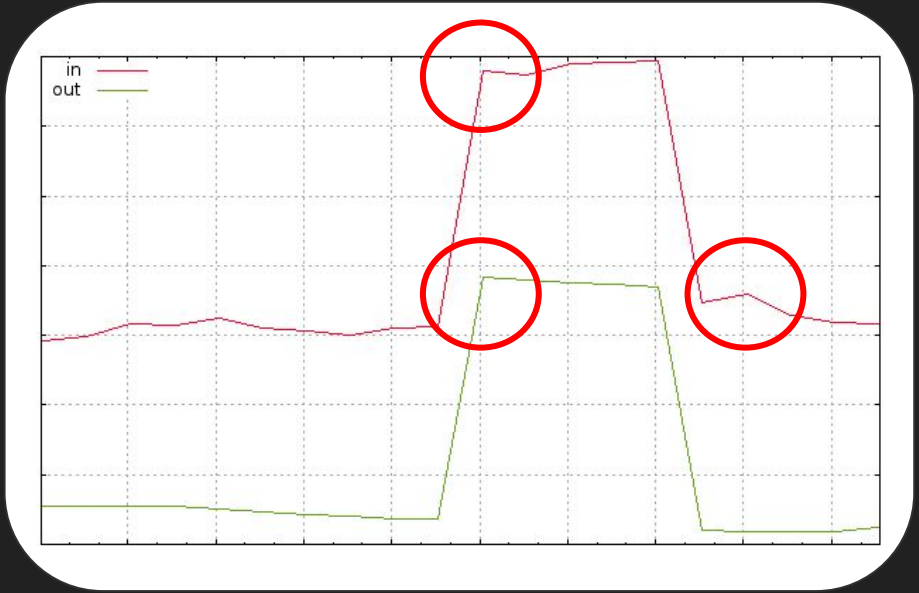
SNMP is dead

Carl Lebsack, Rob Shakir

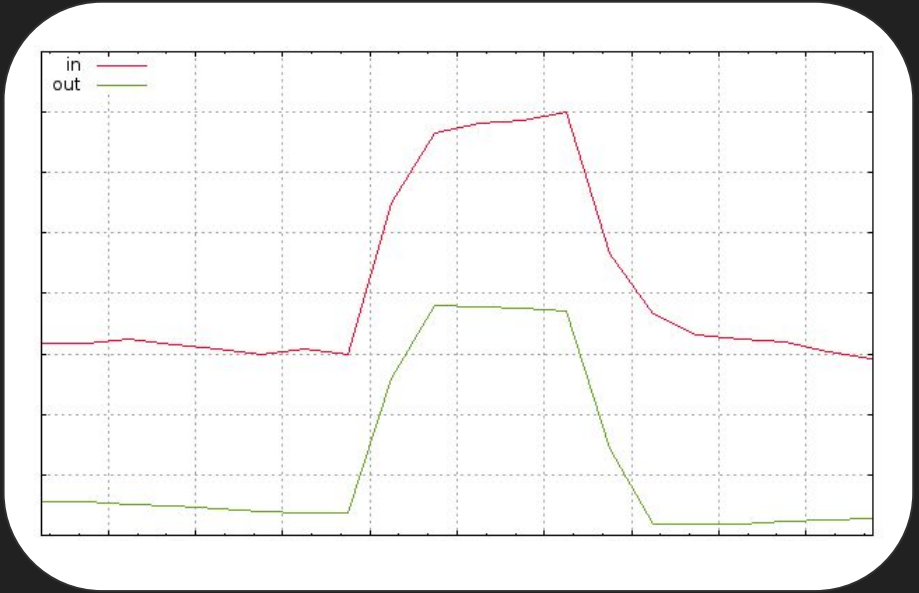
Google

Lack of source timestamp limits SNMP usefulness

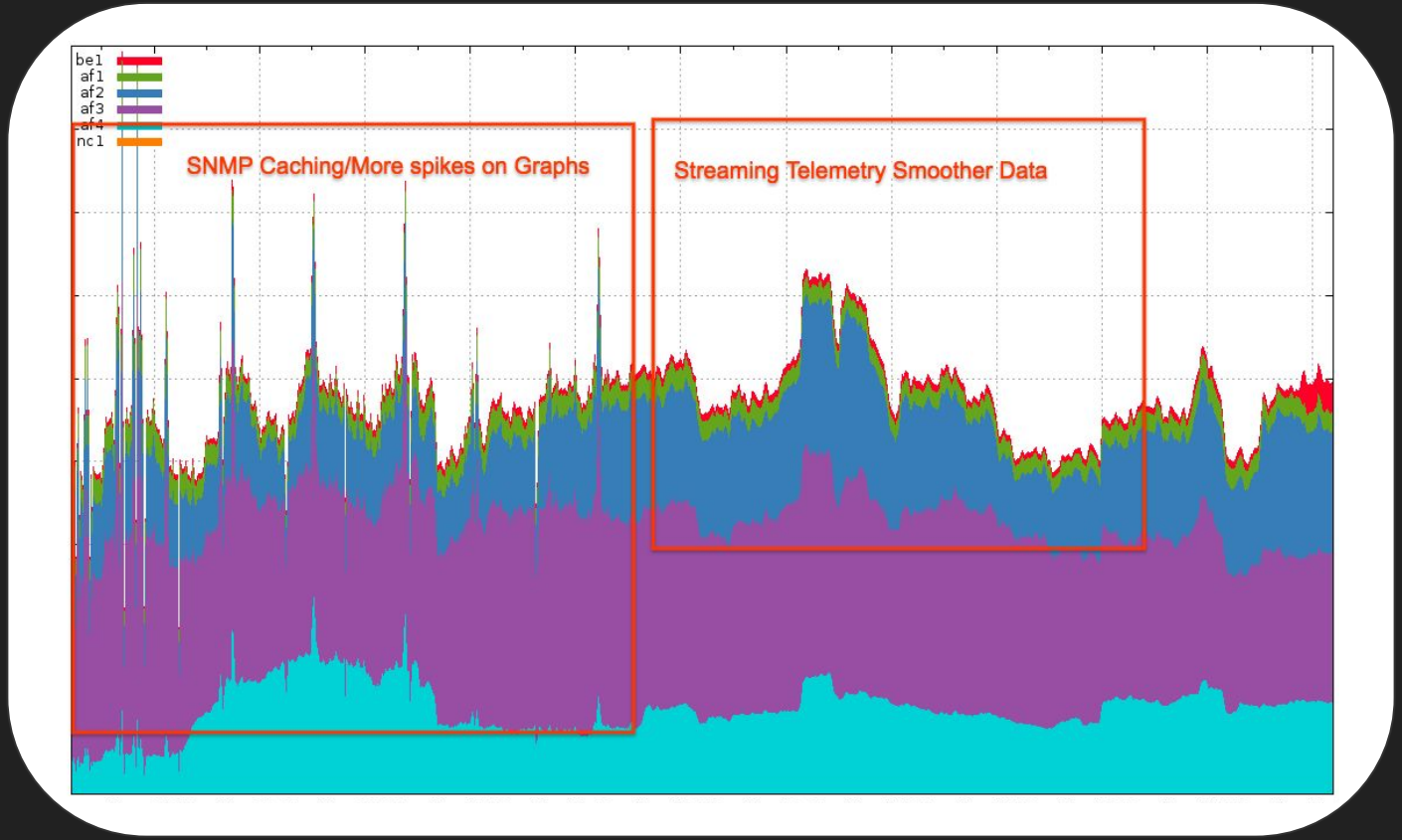
Computing rates on SNMP data introduces artifacts



Streaming Telemetry with source timestamps yields accurate rates

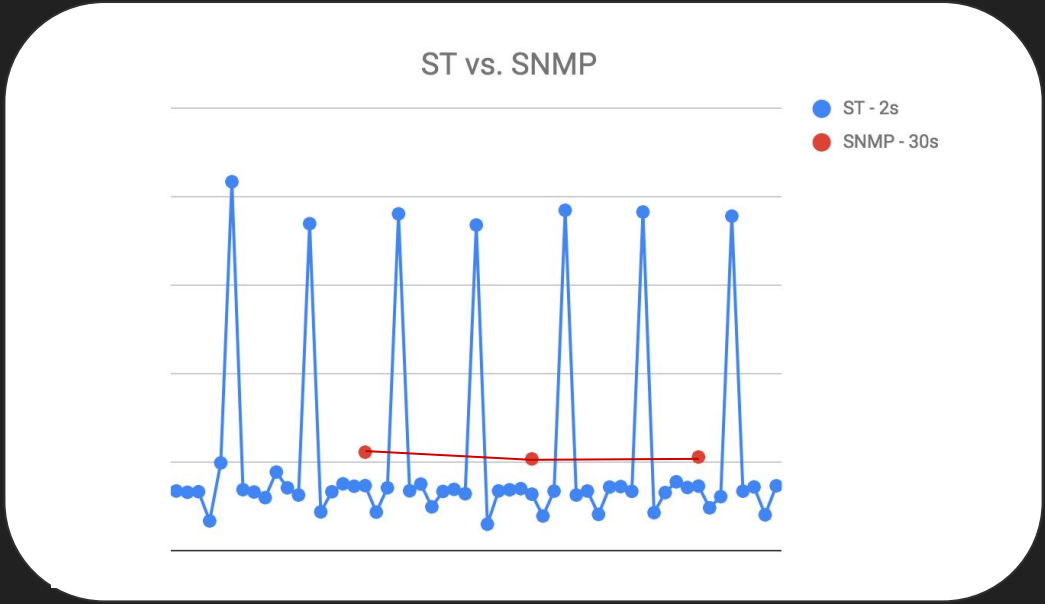


Under heavy load, SNMP artifacts become more severe



SNMP doesn’t scale

- SNMP timeouts on devices with 1/10th full interface count at 30 seconds
- Aim for fidelity at the granularity of seconds



SNMP scale is off by more than two orders of magnitude!

SNMP doesn’t (reliably) enable low-latency updates

Polling is bad

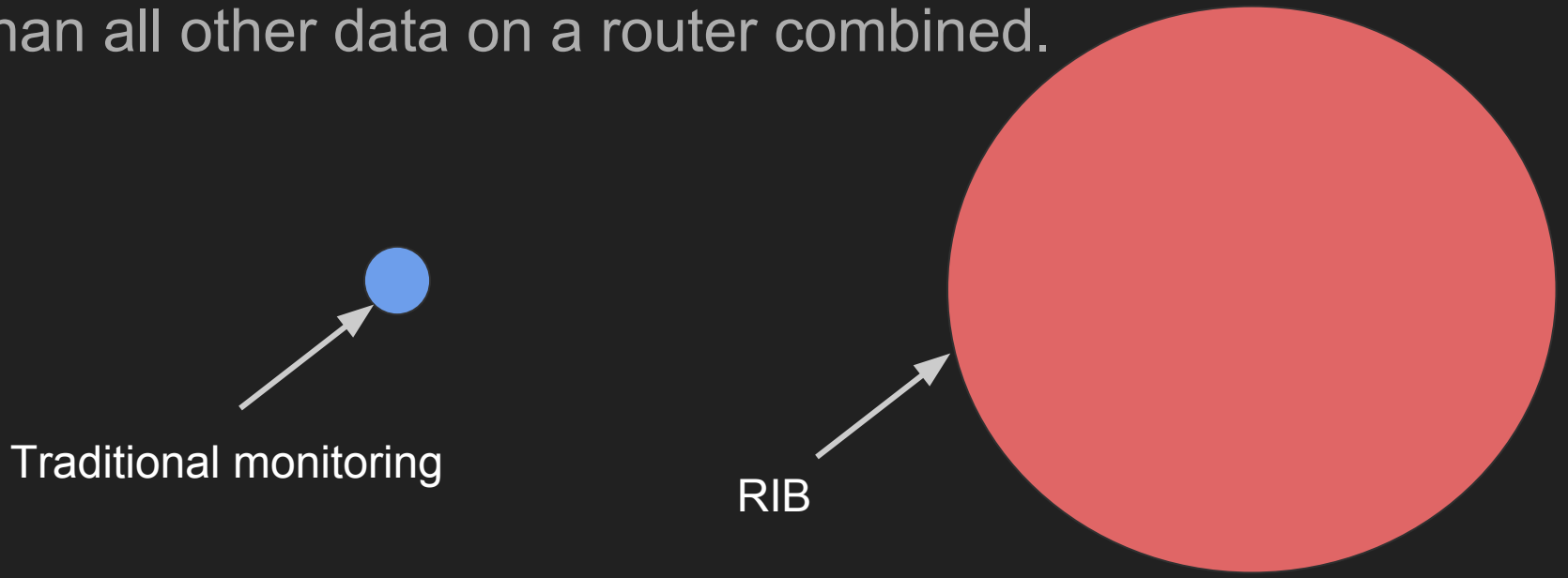
- Servicing multiple asynchronous requests adds latency
- Clients that want to reduce latency poll more frequently

How would one attempt to detect an interface going down within 1 second?

1. Poll every interface every second? **NO - affects scale**
2. SNMP traps? **NO - unreliable delivery**
3. Streaming Telemetry events - **YES!**

SNMP cannot support certain data sets

A routing table can be HUGE, two orders of magnitude larger than all other data on a router combined.



What is Streaming Telemetry?

- Source-timestamped
- Event-driven
- Subscription-based
- Eventually consistent state delivery to multiple clients
- gRPC Network Management Interface (gNMI)
- Part of OpenConfig standardization effort

Streaming Telemetry is real

Deployed at Google

30% of vendor supplied production devices

Millions of updates per second

How is streaming more efficient?

Device scheduled delivery

- Batch collect data from local sources
- Batch send to multiple clients

Event-driven

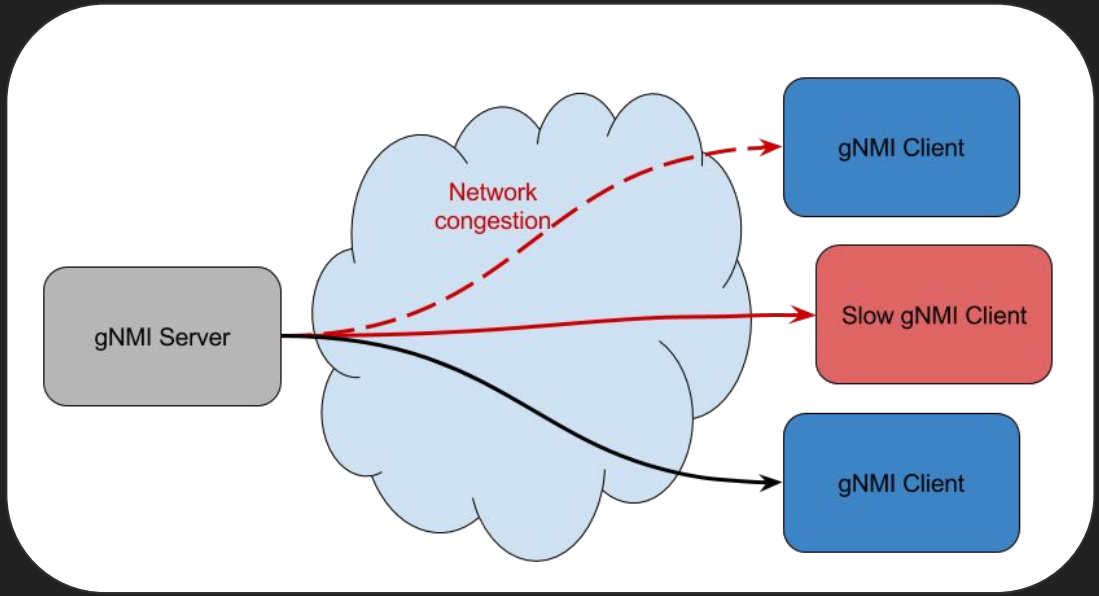
- Push as soon as the data changes (low latency)
- Push only when the data changes (low throughput)

Why implement Streaming Telemetry with gRPC?

- Persistent connection enables event-driven telemetry
- Connection provides implicit heartbeat
- Encrypted over TLS
- Protocol buffers for cross-platform compatibility
- Forward compatible protocol for new data sets

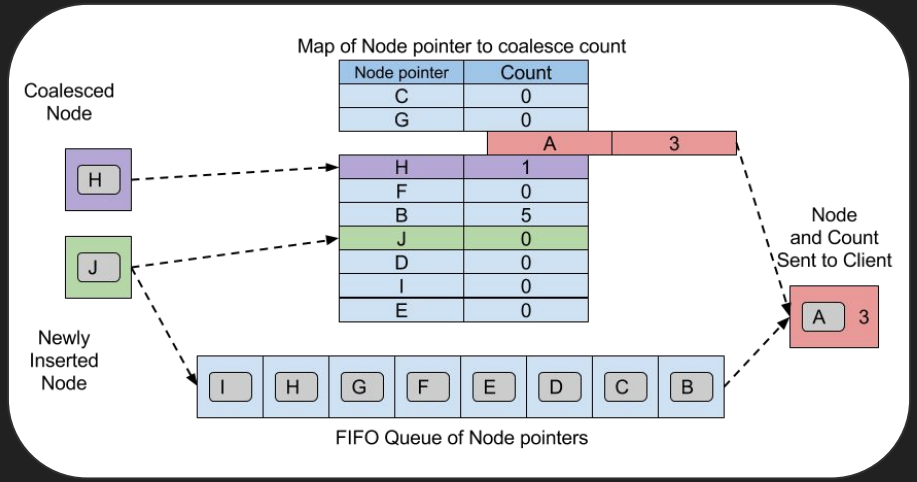
What about congestion?

- UDP has a “nice” property of being discarded when congestion occurs
- TCP (and therefore gRPC) has guaranteed delivery with retry



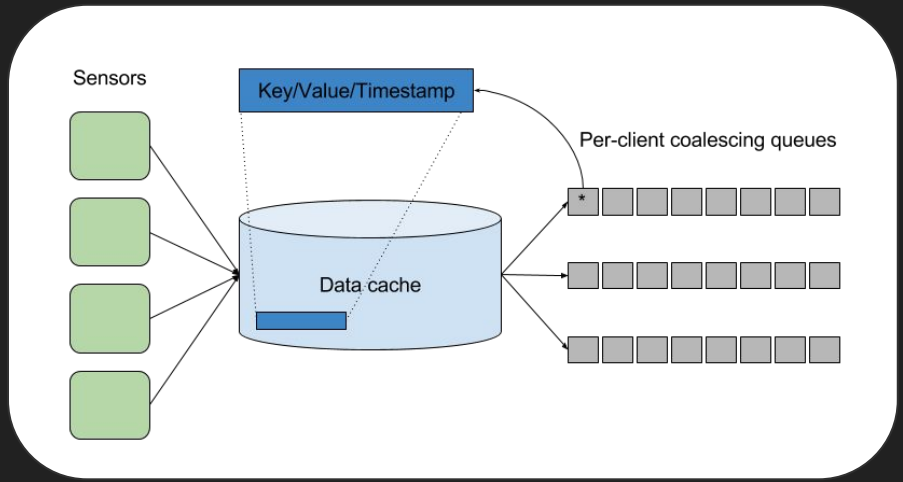
Reliable delivery results in queueing - with negative effects on performance.

Solution: Use application-level queue coalescion



- Detect backpressure with gRPC blocking send
- Deliver only the freshest data
- Maintain a coalescing queue per client

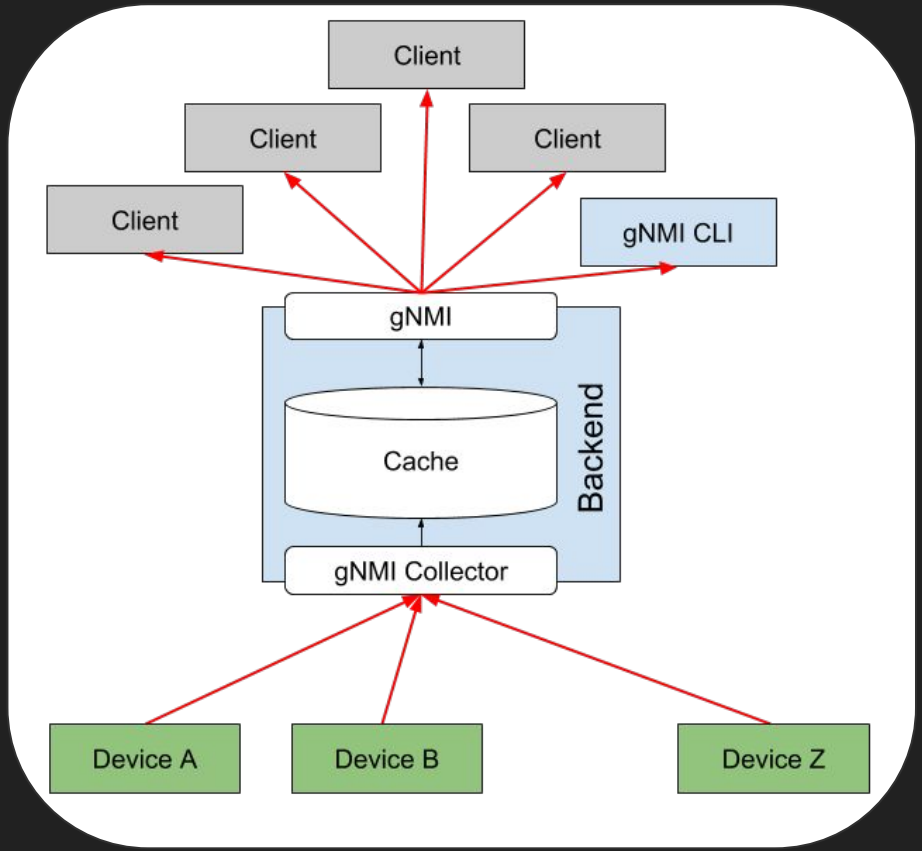
No need to extract details from this slide. There is an Open Source reference implementation!



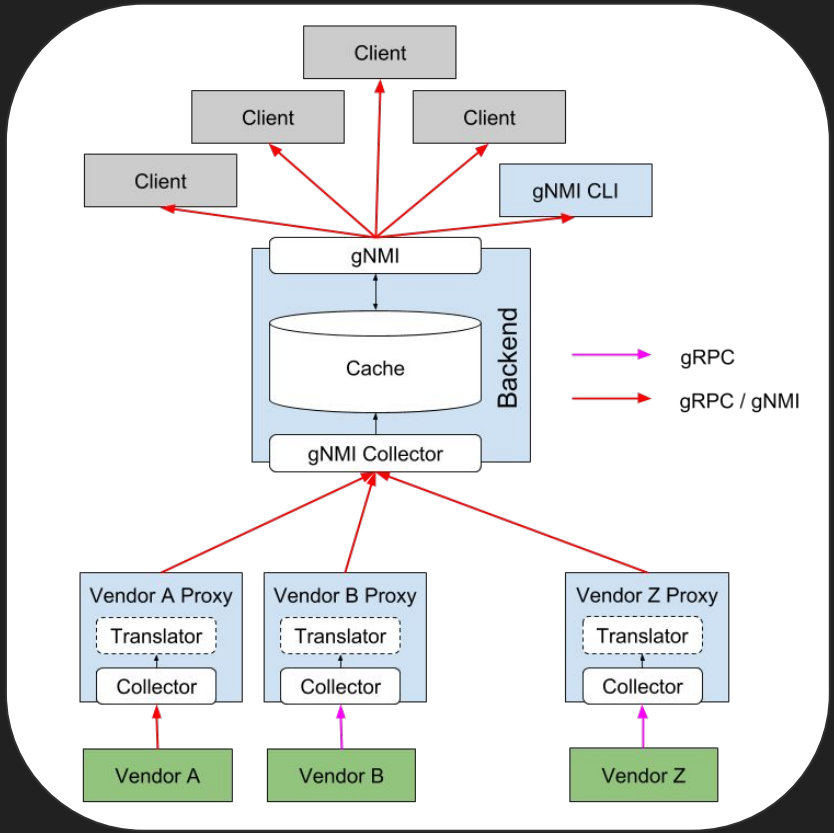
Open source collection framework

http://github.com/openconfig/gnmi

- Command-line interface
- Caching collector
  - Collect from multiple gNMI sources
  - Support multiple gNMI clients
  - Separates client from direct device access



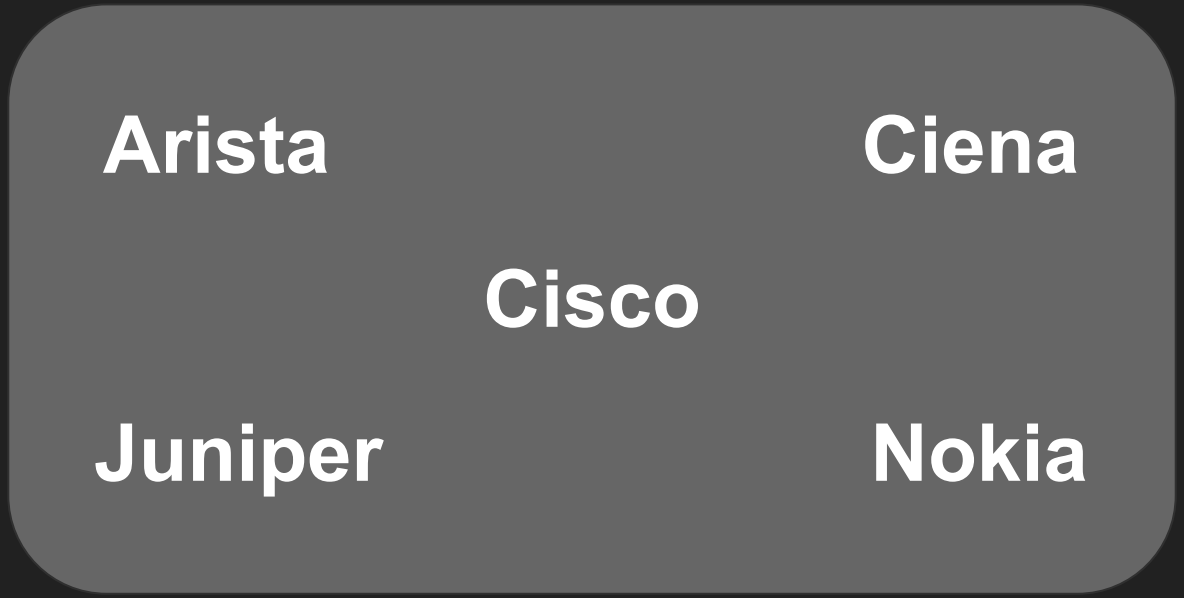
How did we deploy? Chicken and Egg problem



- Translation proxies
- Stateless and stateful mapping of data between vendor-native and OpenConfig models
- Target deployment is proxy-free

Streaming Telemetry via gNMI is available today!

The following vendors have ST via gNMI in release or EFT firmware images.



Thank You

http://openconfig.net

http://github.com/openconfig/gnmi