Network Layer:

intro;

Distance Vector Protocols

Y. Richard Yang

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

11/27/2018
Outline

- Admin and recap
- Network overview
- Network control-plane
  - Routing
Assignment four meeting

- Today:
  - 2:30-3:30 pm
  - 5:00-6:30 pm
- Wednesday

Exam 2 date?
Recap: BW Allocation Framework

- **Forward engineering**: systematically design of
  - objective function
  - distributed alg to achieve objective

- **Science/reverse engineering**: what do TCP/Reno, TCP/Vegas achieve?

<table>
<thead>
<tr>
<th>Objective</th>
<th>Allocation (x1, x2, x3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/Reno</td>
<td>0.26</td>
</tr>
<tr>
<td>TCP/Vegas</td>
<td>1/3</td>
</tr>
<tr>
<td>Max throughput</td>
<td>0</td>
</tr>
<tr>
<td>Max-min</td>
<td>$\frac{1}{2}$</td>
</tr>
<tr>
<td>Max sum log(x)</td>
<td>1/3</td>
</tr>
<tr>
<td>Max sum of $-1/(RTT^2 \times)$</td>
<td>0.26</td>
</tr>
</tbody>
</table>

$$\begin{align*}
\max & \sum_{f \in F} U_f(x_f) \\
\text{subject to } & \sum_{f : f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
& x \geq 0
\end{align*}$$
Recap: Systematic Derivation of Objective Function

- **NBS axioms**
  - Pareto optimality
  - Symmetry
  - Invariance of linear transformation
  - Independence of irrelevant alternatives

- **NBS solution**
  - The rate allocation point is the feasible point which maximizes
    \[ x_1 x_2 \cdots x_F \]
Recap: Systematic Derivation of Alg: Foundation (Strong Dual Theorem)

\[
\begin{align*}
\text{max} & \quad f(x) \\
\text{subject to} & \quad g(x) \leq 0 \\
\text{over} & \quad x \in S
\end{align*}
\]

- \( f(x) \) concave
- \( g(x) \) linear
- \( S \) is a convex set

\[
D(q) = \max_{x \in S} \left( f(x) - qg(x) \right)
\]

- \( -D(q) \) is called the dual;
- \( q \ (\geq 0) \) are called prices in economics
Recap: Primal-Dual Decomposition of Network-Wide Resource Allocation

**SYSTEM(U):**

\[
\max \sum_{f \in F} U_f(x_f) \\
\text{subject to } \sum_{f: f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
\text{over } x \geq 0
\]

**USER\_f:**

\[
\max_{x_f} U_f(x_f) - x_f p_f \\
\text{over } x_f \geq 0
\]

**NETWORK:**

\[
\min_{q \geq 0} \tilde{D}(q) = \sum_l q_l (c_l - \sum_{f: f \text{ uses } l} x_f)
\]
TCP/Reno Dynamics

\[ \Delta x_f \propto U'_f(x_f) - p_f \]

\[ \Delta x = \frac{RTT}{2} x^2 \left( \frac{2}{x^2 RTT^2} - p \right) \]

\[ U'_f(x_f) - p_f \]

\[ \Rightarrow U'_f(x_f) = \left( \frac{\sqrt{2}}{x_f RTT} \right)^2 \]

\[ \Rightarrow U_f(x_f) = -\frac{2}{RTT^2 x_f} \]
TCP/Vegas Dynamics

\[ \Delta x_f \propto U'_f(x_f) - p_f \]

\[ \Delta x = \frac{x}{RTT} \left( \frac{\alpha}{x} - (RTT - RTT_{\text{min}}) \right) \]

\[ U'_f(x_f) - p_f \]

\[ U'_f(x_f) = \frac{\alpha}{x} \quad \Rightarrow \quad U_f(x_f) = \alpha \log(x_f) \]
Outline

Admin and recap

- Network overview
Network Layer

- Transport packets from source to destination
- Network layer in every host, router

Basic functions:
- inter-networking (e.g., fragmentation/assembly)
- routing (determine route(s) taken by packets of a flow), and forwarding (move the packets along the route(s))
Current Internet Network Layer

Network layer functions:

Transport layer

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

Control protocols
• error reporting
  • e.g. ICMP

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

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

Forwarding

Link layer

physical layer
Routing: Overview

**Routing**

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

Graph abstraction for the routing problem:
- graph nodes are routers
- graph edges are physical links
  - links have properties: delay, capacity, $ cost
- compute path on graph
Network Layer: Complexity
Factors/Objectives

- For network providers
  - efficiency of routes
  - policy control on routes
  - scalability

- For users
  - quality of services, e.g.,
    - guaranteed bandwidth?
    - preservation of inter-packet timing (no jitter)?
    - loss-free delivery?
    - in-order delivery?

- Users and network may interact
Routing Design Space

- 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
  - what does routing compute?
    - network cost minimization
    - QoS aware
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing
  - ...

- Robustness
- Optimality
- Simplicity
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?
    - network cost minimization (shortest path)
      - QoS aware
  - how many paths from source s to destination d?
    - 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)
  - ...
Basic Formulation

- Assign link weights
- Compute shortest path
Example: Cisco Proprietary Recommendation on Assigning Link Costs

- **Link metric:**
  
  \[
  \text{metric} = \left[ K1 \times \text{bandwidth}^{-1} + (K2 \times \text{bandwidth}^{-1}) / (256 - \text{load}) + K3 \times \text{delay} \right] \times \left[ K5 / (\text{reliability} + K4) \right]
  \]

  By default, \( k1=k3=1 \) and \( k2=k4=k5=0 \). 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
The bandwidth and delay for an Ethernet interface are 10 Mbps and 1 ms, respectively.

The calculated EIGRP metric is as follows:

- \(256 \times \left[\frac{10^7}{BW\text{ks}} + \text{delay in 10us}\right]\)
- \(= 256 \times \left[\frac{10^7}{10,000} + 100\right]\)
- \(= 256 \times [1000 + 100]\)
- \(= 256,000 + 25,600\)
- \(= 281,600\)
Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Distributed routing computation
Why Study?

- Just as Dijkstra’s Shortest Path algorithm is among the most classical algorithms in algorithm design, distributed shortest path protocols provide many insights in distributed protocol design.

- Please learn not only the protocols, but also the techniques (convergence, global invariants, ...)

Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
Basic Routing Computation Setting

- 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
  - Conceptually, runs for each destination separately
Intuition

\[ d_i \leq d_j + d_{ij}, \text{ for each neighbor } j \]

\[ d_i = \min_{j \in N(i)} (d_{ij} + d_j) \]
Understanding Shortest Path and an Exercise of Primal-Dual

\[
\max d_s - d_D
\]

for any edge \( i \rightarrow j \): \( d_i \leq dj + d_{ij} \)

\( d_i \geq 0 \)

Dual: \( D(x) = \max (d_s - dD - \sum x_{ij}(d_i - d_j - d_{ij})) \)

\( = \sum x_{ij}d_{ij} \)

\( x_{ij} \) is a flow from \( s \) to \( D \)
Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distributed distance vector protocols
Distance Vector Routing: Basic Idea

- Based on Bellman-Ford equation: 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

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - distributed 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

- Admin and recap
- Network overview
- Network control-plane path
  - Routing
    - Link weights assignment
    - Routing computation
      - distributed distance vector protocols
      - synchronous Bellman-Ford (SBF)
      - SBF/$\infty$
SBF/∞

Initialization (time 0):

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

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

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

- 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** $d_i(h) \leq d_i(h-1)$

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

\[
L_i(h+1) = d_i(h+1)
\]
Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distributed distance vector protocols
        - synchronous Bellman-Ford (SBF)
          - SBF/$\infty$
          - SBF/-1
SBF at another
Initial Configuration: 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>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: \(d(0) \leq d(1) \leq d(2) \leq d(3) \leq d(4) \leq d(5) = d(6) = d^*\)
Correctness of SBF/-1

- SBF/-1 converges due to monotonicity

Remaining question:
  - Can we guarantee that SBF/-1 converges to shortest path?
Correctness of SBF/-1

- Common between SBF/$\infty$ and SBF/-1: they solve the Bellman equation
  \[ d_i = \min_{j \in N(i)} (d_{ij} + d_j) \]
  where $d_D = 0$.

- We have proven SBF/$\infty$ is the shortest path solution.
- SBF/-1 computes shortest path if Bellman equation has a unique solution.
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 $d \leq d^*$
**Summary: “Extreme” SBF Initial States**

<table>
<thead>
<tr>
<th>$d_i(0)$</th>
<th>$0$ if $i = \text{dest}$</th>
<th>$\infty$ otherwise</th>
</tr>
</thead>
</table>

| $d_i(0)$ | $0$ if $i = \text{dest}$ | $-1$ otherwise |

- **Nice properties of both cases**
  - Monotonicity
  - Convergence
Discussion

- Will SBF converge under any non-negative initial conditions?
Outline

- Admin and recap
- Network overview
- Network control plane
  - Routing
    - Link weights assignment
  - Routing computation
    - Distributed 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 $i$:
  - We assume that there is an upper bound on the delay of estimate packet
ABF: Example

Below is just one step! The protocol repeats forever!

<table>
<thead>
<tr>
<th>destinations</th>
<th>distance tables from neighbors</th>
<th>computation</th>
<th>E’s routing table</th>
<th>distance table E sends to its neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A: 0 7 ∞</td>
<td>10 15 ∞</td>
<td>A: 10</td>
<td>A: 10</td>
</tr>
<tr>
<td>B</td>
<td>B: 7 0 ∞</td>
<td>17 8 ∞</td>
<td>B: 8</td>
<td>B: 8</td>
</tr>
<tr>
<td>C</td>
<td>C: ∞ 1 2</td>
<td>∞ 9 4</td>
<td>D: 4</td>
<td>C: 4</td>
</tr>
<tr>
<td>D</td>
<td>D: ∞ ∞ 0</td>
<td>∞ ∞ 2</td>
<td>D: 2</td>
<td>D: 2</td>
</tr>
</tbody>
</table>

10 8 2

next hop

distance

E: 0
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 $t$ and connected, ABF will eventually converge to the shortest path!
What is system state?
three types of distance state from node j:

- $d_j$: current distance estimate state at node j
- $d_{ij}$: last $d_j$ that neighbor $i$ received
- $d_{ij}^i$: those $d_j$ that are still in transit to neighbor $i$
ABF Convergence Proof: The Sandwich Technique

**Basic idea:**
- bound system state using extreme states

**Extreme states:**
- $SBF/\infty$; call the sequence $U()$
- $SBF/-1$; call the sequence $L()$
Consider the time when the topology is stabilized at time 0.

$U(0)$ and $L(0)$ provide upper and lower bounds 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 $j$
- $L_j(0) \leq d_{ij} \leq U_j(0)$
- $L_j(0) \leq \text{update messages } d_{ij} \leq U_j(0)$
ABF Convergence

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

- $d^i_j$
  - eventually all $d^i_j$ that are only bounded by $L_j(0)$ and $U_j(0)$ are replaced with in $L_j(1)$ and $U_j(1)$
Distributed, Asynchronous, Routing Protocol: Summary of Features

- **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
Distributed, Asynchronous, Routing Protocol: Summary of Analytical Technique

Tool box: a key technique for analyzing convergence (liveness) of distributed protocols: monotonicity and the bounding-box (sandwich) theorem

- 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
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
  - Routing computation
    - Distance vector protocols (distributed computing)
      - synchronous Bellman-Ford (SBF)
      - asynchronous Bellman-Ford (ABF)
      - properties of DV
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></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Initially</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After 1 exchange</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After 2 exchanges</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After 3 exchanges</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>After 4 exchanges</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Properties of Distance-Vector Algorithms

- Bad news propagate slowly

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

Q: what causes counting-to-infinity?
Counting-To-Infinity is Because of Routing Loop

- Counting-to-infinity is caused by a routing loop, which 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 next hop, B thinks C as next hop, ... E thinks A as next hop.
Discussion

- Why avoid routing loops is hard?

- Any proposals to avoid distributed routing loops?
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
  - Routing computation
    - Distance vector protocols (distributed computing)
      - synchronous Bellman-Ford (SBF)
      - asynchronous Bellman-Ford (ABF)
    - properties of DV
    - distributed protocols w/ safety (loop prevention)
      - reverse poison/split horizon
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>E's distance table</th>
<th>distance table E sends to its neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>To A</td>
<td>To B</td>
</tr>
<tr>
<td>A: $\infty$</td>
<td>B: $\infty$</td>
</tr>
<tr>
<td>B: 8</td>
<td>C: $\infty$</td>
</tr>
<tr>
<td>C: 4</td>
<td>D: $\infty$</td>
</tr>
<tr>
<td>D: $\infty$</td>
<td>E: 0</td>
</tr>
<tr>
<td>A: 1, A</td>
<td>B: 8, B</td>
</tr>
<tr>
<td>B: 8, B</td>
<td>C: 4, D</td>
</tr>
<tr>
<td>C: 4, D</td>
<td>D: 2, E</td>
</tr>
<tr>
<td>D: 2, E</td>
<td>E: 0</td>
</tr>
</tbody>
</table>

Distance through neighbor

c(E,A) c(E,B) c(E,D)

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

<table>
<thead>
<tr>
<th><em>A</em></th>
<th><em>B</em></th>
<th><em>D</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>A</em></td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td><em>B</em></td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td><em>C</em></td>
<td>$\infty$</td>
<td>9</td>
</tr>
<tr>
<td><em>D</em></td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><em>E</em></th>
<th><em>A</em></th>
<th><em>B</em></th>
<th><em>D</em></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>A</em></td>
<td>1, A</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>B</em></td>
<td>8, B</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>C</em></td>
<td>4, D</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>D</em></td>
<td>2, D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Reverse-Poison Example**

Exercise: Can Reverse-poison guarantee no loop for this network?
**DV+RP \(\rightarrow\) RIP**

(Routing Information Protocol)

- 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: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead
  o routes via neighbor invalidated
  o new advertisements sent to neighbors
  o neighbors in turn send out new advertisements (if tables changed)
  o link failure info quickly propagates to entire net
  o reverse-poison used to prevent ping-pong loops
  o set infinite distance = 16 hops (why?)
General Routing Loops and Reverse-poison

Exercise: Can Reverse-poison guarantee no loop for this network?
General Routing Loops and Reverse-poison

- Reverse-poison removes two-node loops but may not remove more-node loops

Unfortunate timing can lead to a loop
- When the link between C and D fails, C will set its distance to D as $\infty$
- 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
Backup Slides
Routing Design Space: User-based, Multipath, Adaptive

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
- what does routing compute?
  - network cost minimization
    - QoS aware
- will routing adapt to network traffic demand?
  - adaptive routing
    - static routing
- ...
User Optimal, Multipath, Adaptive

- User optimal: users pick the shortest routes (selfish routing)

![Diagram showing Braess's paradox](image)

Braess's paradox
Price of Anarchy

For a network with linear latency functions

\[ \rightarrow \]

- total latency of user (selfish) routing for given traffic demand

\[ \leq \frac{4}{3} \]

- total latency of network optimal routing for the traffic demand
Price of Anarchy

For any network with continuous, non-decreasing latency functions \( \Rightarrow \)

\[ \text{total latency of user (selfish) routing for given traffic demand} \]

\[ \leq \]

\[ \text{total latency of network optimal routing for twice traffic demand} \]
Assigning Link Weight: Dynamic Link Costs

- Assign link costs to 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.