Non-Adaptive Routing

Shortest Path Routing
Non-Adaptive Shortest Path Routing

- Static Procedure
  - Network operator generates tables
  - Tables
    - Are loaded when IS operation is initiated and
    - Will not be changed any more

- Characteristics
  - Simple
  - Good results with relatively consistent topology and traffic
  - But
    - Poor performance if traffic volume or topologies change over time
Non-Adaptive Shortest Path Routing

- Spanning Tree and Optimized Route
  - Information about the entire network has to be available
    - i.e. application actually as a benchmark comparison

- Example
  - Link is labeled with distance / weight
  - Node is labeled with distance from source node along best known path (in parentheses)
  - Find the shortest path from A to D
Non-Adaptive Shortest Path Routing

Procedure: e.g. according to Dijkstra

- Find the shortest path from A to D
  - Labels may be permanent or tentative
  - Initially, no paths are known
    - All nodes are labeled with infinity (tentative)
  - Discover the labels that represent shortest possible path from source to any node
  - Make those labels permanent

1. Node A labeled as permanent (filled-in circle)
2. Relabel all directly adjacent nodes with the distance to A (path length, nodes adjacent to source)
3. Examine all tentatively labeled nodes, make the node with the smallest label permanent
4. This node will be the new working node for the iterative procedure (i.e., continue with step 2.)
Non-Adaptive Shortest Path Routing

- Procedure: e.g. according to Dijkstra
- Find the shortest path from A to D:
  1. A flagged as permanent (filled-in circle)
  2. Relabel all directly adjacent nodes with the distance to A
     - (path length, IS adjacent to the source):
Non-Adaptive Shortest Path Routing

- Procedure: e. g. according Dijkstra
- Find the shortest path from A to D:
  - ...
  3. Compare all recent, not firmly flagged IS;
    - flag the one with the lowest number as fixed
  4. This IS is the origin of the iterative procedure
    - (i. e. continue with item 2.)
Non-Adaptive Shortest Path Routing

- Procedure: e.g., according to Dijkstra
- Find the shortest path from A to D:
  1. Node B has been labeled as permanent (filled-in circle)
  2. relabel all directly adjacent nodes with the distance to B (path length, nodes adjacent to source):
     - A (does not apply, because it is the origin),
Non-Adaptive Shortest Path Routing

Procedure: e.g., according to Dijkstra
find the shortest path from A to D:

1. ...
2. ...
3. examine all tentatively labeled nodes;
   - make the node with the smallest label permanent
4. this node will be the new working node for the iterative procedure ...
Non-Adaptive Routing

Flooding
Flooding

- **Principle**
  - IS transmits the received packet to all adjacent IS
    - But generates "an infinite amount" of packets

- **Flood limitations**
  - Hop counter in the packet header
    - Initialize to destination’s distance, or subnet diameter is unknown
    - Decrement per hop, discard packet at 0
  - Keeping history of transferred packets in ISs, delete copies
    - Source router inserts sequence number into each packet
    - ISs keeps per-router history of sequence numbers
    - Old packets are dropped
Selective Flooding

- **Approach**
  - Do not send out on every line
  - IS transmits received packet to adjacent stations, located in the Direction of the Destination
  - With ‘regular’ topologies this makes sense and is an optimization
  - But some topologies do not fit well to this approach

- **Comment**
  - Geographically-oriented routing got recent interest for mobile scenarios
Flooding

Evaluation and use

- In most scenarios impractical because of overhead
- Extremely robust
- Reaches all ISs
- Does not need topology information, no bootstrap
- Always finds the shortest path
Adaptive Routing
Adaptive Routing

- Class Adaptive Algorithms
  - Decisions are based on current network state
    - Measurements / estimates of the topology and the traffic volume
  - Further sub-classification into
    - Centralized algorithms
    - Isolated algorithms
    - Distributed algorithms
Adaptive Routing

Centralized Routing
Adaptive Centralized Routing

- Principle
  - One routing control center (RCC) exists in the network
  - Each IS sends periodically status updates to the RCC
    - Available neighbors
    - Current queue length
    - Line utilization
    -...
  - RCC
    - Collects information
    - Computes the optimal path each IS pair
    - Forms routing tables
    - Distributes tables to ISs
Adaptive Centralized Routing

- Characteristics
  - RCC has complete information
  - IS is free of routing calculations

- But
  - Re-calculations quite often necessary (approx. once/min or more often)
  - Low robustness
  - No correct decisions if network is partitioned
  - ISs receive tables at different times
  - Traffic concentration in RCC proximity
Adaptive Routing

Isolated Routing
Adaptive Isolated Routing through Backward Learning

- Isolated routing
  - Every IS makes decision based on locally gathered information only
    - No exchange of routing information among nodes
    - Only limited adaptation possibility to changed traffic or topology
  - IS ‘learns’ from received packets
    - Source IS
    - Distance estimate by hop count
Adaptive Isolated Routing through Backward Learning

Packet

- From source S
- received on line L
- after C hops
  ⇒ S is reachable on L within C hops

Routing table in IS

- L - table (destination - IS, outgoing line, $C_{min}$)
- Update of the routing table

IS receives packet ( ..., S, C, ... ) on L

if not (S in L-Table)
  then Add(S, L, C)
else if $C < C_{min}$
  then Update(S, L, C)
Adaptive Isolated Routing through Backward Learning

Example: D learns about A

- packet ( ..., source - IS, section counter, ...
  - P1 ( ..., A, 4, ... ) → Add ( A, l1, 4 )
  - P2 ( ..., A, 3, ... ) → Update ( A, l2, 3 )
Adaptive Isolated Routing through Backward Learning

Problem
- Packets use a different route, e.g. because of failures, high load
- Algorithm retains only the old value (because it was "better"),
  - i.e. algorithm does not react to deteriorations

Solution
- Periodic deletion of routing tables (new learning period)
- Table deletion
  - Too often: mainly during the learning phase
  - Not often enough: reaction to deteriorations too slow
Adaptive Routing

Distributed Routing

Distance Vector
Distance Vector Routing

- **Distance-Vector Routing**
  - Group of Distance Vector Routing Algorithms
  - Also known as
    - Distributed Bellman-Ford algorithm, Ford-Fulkerson algorithm

- **Use**
  - Was the original ARPANET routing algorithm
  - Has been used in the Internet as RIP (Routing Information Protocol)

- **Basic principle**
  - IS maintains table (i.e., vector) stating
    - Best known distance to destinations
    - And line to be used
  - ISs update tables
    - By exchanging routing information with their neighbors
Distance Vector Routing

Each IS
- maintains routing table with one entry per router in the subnet
- is assumed to know the ‘distances’ to each neighbor

- IS sends list with estimated distances to each destination periodically to its neighbors

- X receives list $E(Z)$ from neighbor $Y$
  - Distance $X$ to $Y$: $e$
  - Distance $Y$ to $Z$: $E(Z)$
  - Distance $X$ to $Z$ via $Y$: $E(Z)+e$

- IS computes new routing table from the received lists containing
  - Destination IS
  - Preferred outgoing path
  - Distance
Distance Vector Routing

- Previous routing table will not be taken into account
  - Reaction to deteriorations
Distance Vector Routing

- Fast route improvement

- Fast distribution of information about new short paths (with few hops)

- Example
  - initially A unknown
  - later: A connected with distance 1 to B, this will be announced
  - Distribution proportional to topological spread
  - Synchronous (stepwise) update is a simplification
Distance Vector Routing

- Slow distribution of information about new long paths (with many hops)
- “Count to Infinity” problem of DVR

Example: deterioration
- Here: connection destroyed
- A previously known, but now detached
- The values are derived from (incorrect) connections of distant IS

Comment
- Limit "infinite" to a finite value, depending on the metrics, e.g.
  - ‘infinite’ = maximum path length+1
**Distance Vector Routing**

- **Variant: ‘Split Horizon Algorithm’**
- **Objective:** improve the "count to infinity" problem
- **Principle**
  - In general, to publicize the "distance" to each neighbour
  - If neighbor Y exists on the reported route, X reports the response "false" to Y
    - distance X (via Y) according to arbitrary i: $\infty$

- **Example:** deterioration (connection destroyed)
  - B to C: $A = \infty$ (real),
  - C to B: $A = \infty$ (because A is on path), ...

- **But:** still poor, depending on topology, example
  - Connection CD is removed
  - A receives "false information" via B
  - B receives "false information" via A
    - Slow distribution (just as before)
Adaptive Routing

Distributed Routing
Link State Routing
Link State Routing

Basic principle
- IS measures the "distance" to the directly adjacent IS
- Distributes information
- Calculates the ideal route

Procedure
1. Determine the address of adjacent IS
2. Measure the "distance" (delay, ...) to neighbouring IS
3. Organize the local link state information in a packet
4. Distribute the information to all IS
5. Calculate the route based on the information of all IS

Use
- Introduced into the ARPANET in 1979, nowadays most prevalent
- IS-IS (Intermediate System-Intermediate System)
  - developed by DECNET
  - also used as ISO CLNP in NSFNET
  - Novell Netware developed its own variation from this (NLSP)
- OSPF (Open Shortest Path First)
  - since 1990 Internet RFC 1247
Link State Routing

1. Phase: gather information about the adjacent intermediate systems
Link State Routing

1. Phase: gather information about the adjacent intermediate systems

Initialization procedure
- New IS
  - Sends a HELLO message over each L2 channel
- Adjacent IS
  - Responds with its own address, unique within the network
Link State Routing

2. Phase: measure the "distance"
   - Definition of distance needed
     - Usually delay
     - Where to measure?

- Uses & Looks up
  - Data packets
  - Routing Process
  - Topology, link utilization, etc. information
  - Fills & Updates

- Routing table
  - destination | link
  - A       | 0
  - B       | 3
  - C       | 1
  - D       | 4

- Queues
- Forwarding Process
- HELLO
- When to start timer?
2. Phase: measure the "distance"

- Queuing delay
  - Measuring without does not take load into account
  - Measuring with does ⇒ usually better

But
- Possibility for oscillations (route flapping)
- Once per routing table update
Link State Routing

3. Phase: organizing the information as link state packet
- Including own address, sequence number, age, "distance"
- Timing problems: validity and time of sending
  - Periodically
  - In case of major changes

Link State Packets:

- **A**
  - Seq.: 4
  - Age: B
- **B**
  - Seq.: 4
  - Age: A
- **C**
  - Seq.: 2
  - Age: D
- **D**
  - Seq.: 3
  - Age: F
- **E**
  - Seq.: 5
  - Age: C
- **F**
  - Seq.: 6
  - Age: E
Link State Routing

4. Distributing the local information to all IS
   - By applying the flooding procedure (very robust)
     - Therefore sequence number in packets
   - Problem: inconsistency
     - Varying states simultaneously available in the network
     - Indicate and limit the age of packet,
       i. e. IS removes packets that are too old

5. Computing new routes
   - Each IS for itself
   - Possibly larger amount of data available