facebook
Robotron: Top-down Network Management at Facebook Scale

Yu-Wei Eric Sung, Xiaozheng Tie, Starsky H.Y. Wong, Hongyi Zeng
ACM SIGCOMM 2016
August 25, 2016
Scale of Facebook Community

1.7 Billion on Facebook Monthly
1 Billion on Whatsapp Monthly
500 Million on Instagram Monthly
1 Billion on Messenger Monthly
Network Management at Facebook

What’s involved?

- Goals: Build and evolve FB network
- Example tasks: circuit/device turnup, network monitoring
- Human interactions -> outages
Network Management at Facebook

Why is it hard?

• Distributed Configurations
• Multiple Domains
• Versioning
• Dependency
• Vendor Differences
Network Management at Facebook

Early days...

Manual Configuration and Monitoring with ad-hoc scripts
Contribution

Manual Configuration and Monitoring with ad-hoc scripts

2004-2007
2008
2009
2010
2011
2012
2013
2014
2015

Robotron started

• Shed light on
  • Network management tasks
  • Robotron’s usage
  • Evolution of Robotron
  • Our experiences using Robotron

Our Paper
Overview of Facebook’s Network

Lifecycle of user requests
Point of Presence (POP)

• Standardized topology
• Services: LB, Cache
• Common tasks
  • Build/upgrade a cluster
  • Provisioning new peering circuits
**Backbone**

- Irregular, demand-driven topology
- Common tasks:
  - Add/migrate circuits
  - Add/remove routers
Datacenter

- Standardized topology
- Services: Web, Cache, Database
- Common tasks
  - Build/decomm a cluster
  - Cluster capacity upgrade
Overview of Facebook’s Network

Multiple versions of FB cluster architectures co-exist

POP

DC

8 generations
Robotron: “Top-Down” Network Management System@FB

Overview

Network Design → Config Generation → Deployment → Monitoring

FBNet

DB
FBNet: Modeling the Network

Example 4-post POP cluster
FBNet: Modeling the Network

Object

Networkswitch

Linecard

PhysicalInterface

AggregatedInterface

V6Prefix

BgpV6Session

Circuit
FBNet: Modeling the Network

Value

Networkswitch
- et1/1
- et1/2
- et2/1
- et3/1
- PR₁

Linecard
- name=PSWₐ
- slot=1
- model=X

PhysicalInterface
- name=et1/1
- name=et1/2

AggregatedInterface
- name=ae0
- prefix=2001::1

BgpV6Session
- Circuit
  - speed=10G

Circuit
- speed=10G
FBNet: Modeling the Network

It’s complicated
class PhysicalInterface(Interface):
    linecard = models.ForeignKey(Linecard)
    agg_interface = models.ForeignKey(AggregatedInterface)
FBNet Model Snippet

Related models

class PhysicalInterface(Interface):
    linecard = models.ForeignKey(Linecard)
    agg_interface = models.ForeignKey(AggregatedInterface)
FBNet Model Snippet

Model inheritance

class PhysicalInterface(Interface):
    linecard = models.ForeignKey(Linecard)
    agg_interface = models.ForeignKey(AggregatedInterface)
FBNet: Architecture

API Layer

- RPC services
  - Read: fine-grained per-model query
  - Write: task-based
- High Availability: Multiple replicas per DC
FBNet: Architecture

API Layer

• 1 primary, multiple secondary DBs
• Scalability: 1 slave per DC
Robotron’s management life cycle

- Network Design
- Config Generation
- Deployment
- Monitoring

FBNet

DB
Network Design

Design intent → FBNet objects

Template for a POP cluster

Cluster(
    devices={
        PR: DeviceSpec(
            hardware="Router_Vendor1"
            num_devices=2)
        PSW: DeviceSpec(
            hardware="Switch_Vendor2"
            num_devices=4)
    },
    Link_groups=
        LinkGroup(
            a_device=PR,
            z_device=PSW,
            pifs_per_agg=2,
            ip=V6)
)

FBNet objects

BackboneRouters: 2
NetworkSwitches: 4
Circuits: 16
PhysicalInterfaces: 32
AggregatedInterfaces: 16
V6Prefixes: 16
BgpV6Sessions: 8

94 objects across 7 models
**Config Generation**

FBNet objects $\rightarrow$ Device configs

```go
struct Device {
  1: list<AggregatedInterface> aggs,
}
struct AggregatedInterface {
  1: string name,
  2: i32 number,
  3: string v4_prefix,
  4: string v6_prefix,
  5: list<PhysicalInterface> pifs,
}
struct PhysicalInterface {
  1: string name,
}
```
### Config Generation

FBNet objects → Device configs

```plaintext
{% for agg in device.aggs %}
  interface {{agg.name}}
  mtu 9192
  no switchport
  load-interval 30
  {% if agg.v4_prefix %}
  ip addr {{agg.v4_prefix}}
  {% endif %}
  {% if agg.v6_prefix %}
  ipv6 addr {{agg.v6_prefix}}
  {% endif %}
  no shutdown
{% endfor %}
```

- **Vendor 1**
  - Interface template
  - BGP template
  - MPLS template
  - PR1 config
  - PR2 config

- **Vendor 2**
  - Interface template
  - BGP template
  - MPLS template
  - PSW_a config
  - PSW_b config

**FBNet objects**

**Vendor agnostic**

**Vendor Specific**

**Vendor-specific Device Configs**
Config Generation

FBNet objects $\rightarrow$ Device configs

```plaintext
{% for agg in device.aggs %}
  interface {{agg.name}}
  mtu 9192
  no switchport
  load-interval 30
  {% if agg.v4_prefix %}
    ip addr {{agg.v4_prefix}}
  {% endif %}
  {% if agg.v6_prefix %}
    ipv6 addr {{agg.v6_prefix}}
  {% endif %}
  no shutdown
{% endfor %}
```
Config Generation

FBNet objects → Device configs

```plaintext
{% for agg in device.aggs %}
interface {{agg.name}}
  mtu 9192
  no switchport
  load-interval 30
  {%- if agg.v4_prefix %}
  ip addr {{agg.v4_prefix}}
  {%- endif %}
  {%- if agg.v6_prefix %}
  ipv6 addr {{agg.v6_prefix}}
  {%- endif %}
  no shutdown
!
{%- endfor %}
```
Config Generation
FBNet objects → Device configs

```%
{% for agg in device.aggs %}
interface {{agg.name}}
  mtu 9192
  no switchport
  load-interval 30
  {% if agg.v4_prefix %}
  ip addr {{agg.v4_prefix}}
  {% endif %}
  {% if agg.v6_prefix %}
  ipv6 addr {{agg.v6_prefix}}
  {% endif %}
  no shutdown
!
{% endfor %}
```
Usage Statistics

- # of FBNet model change?
- # changed FBNet objects per design change?
- Frequency and size of config change?
FBNet Model Changes

How much does FBNet model change over time?

- Still many changes over time
- Reasons: new models, values, relationships
Design Changes

How many FBNet object are changed per design change?

POP/DC

Backbone
Design Changes

How many FBNet objects are changed per design change?

- POP/DC: bigger design changes
- Backbone: smaller design changes
Configuration Changes

What’s the frequency and size of configuration change?

• Median number of config lines changed per week
  • POP/DC devices: 500 lines
  • Backbone devices: <100 lines

• Avg number of times changes happen per week
  • POP/DC devices: 2.53
  • Backbone devices: 12.46

POP/DC: few bigger config changes
Backbone: many smaller config changes
Evolution of Robotron

Bottom-up, experience driven
Experience: Modeling is laborious

Problem Scenario: new eBGP session configuration

- A new eBGP session needed a proper import policy.
- Robotron was used without proper support.
- Egress link saturated.
- Most development time spent on model changes.

- Lesson: Modeling is hard.
- Open problem: Lack of a network model widely accepted by vendors.
Experience: Coupling changes is key

Problem Scenario: POP cluster switch turnup

1. An engineer updated FBNet to add a new rack, but forgot to generate the config.
2. The engineer pushed the stale config.
3. The rack added never came online.

- Lesson: Network design, config generation, and deployment should be tightly coupled.
- Open problem:
  - Atomicity
  - Conflict resolution
Experience: Fallback is important

Problem Scenario: Robotron-less management

- Engineer bypassed Robotron to manually configure devices
- SSH into device
- Make config change
- Log out
- Needed upon emergencies
- Passively curtail with config monitoring

Lesson: Bypassing mechanism is needed

Open problem:
- How to reliably account for such activities?
- How to safely revert such activities?
Conclusion

• First work sharing experience on a production network management system

• Open research problems:
  • Network modeling
  • Atomicity and conflict resolution across management tasks
  • Make network management system work with manual fallback mechanisms
Questions?

- robotron@fb.com
- Poster session on Thursday
Overview of Facebook’s Network

Backbone: Interconnecting POPs/DCs

- Irregular, demand-driven topology
- PRs/DRs form an iBGP mesh
- Common tasks:
  - Add/migrate circuits
  - Add/remove BBs/PRs/DRs
Overview of Facebook’s Network

Point of Presence (POP)

• Standardized topology
• Services: LB (Proxygen), Cache
• Common tasks
  • Build/upgrade a cluster
  • Provisioning new peering circuits
Overview of Facebook’s Network

Data Center

• Standardized topology
• Services: Web, Cache (TAO), Database
• Common tasks
  • Build/decomm a cluster
  • Cluster capacity upgrade
FBNet: Modeling the Network

Object, Value, and Relationship

[Diagram showing network objects and relationships including PSW_a, et1/1, et1/2, et2/1, et3/1, ae0, ae1, 2001::1, 2001::2, 10G, eBGP session, Linecard, Circuit, PhysicalInterface, AggregatedInterface, V6Prefix, BgpV6Session, a_endpoint, z_endpoint, a_prefix, z_prefix, speed=10G]
Dependencies between FBNet models

CDF across models vs. # of related models
Experience: Fallback is needed

Problem Scenario: manual changes to devices

- Manual config changes on devices are error-prone
- Ideal: All changes made through Robotron
- Reality: Robotron has latency, bugs and missing features. Quick fixes needed upon emergency
- Alternatives to discourage manual changes:
  - Config monitoring
  - Automatic config override after emergency window
Related Work

• Bottom-up config analysis: [Benson11, Sung09, Kim11, …]

• Abstraction-driven design and config generation:
  • Top down config optimization: [Condor, Sun13]
  • Centralized platform for network management: [Onix, Statesman]
  • Template based config generation: [Enck09]
  • Config modeling: [OpenConfig, DMTF]
FBNet: Modeling the Network

Desired versus Derived

![Diagram showing a comparison between Desired and Derived FBNet models.](image-url)
Deployment

Device configs ➔ Devices

- New device: full config replacement
- Existing devices: Incremental “Live” updates
  - Dryrun, Atomic, Phased, etc
Monitoring

Is the network healthy?

• Passive monitoring
• Active monitoring
• Config monitoring