Network Layer: Routing
Distributed, Distance Vector Protocols

Y. Richard Yang

http://zoo.cs.yale.edu/classes/cs433/

4/11/2016
Exam 1 grading issue
- Please stop by Monday/Wednesday 7:30-8:30 pm office hours to fix any issues

Assignment 3 design share
- Wednesday at beginning of class

Assignment 4 issues

Projects
Recap: Primal-Dual Decomposition of Network-Wide Resource Allocation

- **SYSTEM(U):**
  \[
  \begin{align*}
  & \text{max } \sum_{f \in F} U_f(x_f) \\
  \text{subject to } & \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
  \text{over } & x \geq 0
  \end{align*}
  \]

- **USER}_f:**
  \[
  \begin{align*}
  & \text{max } U_f(x_f) - x_f p_f \\
  \text{over } & x_f \geq 0
  \end{align*}
  \]

- **NETWORK:**
  \[
  \min_{q \geq 0} \tilde{D}(q) = \sum_l q_l (c_l - \sum_{f: f \text{ uses } l} x_f)
  \]
Recap: Network Layer

- Transport packet from source to dest.
- Network layer in every host, router

Basic function:
- determine route taken by packets of a flow, and move the packets (potentially with QoS) along the route
Recap: Routing

- Typically routing can be formulated as a graph problem
- Routing has a large design space
  - who decides routing?
    - source routing
    - network routing
      - distributed vs (logically) centralized
  - what does routing compute
    - shortest path or QoS aware
  - how many paths does routing compute?
    - multi-path or single path routing
  - will routing adapt to network traffic demand?
    - adaptive or static routing
Routing Design Space: Internet

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
      - (applications such as overlay and p2p are trying to bypass it)
  - what does routing compute
    - shortest path (mostly, plus policy routing)
      - QoS aware
  - how many paths does routing compute?
    - multi-path routing
      - single path routing (with small amount of multipath)
  - will routing adapt to network traffic demand?
    - adaptive routing
      - static routing (mostly static; adjust in larger timescale)
  - ...
Alternative: Dynamic Link Costs

- 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:

\[ \text{EIGRP}_{\text{metric}} = 256 \times \left\{ \left[ 10^7 / \text{BW}_{\text{min}} \right] + \text{[sum_of_delays]} \right\} \]

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

EIGRP: Enhanced Interior Gateway Routing Protocol
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
  - = 256000
Roadmap: Routing Computation Architecture Spectrum

- Distance Vector
- Data Path
- Distributed Bellman-Ford
- Distributed Link State
- Logical Control
- Notification/Management/Control protocol
Outline

- Admin and recap
- Routing computation
  - Distributed distance vector protocols
Distance Vector Routing

- **Setting**: static (positive) costs assigned to network links
  - The static link costs may be adjusted in a longer time scale: this is called traffic engineering

- **Goal**: distributed computing to compute the shortest path from a source to a destination
  - Based on the Bellman-Ford algorithm (BFA)
  - Conceptually, runs for each destination separately

- **Realization**
  - Basis of RIP, IGRP, EIGRP routing protocols
Distance Vector Routing: Basic Idea

- 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 $j$
Outline

- Network overview
- Routing computation
  - Distance vector protocols
  - synchronous Bellman-Ford (SBF)
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 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
- Routing computation
  - Distance vector protocols
  - Synchronous Bellman-Ford (SBF)
    - SBF/∞
SBF/$\infty$

- Initialization (time 0):

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

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>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
<tr>
<td>$d(1)$</td>
<td>∞</td>
<td>∞</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>

Observation: $d(0) \geq d(1) \geq d(2) \geq d(3) \geq d(4) = d^*$
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 one scenario (\(d\)) than in another scenario (\(d'\)),

- 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.

\[
d_i(h+1) = \min_{j \in N(i)} (d_{ij} + d_j(h))
\]
Correctness of SBF/$\infty$

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., $L_i(h) = d_i(h)$, $L_i(h-1) = d_i(h-1)$, ...
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 \( d_i(h) \leq d_i(h-1) \)

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

\[
L_i(h + 1) = d_i(h + 1)
\]
We referred to the equations as Bellman equations (BE):

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

where \( d_D = 0 \).

- SBF/\( \infty \) 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, $d \geq d^*$

\[
d_i = \min_{j \in N(i)} (d_{ij} + d_j)
\]
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
- Routing computation
  - Distance vector protocols
    - synchronous Bellman-Ford (SBF)
      - SBF/∞
      - SBF/-1
Initialization (time 0):

\[ d_i(0) = \begin{cases} 
0 & \text{i = dest} \\
-1 & \text{otherwise}
\end{cases} \]
**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>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(5)</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>d(6)</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Observation: $d(0) \leq d(1) \leq d(2) \leq d(3) \leq d(4) \leq d(5) = d(6) = d^*$
Bellman Equation and Correctness of SBF/-1

- SBF/-1 converges due to monotonicity.
- At equilibrium, SBF/-1 satisfies the set of equations called Bellman equations (BE)

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

where \( d_D = 0 \).

- Another solution is shortest path solution \( d^* \).
- Since there is a unique solution to the BE equations; thus SBF/-1 converges to shortest path.

Question: will SBF converge under other non-negative initial conditions?

Problems of running synchronous BF?
Outline

- Network overview
- Routing computation
  - 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 $d_{ij}$ from neighbor $j$.

- Asynchronously node $j$ sends its estimate to its neighbor $i$:
  - there is an upper bound on the delay of estimate packet (no worry for out of order)
Asynchronous Bellman-Ford (ABF)

- In general, nodes are using different and possibly inconsistent estimates.
Distance Table: Example

Below is just one step! The protocol repeats forever!

d_{E} ()  |  A  |  B  |  C  |  D  |  computation | E’s distance table | distance table E sends to its neighbors
--|--|--|--|--|---|---|---
A  | 0  | 7  | ∞  |    | 10 | 15 | ∞  | A: 10 | A: 10
B  | 7  | 0  | ∞  |    | 17 | 8  | ∞  | B: 8  | B: 8
C  | ∞  | 1  | 2  |    | ∞  | 9  | 4  | D: 4  | C: 4
D  | ∞  | ∞  | 0  |    | ∞  | ∞  | 2  | D: 2  | D: 2

10 8 2
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
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 $L()$
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 \( j \)
  - \( L_j(0) \leq d^i_j \leq U_j(0) \)
  - \( L_j(0) \leq \text{update messages } d^i_j \leq U_j(0) \)
ABF Convergence

- $d_j$
  - 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
Tool box: a key technique for proving convergence (liveness) of distributed protocols: monotonicity

- Consider two configurations d(t) and d'(t):
  - if d(t) \leq d'(t), then d(t+1) \leq d'(t+1)

- Identify two extreme configurations to sandwich any real configurations
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>1</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>
| 1 | 2  | 3  | 4  | ∞  | Initially
|    |    |    |    |    | After 4 exchanges
|    |    |    |    |    | After 3 exchanges
|    |    |    |    |    | After 2 exchanges
|    |    |    |    |    | After 1 exchange
|    |    |    |    |    | Initially |

Initially
Properties of Distance-Vector Algorithms

- Bad news propagate slowly

<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>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td>Initially</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td>After 1 exchange</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
<td>After 2 exchanges</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td></td>
<td>After 3 exchanges</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td></td>
<td>After 4 exchanges</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td></td>
<td>After 5 exchanges</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td></td>
<td>After 6 exchanges</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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?
The Reverse-Poison (Split-horizon) Hack

If the path to dest is through neighbor h, report $\infty$ to neighbor h for dest.

<table>
<thead>
<tr>
<th></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>D</td>
<td>A</td>
<td>B</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>0</td>
<td>7</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>7</td>
<td>0</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>$\infty$</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>1, A</td>
</tr>
</tbody>
</table>

Note: The distance table E sends to its neighbors is shown in the rightmost column with distances indicated as $\infty$ for paths that do not pass through neighbor h.
Implementation: 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)

<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 I
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?)
An Example Where Split-horizon Fails

- When the link between C and D fails, C will set its distance to D as $\infty$.
- However, unfortunate timing can cause a problem:
  - A receives the bad news ($\infty$) from C, A will use B to go to D.
  - A sends the news to C.
  - C sends the news to B.