Chapter 4
NETWORK LAYER

Computer Networks
Summer 2007

Overview

• Network layer services
• Routing principle: path selection
• Hierarchical routing, scalability
• IP, the Internet Protocol
  – Internet routing protocols reliable transfer
  – Intra-domain
  – Inter-domain
  – Routing convergence
• What’s inside a router?
• Advanced Topics
  – IPv6

Network layer functions

• Transport packet from sending to receiving hosts
• Network layer protocols in every host, router

Three important functions:
• path determination: route taken by packets from source to destination. There are various routing algorithms
• switching: move packets from router’s input to appropriate router output
• call setup: some network architectures require router call setup along path before data flows

Network service model

Q: What service model for “channel” transporting packets from sender to receiver?
• guaranteed bandwidth?
• preservation of inter-packet timing (no jitter)?
• loss-free delivery?
• in-order delivery?
• congestion feedback to sender?

The most important abstraction provided by network layer:

virtual circuit or datagram?
Virtual circuits

"source-to-destination path behaves much like telephone circuit"
- performance-wise
- network actions along source-to-destination path

- call setup, teardown for each call before data can flow
- each packet carries VC identifier (not destination host ID)
- every router on source-dest path maintains "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be allocated to VC
  - to get circuit-like performance

Datagram networks: The Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths

Network layer service models

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no (inferred via loss)</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed yes rate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no congestion</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed no minimum</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Datagram or VC network: why?

Internet
- data exchange among computers
  - “elastic” service, no strict timing req.
- “smart” end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at “edge”
- many link types
  - different characteristics
  - uniform service difficult

ATM
- evolved from telephony
- human conversation
  - strict timing, reliability requirements
  - need for guaranteed service
- “dumb” end systems
  - telephones
  - complexity inside network

Routing
Routing Algorithm classification
Global or decentralized?
Global
- all routers have complete topology, link cost info
- “link state” algorithms
Decentralized
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- “distance vector” algorithms

Static or dynamic?
Static
- routes change slowly over time
Dynamic
- routes change more quickly
  - periodic update
  - in response to link cost changes

Single Source Shortest Paths
- For a given source we want the shortest path to all other nodes
- Optimality principle
  - For each shortest path \( p = (v_0, v_1, \ldots, v_i) \), each subpath \( p' = (v_i, \ldots, v_j) \) is also a shortest path. If this wasn’t the case then there was a shorter path \( p'' \) from \( v_i \) to \( v_j \) which one could use to shortcut path \( p \).
- For a given source \( s \) and a node \( v \) this means
  - There is a node \( u \) such that \( c(sp(s,u)) + c(u,v) = c(sp(s,v)) \), that is, in general \( c(sp(s,u')) + c(u',v) \leq c(sp(s,v)) \).
- The single source shortest path problem results in a tree

Routing Protocol
Goal: determine “good” path (sequence of routers) through network from source to dest.

Graph abstraction for routing
- graph nodes are routers
- graph edges are physical links
  - link cost: delay, $ cost, or congestion level
- “good” path:
  - typically means minimum cost path
  - other definitions possible
Single source shortest path: Intuition

• “Once upon a time, the Chinese Emperor wanted to know the distance and the best routes from Beijing to all the major cities in his country.”
• “At the first day of the summer, a few scouts started in Beijing, taking all the roads leaving Beijing.”
• “Whenever a scout arrives first in a city, he notes the current time and the path he took, and then immediately recruits new scouts that leave the city, taking all the possible roads and trails. Then he returns to Beijing.”
• “Whenever a scout arrives second (or later) in a city, he does nothing and returns to Beijing.”
• This “algorithm” solves the single source shortest path problem… How can one prove that it is correct? How efficient is the algorithm?

A Link-State Routing Algorithm

Dijkstra's algorithm
• net topology, link costs known to all nodes
  – accomplished via “link state broadcast”
  – all nodes have same info
• computes single-source shortest path tree
  – gives routing table for source

Notation
• \( c(i,j) \): link cost from node \( i \) to \( j \).

Can be infinite if not direct neighbors, costs define adjacency matrix
• \( v.\text{distance} \): current value of cost of path from source \( s \) to destination \( v \)
• \( v.\text{visited} \): boolean variable that determines if optimal path to \( v \) was found
• \( v.\text{pred} \): the predecessor node of \( v \) in the routing tree
• \( B \): the set of blue nodes

Algorithm idea

• There are 3 groups of nodes in the network
  – To the green nodes we know the shortest path
  – The blue nodes are directly reachable from the green nodes
  – All other nodes are black

• Idea
  – Start with source \( s \) as the only green node
  – Color the best* blue node green, one after another, until all nodes are green
  (*best = minimum distance to source \( s \) of all blue nodes)

Dijkstra's Algorithm (for source \( s \) and edge costs \( c \))

\[
\begin{align*}
\text{s.visited} & := \text{true}; \ s.\text{distance} := 0; \ s.\text{pred} := s; \ // \text{init source } s \\
\text{for all nodes } v \in V \setminus s & \ do \ // \text{init all other nodes} \\
& \ v.\text{visited} := \text{false}; \ v.\text{distance} := 1; \ v.\text{pred} := \text{undefined}; \\
\text{B} & := \{\}; \ // \text{B is the set of blue nodes, initially all neighbors of } s \\
& \text{for all nodes } v \in V \setminus s \text{ that are direct neighbors of } s \\
& \ B := B + \{v\}; \ v.\text{distance} := c(s,v); \ v.\text{pred} := s; \\
\text{while } B \text{ not empty do } & \ // \text{always choose the best blue node } v \\
& \text{v} := \text{node in } B \text{ with minimum } v.\text{distance}; \\
& \ B := B - \{v\}; \\
& \ v.\text{visited} := \text{true}; \\
& \text{for all neighbors } w \text{ of } v \text{ with } w.\text{visited} = \text{false}; \ // \text{update neighbors of } v \\
& \text{if } w \ not \ in \ B \text{ then} \\
& \ B := B + \{w\}; \ w.\text{distance} := v.\text{distance} + c(v,w); \ w.\text{pred} := v; \\
& \text{if } w \in B \text{ then} \\
& \text{if } (v.\text{distance} + c(v,w) < w.\text{distance}) \text{ then} \\
& \ w.\text{distance} := v.\text{distance} + c(v,w); \ w.\text{pred} := v; \\
\end{align*}
\]
Dijkstra's algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>visited</th>
<th>Set of blue nodes B (with distance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>D (1), B (2), C (5)</td>
</tr>
<tr>
<td>1</td>
<td>A, D</td>
<td>E (2), B (2), C (4)</td>
</tr>
<tr>
<td>2</td>
<td>AD, E</td>
<td>B (2), C (3), F (4)</td>
</tr>
<tr>
<td>3</td>
<td>ADE, B</td>
<td>C (3), F (4)</td>
</tr>
<tr>
<td>4</td>
<td>ADEB, C</td>
<td>F (4)</td>
</tr>
<tr>
<td>5</td>
<td>ADEBC</td>
<td>F (4)</td>
</tr>
</tbody>
</table>

Dijkstra's algorithm: example

Step 0: Visited A, D (1), B (2), C (5)
Step 1: Visited A, D (1), E (2), B (2), C (4)
Step 2: Visited A, D, E (2), B (2), C (3), F (4)
Step 3: Visited A, D, E (2), B (2), C (3), F (4)
Step 4: Visited A, D, E (2), B (2), C (3), F (4)
Step 5: Visited A, D, E (2), B (2), C (3), F (4)

Dijkstra's algorithm, algorithm complexity

- n nodes, m (directed) edges
- Initialization costs $O(n)$ operations
- Each round in the loop visits one unvisited node, that is, there are exactly $n-1$ rounds.
- In each round you have to find and remove the minimum node distance node $v$, and update the neighbors of node $v$.
- You can do both steps in $O(n)$ time, thus $O(n^2)$ total time.

Remark 1: With a Fibonacci-Heap, one can implement the whole algorithm in $O(m + n \log n)$ time.

Remark 2: Some books claim that the algorithm complexity is $O(n \log n)$, which is clearly bogus since at least all the edges have to be examined…

Dijkstra's algorithm, correctness

Oscillations possible
- For example if link costs depend on the amount of carried traffic.
- Example: three flows to node A, with traffic 1, 1, and e (<1)

Initially:
- B and C have better routes
- D, C, B have better routes
- etc.

- How would you prove that Dijkstra's algorithm is optimal for constant (and positive!) link costs? (Not in this course.)

Distance Vector Routing: Intuition

Routing Table of b

<table>
<thead>
<tr>
<th>Destination</th>
<th>Dir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geneva</td>
<td>a</td>
</tr>
<tr>
<td>Zurich</td>
<td>c</td>
</tr>
</tbody>
</table>
Distance Vector Routing

Distance Vector Routing Algorithm

Algorithm is iterative
- continues until no nodes exchange info
- self-terminating: no “signal” to stop asynchronous
- nodes need not to iterate in lock-step distributed
- each node communicates only with direct neighbors

Routing Table with distance info
- each node has one
- a node x has for each neighbor z an entry for each destination y (as in example before); $D^x(y,z) = \text{distance from x to y through z}$
- the best route for a given destination is marked

$$D^x(y) = \min_z D^x(y,z)$$
$$D^x(y,z) = c(x,z) + D^z(y)$$

Distance table gives routing table

Distance Vector Algorithm: example

Distance table

Routing table

Outgoing link to use, cost

Destination | Dir | Dst
---|---|---
Geneva | a | 10
Zurich | c | 4
Distance Vector Routing

Local iteration caused by:
- local link cost change
- Neighbor sends a message saying that (at least) one of its least cost paths changed

Algorithm is distributed:
- each node notifies neighbors only when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary, etc.

Each node executes a loop:
- wait for (change in local link cost or msg from neighbor)
- recompute distance table
- if least cost path to any dest has changed, notify all neighbors

Distance Vector: link cost changes

- What if the cost of a link grows?
- Compare with the count to infinity problem
  (More on this later)

Count to Infinity Problem

“good news travel fast”
Link-State vs. Distance-Vector Routing Algorithms

Message complexity
- LS: with \( n \) nodes, \( m \) links, network flooded with \( O(nm) \) messages
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence
- LS: \( O(m + n \log n) \)
  - may have oscillations
- DV: convergence time varies
  - count-to-infinity problem

Robustness
- what happens if router malfunctions?

LS:
  - node can advertise incorrect link cost
  - each node computes only its own table

DV:
  - DV node can advertise incorrect path cost
  - each node’s table used by others! errors propagate thru network

Hierarchical Routing
So far we studied idealization
- all routers identical, “flat” graph

Reality
- Internet is network of networks
- Each network admin may want to control routing in own network
- You cannot store 200 million destinations in (all) routing tables; routing table exchange too massive…

Idea
- aggregate routers into groups, “autonomous systems” (AS)
- routers in same AS run same routing protocol
  - “intra-AS” routing protocol
  - routers in a different AS can run a different intra-AS routing protocol
- Special gateway routers in AS’s
  - run intra-AS routing protocol with all other routers in AS
  - run inter-AS routing protocol with other gateway routers

Intra-AS and Inter-AS routing
Gateways:
- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

Intra-AS routing within AS A
- We’ll examine specific inter-AS and intra-AS Internet routing protocols shortly
The Internet Network Layer

Host, router network layer functions:

- Transport layer: TCP, UDP
  - Routing protocols: RIP, OSPF, BGP
  - IP protocol: addressing conventions, datagram format, packet handling conventions
  - ICMP protocol: error reporting, router “signaling”

- Link layer
- Physical layer

IP Addressing

- IP address:
  - network part (high order bits)
  - host part (low order bits)
- What’s a (local) network? (from IP address perspective)
  - device interfaces with same network part of IP address
  - can physically reach each other without intervening router

How to find the networks?

- Detach each interface from router or host
- create “islands of isolated networks”

Example on the right

- Interconnected system consisting of six networks
given notion of “network”, let’s re-examine IP addresses

“class-full” addressing

class
A 0 1.0.0.0 to 127.255.255.255
B 10 128.0.0.0 to 191.255.255.255
C 110 192.0.0.0 to 223.255.255.255
D 1110 224.0.0.0 to 239.255.255.255

IP addressing: CIDR

• class-full addressing:
  – inefficient use of address space, address space exhaustion
  – e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network

• CIDR: Classless InterDomain Routing
  – network portion of address of arbitrary length
  – address format: a.b.c.d/x, where x is number of bits in network portion of address

200.23.16.0/23

IP addresses: how to get one?

How do hosts get one? (host portion)

• Either hard-coded by system admin in a file
  – Windows: control-panel→network→configuration→tcp/ip→properties
  – UNIX: /etc/rc.config

• Or DHCP: Dynamic Host Configuration Protocol
  – dynamically get address: “plug-and-play”
  – host broadcasts “DHCP discover” message
  – DHCP server responds with “DHCP offer” message
  – host requests IP address: “DHCP request” message
  – DHCP server sends address: “DHCP ack” message

Network (network portion)

• get allocated portion of ISP’s address space

ISP’s block 11001000_00010111_00010000_00000000 200.23.16.0/20
Organization 0 11001000_00010111_00010000_00000000 200.23.16.0/23
Organization 1 11001000_00010111_00010001_00000000 200.23.18.0/23
Organization 2 11001000_00010111_00010100_00000000 200.23.20.0/23
... 6... 6... 6...
Organization 7 11001000_00010111_00111110_00000000 200.23.30.0/23
Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:

Organization 0
200.23.16.0/23

Organization 1
200.23.18.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

ISP-Rs-Us

"Send me anything with addresses beginning 200.23.16.0/20"

Fly-By-Night-ISP

"Send me anything with addresses beginning 200.23.18.0/23"

ISP-Rs-Us

Organization 0
200.23.16.0/23

Organization 1
200.23.18.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

ISP-Rs-Us

"Send me anything with addresses beginning 199.31.0.0/16"

Hierarchical addressing: more specific routes

What if Organization 1 wants to change the provider?
ISPs-R-Us has a more specific route to Organization 1

Organization 0
200.23.16.0/23

Organization 2
200.23.20.0/23

Organization 7
200.23.30.0/23

ISP-Rs-Us

"Send me anything with addresses beginning 200.23.18.0/23"

Organization 1
200.23.18.0/23

IP addressing: the last word...

• How does an ISP get block of addresses?
  – from another (bigger) ISP or
  – with ICANN: Internet Corporation for Assigned Names and Numbers
    • allocates addresses
    • manages DNS
    • assigns domain names, resolves disputes

• Will there be enough IP addresses, ever?
  – No, there are some hacks around the corner (later)

Getting a datagram from source to destination

Known as “forwarding”

IP datagram:

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>#hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

misc | source IP addr | dest IP addr | data
--- | -------------- | ------------ | ------
A   | 223.1.1.1     |              |       |
B   | 223.1.1.2     | 223.1.1.4    | 2     |
C   | 223.1.1.3     | 223.1.3.27   | 2     |
D   | 223.1.2.1     |              |       |
E   | 223.1.2.2     | 223.1.2.9    | 2     |
F   | 223.1.3.1     |              |       |
G   | 223.1.3.2     |              |       |

• datagram remains unchanged, as it travels from source to destination
• addr fields of interest here
### Getting a datagram from source to destination

#### Starting at A, given IP datagram addressed to B:
- Look up network address of B
- Find B is on same network as A
- Link layer will send datagram directly to B inside link-layer frame
  - A and B are directly connected

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>#hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>223.1.2</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>223.1.3</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

**Diagram:**
- A is connected to 223.1.1, 223.1.2, and 223.1.3.
- B is connected to 223.1.1.
- Link layer sends datagram directly to B.

### Getting a datagram from source to destination

#### Starting at A with destination E:
- Look up network address of E
- E on *different* network
  - A, E not directly attached
- Routing table: next hop router to E is 223.1.1.4
- Link layer sends datagram to router 223.1.1.4 inside link-layer frame
- Datagram arrives at 223.1.1.4

<table>
<thead>
<tr>
<th>Dest. Net.</th>
<th>next router</th>
<th>#hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>223.1.1</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.2</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
<tr>
<td>223.1.3</td>
<td>223.1.1.4</td>
<td>2</td>
</tr>
</tbody>
</table>

**Diagram:**
- A is connected to 223.1.1, 223.1.2, and 223.1.3.
- E is connected to 223.1.1.4.
- Link layer sends datagram to router 223.1.1.4.
- Datagram arrives at 223.1.1.4.

### IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ver</td>
<td>4-bit IP protocol version number</td>
</tr>
<tr>
<td>length</td>
<td>16-bit length of remaining datagram</td>
</tr>
<tr>
<td>data</td>
<td>(variable length, typically a TCP or UDP segment)</td>
</tr>
<tr>
<td>options</td>
<td>Options (if any)</td>
</tr>
<tr>
<td>checksum</td>
<td>16-bit checksum</td>
</tr>
<tr>
<td>ttl</td>
<td>Time to live (decremented at each router)</td>
</tr>
<tr>
<td>id</td>
<td>16-bit source IP address</td>
</tr>
<tr>
<td>flag</td>
<td>Flags (e.g., don't fragment, more fragments)</td>
</tr>
<tr>
<td>fragment</td>
<td>Fragment number (e.g., 1 of 5)</td>
</tr>
<tr>
<td>offset</td>
<td>Offset of fragment within datagram</td>
</tr>
<tr>
<td>protocol</td>
<td>8-bit upper layer protocol type</td>
</tr>
<tr>
<td>src</td>
<td>32-bit source IP address</td>
</tr>
<tr>
<td>dest</td>
<td>32-bit destination IP address</td>
</tr>
<tr>
<td>next router</td>
<td>Next hop router</td>
</tr>
<tr>
<td>interface</td>
<td>Interface (if any)</td>
</tr>
</tbody>
</table>

**Diagram:**
- Datagram format overview with detailed labels for each field.
IP Fragmentation and Reassembly

- network links have MTU
  - max. transmission unit
  - largest possible link-level frame
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
- IP header bits used to identify, order related fragments

Fragmentation:
in: one large datagram
out: 3 smaller datagrams

Reassembly:

ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways to communication network-level information
  - unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
  - ICMP message: type, code plus first 8 bytes of IP datagram causing error

Some typical types/codes

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>

DHCP: Dynamic Host Configuration Protocol

Goals

- allow host to dynamically obtain its IP address from network server when it joins network
- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected and "on")
- Support for mobile users who want to join network (more shortly)

DHCP review

- host broadcasts "DHCP discover" message
- DHCP server responds with "DHCP offer" message
- host requests IP address: "DHCP request" message
- DHCP server sends address: "DHCP ack" message
DHCP client-server scenario

DHCP server: 223.1.2.5

arriving client

dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
yiaddr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255, 68
yiaddr: 223.1.2.4
transaction ID: 655
Lifetime: 3600 secs

• Motivation
– local network uses just one IP address as far as outside world is concerned
– no need to be allocated range of addresses from ISP
– just one IP address is used for all devices
– can change addresses of devices in local network without notifying outside world
– can change ISP without changing addresses of devices in local network
– devices inside local net not explicitly addressable, visible by outside world (a security plus).
– BUT: machines cannot be servers!
Implementation: NAT router must

• outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  – remote clients/servers will respond using (NAT IP address, new port #) as destination addr.

• remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

• incoming datagrams: replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

16-bit port-number field

– 60,000 simultaneous connections with a single LAN-side address!

NAT is controversial

– routers should only process up to layer 3
– violates end-to-end argument
  • NAT possibility must be taken into account by app designers, e.g., P2P applications
– address shortage should instead be solved by IPv6
  • delays deployment of IPv6

Routing in the Internet

• The Global Internet consists of Autonomous Systems (AS) interconnected with each other. There are several “types”
  – Stub AS: small corporation
  – Multihomed AS: large corporation (no transit)
  – Transit AS: provider

• Two-level routing
  – Intra-AS: administrator is responsible for choice
  – Inter-AS: unique standard
Internet AS Hierarchy

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common IGPs:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: number of hops (max = 15 hops)
  - *Can you guess why?*
- Distance vectors: exchanged every 30 sec via Response Message (also called “advertisement”)
- Each advertisement: route to up to 25 destination networks within AS

<table>
<thead>
<tr>
<th>Destination Network</th>
<th>Next Router</th>
<th>Num. of hops to dest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>y</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>B</td>
<td>7</td>
</tr>
<tr>
<td>x</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>

Routing table in D
RIP: Link Failure and Recovery

If no advertisement heard after 180 sec, neighbor/link declared dead
• routes via neighbor invalidated
• new advertisements sent to neighbors
• neighbors in turn send out new advertisements (if tables changed)
• link failure info quickly propagates to entire net
• poison reverse (next slide) used to prevent ping-pong loops (infinite distance = 16 hops)

Distance Vector: poisoned reverse

If Z routes through Y to get to X:
• Z tells Y its (Z’s) distance to X is infinite (so Y won’t route to X via Z)
• will this completely solve count to infinity problem…?

RIP Table processing

• RIP routing tables managed by application-level process called route-d (daemon)
• advertisements sent in UDP packets, periodically repeated

RIP Table example (continued)

• Three attached class C networks (LANs)
• Router only knows routes to attached LANs
• Default router used to “go up”
• Route multicast address: 224.0.0.0
• Loopback interface (for debugging)
OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)

OSPF “advanced” features (not in RIP)

- Security
  - all OSPF messages authenticated
  - therefore no malicious intrusion
  - TCP connections used
- Multiple same-cost paths allowed (only one path in RIP)
  - e.g., satellite link cost set “low” for best effort; high for real time
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF

- Two-level hierarchy: local area or backbone
  - Link-state advertisements only in area
  - each node has detailed area topology but only knows direction (shortest path) to nets in other areas.
- Area border routers
  - “summarize” distances to networks in own area
  - advertise to other area border routers.
- Backbone routers
  - run OSPF routing limited to backbone.
- Boundary routers
  - connect to other ASs.

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- several cost metrics (delay, bandwidth, reliability, load etc)
- uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Algorithm (DUAL) based on diffused computation

Inter-AS routing

BGP does not count to infinity
BGP does not count to infinity

**“withdraw Zurich”**

```
Destination  Dir  Dst  Path
Zurich       a  4  cdeZ
```

BGP Basics Continued

**“announce bfeZ”**

```
Destination  Dir  Dst  Path
Zurich       b  5  bfeZ
```

```
Destination  Dir  Dst  Path
Zurich       f  3  feZ
```

30s

BGP Basics Continued

**“announce bcdeZ”**

```
Destination  Dir  Dst  Path
Zurich       c  4  cdeZ
```

backup
active

```
Destination  Dir  Dst  Path
Zurich       b  4  bfeZ
```

backup
active
BGP (Border Gateway Protocol)

- BGP is the Internet de-facto standard
- Path Vector protocol

5) Receive BGP update (announce or withdrawal) from a neighbor.
6) Update routing table.
7) Does update affect active route? (Loop detection, policy, etc.) If yes, send update to all neighbors that are allowed by policy.

MinRouteAdver: At most 1 announce per neighbor per 30+jitter seconds.
Store the active routes of the neighbors.

Internet Architecture

<table>
<thead>
<tr>
<th>Destination</th>
<th>Dir</th>
<th>Dst</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zurich</td>
<td>c</td>
<td>4</td>
<td>cdeZ</td>
</tr>
<tr>
<td>172.30.160/19</td>
<td>R1</td>
<td>4</td>
<td>1239 1 3561</td>
</tr>
</tbody>
</table>

- iBGP
- Route flap dampening
- Multipath
- Soft configuration
- ...

Robustness of BGP

- We are interested in routes to destination d.
- Nodes a, b, c all have the policy to prefer a 2-hop route through their clockwise neighbor over a direct 1-hop route to destination d.
Cisco bug “withdraw loop” is fixed with IOS release.


Internet Evolution: Today

Experimental Setup

- Analyzed secondary paths of 20x20 AS pairs:
  - Inject and monitor BGP faults.
  - Survey providers on policies.
BGP Convergence Times

- If a link comes up, the convergence time is in the order of time to forward a message on the shortest path.
- If a link goes down, the convergence time is in the order of time to forward a message on the longest path.

Intuition for Slow Convergence
Intuition for Slow Convergence

Convergence in the time to forward a message on the longest path.

Example of BGP Convergence

Example of BGP Convergence

Remember the Example

What might help?

• Idea: Attach a “cause tag” to the withdrawal message identifying the failed link/node (for a given prefix).

• It can be shown that a cause tag reduces the convergence time to the shortest path

• Problems
  – Since BGP is widely deployed, it cannot be changed easily
  – ISP’s (AS’s) don’t like the world to know that it is their link that is not stable, and cause tags do exactly that.
  – Race conditions make the cause tags protocol intricate
Example with BGP-CT (Cause Tags)

Example with BGP-CT

Convergence Time using Cause Tags

O.1s
Convergence Time using Cause Tags

Why are Intra- and Inter-AS routing different?

- **Policy**
  - Inter-AS: admin wants control over how its traffic routed, and who routes through its net.
  - Intra-AS: single admin, so no policy decisions needed
- **Scale**
  - hierarchical routing saves table size, reduced update traffic
- **Performance**
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

Convergence in the time to forward a message on the new shortest path (instead of the longest).

Router Architecture Overview

- Two key router functions
  - run routing algorithms/protocols (RIP, OSPF, BGP)
  - switch datagrams from incoming to outgoing link
Decentralized switching
- given datagram dest., lookup output port using routing table in input port memory
- goal: complete input port processing at “line speed”
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Physical layer
- bit-level reception

Data link layer
- e.g., Ethernet
- see chapter 5

Input Port Functions

Fabric slower that input ports combined
- queueing may occur at input queues

Head-of-the-Line (HOL) blocking
- queued datagram at front of queue prevents others in queue from moving forward
- queuing delay and loss due to input buffer overflow

Input Port Queuing

Three types of switching fabrics

Switching Via Memory

First generation routers
- packet copied by system’s (single) CPU
- speed limited by memory bandwidth (2 bus crossings per datagram)

Modern routers
- input port processor performs lookup, copy into memory
- Cisco Catalyst 8500
Switching Via Bus or Interconnection Network

- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
- 1 Gbps bus, Cisco 1900: sufficient speed for access and enterprise routers (not regional or backbone)

Interconnection Network: overcome bus bandwidth limitations
- Banyan networks, other interconnection nets initially developed to connect processors in multiprocessor
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches Gbps through the interconnection network

Butterfly with Dimension d=4

Output ports

Buffering required when datagrams arrive from fabric faster than the transmission rate
- Scheduling discipline chooses among queued datagrams for transmission

IPv6

- Initial motivation: 32-bit address space completely allocated by 2008.
- Additional motivation
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS (quality of service)
  - new "anycast" address: route to "best" of several replicated servers
- IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed
IPv6 Header

- **Priority**
  - identify priority among datagrams in flow

- **Flow Label**
  - identify datagrams in same “flow” (concept of “flow” not well defined)

- **Next header**
  - identify upper layer protocol for data

Other Changes from IPv4

- **Checksum**
  - removed entirely to reduce processing time at each hop

- **Options**
  - allowed, but outside of header
  - indicated by “Next Header” field

- **ICMPv6**: new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneously
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?

- Two proposed approaches
  - Dual Stack
    - some routers with dual stack (v6, v4) can “translate” between formats
  - Tunneling
    - IPv6 carried as payload in IPv4 datagram among IPv4 routers

Dual Stack Approach
Tunneling

IPv6 inside IPv4 where needed

Distributed Computing Group
Computer Networks
R. Wattenhofer