We Crashed Routers So You Don't Have To

Telemetry, BMP and the AI Data Pipeline

Nicolai Moraru
Director of Software Engineering @ ITcare

COLD OPEN

It was a normal day

cr1	cr2	customers	redundancy	upstream bgp	alarms
UP	**UP**	**OK**	**N+1**	**4/4**	**0**
primary · cpu 14%	backup · cpu 9%	no impact	2 core routers	2 per core router	all quiet

cr2 · syslog — ⚠ RECONSTRUCTED

COLD OPEN

It was a normal day

cr1	cr2	customers	redundancy	upstream bgp	alarms
UP	**UP**	**OK**	**N+1**	**4/4**	**0**
primary · cpu 14%	backup · cpu 9%	no impact	2 core routers	2 per core router	all quiet

cr2 · syslog — ⚠ RECONSTRUCTED

```
09:41:02   bmp          station 192.0.2.10: send queue 48 MB
09:41:19   bmp          station 192.0.2.10: send queue 212 MB
```

It was a normal day

cr1	cr2	customers	redundancy	upstream bgp	alarms
UP	**UP**	**OK**	**N+1**	**4/4**	**0**
primary · cpu 14%	backup · cpu 9%	no impact	2 core routers	2 per core router	all quiet

cr2 · syslog — ⚠ RECONSTRUCTED

```
09:41:02   bmp          station 192.0.2.10: send queue 48 MB
09:41:19   bmp          station 192.0.2.10: send queue 212 MB
09:41:33   routing      memory pressure high, allocation failed
09:41:34   kernel       routing process exited (signal 6)
```

COLD OPEN

It was a normal day

cr1	cr2	customers	redundancy	upstream bgp	alarms
UP	**DOWN**	**OK**	**NONE**	**2/4**	**3**
carrying everything	routing engine lost	no impact	one failure from outage	both of cr2's down	cr2 + 2 sessions

cr2 · syslog — ⚠ RECONSTRUCTED

```
09:41:02   bmp          station 192.0.2.10: send queue 48 MB
09:41:19   bmp          station 192.0.2.10: send queue 212 MB
09:41:33   routing      memory pressure high, allocation failed
09:41:34   kernel       routing process exited (signal 6)
09:41:35   bgp          2 upstream neighbors down, routing engine restarting
```

The cascade

COLD OPEN

The cascade

MONITORING SIDE

01

Monitoring hypervisor

Storage issues, very high IO pressure

COLD OPEN

The cascade

MONITORING SIDE

01

Monitoring hypervisor

Storage issues, very high IO pressure

02

BMP collector + queue

Slows down and stops reading

The cascade

MONITORING SIDE

THE WIRE

01
Monitoring hypervisor
Storage issues, very high IO pressure

02
BMP collector + queue
Slows down and stops reading

03
TCP backpressure
Receive window closes toward the router

The cascade

MONITORING SIDE

01
Monitoring hypervisor
Storage issues, very high IO pressure

▶

02
BMP collector + queue
Slows down and stops reading

THE WIRE

▶

03
TCP backpressure
Receive window closes toward the router

NETWORK SIDE

▶

04
Router BMP session
Unsent data backs up on the router

COLD OPEN

The cascade

MONITORING SIDE

01
Monitoring hypervisor
Storage issues, very high IO pressure

▶

02
BMP collector + queue
Slows down and stops reading

▶

THE WIRE

03
TCP backpressure
Receive window closes toward the router

▶

NETWORK SIDE

04
Router BMP session
Unsent data backs up on the router

▶

05
cr2 crashes
Backup core router: no customer impact, no redundancy

COLD OPEN

The cascade

MONITORING SIDE

01
Monitoring hypervisor
Storage issues, very high IO pressure

02
BMP collector + queue
Slows down and stops reading

THE WIRE

03
TCP backpressure
Receive window closes toward the router

NETWORK SIDE

04
Router BMP session
Unsent data backs up on the router

05
cr2 crashes
Backup core router: no customer impact, no redundancy

BMP off on the crashed router (a firmware bug?), collector back up

The cascade

MONITORING SIDE

01
Monitoring hypervisor
Storage issues, very high IO pressure

▶

02
BMP collector + queue
Slows down and stops reading

▶

THE WIRE

03
TCP backpressure
Receive window closes toward the router

▶

NETWORK SIDE

04
Router BMP session
Unsent data backs up on the router

▶

05
cr2 crashes
Backup core router: no customer impact, no redundancy

▼

BMP off on the crashed router (a firmware bug?), collector back up

again
Another router crashes
Newer firmware, same crash

COLD OPEN

The cascade

MONITORING SIDE

01
Monitoring hypervisor
Storage issues, very high IO pressure

▶

02
BMP collector + queue
Slows down and stops reading

THE WIRE

▶

03
TCP backpressure
Receive window closes toward the router

NETWORK SIDE

▶

04
Router BMP session
Unsent data backs up on the router

▶

05
cr2 crashes
Backup core router: no customer impact, no redundancy

BMP off on the crashed router (a firmware bug?), collector back up

▼

again
Another router crashes
Newer firmware, same crash

A router must never die because its monitoring receiver is slow.

No vendors. No versions. No product names.

Not because of lawyers. Because it doesn’t matter: we saw monitoring-induced crashes across **three major vendors**.

HOW WE GOT HERE

No vendors. No versions. No product names.

Not because of lawyers. Because it doesn't matter: we saw monitoring-induced crashes across **three major vendors**.



Has monitoring ever caused an incident on **your** network?

The pattern

Generation	The promise	What shipped

The pattern

Generation	The promise	What shipped
SNMP	*Standard MIBs, universal monitoring*	**Enterprise MIBs, vendor-specific counters**

The pattern

Generation	The promise	What shipped
SNMP	*Standard MIBs, universal monitoring*	**Enterprise MIBs, vendor-specific counters**
NETCONF	*One standard API for every device*	**Standard transport, vendor-native models**

The pattern

Generation	The promise	What shipped
`SNMP`	*Standard MIBs, universal monitoring*	**Enterprise MIBs, vendor-specific counters**
`NETCONF`	*One standard API for every device*	**Standard transport, vendor-native models**
`YANG / OpenConfig`	*Vendor-neutral models*	**Deviations, augmentations, partial coverage**

The pattern

Generation	The promise	What shipped
`SNMP`	*Standard MIBs, universal monitoring*	**Enterprise MIBs, vendor-specific counters**
`NETCONF`	*One standard API for every device*	**Standard transport, vendor-native models**
`YANG / OpenConfig`	*Vendor-neutral models*	**Deviations, augmentations, partial coverage**
`gNMI streaming`	*Efficient push, on-change everything*	**On-change varies by path, platform, release**

The pattern

Generation	The promise	What shipped
SNMP	*Standard MIBs, universal monitoring*	**Enterprise MIBs, vendor-specific counters**
NETCONF	*One standard API for every device*	**Standard transport, vendor-native models**
YANG / OpenConfig	*Vendor-neutral models*	**Deviations, augmentations, partial coverage**
gNMI streaming	*Efficient push, on-change everything*	**On-change varies by path, platform, release**
BMP	*Clean, passive routing visibility*	**see slide 3 ↩**

The pattern

Generation	The promise	What shipped
SNMP	*Standard MIBs, universal monitoring*	**Enterprise MIBs, vendor-specific counters**
NETCONF	*One standard API for every device*	**Standard transport, vendor-native models**
YANG / OpenConfig	*Vendor-neutral models*	**Deviations, augmentations, partial coverage**
gNMI streaming	*Efficient push, on-change everything*	**On-change varies by path, platform, release**
BMP	*Clean, passive routing visibility*	**see slide 3 ↩**

> **The protocol was never the problem. Implementations and incentives are.**

REALITY TODAY

Off-the-shelf works. In 2005.

`open-source NMS` `SNMP` `scripts/`

```
scripts/
├── check_bgp.sh
├── poll_optics.py
├── fix_poll_v3.sh
└── fix_poll_v3_FINAL2.sh   # it works
```

works, everywhere, boringly

Off-the-shelf works. In 2005.

`open-source NMS` `SNMP` `scripts/`

```
scripts/
├── check_bgp.sh
├── poll_optics.py
├── fix_poll_v3.sh
└── fix_poll_v3_FINAL2.sh   # it works
```

works, everywhere, boringly

`what's missing`

- ✗ Routing visibility
- ✗ Streaming telemetry
- ✗ On-change events

Off-the-shelf works. In 2005.

`open-source NMS` `SNMP` `scripts/`

```
scripts/
├── check_bgp.sh
├── poll_optics.py
├── fix_poll_v3.sh
└── fix_poll_v3_FINAL2.sh   # it works
```

works, everywhere, boringly

`what's missing`

- ✗ Routing visibility
- ✗ Streaming telemetry
- ✗ On-change events

Even the boring path hurts: default discovery and polling, too heavy for old CPUs.

Four ways monitoring hurt us

Four ways monitoring hurt us

01 **Overloads**

Aggressive SNMP polling, struggling on old CPUs.

Four ways monitoring hurt us

01 **Overloads**

Aggressive SNMP polling, struggling on old CPUs.

02 **Isn't there**

Platforms with little or no YANG or BMP support.

Four ways monitoring hurt us

01 **Overloads**

Aggressive SNMP polling, struggling on old CPUs.

02 **Isn't there**

Platforms with little or no YANG or BMP support.

03 **Crashes things**

BMP under TCP backpressure. YANG push that worked for a while, then took the device down.

Four ways monitoring hurt us

01 Overloads

Aggressive SNMP polling, struggling on old CPUs.

02 Isn't there

Platforms with little or no YANG or BMP support.

03 Crashes things

BMP under TCP backpressure. YANG push that worked for a while, then took the device down.

04 Costs more than promised

YANG push used measurably more CPU than SNMPv2 polling. In our setup.

The matrix

FAILURE MODE	FIRMWARE AGE	IMPACT
Crash under BMP backpressure	last ~2 years	routing engine or router
Crash after YANG subscriptions	recent	device down
Higher CPU than SNMP	recent	degraded
Missing YANG / BMP support	any	no data
Overload from polling	old platforms	degraded

The matrix

FAILURE MODE	FIRMWARE AGE	IMPACT
Crash under BMP backpressure	last ~2 years	routing engine or router
Crash after YANG subscriptions	recent	device down
Higher CPU than SNMP	recent	degraded
Missing YANG / BMP support	any	no data
Overload from polling	old platforms	degraded

> **The vendor doesn't matter. The pattern does.**

So your monitoring must know...

So your monitoring must know...

- [x] the platform

So your monitoring must know...

- [x] the platform
- [x] its role in the network

So your monitoring must know...

- ☑ the platform
- ☑ its role in the network
- ☑ the exact firmware version

So your monitoring must know...

- [x] the platform
- [x] its role in the network
- [x] the exact firmware version
- [x] the known bugs in that firmware

So your monitoring must know...

- [x] the platform
- [x] its role in the network
- [x] the exact firmware version
- [x] the known bugs in that firmware
- [x] which protocol is safe on which box

So your monitoring must know…

- ☑ the platform
- ☑ its role in the network
- ☑ the exact firmware version
- ☑ the known bugs in that firmware
- ☑ which protocol is safe on which box

No human can maintain this by hand.

Three rules we run in production

Three rules we run in production

01 Protocol by role

Each device role gets the collection method that fits it.

e.g. BMP where routing visibility matters. Plain SNMP on old access gear.

Three rules we run in production

01 Protocol by role

Each device role gets the collection method that fits it.

e.g. BMP where routing visibility matters. Plain SNMP on old access gear.

02 Features by firmware

Telemetry switched on or off per firmware version, not per platform.

e.g. A subscription allowed on one release, disabled on the one known to crash.

Three rules we run in production

01 Protocol by role

Each device role gets the collection method that fits it.

e.g. BMP where routing visibility matters. Plain SNMP on old access gear.

02 Features by firmware

Telemetry switched on or off per firmware version, not per platform.

e.g. A subscription allowed on one release, disabled on the one known to crash.

03 Isolated collectors

A slow or broken collector never pushes pain back to devices.

e.g. Sized never to fall behind. Their lag is itself an alert.

Three rules we run in production

01 Protocol by role

Each device role gets the collection method that fits it.

e.g. BMP where routing visibility matters. Plain SNMP on old access gear.

02 Features by firmware

Telemetry switched on or off per firmware version, not per platform.

e.g. A subscription allowed on one release, disabled on the one known to crash.

03 Isolated collectors

A slow or broken collector never pushes pain back to devices.

e.g. Sized never to fall behind. Their lag is itself an alert.

Every rule exists because of a story.

"Just give the AI the data"

> NOC Average CPU over the last 6 hours, for all routers in site X?

"Just give the AI the data"

> **NOC** Average CPU over the last 6 hours, for all routers in site X?

`raw data to the model`

RETURNED	**14,400** points · 40 routers × 360 min
SIZE	**150k+** tokens
TOOL CALLS	several: site, names, query, pages
MATH BY	the model, over thousands of numbers

"Just give the AI the data"

> **NOC** Average CPU over the last 6 hours, for all routers in site X?

`raw data to the model`

RETURNED	**14,400** points · 40 routers × 360 min
SIZE	**150k+** tokens
TOOL CALLS	several: site, names, query, pages
MATH BY	the model, over thousands of numbers

`pre-computed tool`

RETURNED	**40** numbers
SIZE	**<1k** tokens
TOOL CALLS	1
MATH BY	the database

Access is not answers

ONE ROUTER, THREE NAMES

source of truth

`X-EDGE-01`

metrics database

`x-edge-01.mgmt.example.net`

log aggregator

`192.0.2.1`

same box? the model has to guess.

One MCP server per API = the model does the joins. Every join is a chance to be wrong.

Access is not answers

ONE ROUTER, THREE NAMES

source of truth

`X-EDGE-01`

metrics database

`x-edge-01.mgmt.example.net`

log aggregator

`192.0.2.1`

same box? the model has to guess.

One MCP server per API = the model does the joins. Every join is a chance to be wrong.

- Different names in every system

Access is not answers

ONE ROUTER, THREE NAMES

source of truth

`X-EDGE-01`

metrics database

`x-edge-01.mgmt.example.net`

log aggregator

`192.0.2.1`

same box? the model has to guess.

One MCP server per API = the model does the joins. Every join is a chance to be wrong.

- Different names in every system
- Counters vs rates, bits vs bytes, time zones

Access is not answers

ONE ROUTER, THREE NAMES

source of truth	`X-EDGE-01`
metrics database	`x-edge-01.mgmt.example.net`
log aggregator	`192.0.2.1`

same box? the model has to guess.

One MCP server per API = the model does the joins. Every join is a chance to be wrong.

- Different names in every system
- Counters vs rates, bits vs bytes, time zones
- Results silently truncated by pagination

Access is not answers

ONE ROUTER, THREE NAMES

source of truth	`X-EDGE-01`
metrics database	`x-edge-01.mgmt.example.net`
log aggregator	`192.0.2.1`

same box? the model has to guess.

One MCP server per API = the model does the joins. Every join is a chance to be wrong.

- Different names in every system
- Counters vs rates, bits vs bytes, time zones
- Results silently truncated by pagination
- The model doing arithmetic over thousands of values

Access is not answers

ONE ROUTER, THREE NAMES

source of truth

`X-EDGE-01`

metrics database

`x-edge-01.mgmt.example.net`

log aggregator

`192.0.2.1`

same box? the model has to guess.

One MCP server per API = the model does the joins. Every join is a chance to be wrong.

- Different names in every system
- Counters vs rates, bits vs bytes, time zones
- Results silently truncated by pagination
- The model doing arithmetic over thousands of values

> *...the same model that can't count the R's in "strawberry"* 🍓

CATEGORIES ONLY · THE LAYER DOES THE JOINS, NOT THE MODEL

The layer we built

Source of truth
devices · sites · roles

Metrics database
SNMP + telemetry

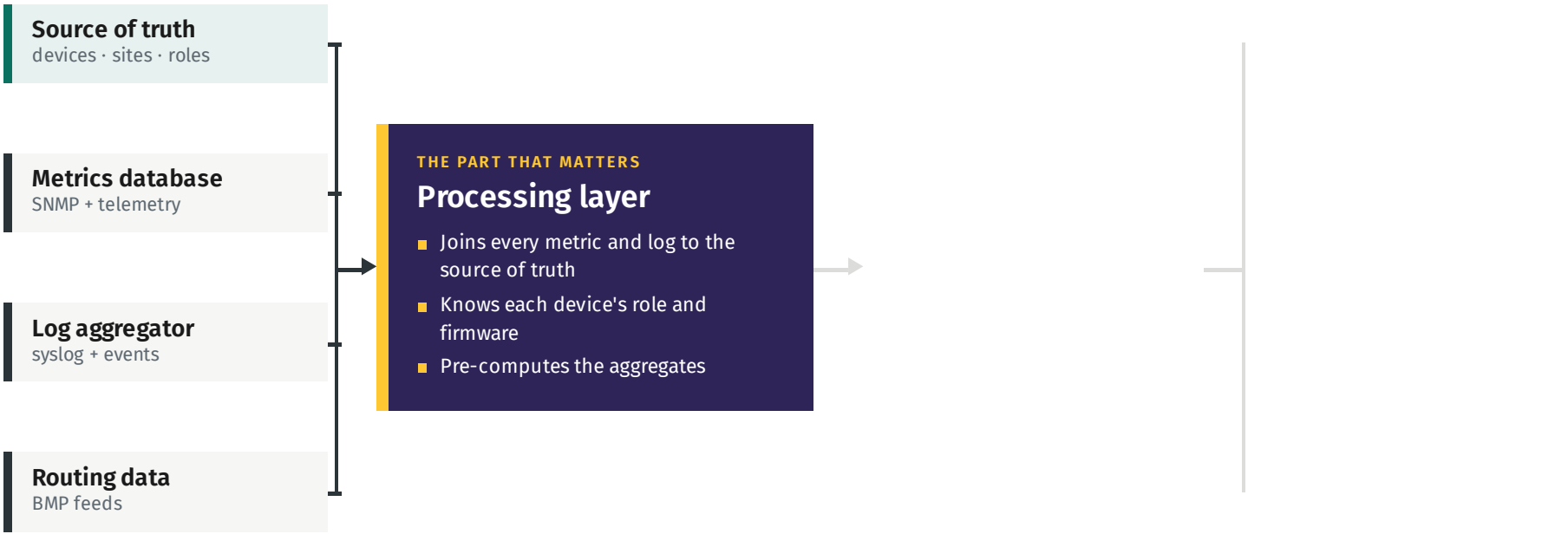
Log aggregator
syslog + events

Routing data
BMP feeds



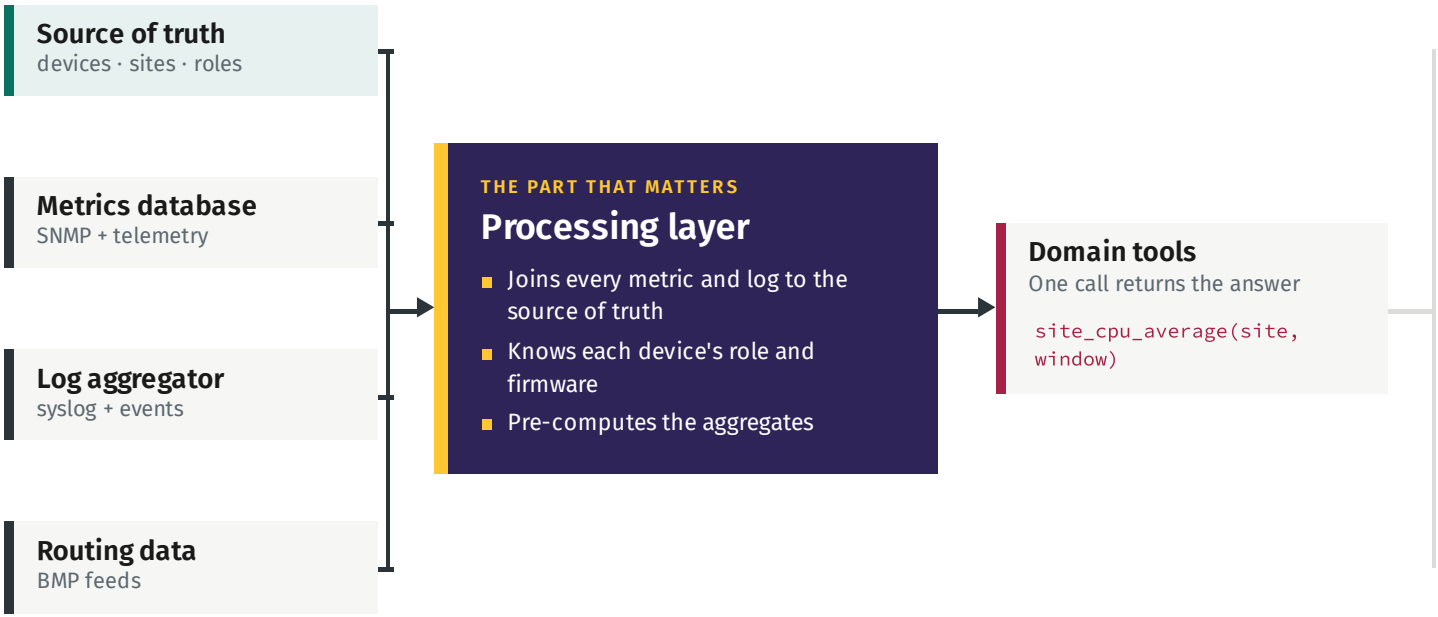
⚠ CATEGORIES ONLY · THE LAYER DOES THE JOINS, NOT THE MODEL

The layer we built



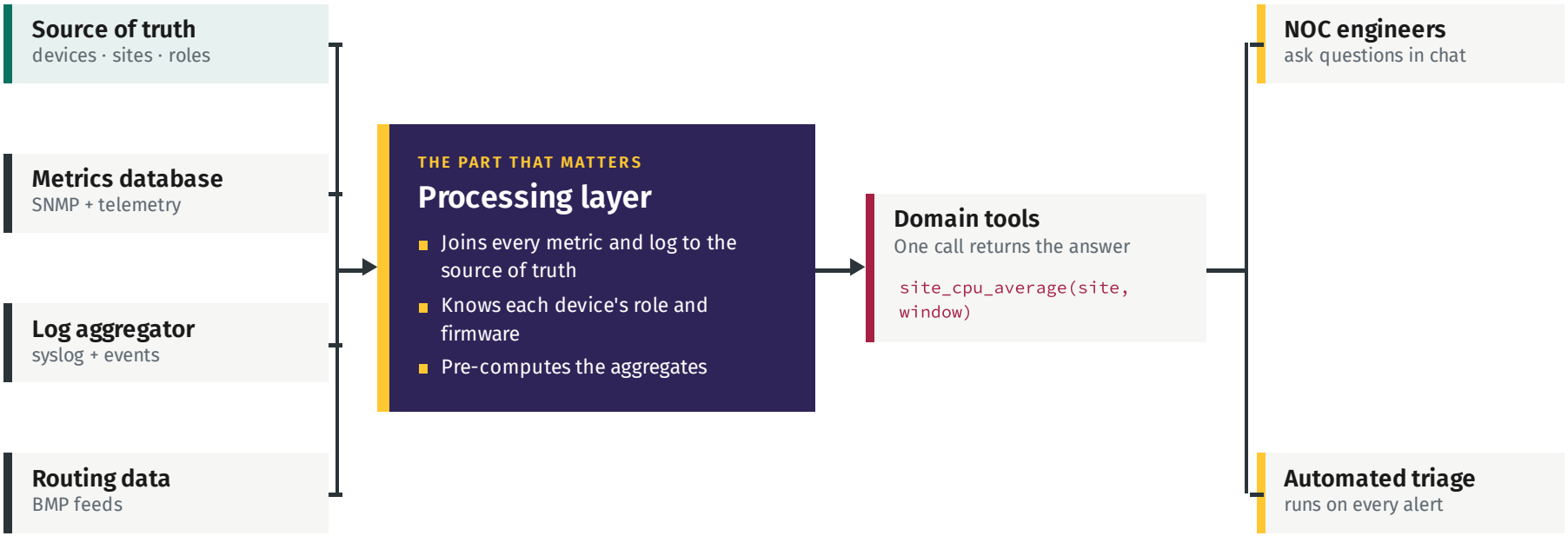
⚠ CATEGORIES ONLY · THE LAYER DOES THE JOINS, NOT THE MODEL

The layer we built



⚠ CATEGORIES ONLY · THE LAYER DOES THE JOINS, NOT THE MODEL

The layer we built



One question, one call

the raw path

```
site  = sot.get_site(name="X")
devs  = sot.list_devices(site=site.id, role="rout
names = [guess_metrics_name(d) for d in devs]
rows  = []
for page in range(1, ???):
    rows += metrics.query("cpu", names, "6h", pag
answer = model_does_math(rows)   # 14,400 values
```

One question, one call

the raw path

```
site  = sot.get_site(name="X")
devs  = sot.list_devices(site=site.id, role="rout
names = [guess_metrics_name(d) for d in devs]
rows  = []
for page in range(1, ???):
    rows += metrics.query("cpu", names, "6h", pag
answer = model_does_math(rows)   # 14,400 values
```

the domain tool

```
site_cpu_average(site="X", window="6h")
```

```
router       avg   max
x-edge-01    23%   41%
x-edge-02    19%   37%
x-core-01    31%   58%
x-core-02    28%   52%
… 36 more rows
```

One question, one call

the raw path

```
site  = sot.get_site(name="X")
devs  = sot.list_devices(site=site.id, role="rout
names = [guess_metrics_name(d) for d in devs]
rows  = []
for page in range(1, ???):
    rows += metrics.query("cpu", names, "6h", pag
answer = model_does_math(rows)   # 14,400 values
```

the domain tool

```
site_cpu_average(site="X", window="6h")
```

```
router        avg   max
x-edge-01     23%   41%
x-edge-02     19%   37%
x-core-01     31%   58%
x-core-02     28%   52%
… 36 more rows
```

AI is only as good as the tools you hand it.

Three things to take home

Three things to take home

01 **Monitoring support is poor**

SNMP plus scripts is a safe ceiling, not a strategy.

Three things to take home

01 Monitoring support is poor

SNMP plus scripts is a safe ceiling, not a strategy.

02 Collect everything, carefully

Decide what runs where by role and firmware. Never let a collector hurt a device.

TAKEAWAYS

Three things to take home

01 Monitoring support is poor

SNMP plus scripts is a safe ceiling, not a strategy.

02 Collect everything, carefully

Decide what runs where by role and firmware. Never let a collector hurt a device.

03 Feed AI answers

Don't pour raw data into a model. Pre-compute, join, then hand it the answer.

MDNOG

ITCARE

Questions?

Nicolai Moraru · Director of Software Engineering @ ITcare