**Routing** is process of discovering the network and finding a way to reach resources. Main Components of decision where are we going to send the packet are name, address. Address is usually assigned to reflect locality in the network. They can be assigned statically where it is hard to keep up with network evolution, and dynamic where that part is ok, but hosts and applications have to cope with continuous changes in their network addresses. Routing is often mixed with forwarding. Routing is the step previous to forwarding which prepares the information used for forwarding. Forwarding is the simple act of executing routing instructions.

**Proactive** routing means that the information required for forwarding is prepared before any communication need. Routing paths are (re-)computed every time the topology or node association’s change, maintaining information to all possible destinations (overhead increase with number of nodes). It requires periodic exchange of routing information among peer nodes. It is better for frequent communications with multiple nodes.  
**Reactive** routing: No routing information is computed before the communication and there is no prior forwarding information. Routing paths are computed when a communication is requested. Maintain information only to destinations of active communications (overhead increase with number of communications). It is better for topologies that change very often.

Routing can be based on address, and name. Address is location, so we use it in routing and the forwarding. It can be used with name that first needs to be resolved (today via DNS) into address. Name based routing allows for more flexible types of routing. It has much simpler and intuitive support for mobility where we can send a message to someone no matter the location. Without specific location routing procedure is much more complicate.

Routing proceeds in 2 steps: first step is the path discovery and second step is pushing the routing information towards forwarding.

For the discovery step, there are generally 3 different families of protocols: Path-vector protocols, Link-state protocols and Distance-vector protocols.

**Path-vector** simply has discovery packets that remember where they are and they were from the start. So if we need route to some resource, we can find it in the path vector. To avoid path vector message looping forever, before forwarding to neighbor, we check if he is already in the path of the vector. Good thing is that node learns route to all the nodes in path vector from the single message.

In Distance-vector protocol each node receives distance information from its neighbors to different nodes in the network. At the beginning node knows only distance to its neighbors. Then it learns distances that its neighbors know. If he gets conflicting distances, he will chose one that is smaller. Protocol converges when there is no updated information to be exchanged. Selecting the next hop to a destination is simple. Node picks the neighbor that is closest to the destination. Problem with distance vectors is anomaly called count to infinity. It happens when… example from the picture. 3 nodes **V**<-2->**U**<-1->**W**<-1->**V** change to **V**<-8->**U**<-1->**W**<-10->**V.** First W notices change and updates table with old distance via U to V that is 3(1+2). Then U notices the mistake and updates distance via W to V 4(1+3). And they are stuck in infinite loop. Solution is that W sends infinity for distance to v to neighbor u. This solution is called poison reverse.

**Link-state** protocols: Each router monitors the state of its direct neighborhood and if there is a topology change broadcasts this information to entire network. This way each router after some time gets the information on neighbors of all the other routers in the network. With that each router knows the topology. Then it can compute the paths using a shortest path algorithm. **Dijkstra's SPF** algorithm has E: evaluated nodes, R: remaining nodes, O: ordered list of paths and SP: shortest path. In every iteration starting form first node, it finds shortest link from O and adds it to SP. After that it adds node on the other side of that link to E, and jumps to next iteration. It gets to the end if all the nodes are in E.

Comparison:

- Link state operates based on global topology knowledge while DV and PV have a partial view of the network.

- Link state and path-vector methods allow seeing the links and in distance vector we have distances between nodes.

-distance vector sometimes can’t see all the links in the graph if they are not preferred. That is bad because it is always nice to have alternative.

After discovery phase we have routing information base with all the routs that we need to transmit to forwarding. That can be done in 2 ways: Source routing and Forwarding Information Base.

**Source routing** functions by storing the route in packet itself. That way each node on the way to the destination knows where to forward the packet. Best thing this approach is that source has full control on of path taken by its packets. That means that only the source needs to perform routing, other nodes simply do forwarding.

**Forwarding Information Base** has path information to all the nodes in the network. It is maintained on each node. In this approach there is no need to have all the steps of the route in table. Only next hop on the path is sufficient. Problem that this approach creates is that such table must be very large for storing next hop for every node. Searching through such table would also be very slow. We can optimize tables using **aggregation of entries**. Fields that can be aggregated need to share the same next hop. This means that addresses that are similar to each other should belong to nodes that are close to each other in the network. Aggregation is not possible if addresses do not reflect the positions in the network. This problem exists in current Internet and is known as the IP address fragmentation problem.

**/////////////////////////////**

**Routing Information Protocol (RIP)** is address based distance-vector protocol. It is proactive, so discovery of routs works independently of current need. It is one of the oldest protocols out there.

**Open Shortest Path First Protocol (OSPF)** is link-state protocol. It uses Link Database Exchange protocol to ensure reliable transfer of the link states over HELLO protocol and Topology change Flooding protocol to broadcast when something is changed in the network. HELLO protocol is also used for leader election to choose designated router and it’s backup. Goal of this is to reduce number of broadcasted messages by restricting communication to talking only with the designated router.

**Optimized Link-State Routing Protocol**

//////////////////////////////

Greedy Routing tries to find out, how would forwarding-function work if the addresses correctly reflected network proximity. If you want to go from point A to B you need to evaluate distance your neighbors from your destination. At every step you check what is the best way to the destination.

We could use **GPS coordinates** but nodes in the same geographic region a do not always have similar network connectivity. For example: 2 neighbors in the same building can be connected to 2 different service providers, or in wireless networks, 2 nodes at the same location that cannot communicate because of an obstacle.

When assigned correctly (without fragmentation), **hierarchical addresses** do reflect a notion of locality in the graph and could be good for greedy routing, but most of the network graphs are not trees. 2 nodes can be close to each other in the network, but their addresses do not need to indicate that.

We can use **Landmark coordinates**. Some nodes in the network are considered landmarks, and all the other nodes calculate their distances in relation to them. Coordinates of a node is its distance vector to the landmarks. Bad thing is that success strongly depends on the placement of the landmarks and that number of landmarks grows with the network.