

Routing Algorithms

The main function of the network layer is routing packets from the source machine to the destination machine. In most subnets, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network. The algorithms that choose the routes and the data structures that they use are a major area of network layer design. The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously-established route. The latter case is sometimes called session routing because a route remains in force for an entire user session (e.g., a login session at a terminal or a file transfer). It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is forwarding. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Regardless of whether routes are chosen independently for each packet or only when new connections are established, certain properties are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness, and optimality. Correctness and simplicity hardly require comment, but the need for robustness may be less obvious at first. Once a major network comes on the air, it may be expected to run continuously for years without systemwide failures. During that period there will be hardware and software failures of all kinds. Hosts, routers, and lines will fail repeatedly, and the topology will change many times. The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network to be rebooted every time some router crashes. Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to equilibrium, no matter how long they run. A stable algorithm reaches equilibrium and stays there. Fairness and optimality may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals.

Conflict between fairness and optimality. Before we can even attempt to find trade-offs between fairness and optimality, we must decide what it is we seek to optimize. Minimizing mean packet delay is an obvious candidate, but so is maximizing total network throughput. Furthermore, these two goals are also in conflict, since operating any queueing system near capacity implies a long queueing delay. As a compromise, many networks attempt to minimize the number of hops a packet must make, because reducing the number of hops tends to improve the delay and also reduce the amount of bandwidth consumed, which tends to improve the throughput as well.

Routing algorithms can be grouped into two major classes: nonadaptive and adaptive. Nonadaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J (for all I and J) is computed in advance, off-line, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get their

information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ΔT sec, when the load changes or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

Shortest path routing:

It is one of the simple static routing algorithms that are widely used for routing in the network. The basic idea of it is to build a graph with each node representing a router and each line representing a communication link. To choose a route between any two nodes in the graph the algorithm simply finds the shortest path between the nodes.

Shortest Path means that the path in which anyone or more metrics is minimized. The metric may be distance, bandwidth, average traffic, communication cost, mean queue length, measured delay or any other factor, Dijkstra Algorithm is used mostly for this purpose. The basic objective of the algorithm is to label each node with its distance from the source node along the best-known path.

Steps of Shortest Path Routing Algorithm:

1. Initially, no path is known – All nodes are labelled with infinity.
2. As the algorithm progresses labels are changed reflecting the better path. The label is called as tentative label.
3. As it is found that the label represents the shortest path then it is made permanent and never changed thereafter.