Evolvable Network Telemetry at Facebook

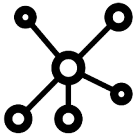
Yang Zhou*, Ying Zhang, Minlan Yu*, Guangyu Wang,
Dexter Cao, Eric Sung, Starsky Wong

Meta

* VE RI TAS HARVARD

Network Telemetry is Critical for Network Management


Alerting


Traffic engineering


Diagnosis

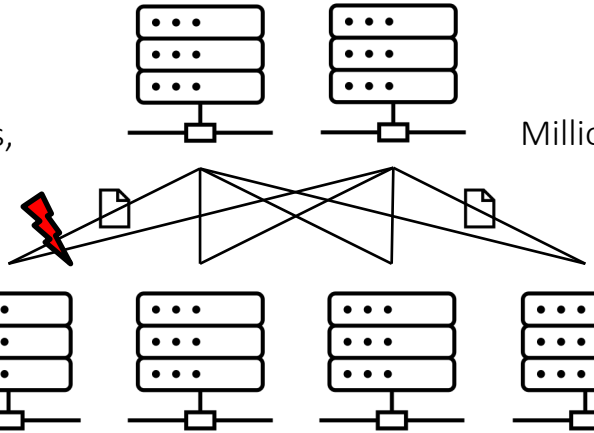

Troubleshooting


Verification


Asset tracking

Network Telemetry

Billions of time-series counters,
Deriving network topology



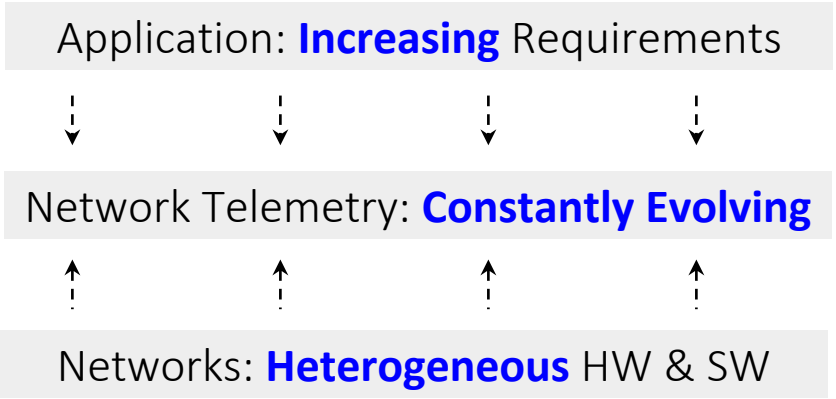
Millions of device interfaces

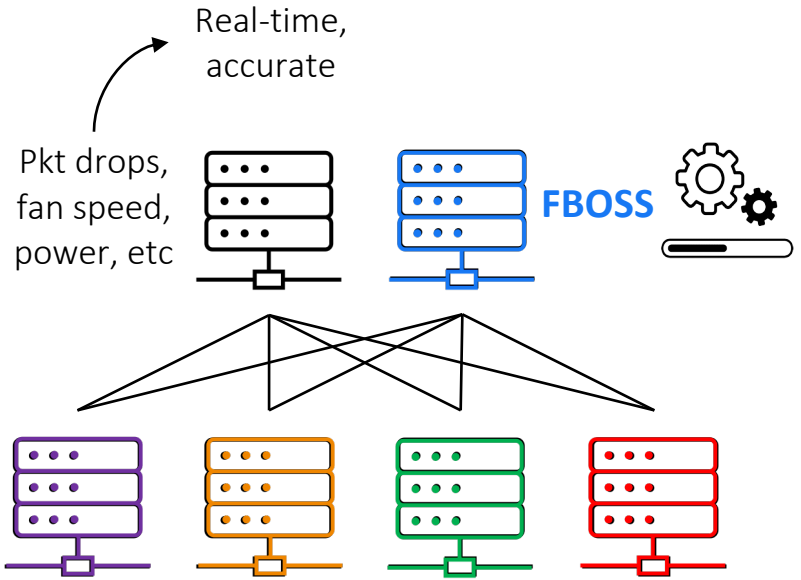
Data Centers,
Backbones, Edges

Hundreds of thousands of
switches/routers

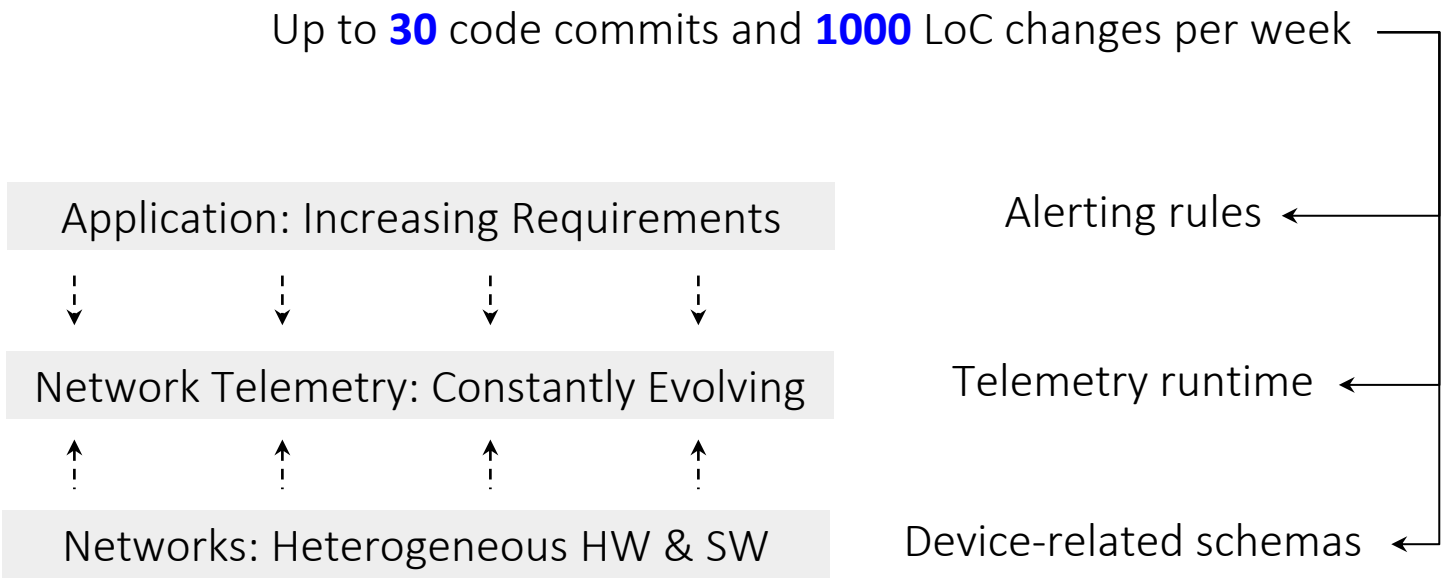
Key Challenge for Telemetry in Production: Evolvability

- Network devices and management applications are constantly evolving.

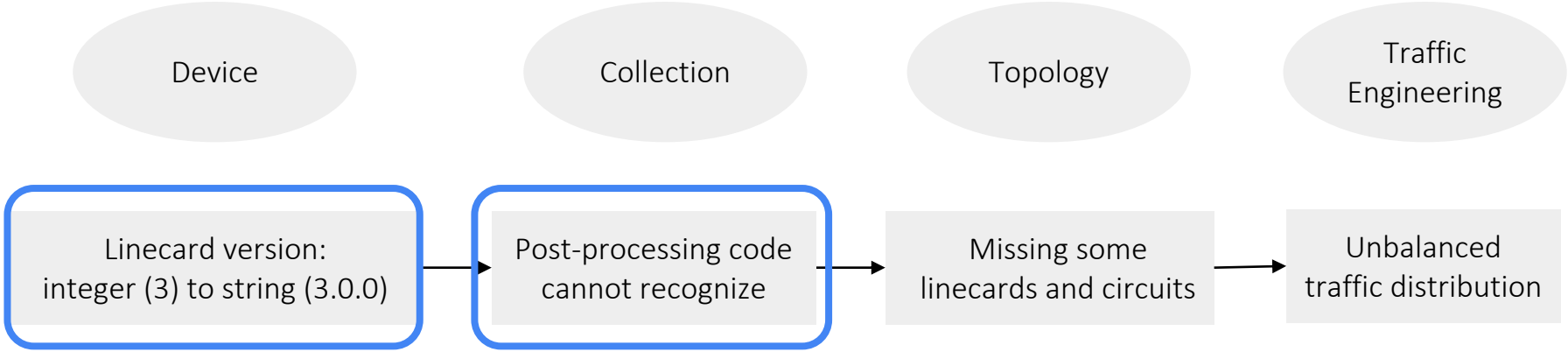




Magnitude of Changes



Incident 1: Changes Affect Many Components

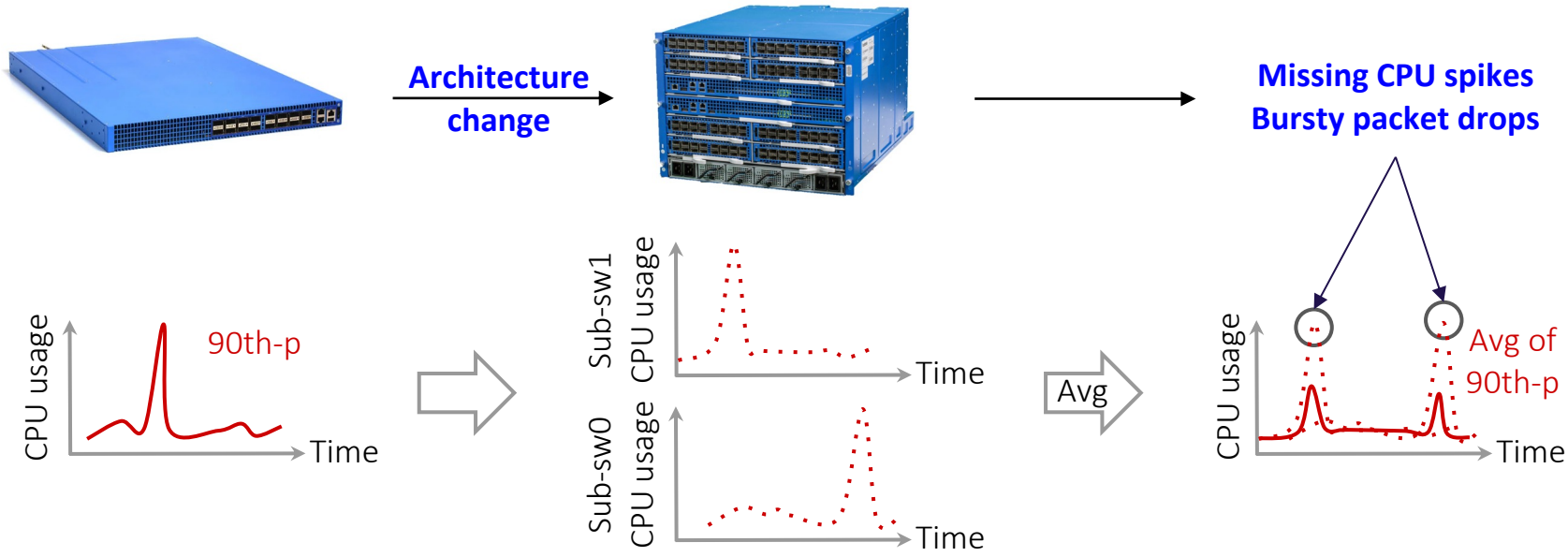


Takes multiple teams over **three days** to diagnose.

Even more frequent with FBOSS open switching

Frequent API *changes* of one component affect many other components

Incident 2: Data Misinterpretation

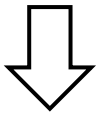


Frequent hardware and software *changes* affect data values and semantics

Bringing Changes to First-Class Citizens in Telemetry

Change Propagation

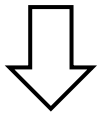
API changes affect many other components



Track API changes across components

Data Misinterpretation

Caused by HW and SW changes



Build **trustful** telemetry data despite changes

PCAT: Production Change-Aware Telemetry System

Change abstraction:

- ❖ Representing changes in a uniform and generic way
  - ✓ Track and use changes easily

Change attribution:

- ❖ Layering design to clearly attribute changes to the right components
  - ✓ Limit change propagation; track changes clearer

Change exploration:

- ❖ Allowing applications to explore change dependencies
  - ✓ Improve timeliness/accuracy

PCAT: Production Change-Aware Telemetry System

Change abstraction:

- ❖ Representing changes in a uniform and generic way
  - ✓ Track and use changes easily

Change attribution:

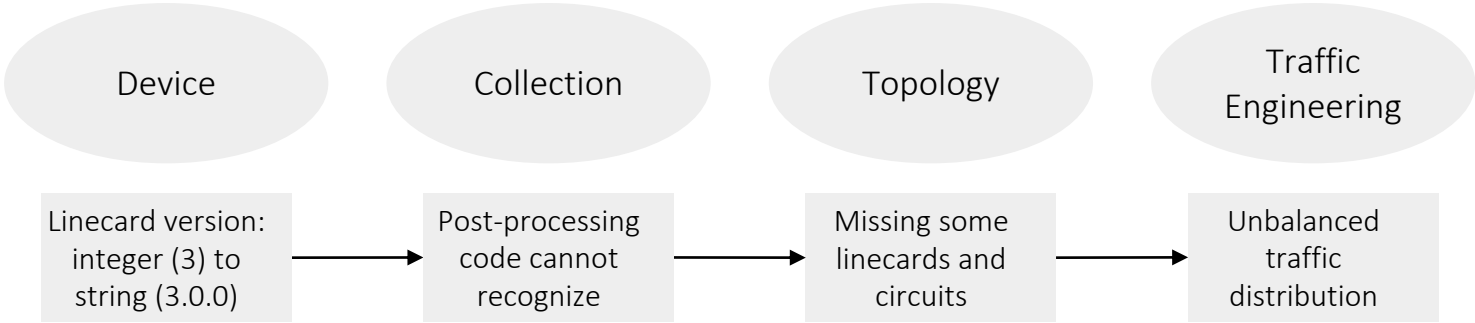
- ❖ Layering design to clearly attribute changes to the right components
  - ✓ Limit change propagation; track changes clearer

Change exploration:

- ❖ Allowing applications to explore change dependencies
  - ✓ Improve timeliness/accuracy

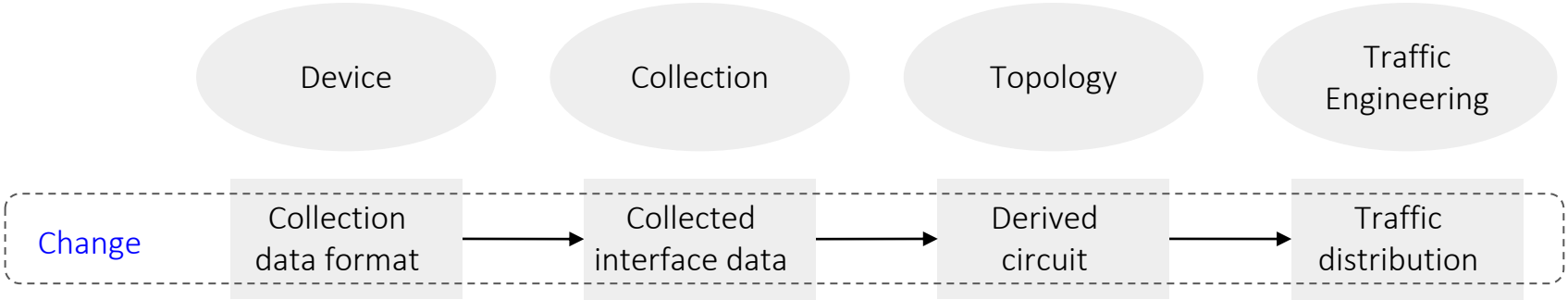
Change Abstraction: Change Cube

Consider the incident 1: Linecard Version Change → Unbalanced Traffic



Change Abstraction: Change Cube

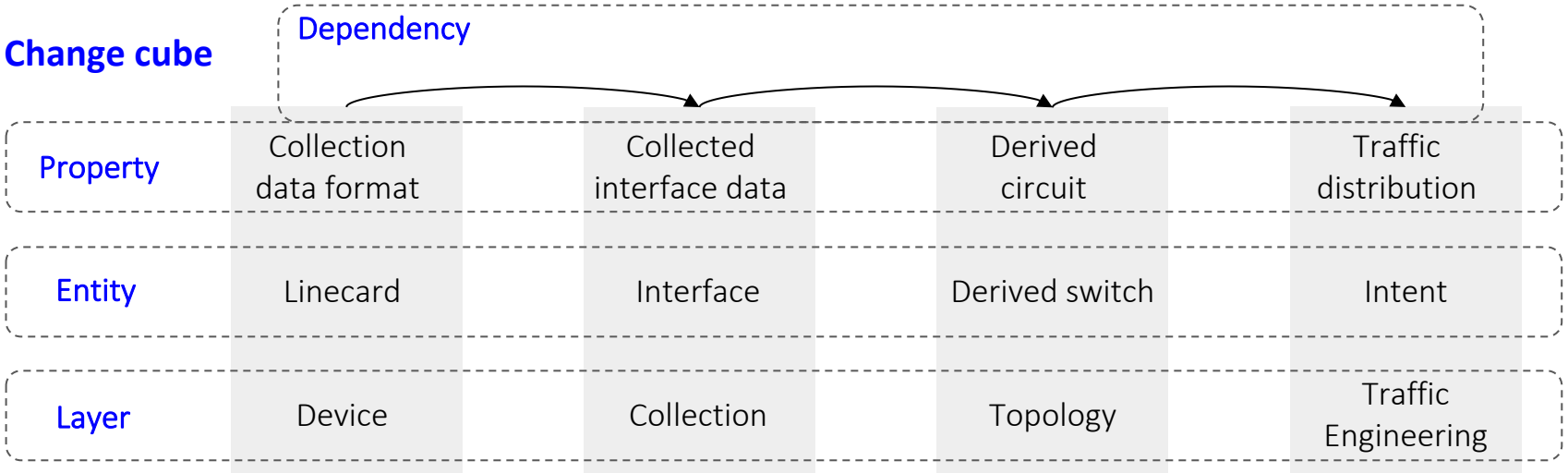
Consider the incident 1: Linecard Version Change → Unbalanced Traffic



Change Abstraction: Change Cube

Consider the incident 1: Linecard Version Change → Unbalanced Traffic

Change cube: <Time, Entity, Property, Layer, Dependency>



PCAT: Production Change-Aware Telemetry System

Change abstraction:

- ❖ Representing changes in a uniform and generic way
  - ✓ Track and use changes easily

Change attribution:

- ❖ Layering design to clearly attribute changes to the right components
  - ✓ Limit change propagation; track changes clearer

Change exploration:

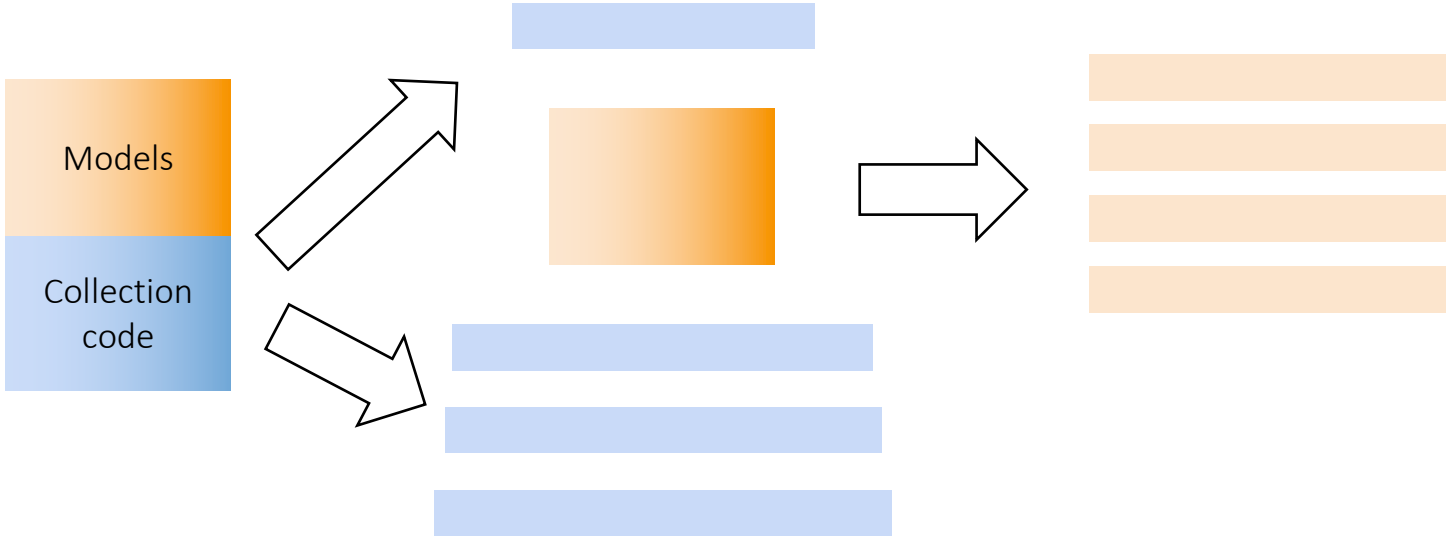
- ❖ Allowing applications to explore change dependencies
  - ✓ Improve timeliness/accuracy

Change Attribution: Three Generations of Telemetry Systems

Gen1: Deeply-coupled script (models + collection code)

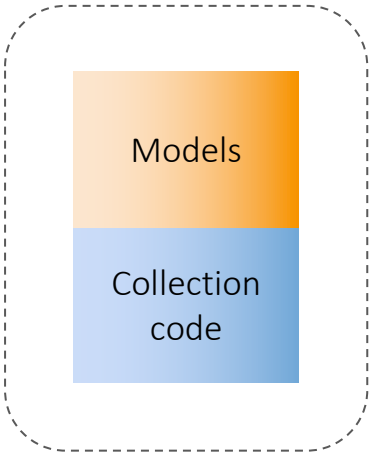
Gen2: Semi-modular

Gen3: Fully modular



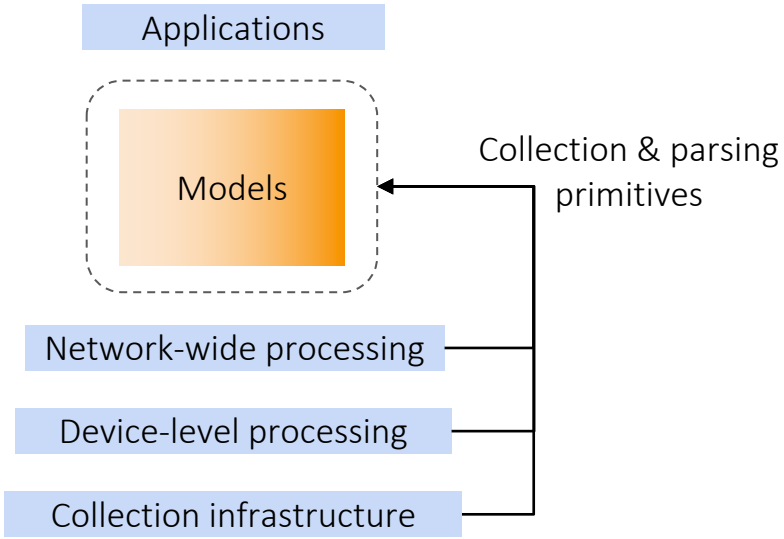
Change Attribution: Gen1 and Gen2

Gen1: Monolithic collection script



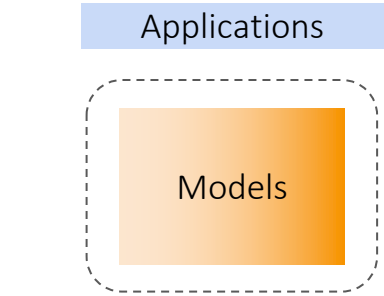
✗ Changes all over the place.

Gen2: Decoupled models from collection code



✓ Confine changes to one of the two layers.
✓ Track changes of two layers separately.

Change Attribution: Gen2's Problems



 # models =
intents ×
vendors ×
device groups

10k =
100 alarm rules ×
10 vendors ×
10 (rack, spine, etc)

✗ Enormous number of models.

✗ Intents are deeply coupled with the vendor-dependent details: models become hard to define and evolve.

Change Attribution: Gen3 (PCAT) Layering Design

Gen3: Limiting the impact of changes

Intent models

```
Alert if Interface.pkt_drops.Rate() > 1k/s
```

Data models

```
ModelDef(name='Interface',
    properties=[
        PropertyDef(name='pkt_drops', type=INT), ...
```

Collection models

```
Vendor1 CLI: show interfaces {$if_name} drops
Vendor2 Thrift: getQueueDrops({$queues})
```

Job models

```
JobDef(model_name='Interface',
    device_group='Rack switches',
    frequency='5min', ...
```

Change Attribution: Gen3 (PCAT) Layering Design

Gen3: Limiting the impact of changes

Intent models

Data models

Collection models

Job models

- ✓ Limit change propagation, eg, easy to adjust collection frequency.
- ✓ Track change clearer, ie, to specific layer.

PCAT: Production Change-Aware Telemetry System

Change abstraction:

- ❖ Representing changes in a uniform and generic way
  - ✓ Track and use changes easily

Change attribution:

- ❖ Layering design to clearly attribute changes to the right components
  - ✓ Limit change propagation; track changes clearer

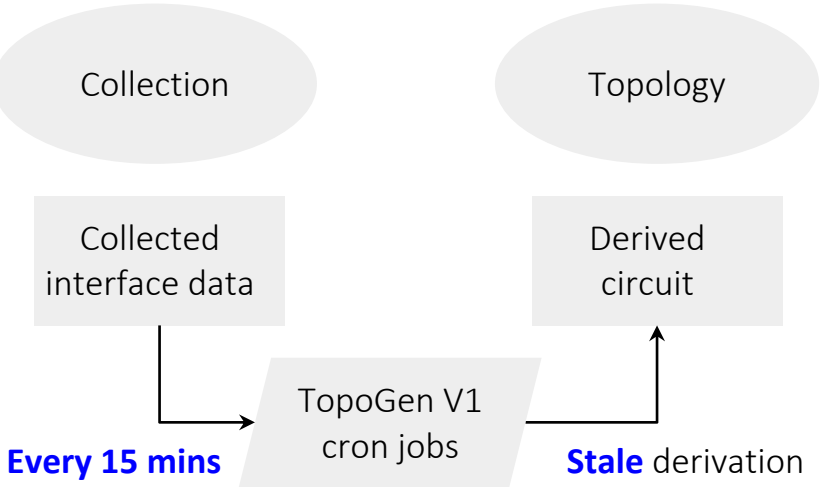
Change exploration:

- ❖ Allowing applications to explore change dependencies
  - ✓ Improve timeliness/accuracy

Change Exploration: Topology Derivation

Creates derived topology from normalized device-level data.

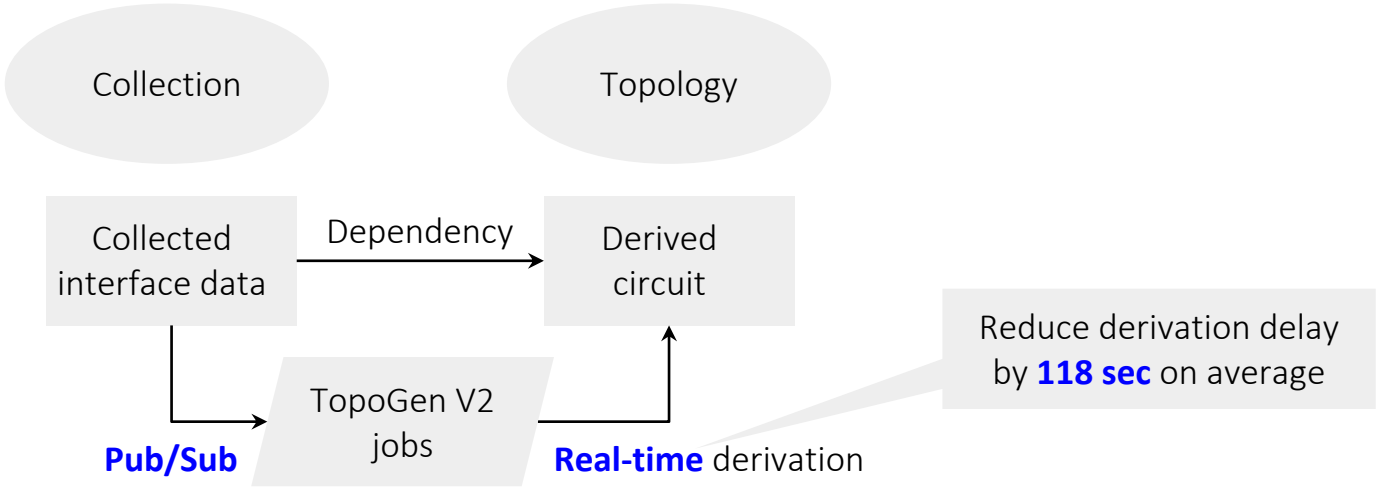
TopoGen V1:



Change Exploration: Topology Derivation

Creates derived topology from normalized device-level data.

TopoGen V2:



PCAT: Production Change-Aware Telemetry System

Change abstraction:

- ❖ Representing changes in a uniform and generic way
  - ✓ Track and use changes easily

Change attribution:

- ❖ Layering design to clearly attribute changes to the right components
  - ✓ Limit change propagation; track changes clearer

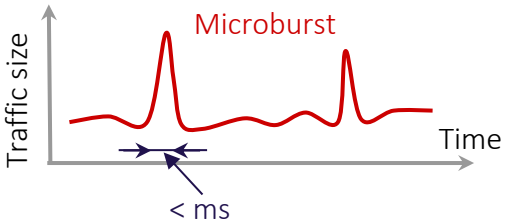
Change exploration:

- ❖ Allowing applications to explore change dependencies
  - ✓ Improve timeliness/accuracy

Open Questions: Adaptive Telemetry Primitives

Efficiency VS. **Adaptivity**

New microburst detection solution



Only work for certain devices/vendors

Vendor A ✗

Vendor B ✓

Vendor C ✗

Partial knowledge causes **complexity** for applications

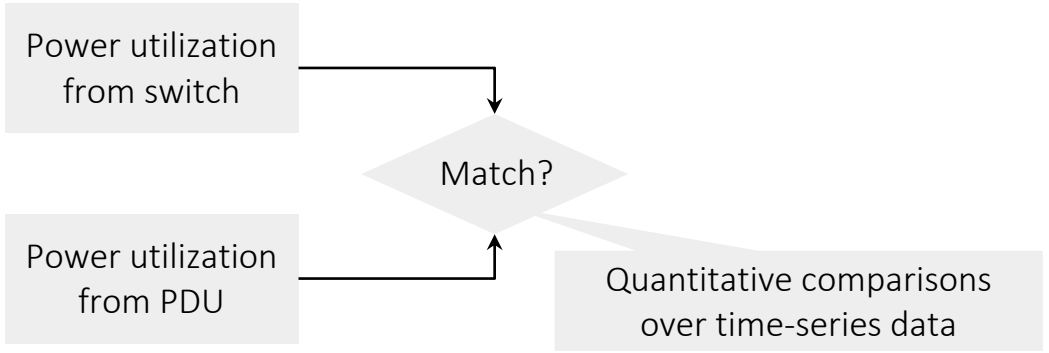
Widely adapt to various resource/programming conditions

Need both efficient and adaptive telemetry primitives

Open Questions: Trustful Telemetry

- Telemetry data may get missed/corrupted in evolving environment.
- Business-critical applications (eg, TE) rely on correct telemetry data.

One opportunity: **cross-validations** for counters

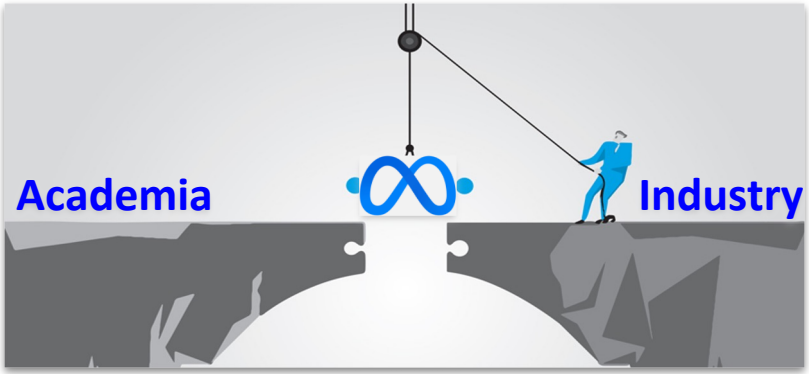


Need telemetry verification and validation

Summary

- ❖ Telemetry is critical for network management.
- ❖ Changes should be first-class citizens in evolvable telemetry.
- ❖ PCAT: Production change-aware telemetry system
  - ✓ **Change abstraction:** change cubes.
  - ✓ **Change attribution:** layering design.
  - ✓ **Change exploration:** change-aware applications.

Algorithm
Language
Verification



Evolvability
Scale
Heterogeneity

Thank You!

Icons from Flaticon.com, pngitem.com, veryicon.com, onlinewebfonts.com