A Northbound API for Sharing SDNs

BROWN

Andrew D. Ferguson
adf@cs.brown.edu

Arjun Guha
arjun@cs.cornell.edu

Chen Liang
chen_liang@brown.edu

Rodrigo Fonseca
rfonseca@cs.brown.edu

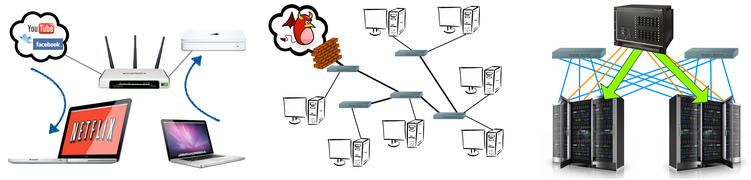
Shriram Krishnamurthi
sk@cs.brown.edu

pane.cs.brown.edu

The Need to Share SDNs

- Applications running in home, campus, datacenter, and wide-area networks can benefit from direct interaction with the control-plane

QoS in a Home | Firewalls in a Campus | Circuits in a Datacenter



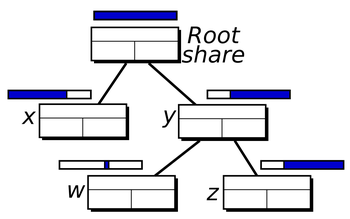
- These applications may require both *read* access to determine current network properties and conditions, and *write* access to adjust the configuration of the network itself
- To prevent anarchy, a northbound API which provides this access must overcome two challenges: 1) how to decompose control and visibility? and 2) how to resolve conflicts?
- Our prototype introduces *participatory networking*, implemented by an OpenFlow controller called **PANE**

Decomposing Network Control and Visibility

- The API uses *shares* to describe a slice of network control
- Each share states who (principals) can say what (privileges) about which flows in the network (flowgroup)

Flowgroup: src=128.12/16 ∧ dst.port ≤1024	
Principals	Privileges
Alice Bob	*deny, allow* *bandwidth: 5Mb/s limit: 10Mb/s* *hint* *query*

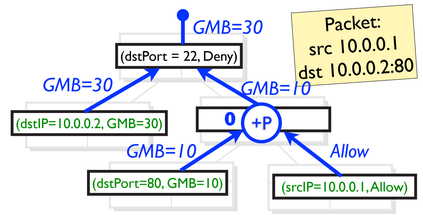
- The privileges PANE exposes are *requests*, *hints*, and *queries*
- Requests are for resources and affect the state of the network, hints provide information the controller may use to improve the network configuration, and queries read the state of the network



- Shares are hierarchically organized in a *ShareTree* that constrains the flowgroups and privileges
- As in a capability system, principals may create new sub-shares and grant access to other principals

Resolving Conflicts

- Requests and hints become *policy atoms*, which affect specified traffic and are placed in the *Policy Tree*
- The Policy Tree specifies the outcome for all packets



- Conflicts between policy atoms are resolved by *conflict-resolution operators* (such as +P above) which exist at all nodes in the tree
- For efficiency, PANE compiles the policy tree into OpenFlow match tables, using a variant of the NetCore algorithm

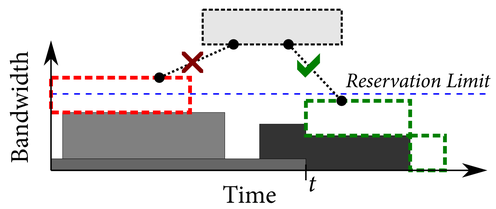
PANE's Northbound API

Share	S	$\in \{P\} \times \{F\} \times \{Priv\}$	A share gives principals some privileges to affect a set of flows.
Principal	P	$::= \langle user, host, app \rangle$	A triple consisting of an *application*, running on a *host* by a *user*.
Flow	F	$::= \langle$ **srcIP**$=n_1$, **dstIP**$=n_2$, **proto**$=n_3$, **srcPort**$=n_4$, **dstPort**$=n_5 \rangle$	A set of packets with shared properties: source and destination IP address, transport protocol, and source and destination transport ports.
Privilege	$Priv$	::= **CanDeny** n | **CanAllow** n	The privileges to allow or deny traffic for up to n seconds (optional).
		| **CanReserve** n | **CanRateLimit** n	The privileges to reserve bandwidth or set rate-limits, up to n MB.
		| **CanWaypoint** $\{IP\}$ | **CanAvoid** $\{IP\}$	The privileges to direct traffic through or around particular IP addresses.
Message	Msg	$::= P : \{F\} : S \rightarrow \langle Req\ Tspec$ | $Hint\ Tspec$ | $Query \rangle$	A message from a principal with a request, hint, or query using a share.
Time Spec	$Tspec$	::= **from** t_1 **until** t_2	An optional specification from time t_1 until t_2.
Request	Req	::= **Allow** | **Deny**	Request to allow/deny traffic.
		| **Reserve** n | **RateLimit** n	Request to reserve n MB or rate-limit to n MB.
		| **Waypoint** IP | **Avoid** IP	Waypoint/avoid traffic through a middlebox with the given IP address.
Query	$Query$	::= **TrafficBetween** $srcIP\ dstIP$ | ...	Query the total traffic between two hosts.
Hint	$Hint$	::= **Duration** t | ...	Hint that the flow's duration is t.

PANE's definitions (top) and end-user API (bottom). An API to create shares and delegate privileges is also provided

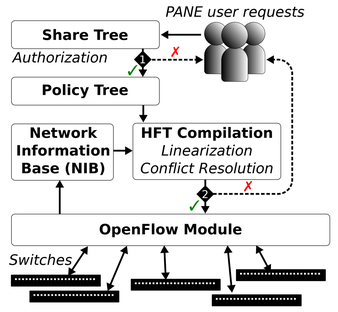
Example: Bandwidth Scheduling

Here, Alice uses PANE to reserve future bandwidth. The network admin only needs to delegate the privilege; PANE does the rest.




```
1 root: NewShare aliceBW for (user=Alice) [reserve <= 10Mb] on rootShare.
2 root: Grant aliceBW to Alice.
3 Alice: reserve(user=Alice,dstPort=80) = 8Mb on aliceBW from now to +10min.
3 Alice: reserve(user=Alice,dstPort=80) = 8Mb on aliceBW from +20min to +30min.
```

Prototype Implementation

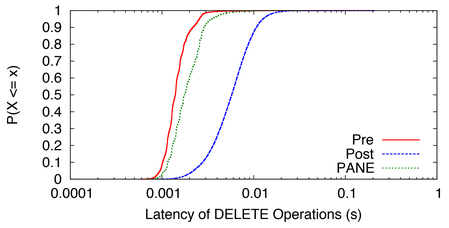


- We have developed a prototype implementation of our API as an OpenFlow-based SDN controller
- The PANE controller implements user requests after authorization (1) and conflict-resolution (2)
- PANE has been running our lab's network since Feb. 2012, and comes with a Java library for client applications (examples on Github)

Evaluation Examples

ZooKeeper: Guaranteed Bandwidth for Lower Latency

- A distributed service providing shared, consistent, available state
- Each operation requires agreement of a quorum of servers



- Configured an ensemble of five ZooKeeper servers connected to a PANE-controlled OpenFlow switch, plus a benchmarking client
- This CDF shows the latency of operations on an isolated network (Pre); when ZooKeeper competes with other traffic (Post); and when ZooKeeper requests guaranteed bandwidth (PANE)

Hadoop: Extending Scheduler Weights into the Network

- An open implementation of MapReduce, Hadoop's scheduler supports weighted fair-sharing across jobs in the cluster
- However, these weights do not extend into the network currently; bandwidth is allocated by TCP's traditional approach
- We augmented Hadoop to reserve bandwidth in proportion to each job's weight, and benchmarked three simultaneous sort jobs weighted 2:1:1. Completion time decreased by 19% for the top-weighted job, and by 9% for the others due to work-conservation