CS3200: Computer Networks

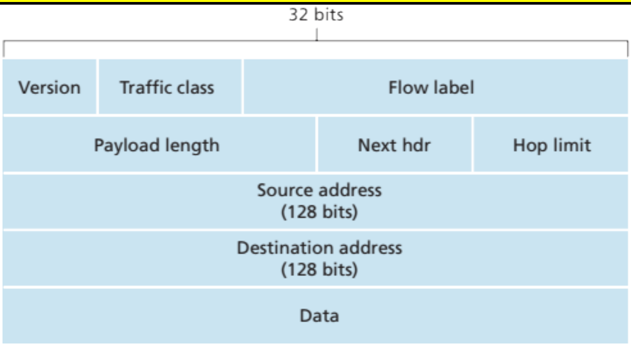
Lecture 17

IIT Palakkad

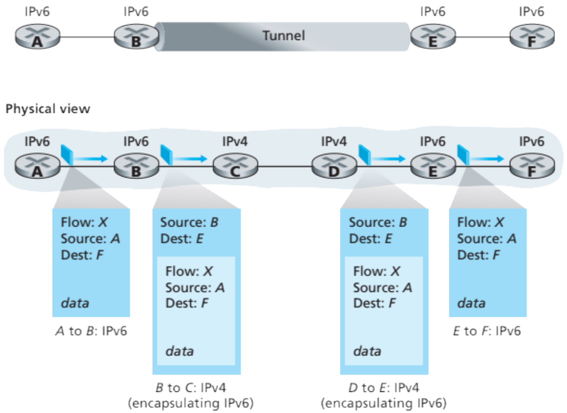
16 Sep, 2019

IPv6 Frame Structure

Here header length is fixed. Traffic tells whether it is UDP, etc. Flow label tells what flow it is. Next hdr: Ignore v4 - v6 tunnelling, everything of v6 to be put inside



Transitioning from IPv4 to IPv6



Routing Algorithms

- Main function of the network layer is routing packets from the source machine to the destination machine.
- Packets may require multiple hops to make the journey.
- Algorithms that choose the routes and the data structures that they use are a major area of network layer design.

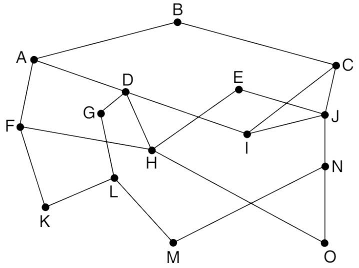
The Optimality Principle

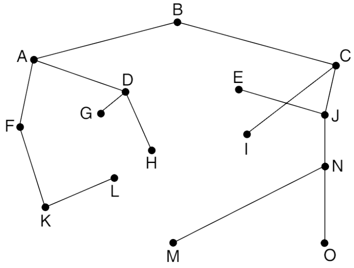
U = set of all vertices except B. Take any vertex from U, find optimal path to B, the remove

The Optimality Principle (Bellman 1957)

If router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route.

As a direct consequence of the optimality principle, the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a sink tree





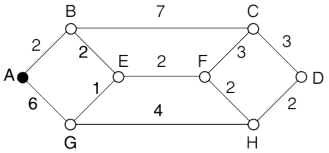
Shortest Path Algorithm

- Computing optimal paths given a complete picture of the network.
- Build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link.
- Each edge is assigned a weight which could be a function of the distance, bandwidth, average traffic, communication cost etc.
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

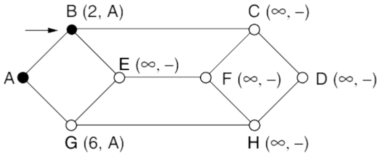
Dijkstra's Algorithm

- Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
- Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
- For the current node, consider all of its unvisited neighbours and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one.
- When we are done considering all of the unvisited neighbours of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
- Otherwise, select the unvisited node that with the smallest tentative distance, set it as the new "current node", and go back to step 3.

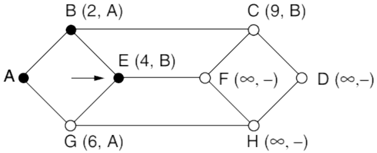
Dijkstra's Algorithm



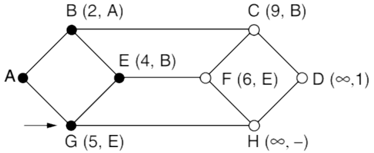
(a)



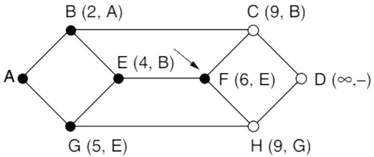
(b)



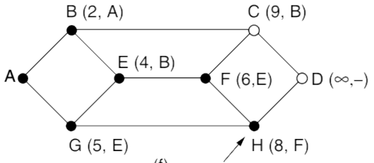
(c)



(d)



(e)



(f)

Dijkstra's Algorithm

- Consider the point where we have just made E permanent.
- Suppose that there were a shorter path than ABE, say $AXYZE$ (for some X and Y).
- If node Z has already been made permanent, then E has already been probed, so the $AXYZE$ path has not escaped our attention and thus cannot be a shorter path.
- Otherwise, if the weight at Z is greater than or equal to that at E, then $AXYZE$ cannot be a shorter path than ABE.
- If the weight is less than that of E, then Z and not E will become permanent first, allowing E to be probed from Z.

Time Complexity of Dijkstra's Algorithm

- Dijkstra's algorithm uses a data structure for storing and querying partial solutions sorted by distance from the start.
- The original algorithm uses a min-priority queue and runs in time $O(|V|^2)$ (where $|V|$ is the number of nodes).
- Fredman and Tarjan 1984 proposed using a Fibonacci heap min-priority queue to optimize the running time complexity to $O(|E| + |V| \log |V|)$, where $|E|$ is the number of edges). This is asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded non-negative weights.