:
  - SNMP Dashboard – Device health & bandwidth.
  - OTel Dashboard – Latency & packet loss.
- Comparison Dashboard – Correlates both in real-time.

Traffic Generation

- Multi-protocol traffic generator:
  - ICMP pings (100/s)
  - HTTP request bursts (curl)
  - UDP streams (64–1500 bytes, 10 Mbps)
  - TCP connection tests
- Each run: 60 minutes under controlled conditions.

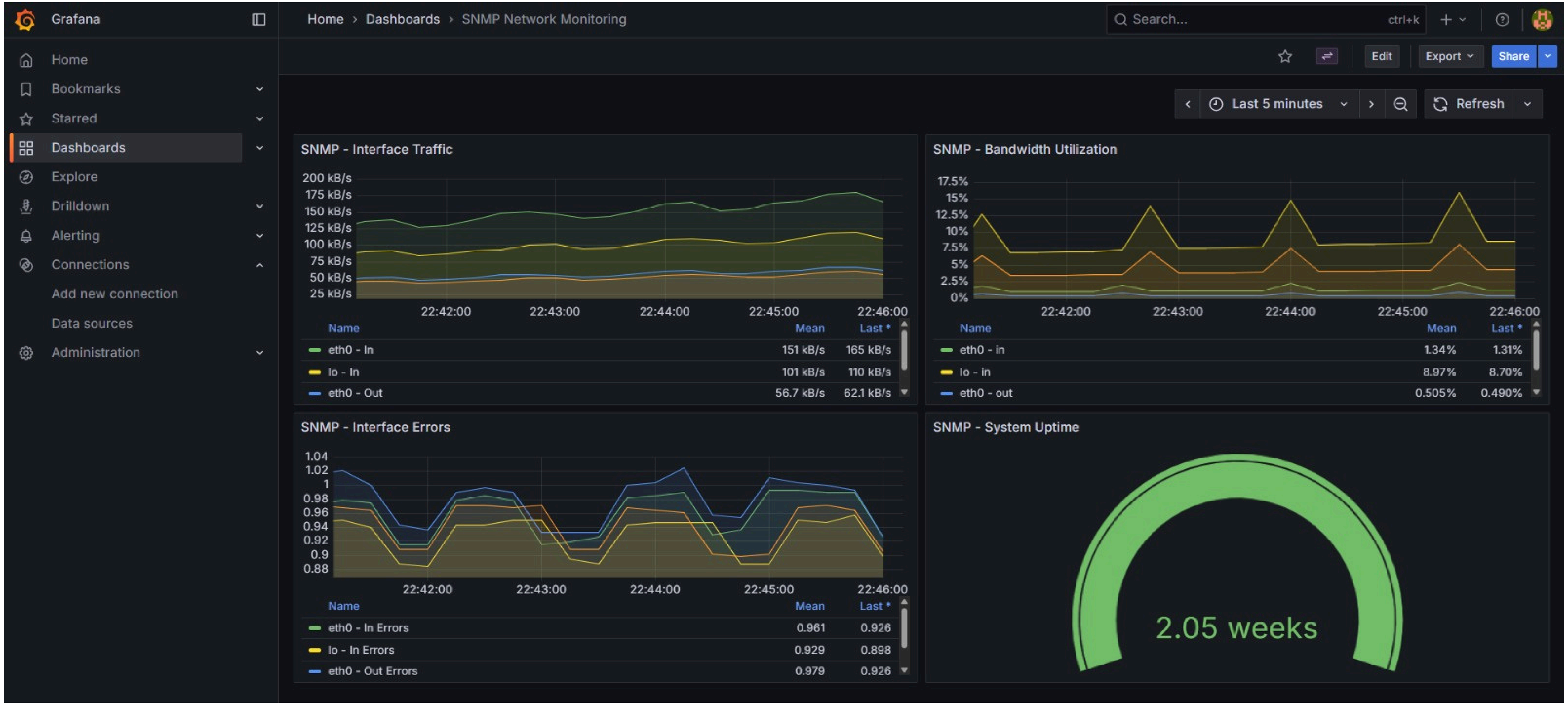
Traffic Generation

- Multi-protocol traffic generator:
    - ICMP pings (100/s)
    - HTTP request bursts (curl)
    - UDP streams (64–1500 bytes, 10 Mbps)
    - TCP connection tests
- Each run: 60 minutes under controlled conditions.

Performance Benchmarking

- Latency Detection:
  - OTel detected sub-second events in real time.
  - SNMP lacked RTT visibility.
- Bandwidth Correlation:
  - High visual correlation between SNMP and OTel metrics.
  - OTel reports ~10–15s faster due to push-based architecture.

Output Visualization



SNMP - Interface Traffic

Name	Mean	Last *
eth0 - In	151 kB/s	165 kB/s
lo - In	101 kB/s	110 kB/s
eth0 - Out	56.7 kB/s	62.1 kB/s

SNMP - Bandwidth Utilization

Name	Mean	Last *
eth0 - in	1.34%	1.31%
lo - in	8.97%	8.70%
eth0 - out	0.505%	0.490%

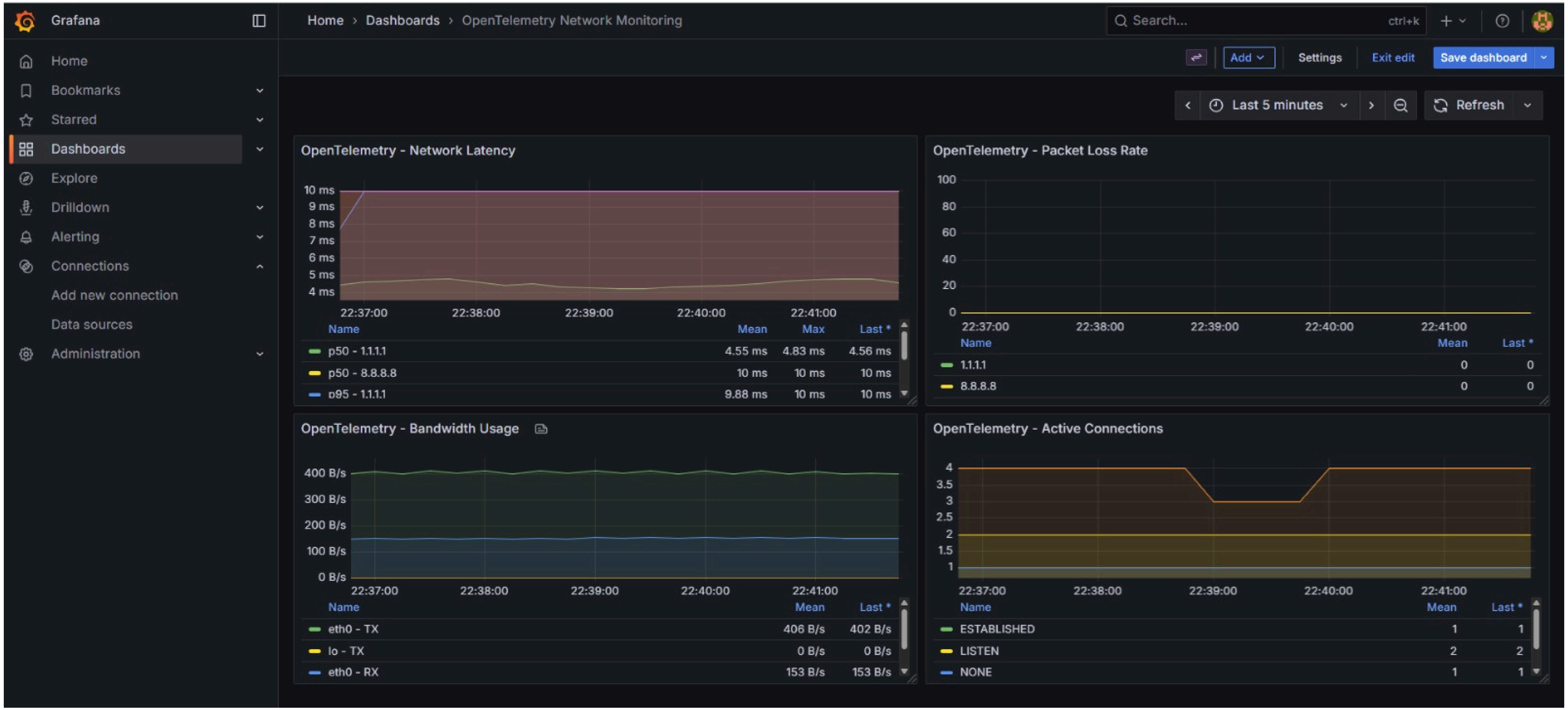
SNMP - Interface Errors

Name	Mean	Last *
eth0 - In Errors	0.961	0.926
lo - In Errors	0.929	0.898
eth0 - Out Errors	0.979	0.926

SNMP - System Uptime

SNMP Dashboard

Output Visualization



Name	Mean	Max	Last *
p50 - 1.1.1.1	4.55 ms	4.83 ms	4.56 ms
p50 - 8.8.8.8	10 ms	10 ms	10 ms
p95 - 1.1.1.1	9.88 ms	10 ms	10 ms

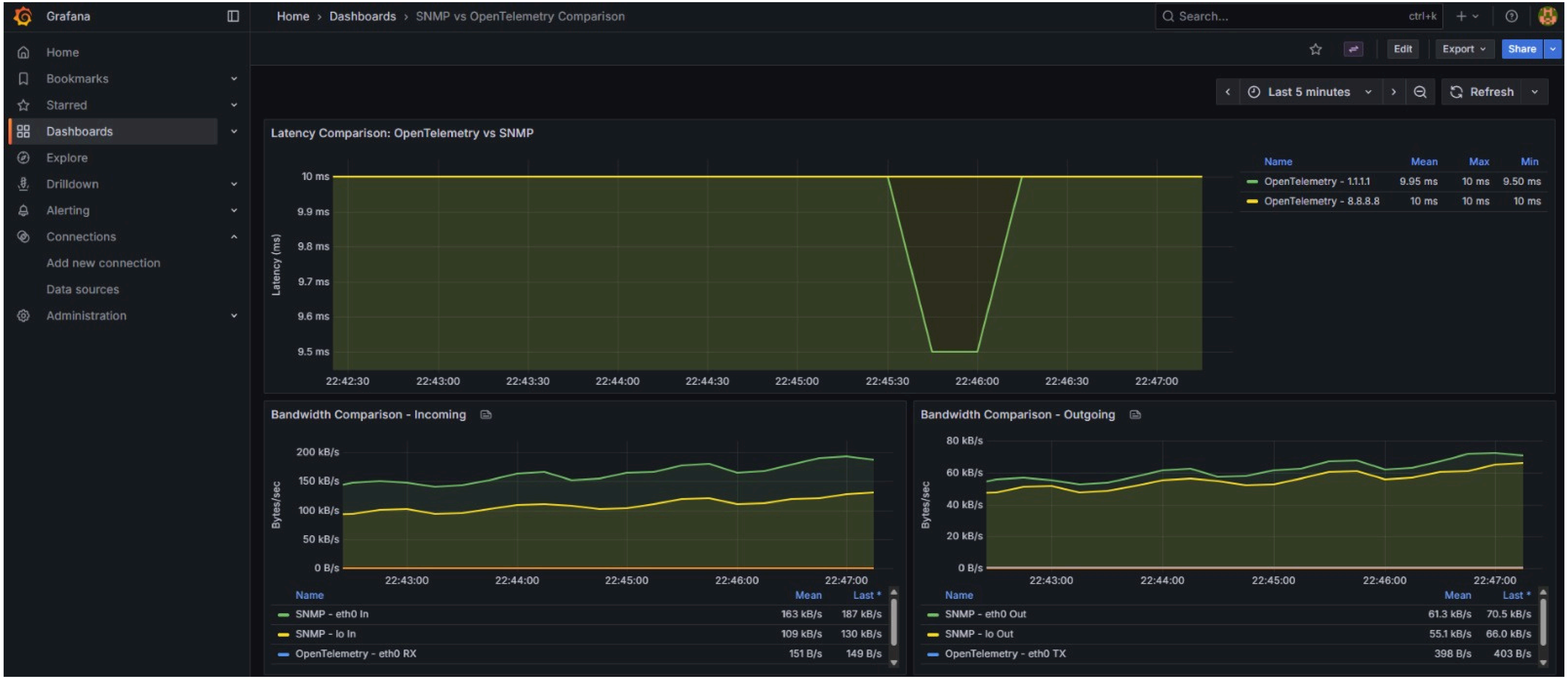
Name	Mean	Last *
1.1.1.1	0	0
8.8.8.8	0	0

Name	Mean	Last *
eth0 - TX	406 B/s	402 B/s
lo - TX	0 B/s	0 B/s
eth0 - RX	153 B/s	153 B/s

Name	Mean	Last *
ESTABLISHED	1	1
LISTEN	2	2
NONE	1	1

OTel Dashboard

Output Visualization



Latency Comparison: OpenTelemetry vs SNMP

Name	Mean	Max	Min
OpenTelemetry - 1.1.1.1	9.95 ms	10 ms	9.50 ms
OpenTelemetry - 8.8.8.8	10 ms	10 ms	10 ms

Bandwidth Comparison - Incoming

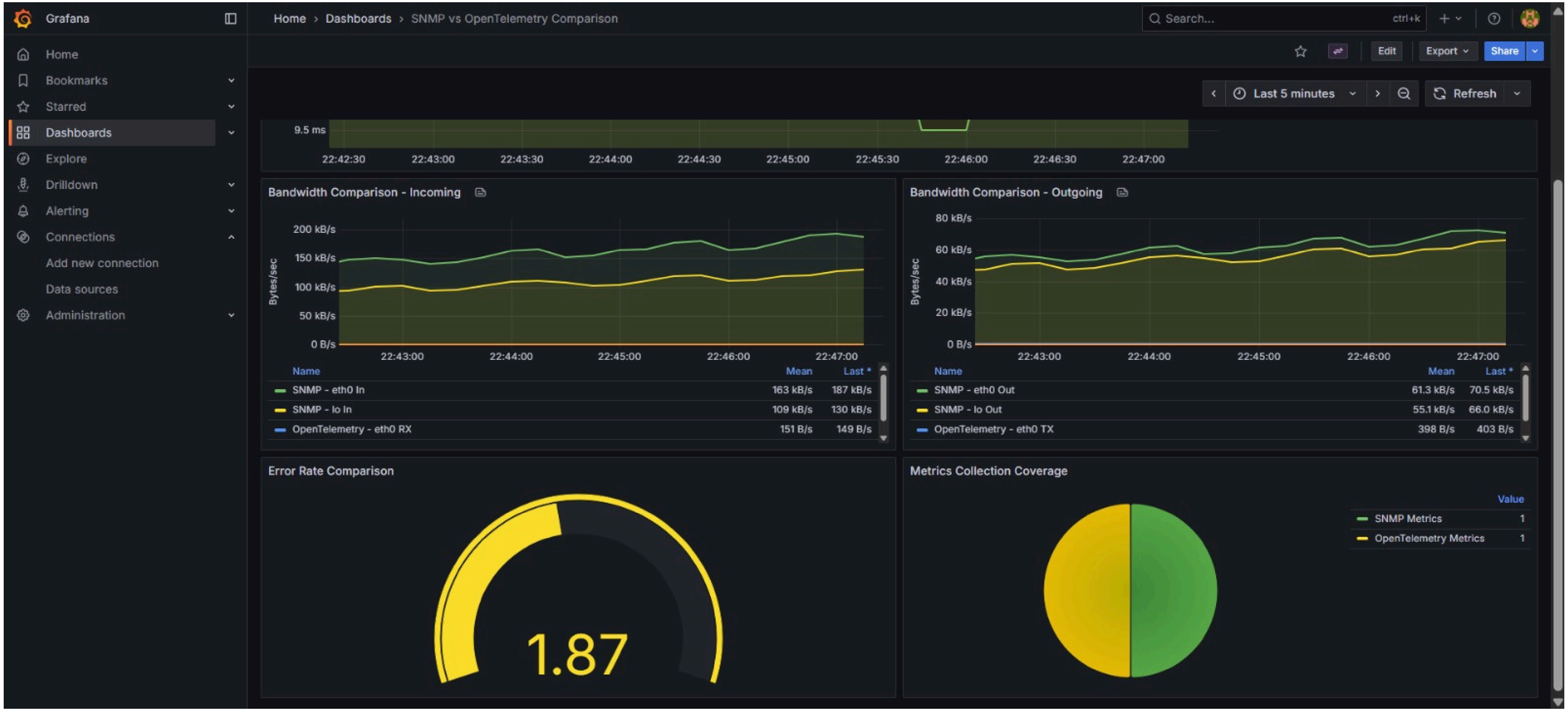
Name	Mean	Last *
SNMP - eth0 In	163 kB/s	187 kB/s
SNMP - lo In	109 kB/s	130 kB/s
OpenTelemetry - eth0 RX	151 B/s	149 B/s

Bandwidth Comparison - Outgoing

Name	Mean	Last *
SNMP - eth0 Out	61.3 kB/s	70.5 kB/s
SNMP - lo Out	55.1 kB/s	66.0 kB/s
OpenTelemetry - eth0 TX	398 B/s	403 B/s

OTel vs SNMP Dashboard

Output Visualization



Name	Mean	Last *
SNMP - eth0 In	163 kB/s	187 kB/s
SNMP - lo In	109 kB/s	130 kB/s
OpenTelemetry - eth0 RX	151 B/s	149 B/s

Name	Mean	Last *
SNMP - eth0 Out	61.3 kB/s	70.5 kB/s
SNMP - lo Out	55.1 kB/s	66.0 kB/s
OpenTelemetry - eth0 TX	398 B/s	403 B/s

	Value
SNMP Metrics	1
OpenTelemetry Metrics	1

OTel vs SNMP Dashboard

Results Summary

Metric	SNMP	OpenTelemetry
Model	Pull	Push
Focus	Infrastructure	Application
Latency Detection	None	Real-time
Data Type	Fixed OIDs	Custom
Tracing	No	Supported
Reporting Speed	15–30s	<1s

Validation and Testing

- SNMP Verification:
  - Cross-checked ifInOctets and ifOutOctets via snmpwalk.
- OTel Validation:
  - Compared dashboard latency with ping logs → accurate p99 spikes.
- Fault Injection Tests:
  - Simulated packet loss (1–2%), latency bursts, and service drops.
  - SNMP detected link degradation; OTel captured latency spikes instantly.

Integration Testing

- Automated test suite (scripts/test.sh) checks:
  - Docker container uptime.
  - API reachability for Prometheus and OTel Collector.
  - Metric ingestion and dashboard provisioning.
- Ensures a consistent environment before each experiment.

Discussion: Architectural Insights

- SNMP: Best for stable, legacy networks with low overhead.
- OTel: Designed for distributed and cloud-native observability.
- Hybrid model enables:
    - Broad infrastructure coverage (SNMP).
    - Deep application visibility (OTel).

Discussion: Architectural Insights

- SNMP: Best for stable, legacy networks with low overhead.
- OTel: Designed for distributed and cloud-native observability.
- Hybrid model enables:
    - Broad infrastructure coverage (SNMP).
    - Deep application visibility (OTel).

Best Practices for Hybrid Monitoring

- Use Prometheus as unified data store.
- Maintain consistent labeling across both pipelines.
- Create role-specific dashboards: network ops vs application devs.
- Stratify alerts:
    - SNMP → hardware faults.
    - OTel → latency or SLA violations.

Conclusion

- SNMP remains reliable for infrastructure-level health.
- OpenTelemetry offers fine-grained, real-time performance metrics.
- Together, they deliver complete network observability from hardware to application layer.
- The hybrid model demonstrated in this project is scalable and adaptable for modern enterprise environments.