Network Layer:
Distance Vector Protocols with Safety; 
Link-State Protocol; 
Global/Internet-Scale Routing

Y. Richard Yang

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

11/29/2018
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
      - Link state protocols (distributed state synchronization)
      - Internet-scale routing
Assignment four meeting
  - Today 2:30 – 4:00; 5:30-7:00 pm
  - Friday: 11:30-12:30 pm; 4:00-5:00 pm

Assignment five posted

Reminder: Exam 2: 7-8:30 pm Tuesday Dec. 11
Recap: 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 (shortest path routing)
  - QoS aware

- will routing adapt to network traffic demand?
  - adaptive routing
  - static routing
Recap: Distributed Distance Vector Routing Protocols

- Basic Idea: Bellman-Ford (BF) update rule

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

- Synchronous BF
  - Two extreme states
    - Monotonicity
    - Convergence

- Asynchronous BF

- Issues of BF
  - counting-to-infinity in settings such as disconnection
Recap: Counting-To-Infinity

- 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).

- Initial solution:
  - Reverse-poison
  - Solution integrated into RIP, the first major routing protocol
Any proposal that can enforce global property (i.e., no loop) using local condition (i.e., decision made by each node locally)?
The No Increase Cost (NIC) Condition

- Claim: If a DV protocol enforces that the cost to a destination cannot increase, there will not be any routing loops
Claim: NIC will NEVER Form a Loop

- Initially no loop (no one has next hop so no loop)
- Derive contradiction if a loop forms after a node processes an update,
  - e.g., when A receives the update from B, A decides to use B as next hop and forms a loop
Analysis Technique: Global Invariants

- **Global Invariant** is a very effective method in understanding distributed asynchronous protocols.
- Invariants are defined over the states of the distributed nodes.

Consider any node $B$ in a DV protocol for a destination $D$.
- What is the state of $B$?
Invariant at a Single Node B

- $[I1] d^B$ is non-increasing
Invariants when A Considers B as Next Hop

- Invariant if A considers B as next hop
  - [I2] $d^B < d^A$
    - because $d^A$ is based on $d^B$ which B sent to A some time ago, $d^B < d^A$ since all link costs are positive;
    - $d^B$ might be decreased after B sent its state (I1)
Loop Freedom of NIC

- Consider a critical moment
  - A starts to consider B as next hop, and we have a loop

- According to invariant I2 for each link in the loop (X considers Y as next hop)
  - $d^Y < d^X$ for each link

- By transition along the loop $d^B > d^B$

Problem of enforcing NIC: achieves safety but stops the protocol when cost is increased?
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 safety and Routing Information Protocol (RIP)
        - No increase cost safety and the Destination-Sequenced DV Protocol (DSDV)
Destination-Sequenced Distance Vector Protocol (DSDV)

- Basic idea: to enforce NIC but still allow link cost increase (e.g., when link broken), use sequence numbers to partition computation
  - tags each route with a sequence number: each node $B$ maintains $(S_B^B, d_B^B)$ for each destination $D$, where $S_B^B$ is the sequence number at $B$ for destination $D$ and $d_B^B$ is the best distance using a neighbor from $B$ to $D$

- External triggered messages
  - Timer-triggered: periodically each destination $D$ increases its seq. by 2 and broadcasts with $(S_D^D, 0)$
  - Link-event triggered: if $B$ is using $C$ as next hop to $D$ and $B$ discovers that $C$ is no longer reachable
    - $B$ increases its sequence number $S_B^B$ by 1, sets $d_B^B$ to infinite $\infty$, and sends $(S_B^B, d_B^B)$ to all neighbors
Example

Will this trigger an update?
Example

Will this trigger an update?
DSDV: Msg-Triggered Update

- Update after receiving a message
  - assume B sends to A its current state \((S^B, d^B)\)
  - when A receives \((S^B, d^B)\)
    - if \(S^B > S^A\), then
      // always update if a higher seq#
      \(S^A = S^B\)
      \(d^A = \infty\) if \(d^B = \infty\); else \(d^A = d^B + d(A,B)\)
    - else if \(S^A == S^B\), then
      \(d^A > d^B + d(A,B)\)
      // update for the same seq# only if better route
      \(d^A = d^B + d(A,B)\) and uses B as next hop
Exercise: update process after D increases its seq# to next even number.
Global Invariants of DSDV: Single Node B

- Some invariants about the state of a node B
  - [I1] $S^B$ is non-decreasing
  - [I2] $d^B$ is non-increasing for the same sequence number
Global Invariants: if A Considers B as Next Hop

- Some invariants if A considers B as next hop
  - [I3] $d^A$ is not $\infty$
  - [I4] $S^B \geq S^A$
    - because A is having the seq# which B last sent to A; B's seq# might be increased after B sent its state
  - [I5] if $S^B = S^A$
    - then $d^B < d^A$ because $d^A$ is based on $d^B$ which B sent to A some time ago, $d^B < d^A$ since all link costs are positive; $d^B$ might be decreased after B sent its state
Loop Freedom of DSDV

- Consider a critical moment
  - A starts to consider B as next hop, and we have a loop
- According to invariant I4 for each link in the loop (X considers Y as next hop): \( S_Y \geq S_X \)
- Two cases:
  - exists \( S_Y > S_X \)
    - by transition along the loop \( S^B > S^B \)
  - all nodes along the loop have the same sequence number
    - apply I5, by transition along the loop \( d^B > d^B \)
When the BC link fails, B loses its path to D, but A actually has C as an alternative (backup) path.

Is there a local condition for A to realize that C is usable?
Issues of DSDV

- DSDV guarantees no loop, but need global recomputation after each cost increase (not using any backup path).
- DSDV does not include a mechanism to use multiple paths (backup, load balancing)
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 safety and Routing Information Protocol (RIP)
        - No increase cost safety and the Destination-Sequenced DV Protocol (DSDV)
        - Feasible Distance safety and Diffusive Update algorithm (DUAL) and EIGRP
Key Idea: Feasible Successors [EIGRP]

- If the **reported distance** of a neighbor n is lower than the lowest ever **total distance** to reach destination D (i.e., neighbor n is closer than this node has ever been), the neighbor n is a feasible successor.

\[ d_n + d_{i \rightarrow n} \geq d_{\text{primary}} + d_{i \rightarrow \text{primary}} > d_n \]
Example

Assume A is destination, consider E

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>Reported Distance</th>
<th>Total Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbor C</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Neighbor D</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Q: Which one is primary path?

Q: If link to primary fail, can E use the back up?
Offline Exercise

- Identify the invariants and prove the safety property of the feasible distance condition
What if a node has no feasible successor?

- Hint: a diffusive distributed computing protocol to compute the next route
  - See Dijkstra-Scholten algorithm
Summary: Distance Vector Routing

- **Basic distance vector**
  - take away: use monotonicity as a technique to understand liveness/convergence
    - highly recommended reading of Bersekas/Gallager chapter; see Schedule page

- **Fix counting-to-infinity problem**
  - Take away: use local condition to enforce global properties; use global invariants to understand/design safety/no routing loops
  - Diffusive computing model can be a powerful tool to design distributed protocols [not fully covered, but highly recommended reading; see schedule page]
Discussion: Distance Vector Routing

- What do you like about distributed, distance vector routing?

- What do you **not** like about distributed, distance vector routing?
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Distance vector protocols (distributed computing)
      - Link state protocols (distributed state synchronization)
Link-State Routing

- Basic idea: Not distributed computing, only distributed state distribution
- Net topology, link costs are distributed to all nodes
  - All nodes have same info
  - Each node computes shortest paths from itself to all other nodes
    - Standard Dijkstra’s algorithm as path compute algorithm
    - Allows multiple same-cost paths
    - Multiple cost metrics per link (for type of service routing)

- Often used by large networks (OSPF by large enterprises; ISIS by service providers)
Example: Link State and Directed Graph (OSPFv2)

Multi-access networks

Stub multi-access networks
Example: Link State and Directed Graph (OSPFv2)

Figure 2: A sample Autonomous System

```
**FROM**

<table>
<thead>
<tr>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>RT</th>
<th>N3</th>
<th>N6</th>
<th>N8</th>
<th>N9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>RT1</td>
<td>RT2</td>
<td>RT3</td>
<td>RT4</td>
<td>RT5</td>
<td>RT6</td>
<td>RT7</td>
<td>RT8</td>
<td>RT9</td>
<td>RT10</td>
<td>RT11</td>
<td>RT12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>RT10</td>
<td></td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>RT11</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RT12</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```

Figure 3: The resulting directed graph
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
  - Routing computation
    - Distance vector protocols (distributed computing)
    - Link state protocols (distributed state synchronization)
      - data structure to be distributed
        - state distribution protocol
Basic event structure at node n

- periodically:
  - broadcast at each interface HELLO msg so that neighbor can discover me

- on discovering new status of a directly connected link e (receive HELLO from a new neighbor or a neighbor misses N HELLO msgs)
  - broadcast new status of LSA[e]

- on receiving an LSA[e]:
  - if (does not have LSA[e])
  - forwards LSA[e] to all links except the incoming link
Link State Broadcast

Node S updates link states connected to it.
To avoid forwarding the same link state announcement (LSA) multiple times (forming a loop), each node remembers the received LSAs.  
- Second LSA[S] received by E from C is discarded  
- Second LSA[S] received by C from E is discarded as well  
- Node H receives LSA[S] from two neighbors, and will discard one of them
Discussion

Issues of the basic link state protocol?
  - Recall: goal is to efficiently distribute to each node to a correct, complete link state map
Link State Broadcast: Issues

- Correctness problem: network partition and then reconnect, basic protocol propagates only the single reconnection link, how to sync across the reconnected components?

- Solution: updates are sent periodically
Link State Broadcast: Issues

- Problem: Broadcast redundancy

https://hal.inria.fr/inria-00072756/document
Summary: Link State

- Basic LS protocol
  - take away: instead of computing routing results using distributed computing, distributed computing is for only link state distribution (synchronization)

- Link state distribution can still have much complexity, e.g., partition and reconnect, scalability
Outline

- Admin and recap
- Network control plane
  - Routing
    - Link weights assignment
    - Routing computation
      - Basic routing computation protocols
        - Distance vector protocols (distributed computing)
        - Link state protocols (distributed state synchronization)
  - Global Internet routing
Discussion

- Does it work to use DV or LS as we discussed for global Internet routing?
Requirements and Solution of Current Global Internet Routing

- **Scalability**: handle network size (#devices) much higher than typical DV or LS can handle
  - Solution: Introduce new abstraction (hierarchy) to reduce network (graph) size

- **Autonomy**: allow each network to have individual preference of routing (full control of its internal routing; control/preference of routing spanning multiple networks)
  - Solution: autonomous, policy routing
New Abstraction: Autonomous Systems (AS)

- Abstract each network as an autonomous system (AS), identified by an AS number (ASN)

- Conceptually the global routing graph consists of only autonomous systems as nodes

Exercise: https://www.ultratools.com/tools/asnInfo
Global Routing: Bigger Picture

- Global Internet routing is divided into inter-AS routing and intra-AS routing
  - Inter-AS routing (also called interdomain routing)
    - A protocol runs among autonomous systems is also called an Exterior Gateway Protocol (EGP)
    - The de facto EGP protocol is BGP
  - Intra-AS routing (also called intradomain routing)
    - A protocol running inside an AS is called an Interior Gateway Protocol (IGP), each AS can choose its own protocol, such as RIP, E/IGRP, OSPF, IS-IS
BGP is a Path Vector protocol
- similar to Distance Vector protocol
- a border gateway sends to a neighbor entire path (i.e., a sequence of ASNs) to a destination, e.g.,
  - gateway X sends to neighbor N its path to dest. Z:
    \[ \text{path (X,Z)} = X,Y_1,Y_2,Y_3,\ldots,Z \]
- if N selects path(X, Z) advertised by X, then:
  \[ \text{path (N,Z)} = N, \text{path (X,Z)} \]
Exercise: Observing BGP Paths

- Use one of the looking glass servers (http://www.bgp4.as/looking-glasses)
  - List of destinations announced by an autonomous system:
    - http://irrexplorer.nlnog.net/search/29
  - Click on AS number of each address to see all paths announced
    - https://lg.de-cix.net/alice/search
**Bigger Picture: Integration of Intra- and Inter-Domain Routing**

**AS C** (RIP intra routing)  
Gateway routers of diff. auto. systems exchange routes using eBGP.

**AS A** (OSPF intra routing)  
Gateway routers of same AS share learned external routes using iBGP.

**AS B** (OSPF intra routing)  
Gateway routers participate in intradomain to learn internal routes.

A potential view: Interdomain routers as an overlay.
Problem: How to choose a route if multiple choices?
BGP Policy Routing Framework:
Decision Components

- Routing cache
- Select best path
- Export path to neighbors

Route selection policy: rank paths
Export policy: which paths export to which neighbors

Networks:
- AT&T
- Qwest
- Internet2
- Yale
BGP Example (1)

Route selection policy:
- Shortest AS Path policy:
- Choose AD using a1

Export policy controls ingress, i.e., who can use I

No export to F (effect?)

Export to E: i->e: I can reach hosts in D; path: IAD

a1->i: I can reach hosts in D; path: AD

a2->a1: I can reach hosts in D; path: D

b->i: I can reach hosts in D; path: BCD

b->i2: I can reach hosts in D; path: BCD

Export to E: i->e: I can reach hosts in D; path: IAD

AS A (OSPF)

AS B (OSPF intra routing)

AS D

AS C

AS I
BGP Example (2)

Selection policy:
- Low local_pref for A
- Shortest AS Path
- Prefer eBGP

Called hot potato (why?)

Outcome: Choose BCD using b

Export to E: i->e: I can reach hosts in D; path: IBCD

i->a1: I can reach hosts in D; my path: AD

a1->i: I can reach hosts in D; path: IBCD

b->i: I can reach hosts in D; my path: BCD

b->i2: I can reach hosts in D; my path: BCD

Export to E: i->e: I can reach hosts in D; path: IBCD

b->a1: I can reach hosts in D; path: D

a2->a1: I can reach hosts in D; path: D

a2->a1: I can reach hosts in D; path: D

Outcome: Choose BCD using b

Called hot potato (why?)
BGP Example (3)

Selection policy:
- Low local pref A
- Shortest AS Path
- Prefer iBGP

Called cold potato (why?)

Outcome: Choose BCD using i2
Routing: Remaining Issue

Naïve design is to send individual host D’s IP. Can we do better?
IP Addressing Scheme: Requirements

- **Uniqueness:** We need an address to uniquely identify each destination.

- **Aggregability:** Routing scalability needs flexibility in aggregation of destination addresses.
  - We want to aggregate as a large set of destinations as possible in BGP announcements.

- **Current:** The unit of routing in the Internet is a classless interdomain routing (CIDR) address.
Classless InterDomain Routing (CIDR) Address: Aggregation

- A CIDR address partitions an IP address into two parts
  - A prefix representing the network portion, and the rest (host part)
  - Address format: \textit{a.b.c.d/x}, where \(x\) is \# bits in network portion of address

\begin{center}
\begin{tikzpicture}[scale=0.8]
\node (network) at (0,0) {network part};
\node (host) at (3,0) {host part};
\draw[->,ultra thick] (network) -- (host);
\end{tikzpicture}
\end{center}

\texttt{11001000 00010111 00010000 00000000}

\texttt{200.23.16.0/23}

Some systems use mask (1’s to indicate network bits), instead of the /x format
CIDR Aggregation in BGP

i->a1: I can reach 130.132.0/22; my path: I

130.132.0.0/24
130.132.1.0/24

130.132.2.0/24

130.132.3.0/24

Intradomain routing uses /24
CIDR Aggregation in BGP

Problem at S: Overlapping routing entries.
Solution: Longest prefix matching (LPM)