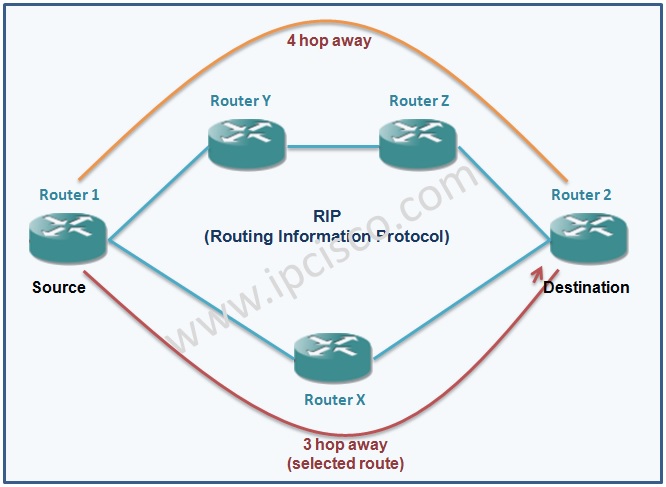
**Routing Information Protocol**

****

The **Routing Information Protocol** (**RIP**) is one of the oldest [distance-vector routing protocols](https://en.wikipedia.org/wiki/Distance-vector_routing_protocols) which employ the [hop count](https://en.wikipedia.org/wiki/Hopcount) as a [routing metric](https://en.wikipedia.org/wiki/Metrics_(networking)). RIP prevents [routing loops](https://en.wikipedia.org/wiki/Routing_loop_problem) by implementing a limit on the number of [hops](https://en.wikipedia.org/wiki/Hop_(telecommunications)) allowed in a path from source to destination. The largest number of hops allowed for RIP is 15, which limits the size of networks that RIP can support.

RIP implements the [split horizon](https://en.wikipedia.org/wiki/Split_horizon), [route poisoning](https://en.wikipedia.org/wiki/Route_poisoning) and [hold down](https://en.wikipedia.org/wiki/Holddown) mechanisms to prevent incorrect routing information from being propagated.

In RIPv1 router broadcast updates with their routing table every 30 seconds. In the early deployments, [routing tables](https://en.wikipedia.org/wiki/Routing_table) were small enough that the traffic was not significant. As networks grew in size, however, it became evident there could be a massive traffic burst every 30 seconds, even if the routers had been initialized at random times.

In most networking environments, RIP is not the preferred choice for [routing](https://en.wikipedia.org/wiki/Routing_protocol) as its [time to converge](https://en.wikipedia.org/wiki/Convergence_(routing)#Convergence_time) and [scalability](https://en.wikipedia.org/wiki/Scale_(computing)) are poor compared to [EIGRP](https://en.wikipedia.org/wiki/Enhanced_Interior_Gateway_Routing_Protocol), [OSPF](https://en.wikipedia.org/wiki/Open_Shortest_Path_First), or [IS-IS](https://en.wikipedia.org/wiki/IS-IS). However, it is easy to configure, because RIP does not require any parameters unlike other protocols.

RIP uses the [User Datagram Protocol](https://en.wikipedia.org/wiki/User_Datagram_Protocol) (UDP) as its transport protocol, and is assigned the reserved [port number](https://en.wikipedia.org/wiki/Port_number) 520.

# Distance vector routing

Distance vector routing is a simple distributed routing protocol. Distance vector routing allows routers to automatically discover the destinations reachable inside the network as well as the shortest path to reach each of these destinations. The shortest path is computed based on metrics or costs that are associated to each link. We use l. cost to represent the metric that has been configured for link l on a router.

Each router maintains a routing table. The routing table R can be modeled as a data structure that stores, for each known destination address d, the following attributes:

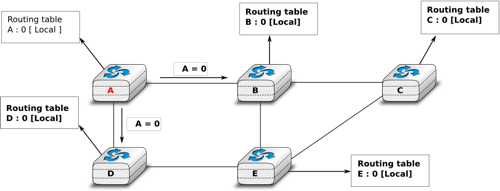
* R[d].link is the outgoing link that the router uses to forward packets towards destination d
* R[d].cost is the sum of the metrics of the links that compose the shortest path to reach destination d
* R[d].time is the timestamp of the last distance vector containing destination d

The router iterates over all addresses included in the distance vector. If the distance vector contains an address that the router does not know, it inserts the destination inside its routing table via link l and at a distance which is the sum between the distance indicated in the distance vector and the cost associated to link l. If the destination was already known by the router, it only updates the corresponding entry in its routing table if either:

* the cost of the new route is smaller than the cost of the already known route ( (V[d].cost+ l. cost) < R[d].cost)
* the new route was learned over the same link as the current best route towards this destination ( R[d].link == l)

The first condition ensures that the router discovers the shortest path towards each destination. The second condition is used to take into account the changes of routes that may occur after a link failure or a change of the metric associated to a link.

To understand the operation of a distance vector protocol, let us consider the network of five routers shown below.



# Link state routing

Link state routing is the second family of routing protocols. While distance vector routers use a distributed algorithm to compute their routing tables, link-state routers exchange messages to allow each router to learn the entire network topology. Based on this learned topology, each router is then able to compute its routing table by using a shortest path computation [[Dijkstra1959]](http://cnp3book.info.ucl.ac.be/bibliography.html#dijkstra1959).

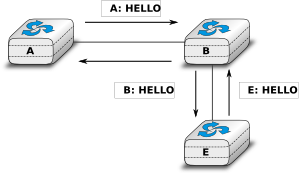
For link-state routing, a network is modeled as a *directed weighted graph*. Each router is a node, and the links between routers are the edges in the graph. A positive weight is associated to each directed edge and routers use the shortest path to reach each destination. In practice, different types of weight can be associated to each directed edge :

* unit weight. If all links have a unit weight, shortest path routing prefers the paths with the least number of intermediate routers.
* weight proportional to the propagation delay on the link. If all link weights are configured this way, shortest path routing uses the paths with the smallest propagation delay.
* weight=\frac{C}{bandwidth}where *C* is a constant larger than the highest link bandwidth in the network. If all link weights are configured this way, shortest path routing prefers higher bandwidth paths over lower bandwidth paths

Usually, the same weight is associated to the two directed edges that correspond to a physical link (i.e. R1 \rightarrow R2and R2 \rightarrow R1). However, nothing in the link state protocols requires this. For example, if the weight is set in function of the link bandwidth, then an asymmetric ADSL link could have a different weight for the upstream and downstream directions. Other variants are possible. Some networks use optimization algorithms to find the best set of weights to minimize congestion inside the network for a given traffic demand [[FRT2002]](http://cnp3book.info.ucl.ac.be/bibliography.html#frt2002).

When a link-state router boots, it first needs to discover to which routers it is directly connected. For this, each router sends a HELLO message every *N* seconds on all of its interfaces. This message contains the router’s address. Each router has a unique address. As its neighboring routers also send HELLO messages, the router automatically discovers to which neighbors it is connected. These HELLO messages are only sent to neighbors who are directly connected to a router, and a router never forwards the HELLO messages that they receive. HELLO messages are also used to detect link and router failures. A link is considered to have failed if no HELLO message has been received from the neighboring router for a period of k \times Nseconds.

The exchange of HELLO messages



Once a router has discovered its neighbors, it must reliably distribute its local links to all routers in the network to allow them to compute their local view of the network topology. For this, each router builds a *link-state packet* (LSP) containing the following information:

* LSP .Router : identification (address) of the sender of the LSP
* LSP .age : age or remaining lifetime of the LSP
* LSP .seq : sequence number of the LSP
* LSP .Links[] : links advertised in the LSP. Each directed link is represented with the following information : - LSP .Links[i].Id : identification of the neighbor – LSP .Links[i].cost : cost of the link