WIKIPEDIA

Link-state routing protocol

Link-state routing protocols are one of the two main classes of routing protocols used in packet switching networks for computer communications, the other being distance-vector routing protocols. Examples of link-state routing protocols include Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS).

The link-state protocol is performed by every *switching node* in the network (i.e., nodes that are prepared to forward packets; in the Internet, these are called routers). The basic concept of link-state routing is that every node constructs a *map* of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical *path* from it to every possible destination in the network. Each collection of best paths will then form each node's routing table.

This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbours, in a link-state protocol the only information passed between nodes is *connectivity related*. Link-state algorithms are sometimes characterized informally as each router, "telling the world about its neighbors."

Contents

History

Distributing maps
- Determining the neighbours of each node
- Distributing the information for the map
- Creating the map
- Notes about this stage

Calculating the routing table
- Calculating the shortest paths
- Filling the routing table

Optimizations to the algorithm
- Partial Recomputation
- Topology Reduction
- Fisheye State Routing

Failure modes

The Optimized Link State Routing Protocol for mobile ad hoc networks

See also

References

Further reading

History

What is believed to be the first adaptive routing network of computers, using link-state routing as its heart, was designed and implemented during 1976-77 by a team from Plessey Radar led by Bernard J Harris; the project was for "Wavell" - a system of computer command and control for the British Army.

The first link-state routing concept was published in 1979 by John M. McQuillan (then at Bolt, Beranek and Newman) as a mechanism that would calculate routes more quickly when network conditions changed, and thus lead to more stable routing.[1][2]

Later work at BBN Technologies showed how to use the link-state technique in a hierarchical system (i.e., one in which the network was divided into areas) so that each switching node does not need a map of the entire network, only the area(s) in which it is included.

The technique was later adapted for use in the contemporary link-state routing protocols IS-IS and OSPF. Cisco literature refers to enhanced interior gateway routing protocol (EIGRP) as a "hybrid" protocol, despite the fact it distributes routing tables instead of topology maps. However, it does synchronize routing tables at start up as OSPF does, and sends specific updates only when topology changes occur.

In 2004, Radia Perlman proposed using link-state routing for layer 2 frame forwarding with devices called routing bridges or Rbridges. The Internet Engineering Task Force has standardized the transparent interconnection of lots of links (TRILL) protocol to accomplish this.[3]

More recently, this hierarchical technique was applied to wireless mesh networks using the optimized link state routing protocol (OLSR). Where a connection can have varying quality, the quality of a connection can be used to select better connections. This is used in some routing protocols that use radio frequency transmission.

In 2012, the IEEE completed and approved the standardization of the use of IS-IS to control Ethernet forwarding with IEEE 802.1aq shortest path bridging (SPB).

Distributing maps

This description covers only the simplest configuration; i.e., one with no areas, so that all nodes have a map of the entire network. The hierarchical case is somewhat more complex; see the various protocol specifications.

As previously mentioned, the first main stage in the link-state algorithm is to give a map of the network to every node. This is done with several subsidiary steps.

Determining the neighbours of each node

First, each node needs to determine what other ports it is connected to, over fully working links; it does this using *reachability protocol* it runs periodically and separately with each of its directly connected neighbours.

Distributing the information for the map

Next, each node periodically (and in case of connectivity changes) sends a short message, the link-state advertisement, which:

- Identifies the node which is producing it.
- Identifies all the other nodes (either routers or networks) to which it is directly connected.
- Includes a 'sequence number', which increases every time the source node makes up a new version of the message.

This message is sent to all the nodes on a network. As a necessary precursor, each node in the network remembers, for every one of *its* neighbors, the sequence number of the last link-state message which it received from that node. When a link-state advertisement is received at a node, the node looks up the sequence number it has stored for the source of that link-state message: if this message is newer (i.e., has a higher sequence number), it is saved, the sequence number is updated, and a copy is sent in turn to each of that node's neighbors. This procedure rapidly gets a copy of the latest version of each node's link-state advertisement to every node in the network.

Networks running link state algorithms can also be segmented into hierarchies which limit the scope of route changes. These features mean that link state algorithms scale better to larger networks.

Creating the map

Finally, with the complete set of link-state advertisements (one from each node in the network) in hand, each node produces the graph for the map of the network. The algorithm iterates over the collection of link-state advertisements; for each one, it makes links on the map of the network, from the node which sent that message, to all the nodes which that message indicates are neighbors of the sending node.

No link is considered to have been correctly reported unless the two ends agree; i.e., if one node reports that it is connected to another, but the other node does not report that it is connected to the first, there is a problem, and the link is not included on the map.

Notes about this stage

The link-state message giving information about the neighbors is recomputed, and then flooded throughout the network, whenever there is a change in the connectivity between the node and its neighbors; e.g., when a link fails. Any such change will be detected by the reachability protocol which each node runs with its neighbors.

Calculating the routing table

As initially mentioned, the second main stage in the link-state algorithm is to produce routing tables, by inspecting the maps. This is again done with several steps.

Calculating the shortest paths

Each node independently runs an algorithm over the map to determine the shortest path from itself to every other node in the network; generally some variant of Dijkstra's algorithm is used. This is based around a link cost across each path which includes available bandwidth among other things.

A node maintains two data structures: a tree containing nodes which are "done", and a list of *candidates*. The algorithm starts with both structures empty; it then adds to the first one the node itself. The variant of a Greedy Algorithm then repetitively does the following:

- All neighbour nodes which are directly connected to the node are just added to the tree (excepting any nodes which are already in either the tree or the candidate list). The rest are added to the second (candidate) list.
- Each node in the candidate list is compared to each of the nodes already in the tree. The candidate node which is closest to any of the nodes already in the tree is itself moved into the tree and attached to the appropriate neighbor node. When a node is moved from the candidate

list into the tree, it is removed from the candidate list and is not considered in subsequent iterations of the algorithm.

The above two steps are repeated as long as there are any nodes left in the candidate list. (When there are none, all the nodes in the network will have been added to the tree.) This procedure ends with the tree containing all the nodes in the network, with the node on which the algorithm is running as the *root* of the tree. The shortest path from that node to any other node is indicated by the list of nodes one traverses to get from the root of the tree, to the desired node in the tree..!

Filling the routing table

With the shortest paths in hand, the next step is to fill in the routing table. For any given destination node, the best path for that destination is the node which is the first step from the root node, down the branch in the shortest-path tree which leads toward the desired destination node. To create the routing table, it is only necessary to walk the tree, remembering the identity of the node at the head of each branch, and filling in the routing table entry for each node one comes across with that identity.

Optimizations to the algorithm

The algorithm described above was made as simple as possible, to aid in ease of understanding. In practice, there are a number of optimizations which are used.

Partial Recomputation

Whenever a change in the connectivity map happens, it is necessary to recompute the shortest-path tree, and then recreate the routing table. Work by BBN Technologies discovered how to recompute only that part of the tree which could have been affected by a given change in the map. Also, the routing table would normally be filled in as the shortest-path tree is computed, instead of making it a separate operation.

Topology Reduction

In some cases it is reasonable to reduce the number of nodes that generate LSA messages. For instance, a node that has only one connection to the network graph does not need to send LSA messages, as the information on its existence could be already included in the LSA message of its only neighbor. For this reason a topology reduction strategy can be applied, in which only a subset of the network nodes generate LSA messages. Two widely studied approaches for topology reduction are:

1. MultiPoint Relays that are at the base of the OLSR protocol but have also been proposed for OSPF[4]
2. Connected Dominating Sets that again have been proposed for OSPF[5]

Fisheye State Routing

With FSR the LSA are sent with different TTL values in order to restrict their diffusion and limit the overhead due to control messages. The same concept is used also in the Hazy Sighted Link State Routing Protocol.

Failure modes

If all the nodes are not working from exactly the same map, *routing loops* can form. These are situations in which, in the simplest form, two neighboring nodes each think the other is the best path to a given destination. Any packet headed to that destination arriving at either node will loop between the two, hence the name. Routing loops involving more than two nodes are also possible.

This can occur since each node computes its shortest-path tree and its routing table without interacting in any way with any other nodes. If two nodes start with different maps, it is possible to have scenarios in which routing loops are created. In certain circumstances, differential loops may be enabled within a multi cloud environment. Variable access nodes across the interface protocol may also bypass the simultaneous access node problem.[6]

The Optimized Link State Routing Protocol for mobile ad hoc networks

The *Optimized Link State Routing Protocol* (OLSR) is a link-state routing protocol optimized for mobile ad hoc networks (which can also be used on other wireless ad hoc networks).[7] OLSR is proactive, it uses Hello and Topology Control (TC) messages to discover and disseminate link state information into the mobile ad hoc network. Using Hello messages each node discovers 2-hop neighbor information and elects a set of *multipoint relays* (MPRs). MPRs makes OLSR unique from other link state routing protocols. Individual nodes use the topology information to compute next hop paths regard to all nodes in the network using shortest hop forwarding paths.

See also

- 802.1aq Shortest Path Bridging

References

1. John M. McQuillan, Isaac Richer and Eric C. Rosen, *ARPANet Routing Algorithm Improvements*, BBN Report No. 3803, Cambridge, April 1978
2. John M. McQuillan, Isaac Richer and Eric C. Rosen, *The New Routing Algorithm for the ARPANet*, IEEE Trans. on Comm., 28(5), pp. 711–719, 1980
3. *Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS*, RFC 7176 (https://tools.ietf.org/html/rfc7176)
4. https://tools.ietf.org/html/rfc5449
5. https://tools.ietf.org/html/rfc5614
6. Wójcik, R (2016). "A survey on methods to provide interdomain multipath transmissions". *Computer Networks*. **108**.
7. RFC 3626

- Josh Seeger and Atul Khanna, *Reducing Routing Overhead in a Growing DDN*, MILCOMM '86, IEEE, 1986
- Radia Perlman “Rbridges: Transparent Routing” (http://www.ieee-infocom.org/2004/Papers/26_1.PDF), Infocom 2004.

Further reading

- Section "Link-State Versus Distance Vector" (http://docwiki.cisco.com/wiki/Routing_Basics#Link-State_Versus_Distance_Vector) in the Chapter "Routing Basics" in the Cisco

"Internetworking Technology Handbook"

Retrieved from "https://en.wikipedia.org/w/index.php?title=Link-state_routing_protocol&oldid=1025105993"