GARNET | Ghanaian Academic and Research Network

DAY 2 · SESSION 2

09:30 — 10:30 · Lecture

SNMP Monitoring Concepts

How Prometheus talks to switches, routers, and other network gear

REALITY CHECK

SNMP isn't going anywhere

Universal on network gear

Every managed switch, router, firewall, AP, UPS, and PDU speaks SNMP. No alternative is as widespread.

Decades of MIBs

Vendors document their devices through SNMP MIBs. Even modern gear with REST APIs still ships SNMP.

Read-only is safe

We only do SNMP GET. No writes, no config changes, no risk to production gear.

But it's slow and quirky

Polling can be slow; some devices time out under load. We design around it, not against it.

VOCAB

SNMP in 60 seconds

OID	A numeric identifier for one piece of data: 1.3.6.1.2.1.2.2.1.10.1 = bytes received on interface 1.
MIB	A document mapping human names (ifInOctets) to OIDs. Vendor-specific or standardised.
Community	An access password in SNMPv1/v2c. 'public' is the (insecure) default; everyone uses something else.
v2c vs v3	v2c uses community strings. v3 adds users, auth, encryption. Use v3 if your gear supports it.
GET / WALK	GET fetches one OID. WALK fetches a subtree (e.g. all interfaces). Exporter does both.

ARCHITECTURE

snmp_exporter is a translator

Key idea: Prometheus has no SNMP code. It just makes an HTTP request like any other.

The exporter does all the SNMP work, applies the MIB, and returns Prometheus-format text.

One snmp_exporter can serve many devices — the target is passed as a URL parameter.

GENERATOR vs EXPORTER

Two tools, one name

GENERATOR

snmp_exporter generator

Build-time tool

- Reads MIB files from vendors
- Reads generator.yml (which OIDs you want)
- Produces snmp.yml — the runtime config
- Run once, then commit snmp.yml to git

EXPORTER

snmp_exporter (the daemon)

Runtime tool

- Reads snmp.yml at startup
- Listens on :9116
- Polls devices on demand when scraped
- What you actually run in production

CONFIGURATION

snmp.yml: modules and metrics

snmp.yml is organised into modules — each module defines which OIDs to walk for one device class.

```
modules:
  if_mib:
    walk:
      - 1.3.6.1.2.1.2     # interfaces table
      - 1.3.6.1.2.1.31    # ifXTable (64-bit counters)
    metrics:
      - name: ifHCInOctets
        oid: 1.3.6.1.2.1.31.1.1.1.6
        type: counter
  cisco_wlc:
    walk: ...               # different OIDs for WLC
```

WHAT TO COLLECT

Metrics every NOC needs

Traffic	`ifHCInOctets / ifHCOutOctets`
Errors	`ifInErrors / ifOutErrors / ifInDiscards`
Status	`ifOperStatus (1=up, 2=down)`
Speed	`ifHighSpeed (Mbps)`
Description	`ifAlias (your description)`
Device	`sysUpTime, sysDescr`
Health	`vendor-specific: CPU, mem, temperature`

Use 64-bit counters (ifHC) wherever available — 32-bit counters wrap in seconds on fast links.*

PRACTICAL

Cardinality on a real network

50 devices × 48 interfaces × 6 metrics = 14,400 series

Comfortably within Prometheus's range. A Pi 4 can handle this.

Healthy practice

- Filter to active interfaces only
- 60-120s scrape interval for SNMP
- Drop unused MIB tables in snmp.yml

What blows things up

- Walking entire MIB on every scrape
- 15s interval against slow gear
- Per-MAC or per-flow tables

FLOW

How a single SNMP scrape works

1	**Prometheus initiates**	Hits snmp_exporter at /snmp?target=10.0.0.1&module=if_mib
2	**Exporter unpacks query**	Reads target IP and module name from URL parameters
3	**Exporter polls device**	Issues SNMP GET / WALK using community/credentials from config
4	**Translation**	Walks the OID subtrees, applies snmp.yml mapping to name metrics
5	**Returns Prometheus text**	ifHCInOctets{ifIndex="1"} 8420213, etc.
6	**Prometheus stores**	Same as any other scrape — into the TSDB

TO THE LAB

Now let's poll a real device

Hands-on: configure snmp_exporter, point it at our three EVE-NG devices, scrape, and graph link traffic.