Routers & Routing

- High-speed router architecture
- Intro to routing protocols
- Assigned reading
  - [McK97] A Fast Switched Backplane for a Gigabit Switched Router
  - Know RIP/OSPF

Speeding up Prefix Match - Alternatives

- Route caches
  - Temporal locality
  - Many packets to same destination
- Other algorithms
  - Waldvogel – Sigcomm 97
    - Binary search on prefixes
    - Works well for larger addresses
  - Bremler-Barr – Sigcomm 99
    - Clue = prefix length matched at previous hop
    - Why is this useful?

Binary Search on Ranges

- Encode each prefix as range and place all range endpoints in binary search table or tree. Need two next hops per entry for > and = case. [Lampson, Srinivasan, Varghese]

- Problem: Slow search ($\log_2 N + 1 = 20$ for a million prefixes) and update ($O(n)$).
Speeding up Prefix Match - Alternatives

- Content addressable memory (CAM)
  - Hardware based route lookup
  - Input = tag, output = value associated with tag
  - Requires exact match with tag
    - Multiple cycles (1 per prefix searched) with single CAM
    - Multiple CAMs (1 per prefix) searched in parallel
  - Ternary CAM
    - 0,1,don’t care values in tag match
    - Priority (i.e. longest prefix) by order of entries in CAM

Outline

- Packet classification
- IP router design
- Routing protocols – distance vector
- Routing protocols – link state

Packet Classification

- Typical uses
  - Identify flows for QoS
  - Firewall filtering
- Requirements
  - Match on multiple fields
  - Strict priority among rules
    - E.g. 1. no traffic from 128.2.*
      2. ok traffic on port 80

Complexity

- N rules and k header fields for k > 2
  - O(log N^{k-1}) time and O(N) space
  - O(log N) time and O(N^k) space
- How many rules?
  - Largest for firewalls & similar → 1700
  - DiffServ/QoS → much larger → 100k (?)
Bit Vectors

Observations [GM99]

- Common rule sets have important/useful characteristics
  - Packets rarely match more than a few rules (rule intersection)
    - E.g., max of 4 rules seen on common databases up to 1700 rules
  - Special cases for $k = 2 \rightarrow$ source and destination
    - $O(\log N)$ time and $O(N)$ space solutions exist

Aggregating Rules [BV01]

- Common case: very few 1’s in bit vector $\rightarrow$ aggregate bits
- OR together A bits at a time $\rightarrow$ N/A bit-long vector
  - A typically chosen to match word-size
  - Can be done hierarchically $\rightarrow$ aggregate the aggregates
- AND of aggregate bits indicates which groups of A rules have a possible match
  - Hopefully only a few 1’s in AND’ed vector
  - AND of aggregated bit vectors may have false positives
- Fetch and AND just bit vectors associated with positive entries
Rearranging Rules [BV01]

- Problem: false positives may be common
- Solution: reorder rules to minimize false positives
  - What about the priority order of rules?
- How to rearrange?
  - Heuristic: sort rules based on single field’s values
    - First sort by prefix length then by value
    - Moves similar rules close together → reduces false positives

Summary: Addressing/Classification

- Global addressing matches design objectives of Internet
- Speed of forwarding lookup/classification of packets is a key challenge for routers
  - Especially high-speed backbone routers
- CIDR provided more structured routing
- Good examples of using common case optimization
  - Routing with a clue
  - Classification with few matching rules

Outline

- Packet classification
- IP router design
- Routing protocols – distance vector
- Routing protocols – link state

What Does a Router Look Like?

- Network controller
- Line cards
- Backplane
**Network Processor**

- Runs routing protocol and downloads forwarding table to line cards
  - Some line cards maintain two forwarding tables to allow easy switchover
- Performs “slow” path processing
  - Handles ICMP error messages
  - Handles IP option processing

**Line Cards**

- Network interface cards
- Provides parallel processing of packets
- Fast path per-packet processing
  - Forwarding lookup (hardware/ASIC vs. software)

**Switch Design Issues**

- Have N inputs and M outputs
  - Multiple packets for same output – output contention
  - Switch contention – switch cannot support arbitrary set of transfers
    - Crossbar
    - Bus
    - High clock/transfer rate needed for bus
    - Banyan net
    - Complex scheduling needed to avoid switch contention
- Solution – buffer packets where needed

**Switch Buffering**

- Input buffering
  - Which inputs are processed each slot – schedule?
  - Head of line packets destined for busy output blocks other packets
- Output buffering
  - Output may receive multiple packets per slot
  - Need speedup proportional to # inputs
- Internal buffering
  - Head of line blocking
  - Amount of buffering needed
Line Card Interconnect

- Virtual output buffering
  - Maintain per output buffer at input
  - Solves head of line blocking problem
  - Each of MxN input buffer places bid for output
- Crossbar connect
- Challenge: map of bids to schedule for crossbar

ISLIP

ISLIP (cont.)

Summary - IP router design

- Different architectures for different types of routers
- High speed routers incorporate large number of processors
- Common case is optimized carefully
Outline

• Packet classification
• IP router design
• Routing protocols – distance vector
• Routing protocols – link state

Factors Affecting Routing

• Routing algorithms view the network as a graph
• Problem: find lowest cost path between two nodes
• Factors
  • Static topology
  • Dynamic load
  • Policy

Two Main Approaches

• Distance-vector (DV) protocols
• Link state (LS) protocols

Distance Vector Protocols

• Employed in the early Arpanet
• Distributed next hop computation
• Unit of information exchange
  • Vector of distances to destinations
• Distributed Bellman-Ford Algorithm
The Bouncing Effect

C Sends Routes to B

B Updates Distance to A

B Sends Routes to C
C Sends Routes to B

How are These Loops Caused?

- Observation 1:
  - B’s metric increases
- Observation 2:
  - C picks B as next hop to A
  - But, the implicit path from C to A includes itself!

Solution 1: Holddowns

- If metric increases, delay propagating information
  - In our example, B delays advertising route
  - C eventually thinks B’s route is gone, picks its own route
  - B then selects C as next hop
  - Adversely affects convergence

Other “Solutions”

- Split horizon
  - C does not advertise route to B
- Poisoned reverse
  - C advertises route to B with infinite distance
  - Works for two node loops
  - Does not work for loops with more nodes
Example Where Split Horizon Fails

- When link breaks, C marks D as unreachable and reports that to A and B
- Suppose A learns it first
  - A now thinks best path to D is through B
  - A reports D unreachable to B and a route of cost=3 to C
- C thinks D is reachable through A at cost 4 and reports that to B
- B reports a cost 5 to A who reports new cost to C
- etc...

Avoiding the Bouncing Effect

- Select loop-free paths
- One way of doing this:
  - Each route advertisement carries entire path
  - If a router sees itself in path, it rejects the route
- BGP does it this way
- Space proportional to diameter

Loop Freedom at Every Instant

- Does bouncing effect avoid loops?
  - No! Transient loops are still possible
  - Why? Because implicit path information may be stale
  - See this in BGP convergence
- Only way to fix this
  - Ensure that you have up-to-date information by explicitly querying

Distance Vector in Practice

- RIP and RIP2
  - Uses split-horizon/poison reverse
- BGP
  - Propagates entire path
  - Path also used for effecting policies
Outline

- Packet classification
- IP router design
- Routing protocols – distance vector
- Routing protocols – link state

Basic Steps

- Start condition
  - Each node assumed to know state of links to its neighbors
- Step 1
  - Each node broadcasts its state to all other nodes
  - Reliable flooding mechanism
- Step 2
  - Each node locally computes shortest paths to all other nodes from global state
  - Dijkstra’s shortest path tree (SPT) algorithm

Link State Packets (LSPs)

- Periodically, each node creates a link state packet containing:
  - Node ID
  - List of neighbors and link cost
  - Sequence number
    - Needed to avoid stale information from flood
  - Time to live (TTL)
  - Node outputs LSP on all its links

Reliable Flooding

- When node J receives LSP from node K
  - If LSP is the most recent LSP from K that J has seen so far, J saves it in database and forwards a copy on all links except link LSP was received on
  - Otherwise, discard LSP
- How to tell more recent
  - Use sequence numbers
  - Same method as sliding window protocols
Link State Characteristics

- With consistent LSDBs, all nodes compute consistent loop-free paths
- Limited by Dijkstra computation overhead, space requirements
- Can still have transient loops

Packet from C→A may loop around BDC if B knows about failure and C & D do not

Summary: LS vs. DV

- Convergence speed:
  - LS: faster – don’t need to process LSPs before forwarding
  - DV: fast with triggered updates
- Space requirements:
  - LS maintains entire topology
  - DV maintains only neighbor state

Summary: LS vs. DV

- Robustness:
  - LS can broadcast incorrect/corrupted LSP
    - Can be made robust since sources are aware of alternate paths
  - DV can advertise incorrect paths to all destinations
    - Incorrect calculation can spread to entire network

- In DV send everything you know to your neighbors
- In LS send info about your neighbors to everyone
- Msg size: small with LS, potentially large with DV
- Msg exchange: LS: O(nE), DV: only to neighbors
Summary: LS vs. DV

- In LS nodes must compute consistent routes independently - must protect against LSDB corruption
- In DV routes are computed relative to other nodes
- Bottom line: no clear winner, but we see more frequent use of LS in the Internet

Next Lecture: Inter-Domain Routing

- Border Gateway Protocol (BGP)
- Assigned reading
  - [LAB00] Delayed Internet Routing Convergence
  - [Nor00] Internet Service Providers and Peering

Distributed Bellman-Ford

- Start Conditions:
  - Each router starts with a vector of (zero) distances to all directly attached networks
- Send step:
  - Each router advertises its current vector to all neighboring routers
- Receive step:
  - Upon receiving vectors from each of its neighbors, router computes its own distance to each neighbor
  - Then, for every network X, router finds that neighbor who is closer to X than any other neighbor
  - Router updates its cost to X
  - After doing this for all X, router goes to send step

Example - Initial Distances
E Receives D’s Routes; Updates Cost

A receives B’s; Updates Cost

A receives E’s routes; Updates Costs

Final Distances
**View From a Node**

- **E’s routing table**
  - **dest** | **Next hop**
  - A | B | D
  - **A** | 1 | 4 | 5
  - **B** | 7 | 8 | 9
  - **C** | 6 | 9 | 4
  - **D** | 4 | 11 | 2

**Final Distances After Link Failure**

- **Info at Node**
  - **Distance to Node**
  - **A** | **B** | **C** | **D** | **E**
  - **A** | 0 | 7 | 8 | 10 | 1
  - **B** | 7 | 0 | 1 | 3 | 8
  - **C** | 8 | 1 | 0 | 2 | 9
  - **D** | 10 | 3 | 2 | 0 | 11
  - **E** | 1 | 8 | 9 | 11 | 0

**SPT Algorithm (Dijkstra)**

- **SPT = {a}**
  - for all nodes **v**
    - if **v** adjacent to **a** then **D(v)** = cost (a, v)
    - else **D(v)** = infinity
  - **Loop**
    - find **w** not in SPT, where **D(w)** is min
    - add **w** in SPT
    - for all **v** adjacent to **w** and not in SPT
      - **D(v)** = min (**D(v)**, **D(w)** + **C(w, v)**)
    - until all nodes are in SPT

**SPT Example**

- **step** | **SPT** | **D(b)**, **P(b)** | **D(c)**, **P(c)** | **D(d)**, **P(d)** | **D(e)**, **P(e)** | **D(f)**, **P(f)**
  - 0 | A | 2, A | 5, A | 1, A | - | -
### Example

<table>
<thead>
<tr>
<th>Step</th>
<th>SPT</th>
<th>D(b), P(b)</th>
<th>D(c), P(c)</th>
<th>D(d), P(d)</th>
<th>D(e), P(e)</th>
<th>D(f), P(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td>1, A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2, A</td>
<td>4, D</td>
<td>2, D</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2, A</td>
<td>3, E</td>
<td>4, E</td>
<td>4, E</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3, E</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

---

### Example

<table>
<thead>
<tr>
<th>Step</th>
<th>SPT</th>
<th>D(b), P(b)</th>
<th>D(c), P(c)</th>
<th>D(d), P(d)</th>
<th>D(e), P(e)</th>
<th>D(f), P(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td>1, A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2, A</td>
<td>4, D</td>
<td>2, D</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2, A</td>
<td>3, E</td>
<td>4, E</td>
<td>4, E</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3, E</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td>4, E</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>step</th>
<th>SPT</th>
<th>D(b), P(b)</th>
<th>D(c), P(c)</th>
<th>D(d), P(d)</th>
<th>D(e), P(e)</th>
<th>D(f), P(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>2, A</td>
<td>5, A</td>
<td>1, A</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>1</td>
<td>AD</td>
<td>2, A</td>
<td>4, D</td>
<td>2, D</td>
<td>~</td>
<td>~</td>
</tr>
<tr>
<td>2</td>
<td>ADE</td>
<td>2, A</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ADEB</td>
<td>3, E</td>
<td>4, E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ADEBC</td>
<td></td>
<td></td>
<td></td>
<td>4, E</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ADEBCF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4, E</td>
</tr>
</tbody>
</table>