meSDN: Mobile Extension of SDN

Mostafa Uddin
Advisor: Dr. Tamer Nadeem
Old Dominion University
Very brief on SDN

• Networks are very hard to manage and evolve.
• Data Plane:
  Fwding state + Packet header → forwarding decision
  Fast(nano-scale) and local.
• Control Plane:
  Compute the forwarding state for the data plane.
  Routing, Isolation, Traffic engineering.
• Control Plane mess is the root cause of SDN.
SDN: A layer of two Control Plane Abstraction

Routing, Access Control etc.

Control Plane

Global Network View (topology)

Network OS

Forwarding Model
Mobile Cloud Application

- Mobile cloud application require guaranteed network performance.

- SDN controller need to provide performance guarantee to clients knowing the app demand from the cloud server.
SDN in Wired and Wireless

Wired
- Point-to-Point full Duplex link.
- End device Tx don't interfere with others.

Wireless
- Shared half-duplex medium.
- Can't control client's uplink Tx.

We can provide performance guarantee by controlling Network edge.

I can't :(
Pushing SDN to Clients

• Existing SDN framework stops at network edge.

• Highly predictable performance for client device.
  • SDN enable AP (OpenRadio and OpenRAN) cannot guarantee wireless resource for uplink from the client.

• End-to-end QoS control.
  • e.g. One client greedily using highest priority can unfairly dominate uplink air-time resource.
Our Solution: meSDN

- **meSDN** (mobile extension of SDN): extend the SDN framework to the end device.

- **meSDN** allows the control-plane of wireless network to be extended to mobile device.

- Provide fundamental software-defined solutions for many applications
  - WLAN virtualization, application-awareness, E2E QoS, and network troubleshooting.
meSDN: Smartness in End-Devices

- **Ground-truth** information about client application information.
- **Monitor and manage** mobile application's traffic flows **real-time**.
- **Guarantee** **airtime resource**.
- **Provide** **end-to-end QoS** service for mobile clients.
Agent @ Mobile endpoints!

- Not a new concept to centrally control the client devices
  - (e.g. PC COE, BYOD solutions, VPN client, mobile WAN acceleration).

- Allows several benefits:
  - Users can have better and predictable network performance.
  - SDN controller can enforce policies directly on the client's traffic.
  - Support enhance network security, end-to-end QoS and WLAN virtualization.
meSDN Architecture

- Flow Manager (e.g. Open vSwitch, OVS).
- Scheduler (e.g. Linux multi Qdisc).
- Local Controller (e.g. Android userspace software/agent).
- Global Controller.
**meSDN: Flow Manager**

- It is a software OpenFlow switch (e.g. OVS)
- Collect Flow statistics:
  - OF Stat extension: burst duration, burst rate and inter-burst time.
- Enforce SDN policies e.g., correct QoS marking
- Interact with the WiFi Driver to configure.
meSDN: Scheduler

• Extension to linux multiq or WiFi Driver that supports 802.11e QoS.

• Receive *time window* from the local controller to start/stop dequeueing.
  – Time Window: e.g. [Start time, active duration, sleep duration]
  – e.g. 05:30:30, 10ms, 30ms
meSDN: Local Controller

- Identify flows correspond to each application.
- Generate flow rules for OVS – Based on per-application policy given by central controller or the user.
- Read per-flow statistics from Flow Manager.
  - OpenFlow extension.
- Control the scheduler.
*meSDN*: Global Controller

Interacts with local controller

- provide per-slice, per-user, per-application policies and QoS profiles.

- Collect 'aggregated' airtime demand of the running applications and QoS requirements.

- Apply proper action back to the local controller (e.g. Scheduling)
meSDN: Use-Cases / Applications

- Realtime detection/analysis of networks flows.
- Network fault diagnosis and trouble shooting.
- WLAN virtualization.
  - Guarantee airtime resource to multiple group of users.
- Dynamic Policy Setting.
WLAN Virtualization

- WLAN virtualization enable effective sharing of wireless resources by a diverse set of users with diverse requirement

Enterprise WLAN: employees, guest

Home WLAN: parents, kids
$p$TDMA: WLAN Virtualization

• $p$TDMA is a simple prototype of meSDN for WLAN virtualization service.

• Manage airtime share between network instances (their clients) that collocate in space and channel
  - Assigning separate airtime slices among different network instances
pTDMA: Scheduling Principles

- Allocate **large enough time window** to transmit and receive multiple packets.

- Schedule **multiple clients in a common slot** to maximize channel utilization.

- The **interval between consecutive time windows** should be based on applications’ traffic pattern & demand.
pTDMA: Prototype Scheduling

• 50:50 airtime share between employee network and guest network.
• Every time window is fixed of 10ms.
$p$TDMA: Implementation

- Prototyped $meSDN$ client-side component on eight Google Nexus 4 Android phones
- Root the device to install OVS and $p$TDMA qdisc kernel modules.
- Re-Build the kernel image
  - To implement the Wi-Fi driver byte limit in Nexus 4 WiFi driver
  - Note: some other phones have Wi-Fi driver as kernel module (e.g. Nexus S)
1. *meSDN* leverage WMM-PS to indirectly confine the downlink traffic to the time window.

2. \(p\)TDMA allows to efficiently utilize the WMM-PS to have more sleeping time without sacrificing the throughput performance.
pTDMA: Implementation Challenges

• Milli-second level synchronization between the phones is needed for effective pTDMA.
  – Achievable by GPS
  – Note: traditional per-packet TDMA requires micro-second level time sync

• Driver buffering delay is large.
  – Bufferbloat: Large ring packet buffer (100 to 300, total bytes >150KB) used by WiFi, Ethernet drivers
    – Byte Queue Limit(BQL) for Ethernet driver in Linux: buffer size limit is dynamically set based on recent “byte” dequeued by the NIC
    – We set hard byte limit in Wi-Fi Driver to 15KB, enough for 10 pkt 802.11 aggregation
We formed two network slices:
- "employee" network with 2 devices
- "guest" network with 6 devices

Applied following pTDMA schedule with 50:50 airtime share between two slices

- 3:1 airtime ratio btw an employee and a guest.
  (but all devices are connected to one AP)
pTDMA: Evaluation (ulink UDP)
Assume the driver goes to sleep state after 5ms of inactivity in WMM-PS

In non-\(p\)TDMA, client sleeps 28% of the time.

In \(p\)TDMA, client sleeps 80% of the time.
Increased transmission time in pTDMA schedule do not adversely effect TCP performance.
Related Work: WLAN infra virtualization

- Multiple SSID networks
  - Don't guarantee wireless resource share to each SSID network
- SDN enable AP (OpenRadio, OpenRAN, Odin, CloudMAC).
  - No control on uplink traffic from client.
- Tuning 802.11e QoS parameters in AP
  - Limit the virtual network to four QoS classes.
Related Work: Client Side Solution

- Per-packet TDMA MAC to virtualize airtime.
  - Clock Synchronization among devices.
  - Hardware control from driver/firmware.

- SplitAP loosely control uplink
  - Under Utilization of airtime

- Deployed OVS to utilize multiple network interface.
In Summary

• Extending SDN capabilities to mobile end device.

• Propose and demonstration of \textit{meSDN} framework.

• As a proof-of-concept, we implement $p$TDMA for WLAN virtualization service.
Thank you

JK Lee

Jean Tourrilhes