CS 5114
Network Programming Languages
Control Plane

Nate Foster
Cornell University
Spring 2013

Based on lecture notes by Jennifer Rexford and Michael Freedman
Announcements

Breakfast pickup and presentations posted to website

**Reviews:** start today!
- Only review one paper but please read them all
- Please submit by CMS in the future

**Homework #1**
- Will go out next Tuesday
- Due 2 weeks later
- Topic: OpenFlow programming
Overview

Host (last time)
- Network discovery and bootstrapping
- Resource allocation and interface to applications

Data plane (next time)
- Streaming algorithms and switch fabric
- Forward, filter, buffer, schedule, mark, monitor, …

Control plane (today)
- Distributed algorithms for computing paths
- Disseminating the addresses of end hosts
## Data, Control, and Management Planes

<table>
<thead>
<tr>
<th>Time-scale</th>
<th>Data</th>
<th>Control</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet (ns)</td>
<td>Event (10 ms to sec)</td>
<td>Human (min to hours)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Data</th>
<th>Control</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forwarding, buffering, filtering, scheduling</td>
<td>Routing, signaling</td>
<td>Analysis, configuration</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>Data</th>
<th>Control</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line-card hardware</td>
<td>Router software</td>
<td>Humans or scripts</td>
<td></td>
</tr>
</tbody>
</table>
Routing vs. Forwarding

**Routing**: control plane
- Computing paths the packets will follow
- Routers talking amongst themselves
- Individual router *creating* a forwarding table

**Forwarding**: data plane
- Directing a data packet to an outgoing link
- Individual router *using* a forwarding table
Data and Control Planes

- Processor
- Switching Fabric
- Line card

Data plane

Control plane
Routing Protocols

What does the protocol compute?
- Spanning tree, shortest path, local policy, arbitrary end-to-end paths?

What algorithm does the protocol run?
- Spanning-tree construction, distance vector, link-state routing, path-vector routing, source routing, end-to-end signaling

How do routers learn end-host locations?
- Learning/flooding, injecting into the routing protocol, dissemination using a different protocol, and directory server
What Does the Protocol Compute?
Different Ways to Represent Paths

Static Model
- The *outcome* of routing computations
- Not how the (distributed) computations are performed

Trade-offs
- State required to represent the paths
- Efficiency of the resulting paths
- Ability to support multiple paths
- Complexity of computing paths
- Which nodes control the computation

Different Settings
- LAN, intradomain, interdomain
Spanning Tree

A tree that connects every node
- Single path between each pair of nodes
- No loops, so supports broadcast easily

Disadvantages
- Paths can sometimes be long
- Some links unused!
Shortest Paths

Shortest path(s) between each pair of nodes
- Separate shortest-path tree rooted at each node
- Minimum hop count (or minimum sum of weights)

Disadvantages
- All nodes must agree on the link metrics
- Multipath routing is limited (e.g., Equal Cost Multipath)
Local Policy at Each Hop

Locally best path
- Each node picks the path it likes best
- … from among the paths selected by its neighbors

Disadvantages
- More complicated to configure and model
End-to-End Path Selection

End-to-end path selection
- Each node picks its own end to end paths
- … independent of what other paths other nodes use

Disadvantages
- More state and complexity in the nodes
- Hop-by-hop destination-based forwarding is not enough
How to Compute Paths?
Spanning Tree Algorithm

Elect a root
- Select switch with the smallest identifier and form a tree

Algorithm
- Repeatedly talk to neighbors
  - “I think node Y is the root”
  - “My distance from Y is d”
- Update state based on neighbors
  - Smaller id as the root
  - Smaller distance d+1
- Disable interfaces not on path

Primarily used in Ethernet-based LANs
Spanning Tree Example: Switch #4

Switch #4 thinks it is the root
- Sends (4, 0, 4) message to 2 and 7

Switch #4 hears from #2
- Receives (2, 0, 2) message from 2
- … and thinks that #2 is the root
- And realizes it is just one hop away

Switch #4 hears from #7
- Receives (2, 1, 7) from 7
- And realizes this is a longer path
- So, prefers its own one-hop path
- And removes 4-7 link from the tree
Shortest-Path Problem

Compute: *path costs* to all nodes

- From a given source u to all other nodes
- Cost of the path through each outgoing link
- Next hop along the least-cost path to s
Link State: Dijkstra’s Algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

**Initialization**

\[ S = \{u\} \]
for all nodes \( v \)
if (\( v \) is adjacent to \( u \))
  \[ D(v) = c(u,v) \]
else \( D(v) = \infty \)

**Loop**

add \( w \) with smallest \( D(w) \) to \( S \)
update \( D(v) \) for all adjacent \( v \):
  \[ D(v) = \min\{D(v), D(w) + c(w,v)\} \]
until all nodes are in \( S \)

Used in OSPF and IS-IS
Link-State Routing Example

Diagram of a network with nodes and links labeled with numbers.
Link-State Routing Example (continued)
Link State: Shortest-Path Tree

Shortest-path tree from u

Forwarding table at u

<table>
<thead>
<tr>
<th>link</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>y</td>
</tr>
<tr>
<td>z</td>
</tr>
<tr>
<td>s</td>
</tr>
<tr>
<td>t</td>
</tr>
</tbody>
</table>
Define distances at each node x
- \( d_x(y) = \) cost of least-cost path from x to y

Update distances based on neighbors
- \( d_x(y) = \min \{c(x,v) + d_v(y)\} \) over all neighbors v

\[
d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}
\]

Used in RIP and EIGRP
Distance Vector: Count to Infinity

Link cost changes:

- Good news travels fast
- Bad news travels slow: “count to infinity” problem!

![Diagram showing link cost changes and algorithm continuation](image)
Path-Vector Routing

Extension of distance-vector routing
- Support flexible routing policies
- Avoid count-to-infinity problem

Key idea: advertise the entire path
- Distance vector: send distance metric per dest d
- Path vector: send the entire path for each dest d

Used in BGP
Path-Vector: Faster Loop Detection

Node can easily detect a loop
- Look for its own node identifier in the path
- E.g., node 1 sees itself in the path “3, 2, 1”

Node can simply discard paths with loops
- E.g., node 1 simply discards the advertisement

```
3 -> 2 -> 1
```

```
“d: path (3,2,1)”
```

```
1
```

```
“d: path (1)”
```

```
2
```

```
“d: path (2,1)”
```
Path-Vector: Flexible Policies

Each node can apply local policies

- Path selection: Which path to use?
- Path export: Which paths to advertise?

Examples

- Node 2 may prefer the path “2, 3, 1” over “2, 1”
- Node 1 may not let node 3 hear the path “1, 2”
End-to-End Signaling

Establish end-to-end path in advance
- Learn the topology (as in link-state routing)
- End host or router computes and signals a path

Routers supports virtual circuits
- Signaling: install entry for each circuit at each hop
- Forwarding: look up the circuit id in the table

Used in MPLS with RSVP
Source Routing

Similar to end-to-end signaling

- But the data packet carries the hops in the path
- … rather than the routers storing big tables

End-host control

- Tell the end host the topology
- Let the end host select the end-to-end path

Variations of source routing

- Strict: specify every hop
- Loose: specify intermediate points

Used in IP source routing (but almost always disabled)
Learning Where the Hosts Are
Finding the Hosts

Building a forwarding table

- Computing paths between network elements
- … and figuring out where the end-hosts are
- … to map a destination address to an outgoing link

How to find the hosts?

- Learning/flooding
- Injecting into routing protocol
- Dissemination via different protocol
- Directory service
Learning and Flooding

When a frame arrives
- Inspect the source address
- Associate address with the incoming interface

When the frame has an unfamiliar destination
- Forward out all interfaces
- … except for the one where the frame arrived

Switch learns how to reach A.

When in doubt, shout!

Used in Ethernet LANs
Inject into Routing Protocol

Treat the end host (or subnet) as a node
- And disseminate in the routing protocol
- E.g., flood information about where addresses attach

Used in OSPF and IS-IS, especially in enterprise networks
Disseminate With Another Protocol

Distribute using another protocol

- One router learns the route
- … and shares the information with other routers

Internal BGP (iBGP) used in backbone networks
Directory Service

Contact a service to learn the location

- Lookup the end-host or subnet address
- … and learn the label to put on the packet
- … to get the traffic to the right egress point

directory

“Host d is at egress e”

Encapsulate packet to send to egress e.

Used in some data centers
Conclusion

Routing is challenging

- Distributed computation
- Challenges with scalability and dynamics

Many different solutions for different environments

- **Ethernet LAN:** spanning tree, MAC learning, flooding
- **Enterprise:** link-state, inject subnet addresses
- **Backbone:** link-state inside, path-vector routing with neighboring domains, and iBGP dissemination
- **Data centers:** many different solutions, still in flux
  - E.g., link-state routing or multiple spanning trees
  - E.g., directory service, inject subnet
“Design Philosophy of the DARPA Internet Protocols”

(ACM SIGCOMM, 1988)

David Clark
Design Goals

Primary goal
- Effective technique for multiplexed utilization of existing interconnected networks (e.g., ARPAnet, packet radio)

Important goals
- Survivability in the face of failure
- Multiple types of communication service
- Wide variety of network technologies

Less important goals
- Distributed management of resources
- Cost effectiveness
- Host attachment with low level of effort
- Accountability of resources
Consequences of the Goals

Effective multiplexed utilization of existing networks
- Packet switching, not circuit switching

Continued communication despite network failures
- Routers don’t store state about ongoing transfers
- End hosts provide key communication services

Support for multiple types of communication service
- Multiple transport protocols (e.g., TCP and UDP)

Accommodation of a variety of different networks
- Simple, best-effort packet delivery service
- Packets may be lost, corrupted, or delivered out of order

Distributed management of network resources
- Multiple institutions managing the network
- Intradomain and interdomain routing protocols
Questions

What if we started with different goals?
- Network management
- Less concern about backwards compatibility
- More concern about security

Can we address new challenges
- Management, security, privacy, sensor nets, …
- Without sacrificing the other goals?
- Without a major change to the architecture?
“End-to-End Routing Behavior in the Internet”

(ACM SIGCOMM, 1996; ToN, 1997)

Vern Paxson
Measurement With Traceroute

Traceroute tool to measure the forwarding path

- Send packets with TTL=1, 2, 3…
- Record the source of the “time exceeded” message

Useful, but introduces many challenges

- Path changes
- Non-participating nodes
- Inaccurate, two-way measurements
Questions

Why can’t we measure the Internet more directly?
- What can we do about it?

Right division of labor between host and network?
- For path selection
- For network monitoring

How do we fix these routing problems?
- In a decentralized, federated network
- How to incentivize better network management
Backup Slides on Paxson Paper
Paxson Study: Forwarding Loops

Forwarding loop
- Packet returns to same router multiple times

May cause traceroute to show a loop
- If loop lasted long enough
- So many packets traverse the loopy path

Traceroute may reveal false loops
- Path change that leads to a longer path
- Causing later probe packets to hit same nodes

Heuristic solution
- Require traceroute to return same path 3 times
Paxson Study: Causes of Loops

Transient vs. persistent
- Transient: routing-protocol convergence
- Persistent: likely configuration problem

Challenges
- Appropriate time boundary between the two?
- What about flaky equipment going up and down?
- Determining the cause of persistent loops?

Anecdote on recent study of persistent loops
- Provider has static route for customer prefix
- Customer has default route to the provider
Paxson Study: Path Fluttering

Rapid changes between paths
- Multiple paths between a pair of hosts
- Load balancing policies inside the network

Packet-based load balancing
- Round-robin or random
- Multiple paths for packets in a single flow

Flow-based load balancing
- Hash of some fields in the packet header
- E.g., IP addresses, port numbers, etc.
- To keep packets in a flow on one path
Paxson Study: Routing Stability

Route prevalence
- Likelihood of observing a particular route
- Relatively easy to measure with sound sampling
- Poisson arrivals see time averages (PASTA)
- Most host pairs have a dominant route

Route persistence
- How long a route endures before a change
- Much harder to measure through active probes
- Look for cases of multiple observations
- Typical host pair has path persistence of a week
Paxson Study: Route Asymmetry

Hot-potato routing

- Customer B
- Provider B
- Customer A
- Provider A
- Multiple peering points
- Early-exit routing

Other causes
- Asymmetric link weights in intradomain routing
- Cold-potato routing, where AS requests traffic enter at particular place

Consequences
- Lots of asymmetry
- One-way delay is not necessarily half of the round-trip time