EASE NETWORK FUNCTION PROGRAMMING IN DISTRIBUTED CONTROL PLANE

Xiaoyan Hong
Computer Science, University of Alabama
http://hong.cs.ua.edu/
Data plane: multiple domains of switches

Control plane: the centralized control, a few controllers, running multiple network services (functions)

Benefits of DCP: large scale; protecting domain privacy

SDN Programming
  * Write service logics as running code at control plane
  * Write the forwarding logics to the data plane

Work with Boyang Zhou, Chunming Wu, Ming Jiang,
Programming DCP

* **Handling Events**
  * **Internal**: by itself, e.g. timeout, link congestion;
  * **External**: by operator, e.g. readjusting the link metrics.

* **Update states in the distributed manner at different switches**
  * control states at the control plane
  * forwarding states at the data plane

---

Initial routes → Load balancing → Topology changes → Link bandwidth changed → New routes

(timeline)

States examples, FIB, ports, packet scheduling, queues...
Problems

Transient State

* Inconsistent view of current control or forwarding states
* Caused by different delays, losses when updating the states
* Lead to losses of in-flight packets (routing loop, black hole, misfunction)

Generic for different services (e.g., routing and firewall)

* Each needs to dealing with it, increases the complexities in developing a service in DCP (code, programmers)
* Duplicate efforts by multiple services in solving the problem, each works on own control states
* Delayed reaction to network dynamics
  * when one service forces the network conditions to change, other services are not able to learn until a performance bottleneck occurs
Research Goal and Approach

Goal: Reduce the complexity for network programming tasks involving DCP

* Offer common interfaces for the complex, yet-similar tasks in dealing with inconsistency in the states, reduce network holding time

Our approach: supervision layer -- proGRAmming Control layer

* Identify service components, sharable common control states; and provide access control (Reusability of control states)
* Safely update the current forwarding states to the new states, without interrupting the data flows (Re-configurability of data plane states)
API guarantees that a service sees consistent data plane states

* DHT and two access models
  * Control states are organized using Chord (a DHT)
  * Active control state reuse
    * proactively pull to synchronize all the states
  * Passive control state reuse
    * passively receive only the changes

* Lossless reconfiguration protocol
  * Three phases of lock steps in updating forwarding rules
  * Buffer in-flight data packets and flow states, and restore
proGRAmming Control (GRACE) layer

Diagram:
- **Controller A**
  - Distributed Services
  - GRACE Layer
  - OpenFlow Protocol Stack

- **Controller B**
  - Distributed Services
  - GRACE Layer
  - OpenFlow Protocol Stack

- **Domain A**
- **Domain B**

Arrows indicate:
- (reuse)
- (reconfigure)
- (Synchronize control plane states)
- (Synchronize domain states)
- (OF control channels)
Case Analysis

Information-Centric Networking (ICN)

Content Distribution Networks (CDN)

Inter-Domain Routing (IDR) service

Evaluated in NS3, ndnSim, and PlanetLab
Related Research

Other work in software defined networking and network virtualization

* Handle dynamic data traffic with uncertainty in the data plane
  * Reduce the overhead in control channel and delay in reaction
* Network programming language: higher level network function abstraction

* Participated in GENI (Global Environment for Network Innovations)
Data-Rich Underwater Networks

- Mobile ad hoc, delay tolerant, wireless networks
- Underwater acomm networks: mobility, DTN routing, map portal
- Trading Computing/Storage/Mobility for Communication
- Named data, searchable and retrievable (Information Centric Networking) for data-rich underwater environment monitoring and information system
Why Software Defined Networking (SDN) for Underwater Comm and Net

- Resource sharing, interoperability
- Evolving PHY, NET, APPL -- double edge
  - Abstraction and interfacing with lower and upper layers
  - Open, standardizing
- Support network experiment and evolution
- Simplify network management
  - Centralized management for resources, abstraction and programmable data plane, reduced complexity in error checking and maintenance
  - Improve network performance, e.g., energy efficiency
Applications and network architectures

- AUVs/gliders, sea floor or 3D sensor net, mobile array
- Static (hierarchical), single hop, multi hop, mobile, data mule
- Data collection, command and control
- Multiple roles: sensing (monitoring), in-situ data processing, communicating, forwarding, moving
- Current SDN: routing choices, performance and priority parameters, commending multiple platforms (UWSN, gliders), etc.

Switching fabric

- Different “switch”, maybe multiple radio fronts
- Current SDN: reconfigurable radios/modems and MAC parameters, shared, open architecture (modems)
Data plane vs control plane, and new components

- Mesh sensor networks, gliders/AUVs, data mules/DTN
- Each device multiple roles: generating data and networking, networking only
- Static, mobile, or coordinated/planned mobility
- In-network processing
- Management tools (resources, power, navigation, etc)
- Current SDN: in-network processing in data plane, adaptive protocol stack, control plane, application plane
Resource sharing (concurrent experimentation), virtualization, multiple radio fronts

- Testbed, multiple applications/experiments shared use
- Current SDN: wireless hypervisor, NFV

Distributed systems under harsh network conditions

- Multiple controllers, distributed, connected, latency, reliability, limited bandwidth
- High programming needs for the increasing number of applications/services, trained in multiple disciplines

E2E secure channel vs data-centric security
Thank you!

Questions?