Computer Networks

Lecture 21

Distance Vector Routing Protocols

11/14/2013

Outline

- Admin and recap
- Distance vector protocols
Recap: Resource Allocation Framework

- Forward (design) engineering:
  - how to determine objective functions
  - given objective, how to design effective alg

\[
\begin{align*}
\text{max} & \quad \sum_{f \in F} U_f(x_f) \\
\text{subject to} & \quad Ax \leq C \\
\text{over} & \quad x \geq 0
\end{align*}
\]

- Reverse (understand) engineering:
  - understand current protocols (what are the objectives of TCP/Reno, TCP/Vegas?)
Recap: Internet Network Layer: Protocols

Network layer functions:

Routing protocols
- path selection
  e.g., RIP, OSPF, BGP

Control protocols
- error reporting
  e.g., ICMP

Network layer protocol (e.g., IP)
- addressing conventions
- packet format
- packet handling conventions

Control protocols
- router “signaling”
  e.g., RSVP

Recap: Data (Forwarding) Plane

Local forwarding table

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet's header
Recap: Routing Design Space

Routing

Goal: determine “good” paths (sequences of routers) thru network from source to dest.

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
  - how many paths from source s to destination d?
    - multi-path routing
    - single path routing
  - does the route(s) provide QoS?
    - QoS
    - best effort, shortest path
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing
Shortest/Lowest Cost Path Routing

- Represent network as a graph, with positive costs assigned to network links
- The path from a source to a destination chosen by the routing protocol is a shortest (lowest cost) path among all possible paths

Routing: Single-Path, Adaptive Routing

- Assume link costs reflect current traffic

Solution: Link costs are a combination of current traffic intensity (dynamic) and topology (static). To improve stability, the static topology part should be large. Thus less sensitive to traffic; thus non-adaptive.
Example: Cisco Proprietary Recommendation on Link Cost

- Link metric:
  - \( \text{metric} = \left[ K_1 \times \text{bandwidth}^{-1} + (K_2 \times \text{bandwidth}^{-1}) / (256 - \text{load}) + K_3 \times \text{delay} \right] \times \left[ K_5 / (\text{reliability} + K_4) \right] \)

By default, \( k_1=k_3=1 \) and \( k_2=k_4=k_5=0 \). The default composite metric for EIGRP, adjusted for scaling factors, is as follows:

\[
\text{EIGRP}_{\text{metric}} = 256 \times \left\{ \left[ \frac{10^7}{\text{BW}_{\text{min}}} \right] + \left[ \text{sum of delays} \right] \right\}
\]

\( \text{BW}_{\text{min}} \) is in kbps and the sum of delays are in 10s of microseconds.

Example: EIGRP Link Cost

- The bandwidth and delay for an Ethernet interface are 10 Mbps and 1 ms, respectively.
- The calculated EIGRP BW metric is as follows:
  - \( 256 \times 10^7 / \text{BW} = 256 \times 10^7 / 10,000 \)
  - \( = 256 \times 10000 \)
  - \( = 2560000 \)
**Outline**

- Admin and recap
  - *Distance vector protocols*

---

**Distance Vector Routing**

- Basis of RIP, IGRP, EIGRP routing protocols

- Distributed alg to compute shortest paths
  - conceptually, runs for each destination separately
    - hence we consider one dest only
  - **state**: each node maintains a current estimate of distance to the destination
    - \(d\), denotes the distance estimation from node \(i\) to dest
  - **update rule**: based on Bellman-Ford alg.
Distance Vector Routing: Update

- At node i, the basic update rule:

\[ d_i = \min_{j \in N(i)} (d_{ij} + d_j) \]

where
- \( d_i \) denotes the distance estimation from i to the destination,
- \( N(i) \) is set of neighbors of node i, and
- \( d_{ij} \) is the distance of the direct link from i to j

Synchronous Bellman-Ford (SBF)

- Nodes update in rounds:
  - there is a global clock;
  - at the beginning of each round, each node sends its estimate to dest to all of its neighbors;
  - at the end of the round, updates its estimation

\[ d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h)) \]
Outline

- Network overview
- Control plane: routing overview
  - Distance vector protocols
    - synchronous Bellman-Ford (SBF)
      - SBF/∞

SBF/∞

- Initialization (time 0):

\[ d_i(0) = \begin{cases} 0 & i = \text{dest} \\ \infty & \text{otherwise} \end{cases} \]
Consider D as destination; \( d(t) \) is a vector consisting of estimation of each node at round \( t \)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d(0) )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>0</td>
</tr>
<tr>
<td>( d(1) )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>( d(2) )</td>
<td>12</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>( d(3) )</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>( d(4) )</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Observations:
- \( d(0) \geq d(1) \geq d(2) \geq d(3) \geq d(4) \)
- After a few iterations \( d(n) = d(n+1) = d^* \)

\[
d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h))
\]

**A Nice Property of SBF: Monotonicity**

- Consider two configurations \( d(t) \) and \( d'(t) \)
- If \( d(t) \geq d'(t) \)
  - i.e., each node has a higher estimate in \( d \) than in \( d' \),
- then \( d(t+1) \geq d'(t+1) \)
  - i.e., each node has a higher estimate in \( d \) than in \( d' \) after one round of synchronous update.
Correctness of SBF/∞

- **Claim:** \(d_i(h)\) is the length \(L_i(h)\) of a shortest path from \(i\) to the destination using \(\leq h\) hops
  - base case: \(h = 0\) is trivially true
  - assume true for \(\leq h\), i.e., \(L_i(h) = d_i(h)\), \(L_i(h-1) = d_i(h-1)\), ...

\[
d_i(h + 1) = \min_{j \in N(i)} (d_{ij} + d_j(h))
\]

Correctness of SBF/∞

- **consider** \(\leq h+1\) hops:
  \[
  L_i(h + 1) = \min(L_i(h), \min_{j \in N(i)} (d_{ij} + L_j(h}))
  = \min(d_i(h), \min_{j \in N(i)} (d_{ij} + d_j(h)))
  = \min(d_i(h), d_i(h + 1))
  \]

  since \(d_i(h) \leq d_i(h-1)\)

\[
d_i(h + 1) = \min_{j \in N(i)} (d_{ij} + d_j(h)) \leq \min_{j \in N(i)} (d_{ij} + d_j(h - 1)) = d_i(h)
\]

\[
L_i(h + 1) = d_i(h + 1)
\]
Bellman Equation

- We referred to the update rules as Bellman equations (BE):
  \[ d_i = \min_{j \in N(i)} (d_{ij} + d_j) \]
  where \( d_D = 0 \).
- SBF/∞ solves the equations in a distributed way
- Does the equation have a unique solution (i.e., the shortest path one)?

Uniqueness of Solution to BE

- Assume another solution \( d \), we will show that \( d = d^* \)
  case 1: we show \( d \geq d^* \)

Since \( d \) is a solution to BE, we can construct paths as follows: for each \( i \), pick a \( j \) which satisfies the equation; since \( d^* \) is shortest, \( d \geq d^* \)
Uniqueness of Solution to BE

Case 2: we show $d \leq d^*$

assume we run SBF with two initial configurations:
- one is $d$
- another is SBF/$\infty$ ($d^\infty$),

$\rightarrow$ monotonicity and convergence of SBF/$\infty$ imply that $d \leq d^*$

Outline

- Network overview
- Control plane: routing overview
  - Distance vector protocols
    - synchronous Bellman-Ford (SBF)
      - SBF/$\infty$
      - SBF/-1
Initialization (time 0):

\[ d_i(0) = \begin{cases} 
0 & i = \text{dest} \\
-1 & \text{otherwise}
\end{cases} \]

Example

Consider D as destination

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>d(0)</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>d(1)</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(2)</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(3)</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(4)</td>
<td>9</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(5)</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(6)</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Observation: \[ d(0) \leq d(1) \leq d(2) \leq d(3) \leq d(4) \leq d(5) = d(6) = d^* \]
**Correctness of SBF/-1**

- SBF/-1 converges due to monotonicity
- At equilibrium, SBF/-1 satisfies Bellman equations:
  \[
  d_i^{(h+1)} = \min_{j \in N(i)} (d_{ij} + d_j^{(h)})
  \]
  where \(d_0 = 0\).
- Another solution is shortest path solution \(d^*\)
- Since there is a unique solution to the BE equations; thus SBF/-1 converges to shortest path

**Discussion**

- How do you prove that SBF converges under other non-negative initial conditions?
**Toolbox**

- A key technique for proving convergence (liveness) of distributed protocols: two extreme configurations to sandwich any real configurations

**Discussion**

- Problem of SBF?

\[ d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h)) \]
Outline

- Network overview
- Control plane: routing overview
  - Distance vector protocols
    - synchronous Bellman-Ford (SBF)
    - asynchronous Bellman-Ford (ABF)

Asynchronous Bellman-Ford (ABF)

- No notion of global iterations
  - each node updates at its own pace
- Asynchronously each node $i$ computes
  \[
  d_i = \min_{j \in N(i)} (d_{ij} + d_j^i)
  \]
  using last received value $d_j^i$ from neighbor $j$.

- Asynchronous node $j$ sends its estimate to its neighbor $i$:
  - there is an upper bound on the delay of estimate packets (no worry for out of order)
## Distance Table: Example

Below is just one step! The protocol repeats forever!

<table>
<thead>
<tr>
<th>Destination</th>
<th>Distance Tables from Neighbors</th>
<th>Computation</th>
<th>E's Distance Table</th>
<th>Distance Table E sends to its neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 7 ∞</td>
<td>10 15 ∞</td>
<td>A: 10</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>7 0 ∞</td>
<td>17 8 ∞</td>
<td>B: 8</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞ 1 2</td>
<td>∞ 9 4</td>
<td>D: 4</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞ ∞ 0</td>
<td>∞ ∞ 2</td>
<td>D: 2</td>
<td></td>
</tr>
</tbody>
</table>

Below is just one step! The protocol repeats forever!

### Asynchronous Bellman-Ford (ABF)

- In general, nodes are using different and possibly inconsistent estimates
Asynchronous Bellman-Ford (ABF)

- ABF will eventually converge to the shortest path
  - links can go down and come up - but if topology is stabilized after some time $t$, ABF will eventually converge to the shortest path!

- If the network is connected, then ABF converges in finite amount of time, if conditions are met

ABF Convergence

- There are too many different “runs” of ABF, so need to use monotonicity

- Consider two sequences:
  - $SBF/\infty$: call the sequence $U()$
  - $SBF/-1$: call the sequence $L()$
System State

where can distance estimate from node j appear?

three types of distance estimates from node j:
- \( d_j \): current distance estimate at node j
- \( d_i^j \): last \( d_j \) that neighbor i received
- \( d_i^j \): those \( d_j \) that are still in transit to neighbor i
**ABF Convergence**

- Consider the time when the topology is stabilized as time 0

- $U(0)$ and $L(0)$ provide upper and lower bound at time 0 on all corresponding elements of states
  - $L_j(0) \leq d_j \leq U_j(0)$ for all $d_j$ state at node $j$
  - $L_j(0) \leq d'_j \leq U_j(0)$
  - $L_j(0) \leq \text{update messages } d'_j \leq U_j(0)$

- $d_j$:
  - after at least one update at node $j$:
    - $d_j$ falls between $L_j(1) \leq d_j \leq U_j(1)$

- $d'_{ij}$:
  - eventually all $d'_{ij}$ that are only bounded by $L_j(0)$ and $U_j(0)$ are replaced with in $L_j(1)$ and $U_j(1)$
Asynchronous Bellman-Ford: Summary

- Distributed: each node communicates its routing table to its directly-attached neighbors
- Iterative: continues periodically or when link changes, e.g. detects a link failure
- Asynchronous: nodes need not exchange info/iterate in lock step!
- Convergence in finite steps, independent of initial condition if network is connected

Properties of Distance-Vector Algorithms

- Good news propagate fast

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initially</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>After 1 exchange</td>
<td>1</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>After 2 exchanges</td>
<td>1</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>After 3 exchanges</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>After 4 exchanges</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>∞</td>
</tr>
</tbody>
</table>
### Properties of Distance-Vector Algorithms

- **Bad news propagate slowly (link A-B broke)**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- This is called the *counting-to-infinity* problem

  Question: why does counting-to-infinity happen?

### What is a Routing Loop?

- A routing loop is a *global state* (consisting of the nodes' local states) at a global moment (observed by an oracle) such that there exist nodes A, B, C, ... E such that A (locally) thinks B as down stream, B thinks C as down stream, ... E thinks A as down stream

- Counting-to-infinity because of routing loops
The Reverse-Poison (Split-horizon) Hack

If the path to dest is through neighbor h, report ∞ to neighbor h for dest.

<table>
<thead>
<tr>
<th>D^E()</th>
<th>A</th>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>7</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>distance from neighbors</th>
<th>computation</th>
<th>E's distance table</th>
<th>distance table E sends to its neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>1, A</td>
<td>1</td>
<td>15</td>
<td>1, A</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8, B</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>4</td>
<td>4, D</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2, D</td>
</tr>
</tbody>
</table>

distance through neighbor

A sends the news to C
C sends the news to B

An Example Where Split-horizon Fails

- C and D fails, C will set its distance to D as ∞
- C sends the bad news (∞) to A
- A switches to use B to go to D
- A sends the news to C
- C sends the news to B

Question: what is the routing loop formed?
Outline

- Admin and recap
  - Distance vector protocols
    - synchronous Bellman-Ford (SBF)
    - asynchronous Bellman-Ford (ABF)
  - RIP, EIGRP

Example: RIP (Routing Information Protocol)

- Distance vector
- Included in BSD-UNIX Distribution in 1982
- Link cost: 1
- Distance metric: # of hops
- Distance vectors
  - exchanged every 30 sec via Response Message (also called advertisement) using UDP
  - each advertisement: route to up to 25 destination nets
RIP (Routing Information Protocol)

Routing table in I

<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>

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
- reverse-poison used to prevent ping-pong loops
- set infinite distance = 16 hops (why?)
**EIGRP Neighbor Discovery**

- EIGRP routers actively establish relationships with their neighbors
  - EIGRP routers establish adjacencies with neighbor routers by using small **hello** packets.
  - The **Hello protocol** uses a multicast address of **224.0.0.10**, and all routers periodically send hellos.
  - Those receiving hellos from each other form adjacencies.

---

**Default Hello Intervals and Hold Time for EIGRP**

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Example Link</th>
<th>Default Hello Interval</th>
<th>Default Hold Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.544 Mbps or less</td>
<td>Multipoint Frame Relay</td>
<td>60 seconds</td>
<td>180 seconds</td>
</tr>
<tr>
<td>Greater than 1.544 Mbps</td>
<td>T1, Ethernet</td>
<td>5 seconds</td>
<td>15 seconds</td>
</tr>
</tbody>
</table>
Neighbor Discovery - 3

Discussion

- Possibilities to avoid routing loops?