15-441 Computer Networking

Intra-Domain Routing, Part II

OSPF (Open Shortest Path First)

A Link-State Routing Algorithm

Dijkstra's algorithm
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives routing table for that node
- iterative: after k iterations, know least cost path to k dest.'s

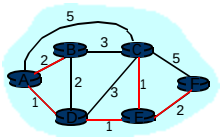
Notation:
- c(i,j): link cost from node i to j. cost infinite if not direct neighbors
- D(v): current value of cost of path from source to dest. V
- p(v): predecessor node along path from source to v, that is next v
- N: set of nodes whose least cost path definitively known

Dijsktra's Algorithm

```
1  Initialization:
2    N = {A}
3    for all nodes v
4      if v adjacent to A
5        then D(v) = c(A,v)
6        else D(v) = infinity
7
8  Loop
9    find w not in N such that D(w) is a minimum
10   add w to N
11   update D(v) for all v adjacent to w and not in N:
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N
```

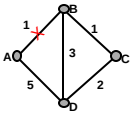

Dijkstra's algorithm: example

Step	start N	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	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



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

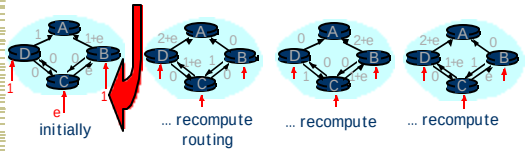
* Lump Sum Death Benefit?
Leisure Studies Data Bank?
Latvijas spriedumu datu bâze?
>> Link State Data Base

Dijkstra's algorithm, discussion

Algorithm complexity: n nodes
- each iteration: need to check all nodes, w, not in N
- n*(n+1)/2 comparisons: O(n**2)
- more efficient implementations possible: O(n log n) (or better)

Oscillations possible:
- e.g., link cost = amount of carried traffic

amount of flow destined for node A from external source



Importance of Cost Metric

- Choice of link cost defines traffic load
  - Low cost = high probability link belongs to SPT and will attract traffic, which increases cost
- Main problem: convergence
  - Avoid oscillations
  - Achieve good network utilization

Metric Choices

- Static metrics (e.g., hop count)
  - Good only if links are homogeneous
  - Definitely not the case in the Internet
- Static metrics do not take into account
  - Link delay
  - Link capacity
  - Link load (hard to measure)

Original ARPANET Metric

- Cost proportional to queue size
  - Instantaneous queue length as delay estimator
- Problems
  - Did not take into account link speed
  - Poor indicator of expected delay due to rapid fluctuations
  - Delay may be longer even if queue size is small due to contention for other resources

Metric 2 - Delay Shortest Path Tree

- Delay = (depart time - arrival time) + transmission time + link propagation delay
  - (Depart time - arrival time) captures queuing
  - Transmission time captures link capacity
  - Link propagation delay captures the physical length of the link
- Measurements averaged over 10 seconds
  - Update sent if difference > threshold, or every 50 seconds

Performance of Metric 2

- Works well for light to moderate load
  - Static values dominate
- Oscillates under heavy load
  - Queuing dominates
- Reason: there is no correlation between original and new values of delay after re-routing!

Specific Problems

- Range is too wide
  - 9.6 Kbps highly loaded link can appear 127 times costlier than 56 Kbps lightly loaded link
  - Can make a 127-hop path look better than 1-hop
- No limit to change between reports
- All nodes calculate routes simultaneously
  - Triggered by link update

Consequences

- Low network utilization (50% in example)
- Congestion can spread elsewhere
- Routes could oscillate between short and long paths
- Large swings lead to frequent route updates
  - More messages
  - Frequent SPT re-calculation

Revised Link Metric

- Better metric: packet delay = f(queueing, transmission, propagation)
- When lightly loaded, transmission and propagation are good predictors
- When heavily loaded queueing delay is dominant and so transmission and propagation are bad predictors

Normalized Metric

- If a loaded link looks very bad then everyone will move off of it
- Want some to stay on to load balance and avoid oscillations
- It is still an OK path for some
- Hop normalized metric diverts routes that have an alternate that is not too much longer
- Also limited relative values and range of values advertised → gradual change

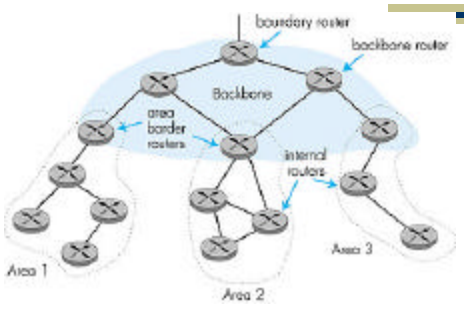
OSPF (Open Shortest Path First)

- "open": publicly available
- Uses Link State algorithm
  - LS packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor router
- Advertisements disseminated to entire AS (via flooding)

OSPF "advanced" features (not in RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion); TCP connections used
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort; high for real time)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

- Two-level hierarchy: local area, backbone.
  - Link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- **Area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.
- **Backbone routers:** run OSPF; routing limited to backbone.
- **Boundary routers:** connect to other ASs.

IGRP (Interior Gateway Routing Protocol)

- CISCO proprietary; successor of RIP (mid 80s)
- Distance Vector, like RIP
- several cost metrics (delay, bandwidth, reliability, load etc)
- uses TCP to exchange routing updates
- Loop-free routing via Distributed Updating Alg. (DUAL) based on *diffused computation*

Comparison of LS and DV algorithms

Message complexity
- LS: with n nodes, E links, O(nE) messages
- DV: exchange between neighbors only

Space requirements:
- LS maintains entire topology
- DV maintains only neighbor state

Speed of Convergence
- LS: O(n log n) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem
  - (faster with triggered updates)

Comparison of LS and DV algorithms

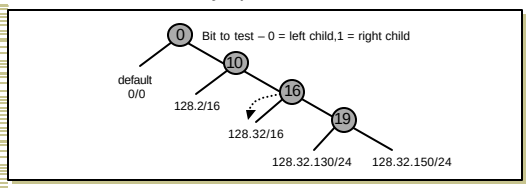
Robustness: what happens if router malfunctions?

LS:
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:
- DV node can advertise incorrect *path* cost
- each node's table used by others
  - errors propagate thru network

How To Do Variable Prefix Match

- Traditional method – Patricia Tree
  - Arrange route entries into a series of bit tests
- Worst case = 32 bit tests
  - Problem: memory speed is a bottleneck



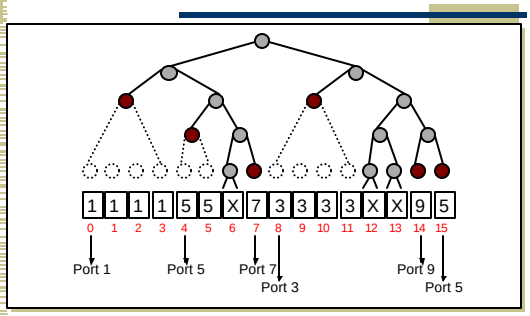
Speeding up Prefix Match - Alternatives

- 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

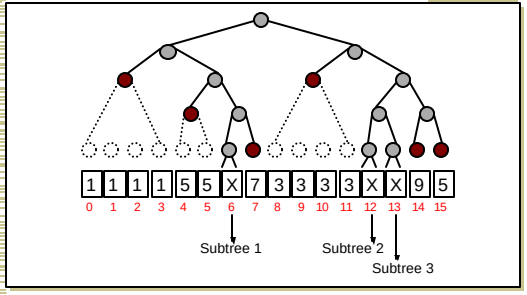
Speeding up Prefix Match

- Cut prefix tree at 16/24/32 bit depth
  - Fill in prefix tree entries by creating extra entries
    - Entries contain output interface for route
  - Add special value to indicate that there are deeper tree entries
    - Only keep 24/32 bit cuts as needed
- Example cut prefix tree at 16 bit depth
  - 64K entries!!
  - Use a variety of clever techniques to compress space taken

Prefix Tree



Prefix Tree



Speeding up Prefix Match

- Scaling issues
  - How would it handle IPv6
- Other possibilities
  - Why were the cuts done at 16/24/32 bits?

Speeding up Prefix Match - Alternatives

- Route caches
  - Packet trains → group of packets belonging to same flow
  - Temporal locality
  - Many packets to same destination
- Other algorithms
  - Bremler-Barr – Sigcomm 99
    - Clue = prefix length matched at previous hop
    - Why is this useful?