**Graph Theory terminology**:

-nodes, edges (undirected, directed, and weighted (capacity, delay, cost …))

Degree is number of edges coming out of one node. D-regular graph is the graph where all nodes have degree D. Graphs can be fully …or partially connected.

Graph can be presented as matrix, where one cell of matrix is link between nodes crossing vertically and horizontally in matrix.

For some set of nodes and edges between them we want to set up network topology. We design topology to maximize reliability, flow and functional aspect.

First we want to **maximize utility**, meaning that we want to have maximal flow (amount of information that can be transferred) in the network for every set of nodes. Flow is feasible if it is smaller or equal than capacity between nodes. Flow conservation rule states that there can’t be change of amount of traffic in the whole graph. Cut capacity is the sum of weights of the directed edges from partition S to partition T. Minimum cut is the graph cut with the minimum capacity.

**Max-Flow Min-Cut:** The minimum cut determines the bottleneck in the flow of information between s and t, so it is an upper bound for the maximum feasible flow. There are number of algorithms that compute maximal flow of the graph. One of them is Ford-Fulkerson algorithm.

**Ford-Fulkerson** algorithm first takes some feasible flow of the graph and tries to find out is it possible to increase it. First we take some path in the graph and from that we make residual graph where firs switch the direction of links taken in the path, and then we make complementing links to the full capacity of each link (we make links from T to S such that every link has flow = (capacity-old taken flow) and old links just switch direction). Then we again try to find a path from S to T now at residual graph. If there is such path we take minimal link flow from it and add it to the links on that path that have same direction and subtract it from the links on that path with opposite direction. We then find minimal cut for new graph and we keep repeating the same algorithm with new flow until the flow stops to rise.

Second goal in of topology design is to **maximize reliability**. Point is to have potential loss of nodes and edges in the graph, without loss of connectivity. For edge connectivity we can use max-flow-min cut approach when we reduce weight of links to one. That way min cut will show the number of links that we need to remove for topology partition. Edge connectivity is can be larger than node connectivity but it can’t be other way around. We can see that when we remove some link and do not have to remove nodes, but if we remove node links coming out of it must be removed. To reduce our solution to node connectivity we have simple algorithm. First, for every node in the graph add a second node and connect them with an edge. Then move all outgoing edges from old node to new node, leaving the incoming edges intact. Then set a weight of 1 on all edges of the graph. Finally apply Max-Flow Min-Cut to find the number of node disjoint paths.

With adding reliability, we create some operational problems that most of the time comes from **loops in the topology**. When sending some message, if we have two paths from one node to another, we have duplicate traffic. Second problem is broadcast storm where after we do a broadcast, for example to find some node, we can get infinite loop that at each side produces duplicate traffic. Another problem is inconsistent state, where one switch at start of the loop learns location of node sending him the message from one side, and after full circle in the loop it learns about the same node from the other side. These problems are simply solved by choosing only one path, and leaving rest of them as backups.

For choosing right path in network dynamically, we have many algorithms. Their main goal is to maximize throughput, minimize delay, remove loops, and probably more. For this they try to make minimal spanning tree (tree that involves all the nodes in the graph and has minimum sum of weights). We have 3 such algorithms in broad use today: Prim ’57 (Also Jarník ’30, Dijkstra ’59), Kruskal ’56 and Perlman ’85 that is most dominant one.

**PJD** First we pick any nod as a starting one and color it. Then we choose closest neighbor, and color it and the link from the first node. We continue finding all the nodes that are not colored over the shortest path to our colored sub-graph and color them and links to them until we have all the nodes colored. Colored sub-graph is at the end minimal spanning tree.

**Kruskal’s** at the start finds cheapest link, and colors it. Then it looks for the cheapest uncolored link in the graph that doesn’t close a colored circuit and color it. This step is repeated until all until all nodes are connected or until we have the same number of colored edges and nodes in the graph. The colored edges form a minimum spanning tree.

**Perlman’s** Spanning Tree Protocol is informational only.

Finding the minimal spanning tree is not a solution for wireless networks where reliability and connectivnes are more important than maximizing throughput. For solving this we use algorithm to find tree with minimal connected dominating set (nodes in spanning tree that are not leaves).

**Multi Point Relay** protocol: Initially every node learns their 2-hop neighborhood, through broadcasting. Then every node creates MPR list with the minimum set of nodes that allow it to see his 2-hop neighborhood, and advertise it.

**Multi Protocol Label Switching** is a mechanism for maintaining the connectivity for a network topology that is used today. Its functionality comes down to adding a label in the traffic between set of nodes so that they can be identified as belonging to certain communication group. It is much faster than IP address longest-prefix match. Ingress router puts a label on the packet, and forwards it. Each node on the path to the Egress router knows what to do if certain label is on the packet so it just forwards it without checking IP address. Distribution of labels is done against the flow of packets. When some router requests the path, packet first gets to the destination and in opposite direction it marks the labels. **802.1Q tunneling** is based on the same principle, but instead of routers we use Ethernet switches.