Scalable Control Plane Substrate

Sachin Katti, John Ousterhout, Guru Parulkar, Marcos Aguilera, Curt Kolovson
ONRC + ON.Lab + RAMCloud, VMWare
Motivation

• Separation of control plane is a common trend: networks/systems
  – Challenging to design, hard requirements on throughput, latency, consistency

• We have been building control planes for our specific systems
  – ONOS, SoftRAN, and RAMCloud Coordinator

• Can we design a generalized scalable control plane with
  – A common foundation that can be customized for different contexts
Examples of Control Planes

• SDN Controllers
  – Data center, enterprise, service provider wired and cellular networks
  – IoT
• VM manager (e.g. Nova in OpenStack)
  – Data Centers
• Name servers (“name node”)
  – distributed file system
• Scheduler
  – Many distributed systems
• Software defined storage
• ...
New “Control” Planes – Drones, Cars

- Controlling fleets of drones and cars
  - Collision avoidance, sharing the airspace
  - Air traffic control for drones
  - Logistics and delivery
Separation of Control Plane

Open Interface(s)

Control Plane

App 1
App 2
App 3

Global View

Open Interface

Separation of Control Plane

Switches

eNBs

Servers

Storage
Logically Centralized Control Plane

- Provides global view
- Makes it easy to program control, management, config apps
- Enables new apps
Key Performance Requirements

High Volume of State:
~500GB-2TB

High Throughput:
~500K-20M ops / second
~100M state ops / second

Low Latency to Events:
1-10s ms

Difficult challenge!

High throughput | Low latency | Consistency | High availability
Control Plane Challenges

- Scalability, High Availability and Performance
- Northbound and Southbound Abstractions
- Modularity
Example Scalable Control Planes

ONOS
SoftRAN
RAMCloud Coordinator
Scale-Out Design w/ Multiple ONOS Instances

Each instance is identical
One can add and remove instances seamlessly
Each instance is a master for a subset of switches
It works like a single system for apps and network devices
ONOS Architecture Tiers

**Northbound Abstraction:**
- network graph
- application intents

**Core:**
- distributed
- protocol independent

**Southbound Abstraction:**
- generalized OpenFlow
- pluggable & extensible

**Northbound - Application Intent Framework**
(policy enforcement, conflict resolution)

**Distributed Core**
 scalability, availability, performance, persistence

**Southbound**
(discover, observe, program, configure)

- OpenFlow
- NetConf
- ...

...
ONOS Application Intent Framework

I want to define “what” I need without worrying about “how”...

“Provision 10G path from Datacenter 1 to Datacenter2 optimized for cost”

Intents translated and Compiled into specific instructions for network devices.
Distributed Core: State Management

- Topology
- Flows
- Intents
- Switch to controller mapping
- Resource allocations
- Network Configuration
- And a plethora of application generated state

Different states have different size and R/W patterns. They can use different consistency, replication, sharding.
Key to Performance/Scalability: Distributed State Management

**Application Intents**
- immutable
- durable & **replicated**

**Global Network View**
- eventually consistent
- fully replicated

**Flow Table Entries**
- strongly consistent
- partitioned

**Switch to Controller**
- strongly consistent
- replicated for durability

**Resource Assignment**
- strongly consistent
- partitioned for scale

**Optimistic Replication**
- gossip based
- anti-entropy

**Master/Backup Replication**
- gossip based
- anti-entropy
- partial ordering

**Consensus for strong consistency**
ONOS Performance Metrics and Results

- **Device & link sensing latency**
  - Less than 100ms
  - ONOS processing less than 10ms

- **Flow rule operations throughput**
  - 500K to 3M ops/sec

- **Intent operations throughput**
  - 200k ops/sec

- **Intent operations latency**
  - Less than 50ms
SoftRAN
**SoftRAN: Software Defined Radio Access Network**

- **eNBs + small cell eNBs + signal strengths from devices to neighboring eNBs**

**So#RAN:** Software Defined Radio Access Network

- **Macrocell**
- **Picocell**
- **Microcell**

**SoftRAN OS**

**Goal:**
- Maximize efficiency of precious spectrum
- Create personalized user experience
- Minimize capex/opex with rapidly increasing devices and their bandwidth requirements

- **LTE-A**
- **LTE-IoT**
- **MVNO**

As devices move, need to handover 50 Macro eNBs, 600 Micro eNBs. Signal strengths constantly change – with interference.
Control Plane Requirements for SoftRAN

Global Network View
- eNBs + small cell eNBs + signal strengths from devices to neighboring eNBs

Critical decisions:
- Handover
- Device to eNB assignment
- Power assignment

Allocation of 50 resource blocks per eNBs to mobile devices every ms

- 50 Macro eNBs, 600 Micro eNBs
- Up to 100K mobile devices; few flows per device
SoftRAN Key Performance Requirements

High Volume of State:
~500GB-2TB

High Throughput:
~500K-5M ops / second
~20M state ops / second

Low Latency to Events:
1-10s ms

Difficult challenge!

Order of magnitude higher throughput and lower latency requirement
Control Plane Challenges for SoftRAN: Simplification and Optimization

- LTE-A
- LTE-IoT
- MVNO

Global Network View
Multiple TBs
Updated every few ms

Opportunity for Parallelism
State management/processing can be done independently – not as tightly coupled

Only devices in transition require immediate processing

- 50 Macro eNBs, 600 Micro eNBs
- Up to 100K mobile devices; few flows per device
RAMCloud Coordinator
RAMCloud Architecture: Control Plane

1000 – 100,000 Application Servers

- Appl. Library

Datacenter Network

1000 – 10,000 Storage Servers

- Meta-state: mapping of key space to tables on servers, list of active servers, list of active client-leases
- App servers cache the state and so coordinator not in the critical path
RAMCloud Coordinator: Requirements

High Volume of State:
~500GB

Medium Throughput:
~M updates per second

Low Latency to Response:
usec to msec

100,000 App Servers
10,000 Servers
Generalized and Customizable Scalable Control Plane
What is domain specific and what isn’t?

- **Domain independent**
  - State management with configurable consistency, availability
  - Streaming event processing
  - Graph abstractions

- **Domain specific**
  - Northbound and southbound interfaces
  - Notification APIs
Generalized and Customizable Scalable Control Plane

Northbound Abstraction:
- Provide different APIs
- Customize for the context

Core:
- distributed
- context independent

Southbound Abstraction:
- Plug-ins for different contexts

Northbound Abstractions/APIs
(C/C++, Declarative Programming, REST)
Strongly consistent, transaction semantics?

Distributed Core
Cluster of Servers, 10-100Gbps, low latency RPC
Distributed State Management Primitives

Southbound
Plug-in for different contexts

Apps

OpenFlow/NetConf
RAN X2 Interface
RPC

Switches
eNBs
Servers
Storage
Time Varying Streaming Graph Abstraction

• Common state abstraction for many control planes ⇒ Graph
  – E.g: In networks, graphs of nodes (routers, basestations, phones) with state changing frequently (link load, cell connectivity)
  – O(1 million) nodes in graph, relatively small compared to web graphs

• Updates to graphs in streaming fashion & highly dynamic
  – E.g: In cellular networks, every edge updated every 3-5ms ⇒ O(2-5 million) updates every second
Complex Event Processing

• Compute abstraction: Updates to graphs (events) trigger complex processing that’s domain specific
  – E.g: link down event triggering a route update to maintain connectivity

• Triggered event processing can have tight latency constraints
  – E.g: link down event can cause disconnection, need fast action (~10-100ms) to keep the network connected

• Ideally, one should be able to write the event logic in a high level declarative fashion
Summary

• Separation of control plane is a common trend: networks/systems
• We have been building control planes for our specific systems
  – ONOS, SoftRAN, and RAMCloud Coordinator
• Can we design a generalized scalable control plane with
  – A common foundation that can be customized for different contexts
Backup
Generalized and Customizable Scalable Control Plane

- A common foundation for all control planes
  - A cluster of commodity servers
  - High performance network: 10-100Gbps, speed of light latency
  - Low latency RPC: ~5usec end to end latency
  - Customized distributed state management
  - Different consistency, replication, and sharding models for different state

- Customizable for specific contexts
  - Allow the specific system to pick distributed state managements
  - App programming abstractions/models/APIs
    - C/C++, Declarative Programming, REST APIs
  - Support different consistency models
    - Can we get away with everything being strongly consistent?
Outline

• Example control planes: requirements and designs
  – ONOS for service provider networks
  – SoftRAN for cellular RANs
  – Coordinator for RAMCloud
  – Other examples

• Opportunity for a unified scalable control plane
  – What makes us believe this is possible
  – What are the challenges

• High level architecture of a unified scalable control plane
  – Common building blocks
  – Customization

• Next steps
**SoftRAN: Data Plane Abstraction**

- **Flow-graph**: the pipeline/graph of packet processing atoms that need to be executed on all data/packets matching the flow
- Flow is flexibly defined (e.g. IP flow, UE bearer, QoS class)
- Atoms are simple packet processing functions (e.g. header compression, segmentation, FFT)
- Flow-graphs can have latency and compute throughput requirements associated with them

**Match**: If Pkt $\rightarrow$ Flow 1
**Action**: Execute Flow-Graph F
SoftRAN: Control Plane Abstraction

- **Update/Reduce** on control-graphs that represent network/application state
  - Control-graph is defined flexibly (e.g. network graph with eNBs, UEs and edges annotated with link strength, flow records, CDN graph with content nodes)
  - Local update computation at each node in the graph, message passing on edges and global reduce functions
- Simple abstraction is sufficient to express a large range of control functions

![Diagram](image)
SoftRAN: Expressing network services using flow graphs & control graphs

SoftRAN OS: Platform to execute any combination of flow and control graphs expressed in high level programs on COTS distributed hardware while meeting latency & throughput requirements
## Control Plane for Service Provider Networks

### ONOS
- **Network state**
  - 1TB
  - Changes more slowly
- **Control ops**
  - 1-3M ops/sec
- **Latency for processing**
  - 10 ms
- **High availability**

### SoftRAN
- **Network state**
  - Multiple TBs
  - Changes every few ms
- **Control ops**
  - 20 M ops/sec
- **Latency for processing**
  - 1 ms
- **High availability**

SoftRAN offers opportunities for simplification and optimizations.
RAMCloud Architecture

1000 – 100,000 Application Servers

Commodity
Servers

1000 – 10,000 Storage Servers

Datacenter
Network

Appl.
Library

Master
Backup

Applic.
Library

Master
Backup

High-speed networking:
• 5 µs round-trip
• Full bisection bandwidth

Coordinator

Coordinator
Standby

External
Storage
(ZooKeeper)

External
Storage
(ZooKeeper)

64–256 GB
per server

RAMCloud Control Plane
Scalable Control Plane: Breakout Session Summary

• Heard positive comments about the overall goals and project
  – with a reminder that we are taking on something challenging
  – general and customizable and high performance re obviously at odds with each other

• Scalable control planes for networking — easier to see from IoT to RAN to other networks
  – Are they in a different category compared to scalable control plane for distributed systems

• Examples of scalable control planes for distributed systems
  – Schedulers: clusters and multiple clusters
  – Scheduling of micro-services or associated workload
What is domain or context specific and what is domain/context independent

- Distributed state management is domain independent
- Application abstractions and APIs and interfaces to devices being controlled are domain specific

What is the right model or paradigm for the domain independent part

- Distributed state management but there are other models
- Time varying network graph as a data base and standing queries
- Streaming database with ability to customize: consistency, replication, sharding
- Complex event processing on streaming database

Application abstractions and APIs

- Declarative/easy to program