Lecture 8: IP Routing Protocols I

Dr. Mohd Nazri Bin Mohd Warip

High Performance Broadband Networks Research Group
Embedded, Networks and Advanced Computing Research Cluster
School of Computer and Communication Engineering
Universiti Malaysia Perlis

Session 2013/2014
Lecture Outline

- Overview: Forwarding vs Routing
- Static vs Dynamic Routing
- Static Route
- RIP
- Link State
- OSPF
- Distance Vector
- EIGRP
Overview: Routing and Forwarding

- **Routing:**
  - How to determine the routing table entries
  - carried out by routing daemon

- **Forwarding:**
  - Look up routing table & forward packet from input to output port.
  - carried out by IP layer

- Routers exchange information using routing protocols to develop the routing tables
# Forwarding Table vs Routing Table

<table>
<thead>
<tr>
<th><strong>Forwarding Table</strong></th>
<th><strong>Routing Table</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Used when a packet is being forwarded and so must contain enough information to accomplish the forwarding function</td>
<td>Built by the <em>routing algorithm</em> as a precursor to build the forwarding table</td>
</tr>
<tr>
<td>A row in the forwarding table contains the <em>mapping</em> from a network number to an outgoing interface and some MAC information, such as Ethernet Address of the next hop</td>
<td>Generally contains <em>mapping</em> from network numbers to next hops</td>
</tr>
</tbody>
</table>
Key Role of Routing

- How to get packet from here to there?
- Decentralized nature of Internet makes routing a major challenge
  - Interior gateway protocols (IGPs) are used to determine routes within a domain
  - Exterior gateway protocols (EGPs) are used to determine routes across domains
  - Routes must be consistent & produce stable flows
- Forwarding:
  - Look up routing table & forward packet from input to output port.
  - carried out by IP layer
- Routers exchange information using routing protocols to
# Static vs. Dynamic Routes

<table>
<thead>
<tr>
<th>Static Routing</th>
<th>Dynamic Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set up manually, do not change; requires administration</td>
<td>Adapt to changes in network conditions</td>
</tr>
<tr>
<td>Works when traffic predictable &amp; network is simple</td>
<td>Uses a route that a network routing protocol adjusts automatically for topology or traffic changes</td>
</tr>
<tr>
<td>Used to override some routes set by dynamic algorithm</td>
<td>Calculates routes based on received updated network state information</td>
</tr>
<tr>
<td>Used to provide default router</td>
<td></td>
</tr>
</tbody>
</table>
IP Routing Protocols

- IP routing protocols are characterized by:
  - Classful or classless IP addressing/routing
  - Static and dynamic routing (distance vector and link state)
  - Routing metrics
  - Variable-Length Subnet Mask (VLSM) to conserve and use efficiently the total IP addresses allocation
  - Route summarization across network boundaries
  - Timers
IP Routing Protocols

- IP routing provides a mechanism to route packets from different network addresses classified as an inter-domain or Interior Gateway Protocol (IGP) and intra-domain or Exterior Gateway Protocol (EGP) routing protocols.

- IGP is designed to distribute routing information within an Autonomous System (AS) and uses the IP address to establish the route, such as Routing Information Protocol (RIP), Interior Gateway Routing Protocol (IGRP), and Open Shortest Path First (OSPF).

- EGP is also used to exchange routing information among different Autonomous Systems (ASs) and depends on an AS number to construct paths, for example, Border Gateway Protocol (BGP).
Autonomous Systems: Interior or Exterior Routing Protocols

- An autonomous system is a collection of networks under a common administrative domain.
- IGPs operate within an autonomous system.
- EGPs connect different autonomous systems.

RIP, IGRP, EIGRP, OSPF

EGPs: BGP

Autonomous System 100

Autonomous System 200
Routing Metrics

- **Bandwidth:**
  - The data capacity of a link (the connection between two network equipments)

- **Delay:**
  - The length of time required to move a packet along each link from source to destination.

- **Hopcount**
  - The number of routers that a packet must travel through before reaching its destination

- **Cost**
  - An arbitrary value assigned by a network administrator, usually based on bandwidth, administrator preference or other measurement.
Static Routes

- Configure unidirectional static routes to and from a stub network to allow communications to occur.
RIP Features

- Maximum is 16 equal-cost paths (default = 4)
- Hop-count metric selects the path
- Routes update every 30 seconds.
# RIPv1 and RIPv2 Comparison

<table>
<thead>
<tr>
<th></th>
<th>RIPv1</th>
<th>RIPv2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing protocol</td>
<td>Classful</td>
<td>Classless</td>
</tr>
<tr>
<td>Supports variable-length subnet mask?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sends the subnet mask along with the routing update?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Addressing type</td>
<td>Broadcast</td>
<td>Multicast</td>
</tr>
<tr>
<td>Defined in …</td>
<td>RFC 1058</td>
<td>RFCs 1721, 1722, and 2453</td>
</tr>
<tr>
<td>Supports manual route summarization?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Authentication support?</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Link-State Routing Protocols

- After initial flood, passes small event-triggered link-state updates to all other routers.
Link-State Protocols

The diagram illustrates the process of Link-State Protocols:

1. **LSA Updates:** Information is exchanged through Link-State Advertisements (LSAs).
2. **Link-State Database:** Each router maintains a database containing information about the network's links and nodes.
3. **Dijkstra's Algorithm:** The SPF (Shortest Path First) algorithm is used to calculate the shortest paths from the router to all other nodes.
4. **Routing Table:** Based on the SPF Tree, the router updates its routing table with the best paths to reach each destination.

This process ensures that each router has a comprehensive view of the network and can efficiently route data to its destinations.
Link-State Protocol Data Structures

- Link-state routers recognize more information about the network than their distance vector counterparts.
  - Neighbour table: also known as the adjacency database
  - Topology table: referred as the LSDB
  - Routing table: also known as the forwarding database

- Each router has a full picture of the topology.
- Link-state routers tend to make more accurate decisions
## Benefits and Drawbacks of Link-State Routing

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Fast convergence:</td>
<td>- Significant demands for resources:</td>
</tr>
<tr>
<td>- Changes are reported immediately by the affected</td>
<td>- Memory (three tables: adjacency, topology, forwarding).</td>
</tr>
<tr>
<td>source.</td>
<td>- CPU (Dijkstra’s algorithm can be intensive, especially when there</td>
</tr>
<tr>
<td>- Robustness against routing loops:</td>
<td>are many instabilities).</td>
</tr>
<tr>
<td>- Routers know the topology.</td>
<td>- Requires very strict network design.</td>
</tr>
<tr>
<td>- Link-state packets are sequenced and acknowledged.</td>
<td>- Configuration can be complex when tuning various parameters and when</td>
</tr>
<tr>
<td>- Hierarchical network design enables optimization of</td>
<td>design is complex.</td>
</tr>
<tr>
<td>resources.</td>
<td></td>
</tr>
</tbody>
</table>

- Memory (three tables: adjacency, topology, forwarding).
- CPU (Dijkstra’s algorithm can be intensive, especially when there are many instabilities).
- Requires very strict network design.
- Configuration can be complex when tuning various parameters and when design is complex.
### Elements of Routing Techniques

<table>
<thead>
<tr>
<th>Elements</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Criteria</td>
<td>- Number of nodes</td>
</tr>
<tr>
<td></td>
<td>- Cost</td>
</tr>
<tr>
<td></td>
<td>- Delay</td>
</tr>
<tr>
<td></td>
<td>- Throughput</td>
</tr>
<tr>
<td>Decision Time</td>
<td>- Packet (datagram)</td>
</tr>
<tr>
<td></td>
<td>- Session (Virtual circuit)</td>
</tr>
<tr>
<td>Decision Place</td>
<td>- Each node (distributed),</td>
</tr>
<tr>
<td></td>
<td>- Central node (centralized)</td>
</tr>
<tr>
<td></td>
<td>- Originating node (source)</td>
</tr>
<tr>
<td>Network Information Source</td>
<td>- None</td>
</tr>
<tr>
<td></td>
<td>- Local</td>
</tr>
<tr>
<td></td>
<td>- Adjacent node</td>
</tr>
<tr>
<td></td>
<td>- Nodes along route</td>
</tr>
<tr>
<td></td>
<td>- All nodes</td>
</tr>
<tr>
<td>Network Information Update Timing</td>
<td>- Continuous</td>
</tr>
<tr>
<td></td>
<td>- Periodic</td>
</tr>
<tr>
<td></td>
<td>- Major load change</td>
</tr>
<tr>
<td></td>
<td>- Topology change</td>
</tr>
</tbody>
</table>
Routing Algorithm Requirements

- **Responsiveness to changes**
  - Topology or bandwidth changes, congestion
  - Rapid convergence of routers to consistent set of routes
  - Freedom from persistent loops

- **Optimality**
  - Resource utilization, path length

- **Robustness**
  - Continues working under high load, congestion, faults, equipment failures, incorrect implementations

- **Simplicity**
  - Efficient software implementation, reasonable processing load
Shortest Path Algorithm (1)

- Dijkstra’s algorithm computes a sink tree on the graph:
  - Each link is assigned a non-negative weight/distance
  - Relax distance to other nodes
  - Using weights of 1 gives paths with fewest hops

- Algorithm:
  - Start with sink, set distance at other nodes to infinity
  - Shortest path is the one with lowest total weight
  - Pick the lowest distance node, add it to sink tree
  - Repeat until all nodes are in the sink tree
Dijkstra’s Algorithm

- finds shortest paths from given source nodes to all other nodes
- develop paths in order of increasing path length
- algorithm runs in stages
  - each time adding node with next shortest path
- algorithm terminates when all nodes have been added to $T$
# Dijkstra’s Algorithm Method

## Step 1 [Initialization]

- **T = {s}** Set of nodes so far incorporated
- **L(n) = w(s, n) for n ≠ s**
- **initial path costs to neighboring nodes are simply link costs**

## Step 2 [Get Next Node]

- **find neighboring node not in T with least-cost path from s**
- **incorporate node into T**
- **also incorporate the edge that is incident on that node and a node in T that contributes to the path**

## Step 3 [Update Least-Cost Paths]

- **L(n) = min[L(n), L(x) + w(x, n)] for all n ∉ T**
- **if latter term is minimum, path from s to n is path from s to x concatenated with edge from x to n**
Dijkstra’s Algorithm Example 1

- Figure 1 illustrates a network in which the two arrowed lines between a pair of nodes represent a link between theses nodes, and the corresponding numbers represents the current link cost in each direction. The shortest path (fewest hops) from node 1 to node 6 is 1-3-6 (cost = 5 + 5 = 10), but the least-cost path is 1-4-5-6 (cost = 1 + 1 + 2 = 4)
Packet Switched Network Example

Figure 1
Dijkstra’s Algorithm Example

- finds Table 1 and Figure 2 show the result of applying this algorithm to the graph of Figure 1, using $s=1$. The shaded edges define the spanning tree for the graph. The values in each circle are the current estimates of $L(x)$ for each node $x$. A node is shade when it is added to $T$. Note that at each step the path to each node plus the total cost of that path is generated. After the final iteration, the least-cost path to each node and the cost of that path have been developed. The same procedure can be used with node 2 as source node, and so on.
Dijkstra’s Algorithm Example

Figure 2
## Dijkstra’s Algorithm Example

### Table 1 s=1

<table>
<thead>
<tr>
<th>Iter</th>
<th>T</th>
<th>L(2)</th>
<th>Path</th>
<th>L(3)</th>
<th>Path</th>
<th>L(4)</th>
<th>Path</th>
<th>L(5)</th>
<th>Path</th>
<th>L(6)</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1}</td>
<td>2</td>
<td>1–2</td>
<td>5</td>
<td>1-3</td>
<td>1</td>
<td>1–4</td>
<td>∞</td>
<td>-</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>{1,4}</td>
<td>2</td>
<td>1–2</td>
<td>4</td>
<td>1-4-3</td>
<td>1</td>
<td>1–4</td>
<td>2</td>
<td>1-4-5</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>{1, 2, 4}</td>
<td>2</td>
<td>1–2</td>
<td>4</td>
<td>1-4-3</td>
<td>1</td>
<td>1–4</td>
<td>2</td>
<td>1-4-5</td>
<td>∞</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>{1, 2, 4, 5}</td>
<td>2</td>
<td>1–2</td>
<td>3</td>
<td>1-4-5-3</td>
<td>1</td>
<td>1–4</td>
<td>2</td>
<td>1-4-5</td>
<td>4</td>
<td>1-4-5-6</td>
</tr>
<tr>
<td>5</td>
<td>{1, 2, 3, 4, 5}</td>
<td>2</td>
<td>1–2</td>
<td>3</td>
<td>1-4-5-3</td>
<td>1</td>
<td>1–4</td>
<td>2</td>
<td>1-4-5</td>
<td>4</td>
<td>1-4-5-6</td>
</tr>
<tr>
<td>6</td>
<td>{1, 2, 3, 4, 5, 6}</td>
<td>2</td>
<td>1–2</td>
<td>3</td>
<td>1-4-5-3</td>
<td>1</td>
<td>1-4</td>
<td>2</td>
<td>1-4-5</td>
<td>4</td>
<td>1-4-5-6</td>
</tr>
</tbody>
</table>
Link-State Algorithm

- Basic idea: two step procedure
  - Each source node gets a map of all nodes and link metrics (link state) of the entire network
  - Find the shortest path on the map from the source node to all destination nodes

- Broadcast of link-state information
  - Every node $i$ in the network broadcasts to every other node in the network:
    - ID’s of its neighbours: $\mathcal{N}_i$=set of neighbours of $i$
    - Distances to its neighbours: $\{C_{ij} \mid j \in \mathcal{N}_i\}$
  - Flooding is a popular method of broadcasting packets
Dijkstra Algorithm: Finding shortest paths in order

- Find shortest paths from source $s$ to all other destinations
  - Closest node to $s$ is 1 hop away
  - 2nd closest node to $s$ is 1 hop away from $s$ or $w''$
  - 3rd closest node to $s$ is 1 hop away from $s$, $w''$, or $\chi$
Dijkstra’s algorithm

- **N**: set of nodes for which shortest path already found

**Initialization**: (Start with source node s)
- \( N = \{s\}, D_s = 0, \text{“s is distance zero from itself”} \)
- \( D_j = C_{sj} \text{ for all } j \neq s, \text{ distances of directly-connected neighbours} \)

**Step A**: (Find next closest node i)
- Find \( i \notin N \) such that
  - \( D_i = \min D_j \text{ for } j \notin N \)
  - Add i to N
  - If N contains all the nodes, stop

**Step B**: (update minimum costs)
- For each node \( j \notin N \)
  - \( D_j = \min (D_j, D_i + C_{ij}) \)
  - Go to Step A

*Minimum distance from s to j through node i in N*
Example 2: Dijkstra’s algorithm

<table>
<thead>
<tr>
<th>Iteration</th>
<th>N</th>
<th>D₂</th>
<th>D₃</th>
<th>D₄</th>
<th>D₅</th>
<th>D₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>{1}</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>1</td>
<td>{1,3}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>∞</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>{1,2,3}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>{1,2,3,6}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>{1,2,3,4,6}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>{1,2,3,4,5,6}</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>
Shortest Paths in Dijkstra’s Algorithm
Example 3: Shortest Path Algorithm

- A network and first five steps in computing the shortest paths from A to D. Pink arrows show the sink tree so far.
Shortest Path Algorithm

```c
for (p = &state[0]; p < &state[n]; p++) {
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;
do {
    for (i = 0; i < n; i++)
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }
    ...
}
```

Start with the sink, all other nodes are unreachable

Relaxation step. Lower distance to nodes linked to newest member of the sink tree
Shortest Path Algorithm

... 

k = 0; min = INFINITY;
for (i = 0; i < n; i++)
    if (state[i].label == tentative && state[i].length < min) {
        min = state[i].length;
        k = i;
    }
state[k].label = permanent;
} while (k != s);

Find the lowest distance, add it to the sink tree, and repeat until done
OSPF Overview

- Creates a neighbour relationship by exchanging hello packets
- Propagates LSAs rather than routing table updates
  - Link: Router interface
  - State: Description of an interface and its relationship to neighbouring routers
- Floods LSAs (Link-State Advertisements) to all OSPF routers in the area, not just directly connected routers
- Pieces together all the LSAs generated by the OSPF routers to create the OSPF link-state database
- Uses the SPF algorithm to calculate the shortest path to each destination and places it in the routing table
**OSPF Areas**

- Link-state routing requires a hierarchical network structure
- This two-level hierarchy consists of the following:
  - Transit area (backbone or area 0)
  - Normal areas (non-backbone areas)
OSPF Hierarchical Routing

- Consists of areas and autonomous systems
- Minimizes routing update traffic
Area Terminology and Router Types

- ABR: Area Border Router
- ASBR: Autonomous System Boundary Router
- R5, R6: Internal routers
- R1: Backbone router
Planning for OSPF

- Assess the requirements and options:
  - IP addressing plan
  - Network topology
    - Primary vs. backup links
    - WAN bandwidth utilization

- Define hierarchical network design and areas
  - Evaluate OSPF scaling options
    - Summarization - where necessary
    - Define stub areas
OSPF Functions (multi-area)

- High-level functions of OSPF include the following:
  - Discover neighbours and form adjacencies
  - Flood link-state database (LSDB) information
  - Compute the shortest path
  - Install routes in the route-forwarding table

- Additional functions of OSPF include the following:
  - Detect changes in the link state
  - Propagate changes to maintain link-state database synchronization

- Several OSPF packet types are involved
OSPF Packet Header Format
OSPF Packet Types

- OSPF uses five types of routing protocol packets.

1. Hello
2. Database Description
3. Link-State Request
4. Link-State Update
5. Link-State Acknowledgment
Neighbour Relationship: The Hello Packet

- Entries must match on neighbouring routers
Neighbor Adjacencies: The Hello Packet

- Neighbor routers (A, B, C, D, E)
- Hello packet exchanged
- Hello message includes:
  - Router ID
  - Hello and dead intervals
  - Neighbors
  - Area ID
  - Router priority
  - DR IP address
  - BDR IP address
  - Authentication IP address
  - Stub area flag

* Entry must match on neighboring routers
OSPF Routing Update Packets

- Use of Multicast and unicast IP address
- Four types of update packets
- LSDB synchronization process
  - Discover neighbour
  - Establish bidirectional communication
  - Elect a designated router, if desired
  - Form an adjacency
  - Discover the network routes
  - Update and synchronize link-state databases
Establishing Bidirectional Communication

Dr. Mohd Nazri Mohd Warip
47 November 2013 / EKT 544 Lecture 8

R1
172.16.5.1/24
E0

R2
172.16.5.2/24
E1

R3
172.16.6.0/24

Hello
I am router ID 172.16.5.1, and I see no one.

Init State

Router R2
Neighbors List
172.16.5.1/24, Int E1

to 244.0.0.5

Router R1
Neighbors List
172.16.5.2/24, Int E0

Unicast to R1
I am router ID 172.16.5.2, and I see 172.16.5.1.

Down State

Two-Way State

Cisco

UniMAP
Building the LSDB

- The Hello protocol is used to define neighbours
- Adjacency is established
- Adjacent routers exchange LSAs
- Each router builds an LSDB using LSAs
Discovering the Network Routes

Exstart State

I will start exchange because I have router ID 172.16.5.1

Exchange State

No, I will start exchange because I have a higher router ID.

Here is a summary of my LSDB.

Here is a summary of my LSDB.
Adding the Link-State Entries

1. R1 and R2 exchange LSAck messages.
2. R2 requests a complete entry for network 172.16.6.0/24.
3. R1 sends the full entry for network 172.16.6.0/24.
4. R2 acknowledges the receipt of the full entry.

Transmit (Tx) and Receive (Rx) information is indicated.
Flooding Changes in Topology

- Router R1 that detects a topology change adjusts its LSA and floods the LSA:
  - Router R1 notifies all OSPF neighbours using 224.0.0.5, or, on LAN links, all OSPF DRs and BDRs using 224.0.0.6.
  - The DR notifies others on 224.0.0.5.

- The LSDBs of all routers must be synchronized.
Link-State Data Structures: LSA Operation

1. Is entry in link-state database?
   - Yes: Is sequence no. the same?
     - Yes: Ignore LSA
     - No: Add to database
       - Send LSAck
       - Flood LSA
       - Run SPF to calculate new routing table
       - End
   - No: Go to A

2. Is sequence no. higher?
   - Yes: Send LSU with newer information to source
   - No: End
# LSA Types

<table>
<thead>
<tr>
<th>LSA Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Router LSAs</td>
</tr>
<tr>
<td>2</td>
<td>Network LSAs</td>
</tr>
<tr>
<td>3 or 4</td>
<td>Summary LSAs</td>
</tr>
<tr>
<td>5</td>
<td>Autonomous system external LSAs</td>
</tr>
<tr>
<td>6</td>
<td>Multicast OSPF LSAs</td>
</tr>
<tr>
<td>7</td>
<td>LSAs defined for not-so-stubby areas</td>
</tr>
<tr>
<td>8</td>
<td>External attribute LSAs for Border Gateway Protocol (BGP)</td>
</tr>
<tr>
<td>9, 10, 11</td>
<td>Opaque LSAs</td>
</tr>
</tbody>
</table>
LSA Type 1: Router LSA

- One router LSA for every router in an area (intra-area)
  - Includes a list of directly attached links
  - Links identified by the IP prefix and link type

- LSA identified by the router ID of the originating router.
- Floods within its area only; does not cross an ABR
LSA Type 2: Network LSA

- One network LSA for each transit broadcast or NBMA (Non-Broadcast Multi-Access) network in an area (intra-area)
  - Includes a list of attached routers on the transit link
  - Includes a subnet mask of the link

- Advertised by the DR (Designated Router) of the broadcast network
- Floods within its area only; does not cross an ABR (Area Border Router)
LSA Type 3: Summary LSA

- Used to flood network information to areas outside the originating area (interarea)
  - Describes the network number and mask of the link
- Advertised by the ABR of the originating area, regenerated by all subsequent ABRs to flood throughout the AS
- Advertised for every subnet and not summarized, by default
LSA Type 3: Summary LSA

- Used to flood network information to areas outside the originating area (interarea)
  - Describes the network number and mask of the link
- Advertised by the ABR of the originating area, regenerated by all subsequent ABRs to flood throughout the AS
- Advertised for every subnet and not summarized, by default
LSA Type 4: ASBR Summary LSA

- Used to advertise a metric to the ASBR, which is used for path selection
- Generated by the ABR of the originating area
- Regenerated by all subsequent ABRs to flood throughout the AS
- Contain the router ID of the ASBR (Autonomous Border Router)
LSA Type 5: External LSA

- Used to advertise networks from other ASs
- Advertised and owned by the originating ASBR
- Flooded throughout the entire AS
- The advertising router ID (ASBR) is unchanged throughout the AS
- A type 4 LSA is needed to find the ASBR
- By default, routes are not summarized
LSA Type 7: NSSA External LSA

- Used to advertise networks from other ASs injected into the NSSA (Not-So-Stubby Area).
- Have the same format as a type 5 external LSA.
- Advertised and owned by the originating ASBR.
- Translated to LSA type 5 on first NSSA subsequent ABR.
- By default, routes are not summarized.
OSPF Metric (1)

- The cost, or metric, is an indication of the overhead to send packets over an interface.
- OSPF cost is used as the route selection criteria.
- Dijkstra’s algorithm determines the best path by adding all link costs along a path.
- OSPF cost is computed automatically.
  - Cost = $10^8$ / Bandwidth (in b/s)
    - Bandwidth is specified on the interface with the bandwidth command.
- OSPF cost is recomputed after every bandwidth change.
OSPF Metric (2)

- Also called “cost”
- Defined per interface, but may be altered
- Inversely proportional to the bandwidth of that interface
- \[ \text{COST} = \frac{100,000,000}{\text{bandwidth [b/s]}} \]

<table>
<thead>
<tr>
<th>Link Type</th>
<th>Default Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-kb/s serial link</td>
<td>1562</td>
</tr>
<tr>
<td>T1 (1.544-Mb/s serial link)</td>
<td>64</td>
</tr>
<tr>
<td>E1 (2.048-Mb/s serial link)</td>
<td>48</td>
</tr>
<tr>
<td>Ethernet</td>
<td>10</td>
</tr>
<tr>
<td>Fast Ethernet</td>
<td>1</td>
</tr>
<tr>
<td>ATM</td>
<td>1</td>
</tr>
</tbody>
</table>
OSPF Area Types and Structure

- OSPF is based on a two-level hierarchical area structure
- Each area has its own topology database
- Area Types
  - Backbone area: Connects all other areas
  - Normal area: Contains all of the internal and external routing information
  - Stub area: Contains internal and area routing information, but not external routing information
  - Totally stubby area: Contains area routing information only; Cisco proprietary
  - Not-so-stubby area: Contains area and external routing information
Types of Areas

- **NSSA Area**: Does not accept external LSAs. Allows ASBR.
- **Totally NSSA Area**: Does not accept external or summary LSAs, allows ASBR.
- **Stub Area**: Does not accept external LSAs.
- **Totally Stubby Area**: Does not accept external or summary LSAs.
- **Backbone Area 0**: Accepts link updates, summaries, and external routes.
- **Normal Area**: Does not accept external LSAs.
OSPF Router and LSA Types

- ABR is generating Summary LSAs
- ASBR is generating External LSAs
- Summary and External LSAs can be blocked and default route is sent instead
An area can be stub or totally stub if:

- There is one ABR or more
- All routers that are members of the stub area are configured as stub routers
- There is no ASBR in the area
- The area is not an area 0
- No virtual links go through the area
OSPF Stub Areas

- External LSAs are stopped.
- The default route is advertised into the stub area by the ABR.
OSPF Totally Stubby Areas

- External and Summary LSAs are stopped
- The default route is sent instead
- Cisco proprietary feature
SPF Algorithm

- Places each router at the root of a tree and calculates the shortest path to each destination based on the cumulative cost.
- Cost = Reference Bandwidth / Interface Bandwidth (b/s)
OSPF Calculation

- Routers find the best paths to destinations by applying Dijkstra’s SPF algorithm to the LSDB.
- The best path is calculated based on the lowest total cost and sent to the routing table.
Link-State Routing Protocol Algorithms
References