ÆtherFlow: Principled Wireless Support in SDN

Muxi Yan, Jasson Casey, Prithviraj Shome, and Alex Sprintson
Texas A&M University

Andrew Sutton
University of Akron
Motivation:

• Current SDN ecosystem does not support wireless (WiFi) networks
  – No support for wireless in published SDN standards
• No true network visibility with respect to wireless protocols
• Current distributed AP functionality is limited,
  – difficult to provide certain behavior
Goals:

• Bring wireless networking into the SDN ecosystem
• Provide a unified interface for both wireless and wireline components
• Support diverse wireless services
  – Multiple SSID, Unified AAA
  – User/Group/World based treatment
  – Facilitated mobility
Our approach

• Principled extensions to OpenFlow
  – General definition of wireless abstractions within an existing SDN framework.
  – Incremental modifications to the existing SDN network elements.

• Present a solution which is
  – Easily extended to multiple wireless technologies
  – Not tailored to a single application
Related Work

- OpenRoads (Yap et al. 09’)

- Odin (Suresh et al. 12’) OpenSDWN (Schulz-Zander et al. 15’)

- SoftCell (Jin et al. 13’), SoftRAN (Gudipati et al. 13’)

- OpenRadio (Bansal et al. 12)

- ONF Wireless and Mobile Working Group
Extensions to OpenFlow

• Switch extensions
  – Abstraction of wireless interfaces
  – Control API for wireless interfaces

• Protocol extensions
  – Add extended messages to protocol
Switch Abstractions

• TinyNBI Model (Casey et al. 14’)
  – Data plane components
Switch Abstractions

**Capabilities**
- Abilities/limitations of this abstraction; read only

**Events**
- Notifications sent by this abstraction

**Configuration**
- Configured state of this application; readable or writable

**Statistics**
- Observed metrics for this abstraction; read only
Switch Abstractions

• The controller interact with a component with its provided API
  – FeatureRequest - Datapath / capabilities
  – FlowMod - Flow table / configuration
  – PortMod - Port / event
  – MultipartRequest - * / statistics
Wireless Interface Abstraction

- New data plane component
  - *Wireless* physical port
    - Corresponds to a radio interface
  - *Wireless* logical port
    - Corresponds to an AP
    - Maps to the corresponding wireless physical port
Radio Physical Port

• **Capabilities**
  – Supported IEEE 802.11 versions (b/g/n/ac/…)
  – Supported channels
  – Supported transmission power
  – Supported encryption and authentication methods

• **Configuration**
  – IEEE 802.11 version
  – Channel
  – Transmission power
  – Add/Remove AP

• **Statistics**
  – Tx/Rx bytes
  – Tx/Rx packets
  – … (Same as port counters in OpenFlow)
802.11 MAC Logical Port

- **Capabilities/Configuration**
  - Physical port
  - BSSID, SSID
  - Encryption/Authentication method

- **Statistics**
  - Client association state
  - Client RSSI
  - Tx/Rx bytes
  - Tx/Rx packets

- **Events**
  - Probe
  - Authentication/Deauthentication
  - Association/Disassociation/Reassociation
  - 802.1X authorization completed
Protocol Extension

• Allow controllers to invoke APIs provided by new data components
• Use Experimenter messages
• Define messages corresponding to each API function and error
Benefits of ÆtherFlow

• Provides explicit support for wireless radio interfaces and wireless access points.
• Transparent to mobile stations (user devices)
• Provides an ability to
  – handle wireless packets through an OpenFlow data path
  – remotely configure access points
  – query mobile station capabilities and statistics
  – report mobile station events
Implementation

- TP-LINK WR1043ND v2
  - Nice OpenWRT support
- OpenWRT (Barrier Breaker)
- CPqD Software Switch (OF v1.3)
  - User space software switch
Implementation

- **ÆtherFlow**
  - Connect hostapd and softswitch
  - hostapd still manages mobile station
  - ÆtherFlow API related information are forwarded between Soft Switch and hostapd
Experiment

• Layer 2 Handoff
  – Client move from AP1 coverage to AP2 coverage

• Compare performance
  – Bridge based configuration
  – Prediction-based configuration
Prediction Based Handoff

• Handoff prediction
  – Signal strength
  – History event records

• Multicast
  – Packets forwarded to both APs prior to handoff

• Redirect
  – Redirect packets based on whether handoff occurs or not
Experiment

- **Settings**
  - 3 AÆtherFlow switches
  - All switches connect to a OF controller (not depicted)
  - Two APs are spatially separated apart
  - STA uses open authentication
  - STA and TG use static IP address
Experiment

• Settings
  – Traffic Generator sends UDP packets
    • use iperf
    • transmit at 9Mbps
  – STA receives packets and make measurements
    • use iperf
    • at 0.5s interval
Experiment

- Prediction is out of scope
  - Assume controller predicts that a handoff occurs at a time $t$ after experiment starts
  - STA moves at time $t - 1$ seconds after experiment starts
Experiment

• Procedure
  – 5 rounds of experiments for each configuration
    • Linux bridge based
    • Prediction based

• Performance metrics
  – Average *handoff duration*
  – Median rounds
    • Throughput
    • Packet loss
Results

- Average handoff duration
  - Handoff duration: the duration where throughputs drop below 8Mbps
  - Average over 5 rounds
    - Bridge: 7.1s
    - ÆtherFlow: 5.9s
Results

- Median rounds throughput and packet loss rates
Conclusion

• Principled wireless extensions to SDN
  – Switches
  – Protocols

• Implementation
  – Commercial wireless AP
  – Software OpenFlow Switch

• Experiment
  – The proposed system provides wireless interface controlling capabilities to SDN controller
Thank you!

Questions?