Network Telemetry Architecture at Roblox

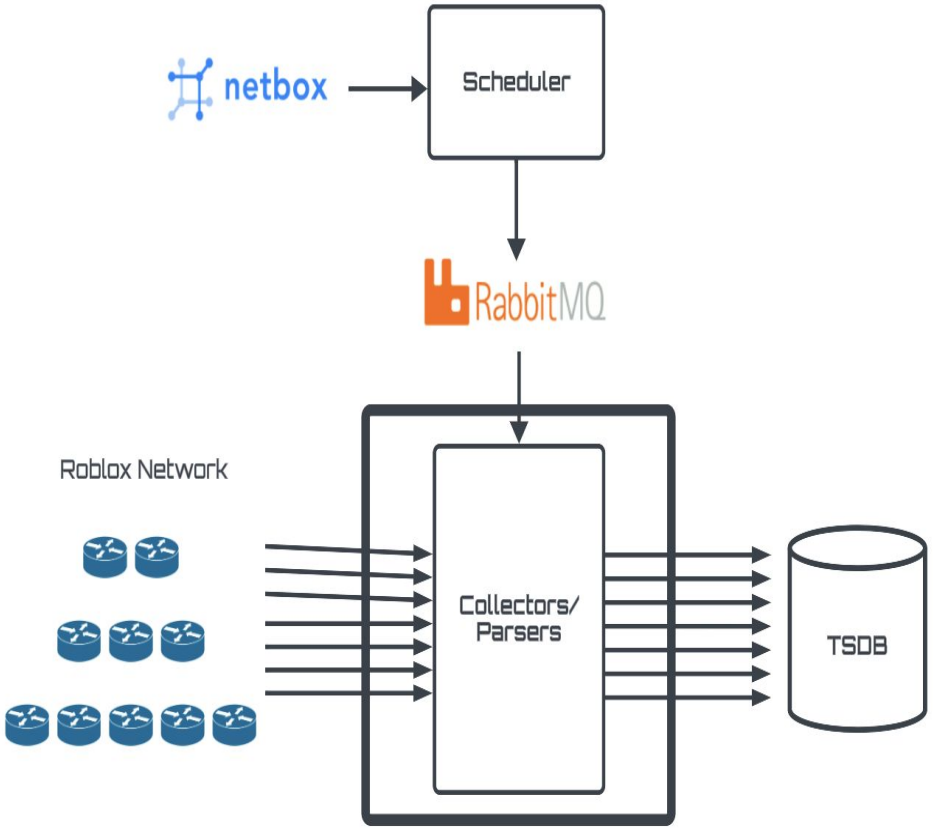
ROBLOX

Table of Contents

Legacy Architecture

Legacy Architecture

- Scheduler distributes unique device list across collectors to prevent split brain problem
- Collectors fetch metrics from network devices
- Parses intended metrics using user defined yaml files
- Sends it to time-series database



Legacy Architecture

- Written in Python
- Collection Interval - 2mins
- Collection Types
  - REST API and Netconf
- Concurrency using threading
- Keeps a connection open with rabbitmq to fetch unique device list

Legacy Architecture

What pushed us to explore a new system?

- Need of a scalable system that can handle the rapidly growing Roblox network
- Network team wanted granular collections to reduce MTTD of network failures
- We needed a reliable system to prevent data gaps in time series database

What are the requirements for our scalable collector?

- Scalable architecture (duh!)

What are the requirements for our scalable collector?

- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality

What are the requirements for our scalable collector?

- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality
- 30 second/one minute collections

What are the requirements for our scalable collector?

- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality
- 30 second/one minute collections
- Compiled Language to increase efficiency with expertise in the team

What are the requirements for our scalable collector?

- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality
- 30 second/one minute collections
- Compiled Language to increase efficiency with expertise in the team
- Good concurrency

What are the requirements for our scalable collector?

- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality
- 30 second/one minute collections
- Compiled Language to increase efficiency with expertise in the team
- Good concurrency
- Better instrumentation and easier to maintain

What are the requirements for our scalable collector?

- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality
- 30 second/one minute collections
- Compiled Language to increase efficiency with expertise in the team
- Good concurrency
- Better instrumentation and easier to maintain
  - statically typed, rich libraries, prioritize error handling, expose extensive metrics

What are the requirements for our scalable collector?

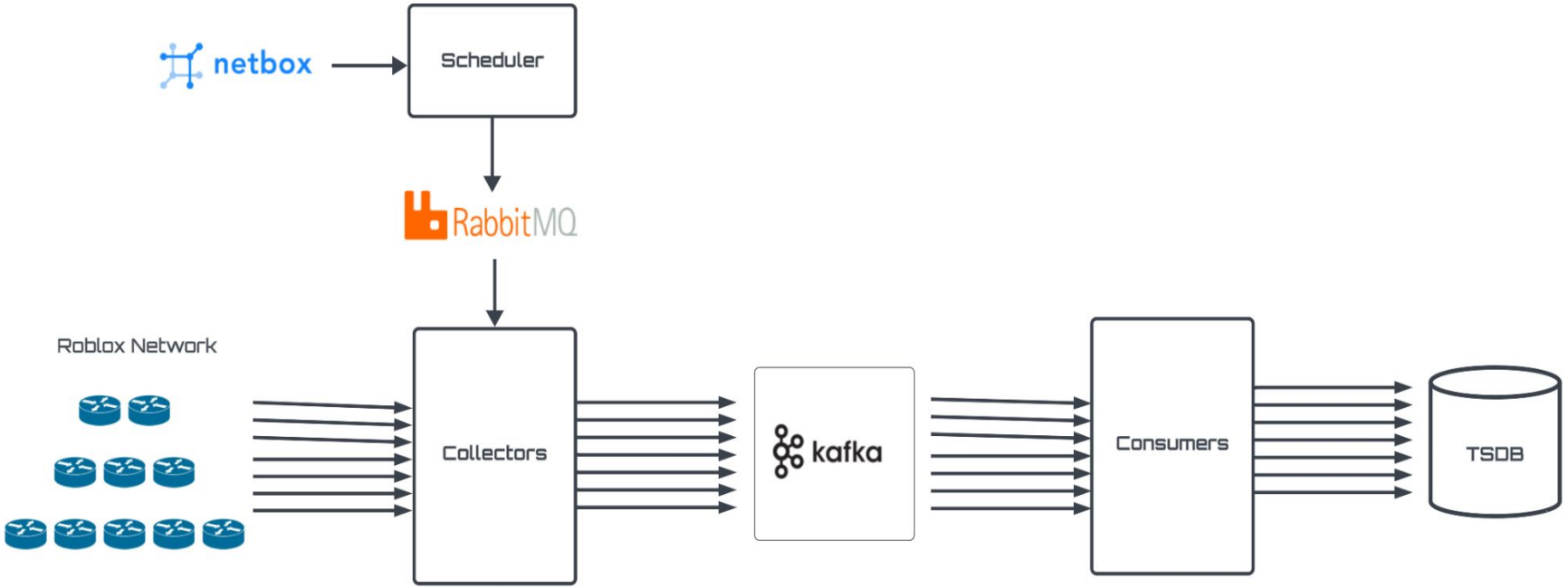
- Scalable architecture (duh!)
  - disaggregate metric collection and parser functionality
- 30 second/one minute collections
- Compiled language to increase efficiency with expertise in the team
- Good concurrency
- Better instrumentation and easier to maintain
  - statically typed, rich libraries, prioritize error handling, expose extensive metrics

Kafka

Go programming

Next-Gen Architecture

Next-Gen Architecture



Next-Gen Architecture

Collection Types

- gNMI sample mode
- REST API
- Netconf

Why are we using three collection types?

- gNMI does not support all the operational data we need
- native gNMI models do not follow openconfig convention and standards

Next-Gen Architecture

Collector

- Reads a config file with all the show commands/paths that needs to be run for network devices
- Fetches metrics from network devices using different collection types and sends it to kafka

Next-Gen Architecture

Collector Command Configuration

```
vendor_interface_stats:
    paths: [/interfaces/interface/state]
    type: subscribe
    tags: [vendor]
    topic: gnmi.interface.counters
    interval: 30s
```

Next-Gen Architecture

Kafka

- Each kafka topic corresponds to one show command/path
- Each kafka topic can be scaled using partitions

Next-Gen Architecture

Consumer

- Fetches metrics from all the subscribed topics in kafka
- Each parser corresponds to one kafka topic
- Parses the intended metrics using a parser yaml file and pushes it to time series database

Next-Gen Architecture

Consumer Configuration

```
[backends.influx.topics."gnmi.interface.counters"]
  parser = "interface_counters"
  measurement = "interface_counters"
  parsertype = "gnmi"
```

Next-Gen Architecture

Pros

- Faster collections and granular data
- Reliable signals for network events
- Reliable system that can handle hundreds of devices

Cons

- More components in the new architecture
- Slower deployment

Takeaways

Takeaways

- Evaluate your telemetry systems regularly to see if they can incorporate future growth

Takeaways

- Evaluate your telemetry systems regularly to see if they can incorporate future growth
- Do not rely on gNMI for all your data. It’s adoption is still in its early days

Takeaways

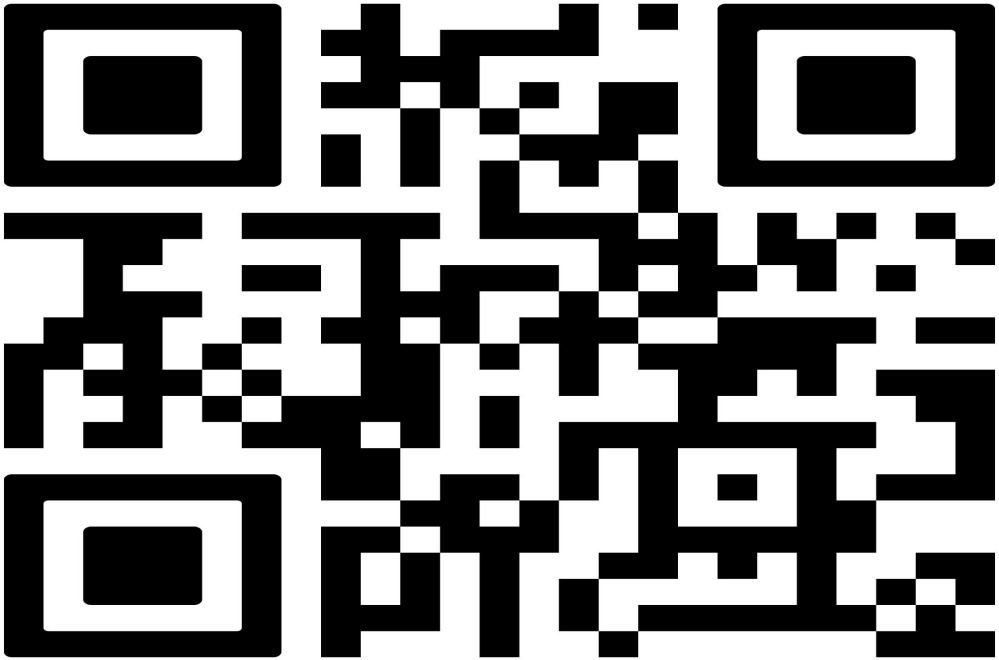
- Evaluate your telemetry systems regularly to see if they can incorporate future growth
- Do not rely on gNMI for all your data. It’s adoption is still in its early days
- Maintaining software becomes easier if you have better instrumentation and tests

Shoutout

- **Mayuresh Gaitonde** - Principal Network Reliability Engineer
  - Initiated the telemetry system project
  - Mentored me throughout the project

- **Brandon Bennett** - Principal Software Engineer
  - Created an open source netconf library in golang that is being actively used in production

https://github.com/nemith/netconf



Q&A

ROBLOX