OSPF (Open Shortest Path First)

- “Open”: Specification publicly available
  - RFC 1247, RFC 2328
  - Working group formed in 1988
  - Goals:
    - Large, heterogeneous internetworks

- Uses the Link State algorithm
  - Topology map at each node
  - Route computation using Dijkstra’s algorithm
Routing tasks: OSPF

- Neighbor?
  - Discovery
  - Maintenance

- Database?
  - Granularity
  - Maintenance – updates
  - Synchronization

- Routing table?
  - Metric
  - Calculation
  - Update
OSPF “advanced” features (not in RIP)

- **Security**: All OSPF messages are authenticated (to prevent malicious intrusion); UDP 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 database as OSPF
- **Hierarchical** OSPF for large domains
OSPFv2: Components

- Hello Protocol: “Who is my neighbor?”
- Designated router/Backup designated router (DR/BDR) election: “With whom I want to talk?”
- Database Synch: “What info am I missing?”
- Reliable flooding alg: “How do I distribute info?”
- Route computation
  - From link state database
  - Using Dijkstra’s algorithm
  - Supporting equal-cost path routing
Neighbor discovery and maintenance

- **Hello Protocol**
  - Ensures that neighbors can send packets to and receive packets from the other side: bi-directional communication
  - Ensures that neighbors agree on parameters (HelloInterval and RouterDeadInterval)

- **How**
  - Hello packet to fixed well-known multicast address
  - Periodic Hellos
  - Broadcast network: Electing designated router
Some multicast addresses

- 224.0.0.5 AllSPFRouters OSPF-ALL.MCAST.NET
- 224.0.0.6 AllDRouters OSPF-DSIG.MCAST.NET
- FF02::5 and FF02::6, respectively for OSPFv3.

- While we are at it:
  - 224.0.0.1 ALL- SYSTEMS. MCAST. NET
  - 224.0.0.2 ALL- ROUTERS. MCAST. NET
  - 224.0.0.9 RIP2- ROUTERS. MCAST. NET
  - 224.0.0.10 IGRP- ROUTERS. MCAST. NET
  - Look up some more (with dig –x address).
Hello Protocol: 3 phases

- **Down**
  - Neighbor is supposed to be “dead”
  - No communication at all

- **Init**
  - “I have heard of a Neighbor”
  - Uni-directional communication

- **ExStart or TwoWay**
  - Communication is bi-directional
Hello Protocol: Packet

- **Hello Interval**: 10 seconds (typical default)
- **RouterDeadInterval**: 4 * Hello Interval (typical default)
OSPF packet

- IP Protocol #89
- Directly to neighbors using Multicast address
  - TTL 1
- Five packet types
  - Hello
  - Database Description
  - Link State Request
  - Link State Update
  - Link State Acknowledgement
Link state database

- Based on link-state technology
  - Local view of topology in a database

- Database
  - Consists of Link State Advertisements (LSA)
  - LSA: Data unit describing local state of a network/router
  - Must kept synchronized to react to routing failures
Example network
## Link state database: Example

<table>
<thead>
<tr>
<th>LS-Type</th>
<th>Link State ID</th>
<th>Adv. Router</th>
<th>Checksum</th>
<th>Seq. No.</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router-LSA</td>
<td>10.1.1.1</td>
<td>10.1.1.1</td>
<td>0x9b47</td>
<td>0x80000006</td>
<td>0</td>
</tr>
<tr>
<td>Router-LSA</td>
<td>10.1.1.2</td>
<td>10.1.1.2</td>
<td>0x219e</td>
<td>0x80000007</td>
<td>1618</td>
</tr>
<tr>
<td>Router-LSA</td>
<td>10.1.1.3</td>
<td>10.1.1.3</td>
<td>0x6b53</td>
<td>0x80000003</td>
<td>1712</td>
</tr>
<tr>
<td>Router-LSA</td>
<td>10.1.1.4</td>
<td>10.1.1.4</td>
<td>0xe39a</td>
<td>0x8000003a</td>
<td>20</td>
</tr>
<tr>
<td>Router-LSA</td>
<td>10.1.1.5</td>
<td>10.1.1.5</td>
<td>0xd2a6</td>
<td>0x80000038</td>
<td>18</td>
</tr>
<tr>
<td>Router-LSA</td>
<td>10.1.1.6</td>
<td>10.1.1.6</td>
<td>0x05c3</td>
<td>0x80000005</td>
<td>1680</td>
</tr>
</tbody>
</table>
**LSAs**

- Consists of a Header and a Body
- Header size is 20 Byte and consists of

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS Age</td>
<td>Options</td>
<td>LS Type</td>
<td></td>
</tr>
<tr>
<td>Link State ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advertising Router</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS Checksum</td>
<td>Length</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LSAs (2.)

- Identifying LSAs
  - LS Type Field
  - Link State ID Field
  - Advertising Router Field

- Verifying LSA Contents
  - LS Checksum Field

- Identifying LSA Instances
  (keeping in mind that the topology changes)
  - LS Sequence Number Field
    - Linear sequence space
    - Max Seq $\Rightarrow$ new instance
LSAs (3.)

- LS Age Field
  (to ensure consistency)
  - Goal: new sequence number every 30 minutes
  - Maximum value 1 hour
  - Age > 1 hour $\Rightarrow$ invalid $\Rightarrow$ removal
  - Enables premature aging
  - Ensures removal of outdated information
# Example LSA: Router-LSA

<table>
<thead>
<tr>
<th>LS Age</th>
<th>Options</th>
<th>LS Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link State ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advertising Router</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS sequence number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS Checksum</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>V</td>
<td>E</td>
</tr>
<tr>
<td>Link ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td># TOS</td>
<td>Metric</td>
</tr>
<tr>
<td>............</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Router LSA

- Link-Cost: Integers (configured)

<table>
<thead>
<tr>
<th>32 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age = 0</th>
<th>Options</th>
<th>Type = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Link State ID = 10.1.1.1
- Advertising Router = 10.1.1.1
- Sequence Number = 0x80000006
- Checksum = 0x9b47
- Length = 60

- Number of Links = 3

- Link ID = 10.1.1.2
- Link Data = Interf. Index 1
- Link Typ = 1
  - # TOS = 0
  - Link-Cost = 3

- Link ID = 10.1.1.3
- Link Data = Interf. Index 2
- Link Typ = 1
  - # TOS = 0
  - Link-Cost = 5

- Link ID = 10.1.1.1
- Link Data = 255.255.255.255
- Link Typ = 3
  - # TOS = 0
  - Link-Cost = 0

Link Typ 1: Peer-to-peer
Link Typ 3: Stub Network
Link state database (2.)

- Is the database synchronized?
  - Same number of LSAs?
  - Sums of LSA LS Checksums are equal?
Database synchronization

- Central aspect: all routers need to have identical databases!

- 2 types of synchronization
  - Initial synchronization
    - After hello
  - Continuous synchronization
    - Flooding
Initial synchronization

- Explicit transfer of the database upon establishment of neighbor ship
- Once bi-directional communication exists
- Send all LS header from database to neighbor
  - OSPF database description packets (DD pkt)
  - Flood all future LSA’s
Initial synchronization (2.)

- Database description (DD) exchange
  - Only one DD at a time
  - Wait for Ack

- Control of DD exchange
  - Determine Master/Slave for DD exchange
  - Determine which LSA’s are missing in own DB
  - Request those via link state request packets
  - Neighbor sends these in link state update packets

- Result:
  - Fully adjacent OSPF neighbors
Example: Database synchronization

OSPF Hello

OSPF Hello: I heard 10.1.1.6

Database Description: Sequence = x

DD: Sequence = x, 5 LSA Headers =
(router-LSA, 10.1.1.1, 0x80000004),
(router-LSA, 10.1.1.2, 0x80000007),
(router-LSA, 10.1.1.3, 0x80000003),
(router-LSA, 10.1.1.4, 0x8000003b),
(router-LSA, 10.1.1.5, 0x80000039),
(router-LSA, 10.1.1.6, 0x80000005)

DD: Sequence = x+1, 1 LSA Header =
(router-LSA, 10.1.1.6, 0x80000001)

~ Router from previous example are synchronized
~ 10.1.1.6 is restarted
Reliable flooding

- 10.1.1.3 sends LS Update
- Same copy of an LSA is an implicit Ack
- Use delayed Ack ‘s
- All LSA ‘s must be acknowledged either implicit or explicit
Robustness of flooding

- More robust than a spanning tree
- LSA refreshes every 30 minutes
- LSAs have checksums
- LSAs are aged
- LSAs cannot be send at arbitrary rate: There are timers
OSPF LSA timers

- MinLSArrival: 1 second
- MinLSInterval: 5 seconds
- CheckAge: 5 minutes
- MaxAgeDiff: 15 minutes
- LSRefreshTime: 30 minutes
- MaxAge: 1 hour
Calculation of routing table

- Link state database is a directed graph with costs for each link
- Dijkstra’s SPF algorithms
  - Add all routers to shortest-path-tree
  - Add all neighbors to candidate list
  - Add routers with the smallest cost to tree
  - Add neighbors of this router to candidate list
    - If not yet on it
    - If cost smaller
  - Continue until candidate list empty
Example

10.1.1.4 (4, 10.1.1.5/2)
10.1.1.1 (5, 10.1.1.1)
10.1.1.6 (11, 10.1.1.5)
10.1.1.6 (10, 10.1.1.5/2)
10.1.1.1 (5, 10.1.1.1/2)
10.1.1.5 (1, 10.1.1.5)
10.1.1.2 (3, 10.1.1.2)
10.1.1.1 (5, 10.1.1.1)
10.1.1.2 (3, 10.1.1.2)
10.1.1.4 (4, 10.1.1.5)
10.1.1.1 (5, 10.1.1.1)
10.1.1.6 (11, 10.1.1.5)
Network types

- So far only point-to-point
- Many other technologies
- Specific requirements for OSPF
  - Neighbor relations
  - Synchronization
  - Representation in DB
- Kinds
  - Point-to-point
  - Broadcast
  - Nonbroadcast multiaccess
  - Point-to-multipoint
Adjacencies on broadcast networks

If \( n \) routers are on a broadcast link, \( n(n-1)/2 \) adjacencies can be formed.
Adjacencies (2.)

- If routers formed pair wise adjacencies:
  - Each would originate \((n-1)+1=n\) LSAs for the link.
  - Out of the network, \(n^2\) LSAs would be emanating.

- Routers also send received LSAs to their neighbors
  - \((n-1)\) copies of each LSA present on the network
  - Even with multicast: \((n-1)\) responses

- Solution: Elect Designated Router (DR)
  - Routers form adjacencies only with DR:
  - Link acts as a (multi-interface) virtual router to the rest of the area
Designated router election

- When router joins:
  - Listen to hellos; if DR and BDR advertised, accept them
    - All Hello packets agree on who the DR and BDR are
    - Status quo is not disturbed
- If there is no elected BDR, router with highest priority becomes BDR
- Ties are broken by highest RouterID
  - RouterIDs are unique (IP address of interface)
- If there is no DR, BDR is promoted to DR
- Elect new BDR
Network LSA ‘s

- A network LSA represents a broadcast subnet
- Router LSA ‘s have links to network LSA
- Reduction of links
- DR responsible for network LSA
- Link State ID = IP-address of DR
OSPF interface state machine
Hierarchical OSPF
Hierarchical OSPF

- **Two-level hierarchy:** local area and backbone.
  - Link-state advertisements do not leave respective areas.
  - Nodes in each area have detailed area topology; they only know direction (shortest path) to networks in other areas.

- **Area Border routers:** “summarize” distances to networks in the area and advertise them to other Area Border routers.

- **Backbone routers:** run an OSPF routing algorithm limited to the backbone.

- **Boundary routers:** connect to other ASs.
Areas

- An AS (or Routing Domain) is divided into areas.
- Group of routers
- “Close” to each other.
- Reduce the extend of LSA flooding
- Intra-area traffic
- Inter-area traffic
- External traffic: Injected from a different AS
- OSPF requires a backbone area (Area 0)
  - Routing between areas only via backbone area
  - Strict area hierarchy (no loops allowed)
Area partitions

- Link and router failures can cause areas to be partitioned
- Some partitions are healed automatically
- Some need manual intervention.
  - Virtual Links.
- Isolated area: Link failure results in no path to the rest of the network
  - Obviously, cannot be healed at all
  - Redundancy is important!
OSPF: Summary

- Neighbors
  - Discovery: Multicast group
  - Maintenance: Hello protocol

- Database
  - Granularity: Link state advertisements (LSA)
  - Maintenance: LSA-updates
  - Synchronization: Synchronization protocol

- Routing table
  - Metric: Fixed values
  - Calculation: Local shortest path calculation