Extending SDN for mobile device

Jeongkeun Lee,[1] Mostafa Uddin,[2] Jean Tourrilhes,[1] Souvik Sen,[1] Sujata Banerjee,[1]
Manfred Arndt,[3] Tamer Nadeem[2]

[1]HP Labs, [2]Old Dominion University, [3]HP Networking

1. INTRODUCTION

Large number of mobile devices use numerous apps that access internet through wireless. With significant amount of traffic growth and variability, it is necessary to have greater visibility and control over the traffic generated from the client devices, such that we can provide better performance guarantees to multiple types of users on a shared wireless infrastructure. In a wired infrastructure, network virtualization is a means to deliver such performance guarantees. Wired virtualization solutions use Software-Defined Networking (SDN) APIs to dynamically coordinate network edges (e.g. routers, switch etc.); but don't require a change of client device behavior because the last hop between the network edge and the wired end device is an isolated full-duplex p2p link, e.g., Ethernet. However, this is not the case with wireless LANs (WLAN) as the last hop between the mobile device and the access points is a shared medium. Moreover the current WiFi MAC protocol does not allow edge access points (APs) to control client uplink transmissions and their 802.11 quality of service (QoS) settings.

We argue that the SDN framework needs to be extended to the client devices to support several interesting capabilities and services such as guaranteeing airtime resource to each virtualized WiFi network slice. In addition, by integrating SDN APIs in the client device, we can manage the uplink TX and the QoS over the shared wireless medium, and provide end-to-end QoS control.

2. *meSDN*

Our solution is meSDN – mobile extension of SDN. As a proof-of-concept, we design and implement a new WLAN virtualization service that slices mobile devices via a Time Division Multiple Access (TDMA) like scheduling, named pseudo-TDMA (pTDMA). By using Linux Qdisc on end devices, pTDMA virtualizes (separates) airtime resource between network slices while minimizing contention between clients within a slice. pTDMA also allows client wireless interfaces to improved power-efficiency utilizing their active time and to sleep longer outside of the given transmission windows.

As shown in Fig. 1, *meSDN* has three components in clients: (1) airtime scheduler (Linux qdisc) (2) flow manager (e.g., Open vSwitch, OVS), and (3) local controller. There is also a global network controller that talks with the client local controllers.

Flow Manager is a software OpenFlow switch, e.g. Open vSwitch [1], that monitors and manages mobile's application traffic. To better support various mobile apps' needs, we extend OpenFlow statistics APIs and measure burst duration & rate and inter-burst time, and feeds them to the local controller for airtime scheduling. OVS also takes per-flow QoS and access control actions. meSDN extends OVS further to interact with WiFi driver; this "Wireless Extension" enables the control plane to better monitor and manage airtime resource.

Scheduler is a Linux Qdisc that implements airtime scheduling. It starts/stops dequeueing of the outgoing flow based on the airtime schedule given by the Local controller. The Qdisc also applies prioritization and rate limiting to application flows, as instructed by the control plane.

Local controller is userspace software that controls Flow Manager and Scheduler. The local controller also provides application awareness and generates flow-to-application mappings by monitoring `netstat` logs; it can easily apply appropriate application-specific SDN policy through OVS or Qdisc.

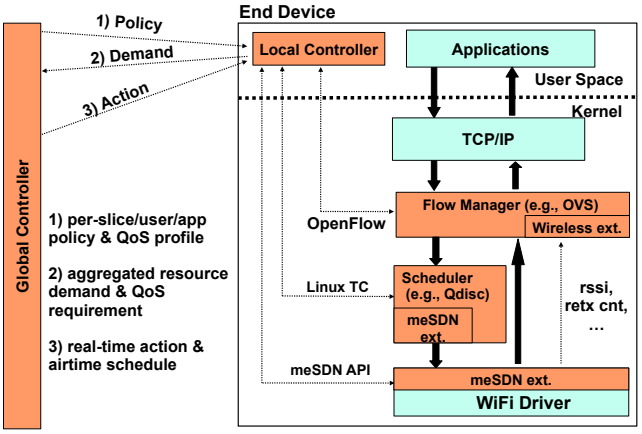

Figure 1: *meSDN* Architecture.

Global controller coordinates with the local controller in three steps (also see Fig. 1). First, the global controller provides per-slice, per-user, per-application policies and QoS profiles to the local controllers, which then can apply the policies to application flows without querying the global controller for every new flow. The local controller send the resource and QoS requirements to the global controller airtime scheduler (the 2nd step in Fig. 1). Finally, the global controller computes airtime schedules based on the received per-client demands, and provides the schedules to the local controllers (the 3rd step).

We envision that the flexible meSDN architecture will enable more interesting services including e2e QoS controls, enhanced security, fault diagnosis, etc. We believe, the ability to truly inspect flows end-to-end and conduct diagnostic tests at the endpoints may be one of the killer applications of meSDN.

3. REFERENCES

[1] Open vSwitch. `http://openvswitch.org/`.