CS 352
Software-Defined Networking: Intro

CS 352, Lecture 21.1
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
The main function of the network layer is to move packets from one endpoint to another.
Review: Network layer functions

• **Forwarding**: move packets from router’s input to appropriate router output

• **Routing**: determine route taken by packets from source to destination

• **Data Plane**

• **Control Plane**

• **Two kinds of control planes**:  
  • Distributed per-router control  
  • Logically centralized
Routing Algorithm

4.1 • \textsc{Ov}er\textsc{V}iew\textsc{O}f\textsc{Net}work\textsc{L}ayer

tables. In this example, a routing algorithm runs in each and every router and both forwarding and routing functions are contained within a router. As we’ll see in Sections 5.3 and 5.4, the routing algorithm function in one router communicates with the routing algorithm function in other routers to compute the values for its forwarding table. How is this communication performed? By exchanging routing messages containing routing information according to a routing protocol! We’ll cover routing algorithms and protocols in Sections 5.2 through 5.4.

The distinct and different purposes of the forwarding and routing functions can be further illustrated by considering the hypothetical (and unrealistic, but technically feasible) case of a network in which all forwarding tables are configured directly by human network operators physically present at the routers. In this case, no routing protocols would be required! Of course, the human operators would need to interact with each other to ensure that the forwarding tables were configured in such a way that packets reached their intended destinations. It’s also likely that human configuration would be more error-prone and much slower to respond to changes in the network topology than a routing protocol. We’re thus fortunate that all networks have both a forwarding and a routing function!

Values in arriving packet’s header

<table>
<thead>
<tr>
<th>header</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0110</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.2 ♦ Routing algorithms determine values in forward tables

Review: Traditional distributed ctrl planes

\textbf{Distributed control plane:}\nComponents in every router interact with other components to produce a routing outcome.

\textbf{Data plane}\nper-packet processing, moving packet from input port to output port

values in arriving packet header, i.e., destination IP address
Traditional distributed control plane
Software-defined network (SDN)

(1) Logically-centralized control plane

(2) Open, well-defined control-data-plane interface

Control plane
Data plane

Control plane
Data plane

Control plane
Data plane

Control plane
Data plane
Two components of an SDN

• Logically centralized control plane: the SDN controller
  • Function implemented in software: software-defined
  • General-purpose software to compute forwarding paths and policies
  • Not subject to restrictions of distributed protocols (more soon)

• Open control-data interface
  • Routers expose a general instruction set (next module)
  • Statistics, events (link changes, interface changes), control messages, etc.
Why SDN?

A reaction to two trends
Routers hard to change (~2005)

- Router vendors packaged routers with hardware for forwarding and software for the control plane processor
  - Proprietary software and interfaces to hardware
  - ISP operators relegated to a rigid, high-level command line interface

- The behavior of the routers (owned by an ISP) couldn’t be changed by the ISP’s network operators
  - Require vendor support

- Vendor software bloated and buggy
  - Including bugs from features that were never used by the ISP
Operators need better abstractions

- Larger networks come with harder problems
  - more devices to manage
  - higher data volumes
  - more failures
Operators need better abstractions

• Example 1: Choosing network paths freely
• Distributed intra-domain routing protocols compute least cost paths from link metrics
• Q: How to forward $u \rightarrow z$ data traffic along the path shown?
• Link weight inference problem: set link weights such that least cost paths are those that are desired
Operators need better abstractions

• Example 2: More flexible forwarding decisions
• Distributed routing protocols and data planes forward packets using destination address
• But might want to forward based on other criteria
  • Example: by both source and destination, to balance link usage
• Traditional routing: can’t use different paths for traffic to the same destination

Requirement at node w:
Originate from u, dst z: 
  \( w \rightarrow z \)
Originate from v, dst z: 
  \( w \rightarrow y \rightarrow z \)
Operators need better abstractions

• Example 3: Reaction to network failures
  - When a link fails, traditional routing protocols re-converge to an outcome based on link metrics on new network
• Two problems with this approach:
  - Slow convergence
  - Non-deterministic outcomes
Benefits of SDN

• Software controls the behavior, not distributed protocols
  • Implement flexible forwarding tables
• Interface to data plane is open and well-defined
  • Forward packets using a flexible set of fields
• Impact: like moving from mainframes to commodity PCs!
Design questions in SDN

Apps

OS

Hardware design

Applications

Network operating system

Router hardware design
Design questions in SDN

- Applications
- Network operating system
- Hardware design

Subject of the next module.

Abstraction?

Instruction set
Forwarding with Match-Action

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana
Review: SDN

(1) Logically-centralized control plane (SDN controller)

This module: What’s the instruction set exposed by the data plane?

(2) Open, well-defined control-data-plane interface
Data plane instruction set: Goals

• Want to match patterns on any part of the packet header
  • Flexible forwarding using any field (much more than dest address!)
  • Match a whole or a subset of each field
  • Allow any forwarding granularity (e.g., src-dst, application, etc.)

• Want to act on packets that match the pattern
  • Forward out a specific port
  • Modify, remove, or add fields
  • Drop packets
  • Count packets
Match-action flow tables

- A generalization of forwarding tables
Match-action flow tables

• For concreteness, let’s consider **Openflow 1.0**
  • A specification of applicable match **patterns**
  • A specification of actions available in the data plane

• Flow table contains match-action **rules**
• Match: can occur on any subset of 12 fields

```
Flow table

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
Ingress Port  Src MAC  Dst MAC  Eth Type  VLAN ID  VLAN Pri  IP Src  IP Dst  IP Proto  IP ToS  TCP/UDP Src Port  TCP/UDP Dst Port

Link layer  Network layer  Transport layer
```
Match-action flow tables

• Examples of matches:
  • Dstip = 192.168.0.1
  • Srcip = 10.0.0/24, IP protocol = TCP, TCP dstport=80
  • DstMAC = 01:4f:3d:10:33:a4

• Any bit among those not listed is wildcarded
  • All values of those bits are accepted when matching
Match-action flow tables

• What if multiple rules match a packet?
  • Srcip = 10.0.0/24, IP proto=TCP, dstport=80
  • Dstip = 192.168.0.1
  • Can both match the same packet!

• Rule priority indicates the rule taking precedence
Match-action flow tables

- **Actions** in Openflow 1.0 are simple
  
  - Forward a packet through a specified port
  - Drop the packet
  - Send the packet to the SDN controller
Other match-action specifications

• Newer, more general flow table specifications exist

• Openflow 1.0 matches packets with a single table and a fixed set of packet fields (TCP/IP stack)
  • Later versions of Openflow: match using multiple tables

• P4: the ability to match on user-defined packet fields
  • Parse packets in a high-speed router using a program you wrote!
  • Highly flexible: introduce new protocol formats on packets in a network
  • Much more flexible actions: rewrite packets, do arithmetic on fields
Impact of SDN

• Real systems and deployments have been built using SDN principles
• Google: inter-cluster backbone network
  • High network utilization
  • Deterministic network behavior upon failures
  • Ability to pre-plan and test code by mirroring event streams from production networks
• Microsoft: Virtual Filtering Platform (VFP) to enforce policies on VM network traffic
  • Easily “serviceable” using software upgrades
  • High performance with hardware implementations