Routing

Fundamentals of Computer Networks
Guevara Noubir

Outline

- Introduction
- Broadcasting and Multicasting
- Shortest Path Unicast Routing
- Link Weights and Stability

Introduction to Wide Area Routing

- Routing is implemented within layer 3
- Main issues:
  - Selection of routes for (origin, destination) pairs
  - Delivery of messages to their correct destination
- Performance measures: throughput, average delay
- Focus of the lecture:
  - Selection of routes for fixed networks (no mobility)

Classification of Routing Algorithms

- Centralized versus Distributed
  - Algorithms may be equivalent at some abstraction level
- Static versus Adaptive
  - In static routing the path used by a session is fixed regardless of traffic conditions (e.g., congestion)
  - The routing algorithm changes the path if a congestion occurs on some of the used links

Broadcasting and Multicasting

- Broadcasting:
  - Sending a message from one node to all nodes in the network
- Multicasting:
  - Sending a message from one node to a set of nodes (multicast group)
- Unicast: Sending a message from one node to another node
- Advantages of broadcasting and multicasting:
  - Saves bandwidth, and increases overall throughput
- Use of broadcasting: e.g., update the nodes on link status
- Use of multicasting: newspaper, stock-market info, video streaming, etc.
- Difficulty of multicasting:
  - reliability, optimality, response to group dynamic

Broadcasting (Flooding)

- Simple broadcast algorithms: flooding, spanning tree
- Simple flooding algorithm:
  - Source sends out the message to each of its adjacent nodes
  - When a node receives a message it sends it out to all its adjacent nodes (except to the one from which it received it)
- Problem?
- How to solve it?

Broadcasting (Flooding)

- Complexity of flooding:
  - Time complexity:
  - Message complexity (communication complexity):

Broadcasting (Spanning Tree)

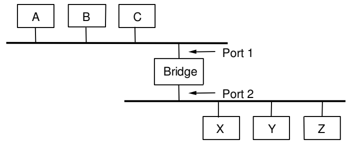
- A spanning tree is a connected sub-graph that contains all nodes and has no cycles
- A spanning tree can implement broadcasting with no use of sequence numbers
- Complexity of spanning tree: time, message
- Constructing a spanning tree that minimizes the diameter or total weight (delay/cost) in a distributed manner is difficult

Broadcasting (Spanning Tree)

- Minimum Spanning Tree:
  - Minimum cost tree which spans all nodes (Prim-Dijkstra's algorithm: add nearest members one by one to the tree)
  - Example:
- Building a spanning tree:
  - A node sends a "spanning tree" message to all its neighbors
  - Whenever a node *u* receives such a message from *v*, *u* sets *v* as its parent
  - Whenever a node that has already determined its parent receives such a message, it ignores it

Bridges and Extended LANs

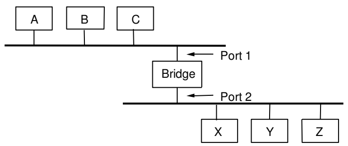
- One use of spanning trees at the LAN level in Bridges
- LANs have physical limitations (e.g., 2500m)
- Connect two or more LANs with a *bridge*
  - Accept and forward strategy
  - Level 2 connection (does not add packet header)



- Ethernet Switch is a LAN Switch = Bridge

Learning Bridges

- Do not forward when unnecessary
- Maintain forwarding table

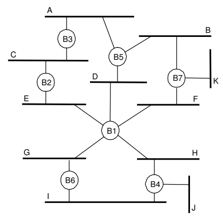


Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

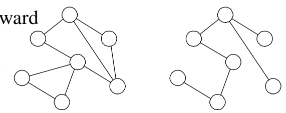
- Learn table entries based on source address
- Table is an optimization; need not be complete
- Always forward broadcast frames

Spanning Tree Algorithm

- Problem: loops



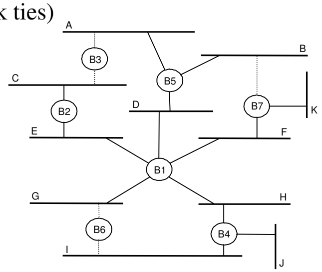
- Bridges run a distributed spanning tree algorithm
  - Select which bridges actively forward
  - Developed by Radia Perlman
  - Now in IEEE 802.1 specification



Algorithm Overview

- Each bridge has unique id (e.g., B1, B2, B3)
- Select bridge with smallest id as root
- Select bridge on each LAN closest to root as designated bridge (use id to break ties)

Each bridge forwards frames over each LAN for which it is the designated bridge



Algorithm Details

- Bridges exchange configuration messages
  - Id for bridge sending the message
  - Id for what the sending bridge believes to be root bridge
  - Distance (hops) from sending bridge to root bridge
- Each bridge records current best configuration message for each port
- Initially, each bridge believes it is the root

Algorithm Detail (cont)

- When learn not root, stop generating config messages
  - in steady state, only root generates configuration messages
- When learn not designated bridge, stop forwarding config messages
  - in steady state, only designated bridges forward config messages
- Root continues to periodically send config messages
- If any bridge does not receive config message after a period of time, it starts generating config messages claiming to be the root

Broadcast and Multicast

- Forward all broadcast/multicast frames
  - current practice
- Learn when no group members downstream
- Accomplished by having each member of group G send a frame to bridge multicast address with G in source field

Limitations of Bridges

- Do not scale
  - Spanning tree algorithm does not scale
  - Broadcast does not scale
- Do not accommodate heterogeneity
- Caution: beware of transparency
  - Bridged LANs do not always behave as single shared medium LAN: they drop packets when congested, higher latency

Shortest Path Unicast Routing

- Each link is assigned a number called *length*
  - The length depends on the direction (asymmetric links)
  - The length may depend on the link bandwidth, delay, congestion, etc.
  - In general the length may change with time (ignored during the next slides)
- Two famous algorithms:
  - Bellman-Ford's algorithm (Routing Information Protocol:RIP)
  - Dijkstra's algorithm (Open Shortest Path First: OSPF)

Bellman-Ford's Algorithm

- Goal: computer the shortest path from any node *u* to *t*
- Start: $D_u^0 = \infty$
- Iteration *i*:
  - Invariant: each node *u* has determined a shortest path to *t* using at most *i* hops
  - $D_u^{i+1} = \min_v \{d(u,v) + D_v^i\}$
- Illustration on a graph:
- Complexity:
  - Time = |*V*|-1 iterations; Computations = at most O(|*V*|^3),

Distributed Bellman-Ford Alg.

- Synchronous environment: easy
- Asynchronous:
  - Even if the nodes are not synchronized (different iterations)
  - Even if the starting value of nodes different from the destination are not accurate
  - Even if the weights change
  - The Bellman-Ford will converge
- Illustration on a graph with non accurate starting values

Routing Information Protocol

- Uses Bellman-Ford's algorithm
- Protocol over UDP, port 520
- Distance-vector protocol
- Protocol overview:
  - Init: send a request packet over all interfaces
  - On response reception: update the routing table
  - On request reception:
    - if request for complete table (*address family*=0) send the complete table
    - else send reply for the specified address (infinity=16)
  - Regular routing updates:
    - every 30 seconds part/entire routing table is sent (broadcast) to neighboring routers
  - Triggered updates: on metric change for a route
  - Simple authentication scheme

Dijkstra's Algorithm

- Assumption: all weights are non-negative
- Goal: grow a tree with root the destination node
  - Start: tree T = destination node
  - Iterate:
    - Add to the tree a node u such that $D_u = \min_{v \notin T} D_v$
    - Every node $v \notin T$ updates $D_v = \min_{w \in T} \{D_w + d(\mathrm{v},\mathrm{w})\}$
- Efficient implementation:
  - Every node $v \notin T$ updates $D_v = \min \{D_v, D_u + d(v, u)\}$
- Complexity:
  - Number of iterations = $|V|-1$; each iteration takes $O(|V|)$
  - Thus the running time is: $O(|V|^2)$

Open Shortest Path First

- IP protocol (not over UDP), reliable (sequence numbers, acks)
- Protocol overview: link state protocol
  - The link status (cost) is sent/forwarded to all routers (LSP)
  - Each router knows the exact topology of the network
  - Each router can compute a route to any address
  - simple authentication scheme
- Advantages over RIP
  - Faster to converge
  - The router can compute multiple routes (e.g., depending on the type of services, load balancing)
  - Use of multicasting instead of broadcasting (concentrate on OSPF routers)

Routing using Dijkstra's Alg. = Forward Search (OSPF)

- Each node *s*:
  - Collects the LSPs for the whole network
  - Maintains two lists: *tentative, confirmed*

1. Confirmed = {(*s*, 0, s)}
2. Last added node to *confirmed* list is called *Next*,
3. For each *Neighbor* of *Next*: cost = d(*s,Next*)+d(*Next, Neighbor*)
   1. If *Neighbor* is neither on tentative nor on confirmed lists: add (*Neighbor, cost, NextHop*) to tentative list
   2. If *Neighbor* is on the tentative list: update the list
4. If the *tentative* list is empty stop. Otherwise, pick the entry from the *tentative* list with the lowest cost and move it to the *confirmed* list

Link Length and Stability Issues

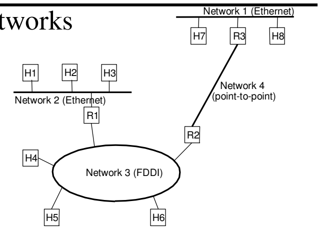
- Issues:
  - Dynamic link status,
  - Computation of the link weights: fct(latency, bandwidth, congestion)
- Static costs: fct(latency, capacity) = *hop metric*
  - Problems in a heavily loaded network
- Dynamic costs: fct(average queue size)
  - Problem of oscillation
- Hybrid costs:
  1. f1(latency, capacity) + α* f2(average queue size)
  2. Average dynamic costs over more then one shortest path update

Unicast Routing Protocols

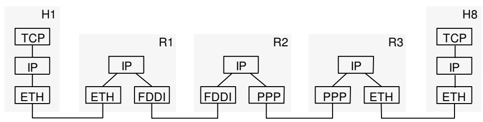
- Internet is constituted of Autonomous Systems (AS)
- Interior Gateway Protocols (IGP) inside an AS
  - Routing Information Protocol (RIP: RFC1388)
  - Open Shortest Path First (OSPF: RFC1247)
- Exterior Gateway Protocols (EGP) between AS
  - Border Gateway Protocol (BGP: RFC1267)
- Classless Inter-domain Routing (CIDR: RFC1518, RFC1519)

IP Internet

- Concatenation of Networks

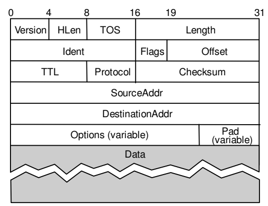


- Protocol Stack



Service Model

- Connectionless (datagram-based)
- Best-effort delivery (unreliable service)
  - packets are lost
  - packets are delivered out of order
  - duplicate copies of a packet are delivered
  - packets can be delayed for a long time
- Datagram format

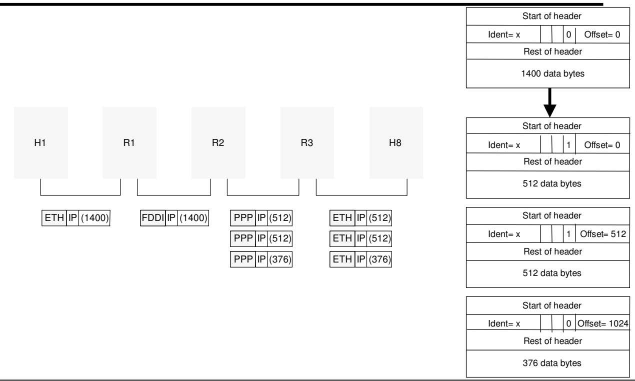


Fragmentation and Reassembly

- Each network has some MTU
- Strategy
  - fragment when necessary (MTU < Datagram)
  - re-fragmentation is possible
  - fragments are self-contained datagrams
  - use CS-PDU (not cells) for ATM
  - delay reassembly until destination host
  - do not recover from lost fragments

  - hosts are encouraged to perform “path MTU discovery”

Example

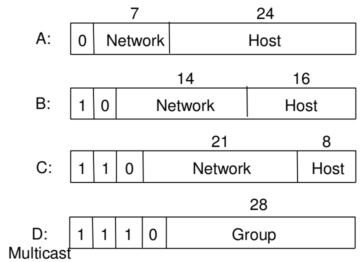


Internet Control Message Protocol (ICMP) RFC 792

- Integral part of IP but runs as ProtocolType = 1 using an IP packet
- Codes:
  - Echo (ping)
  - Redirect (from router to source host)
  - Destination unreachable (protocol, port, or host)
  - TTL exceeded (so datagrams don’t cycle forever)
  - Checksum failed
  - Reassembly failed

Global Addresses

- Properties
  - globally unique
  - hierarchical: network + host
- Dot Notation
  - 10.3.2.4
  - 128.96.33.81
  - 192.12.69.77



Datagram Forwarding

- Strategy
  - every datagram contains destination’s address
  - if directly connected to destination network, then forward to host
  - if not directly connected to destination network, then forward to some router
  - forwarding table maps network number into next hop
  - each host has a default router
  - each router maintains a forwarding table
- Example (R2)

Network Number	Next Hop
1	R3
2	R1
3	interface 1
4	interface 0

Address Translation

- Map IP addresses into physical addresses
  - destination host
  - next hop router
- Techniques
  - encode physical address in host part of IP address
  - table-based
- ARP
  - table of IP to physical address bindings
  - broadcast request if IP address not in table
  - target machine responds with its physical address
  - table entries are discarded if not refreshed

ARP Details

- Request Format
  - HardwareType: type of physical network (e.g., Ethernet)
  - ProtocolType: type of higher layer protocol (e.g., IP)
  - HLEN & PLEN: length of physical and protocol addresses
  - Operation: request or response
  - Source/Target-Physical/Protocol addresses
- Notes
  - table entries timeout in about 15 minutes
  - update table with source when you are the target
  - update table if already have an entry
  - do not refresh table entries upon reference

ARP Packet Format

<table>
<tr><td>0</td><td>8</td><td>16</td><td>31</td></tr>
<tr><td colspan="2">Hardware type = 1</td><td colspan="2">ProtocolT ype = 0x0800</td></tr>
<tr><td>HLen = 48</td><td>PLen = 32</td><td colspan="2">Operation</td></tr>
<tr><td colspan="4">SourceHardwareAddr (bytes 0 – 3)</td></tr>
<tr><td colspan="2">SourceHardwareAddr (bytes 4 – 5)</td><td colspan="2">SourceProtocolAddr (bytes 0 – 1)</td></tr>
<tr><td colspan="2">SourceProtocolAddr (bytes 2 – 3)</td><td colspan="2">TargetHardwareAddr (bytes 0 – 1)</td></tr>
<tr><td colspan="4">TargetHardwareAddr (bytes 2 – 5)</td></tr>
<tr><td colspan="4">TargetProtocolAddr (bytes 0 – 3)</td></tr>
</table>