Computer Networks

BCST -502 BCSP- 502

B.Tech (CSE) 5th Semester

Course Instructor: Dr Bishwajeet Pandey

New 2020 Syllabus

Unit –I

Computer Network: Definitions, goals, components, Architecture, Classifications & Types.Layered Architecture: Protocol hierarchy, Design Issues, Interfaces and Services, ConnectionOriented & Connectionless Services, Service primitives, Design issues & its functionality. ISOOSI Reference Model: Principle, Model, Descriptions of various layers and its comparison with TCP/IP. Principals of physical layer: Media, Bandwidth, Data rate and Modulations

Unit-II

Data Link Layer: Need, Services Provided, Framing, Flow Control, Error control. Data Link Layer Protocol: Elementary &Sliding Window protocol: 1-bit, Go-Back-N, Selective Repeat, Hybrid ARQ. Protocol verification: Finite State Machine Models & Petri net models. ARP/RARP/GARP

Unit-III

MAC Sub layer: MAC Addressing, Binary Exponential Back-off (BEB) Algorithm, Distributed Random Access Schemes/Contention Schemes: for Data Services (ALOHA and Slotted- ALOHA), for Local-Area Networks (CSMA, CSMA/CD, CSMA/CA), CollisionFree Protocols: Basic Bit Map, BRAP, Binary Count Down, MLMA Limited Contention Protocols: Adaptive Tree Walk, Performance Measuring Metrics. IEEE Standards 802 series & their variant.

BIRLA INSTITUTE OF APPLIED SCIENCES
BIAS

New 2020 Syllabus

Unit-IV

Network Layer: Need, Services Provided, Design issues, Routing algorithms: Least CostRouting algorithm, Dijkstra's algorithm, Bellman-ford algorithm, Hierarchical Routing,Broadcast Routing, Multicast Routing. IP Addresses, Header format, Packet forwarding,Fragmentation and reassembly, ICMP, Comparative study of IPv4 & IPv6

Unit-V

Transport Layer: Design Issues, UDP: Header Format, Per-Segment Checksum, CarryingUnicast/Multicast Real-Time Traffic, TCP: Connection Management, Reliability of DataTransfers, TCP Flow Control, TCP Congestion Control, TCP Header Format, TCP TimerManagement. Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).

About Course Instructor

- PhD from Gran Sasso Science Institute, Italy
- PhD Supervisor Prof Paolo Prinetto from Politecnico Di Torino, World Rank 13 in Electrical Engineering
- MTech from Indian Institute of Information Technology, Gwalior
- Scopus Profile: https://www.scopus.com/authid/detail.uri?authorId=57203239026
- Google Scholar: https://scholar.google.com/citations?user=UZ_8yAMAAAAJ&hl=hi
- Contact: gyancity@gyancity.com, +91-7428640820 (For help in this Subject @ BIAS and Guidance for future MS from Europe and USA after BIAS)

About Course Outline

- UNIT 1: Lecture No 1-4
- UNIT 2: Lecture No 5-11 (Including Lab on Vivado)
- UNIT 3: Lecture No 14-18
- UNIT 4: Lecture No 19-21, Lecture 12-13
- UNIT 5: Lecture No 22-28 (Including Lab on Packet Tracer)
- Lecture No 29-35: Discuss Previous Year Question of UKTU
- Out of 35 Lectures: Some will delivered by Professor From Foreign University

OUTLINE OF LECTURE 19

- Need of Network Layers
- Service Provided by Network Layer
- Design Issues in Network Layer

OUTLINE OF LECTURE 20

- **Routing Protocols**
  - Least Cost Routing algorithm
    - Dijkstra's algorithm
    - Bellman-ford algorithm

Least Cost Routing algorithm

- Given a network of nodes connected by bidirectional links, where each link has a **cost** associated with it in each direction.

- Define the **cost** of a path between two nodes as the sum of the **costs** of the links traversed.
- For each pair of nodes, find a path with the **least cost**

Least Cost Routing Algorithm

- Several least-cost-path algorithms have been developed for packet-switched networks.

- In particular, Dijkstra's algorithm and the Bellman-Ford algorithm are the most effective and widely used algorithms.

Dijkstra's algorithm

- Dijkstra's algorithm is a centralized routing algorithm that maintains information in a central location.

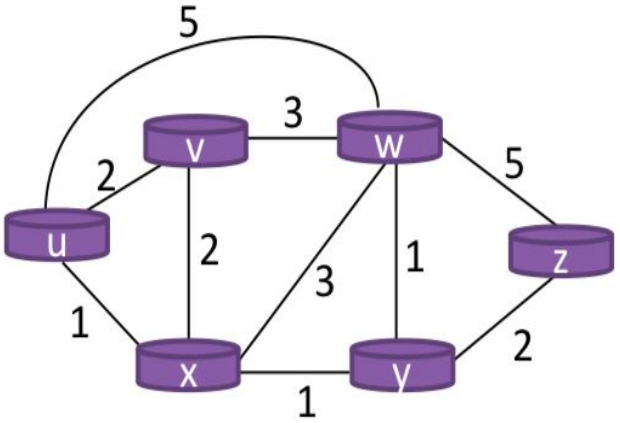
- This algorithm determines least-cost paths from a source node to a destination node by optimizing the cost in multiple iterations.

Dijkstra's algorithm

- Compute the least-cost path from one node to all other nodes in the network in O(VLogV) time.
- Iterative algorithm.
  – After the kth iteration, the least-cost paths for k destination nodes are found.
- D(v): cost (distance) of the least-cost path from source node to destination v
- p(v): previous node of v along the least-cost path from source.
- N’: set of nodes to which the least-cost path is found.

Dijkstra's algorithm: Example

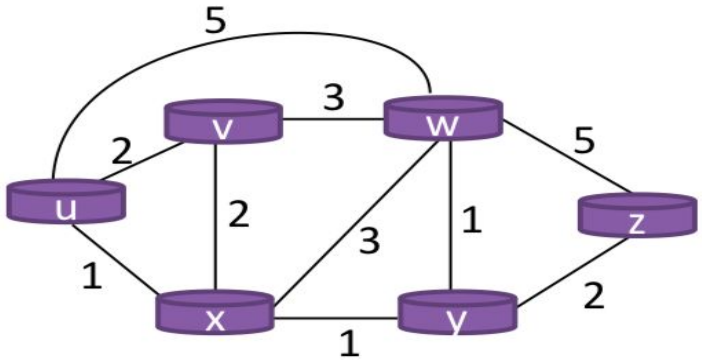
- Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞

Dijkstra's algorithm: Example

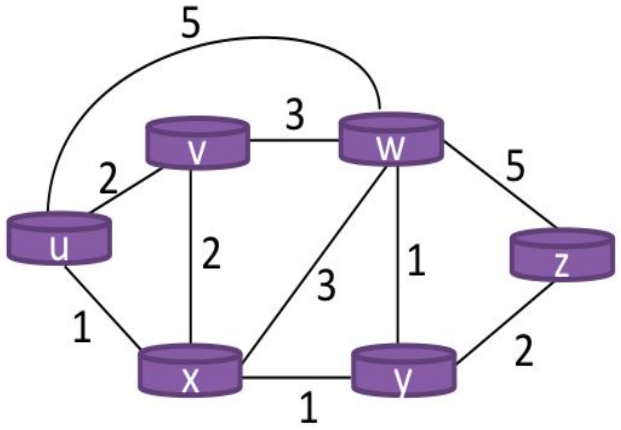
- Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞

Dijkstra's algorithm: Example

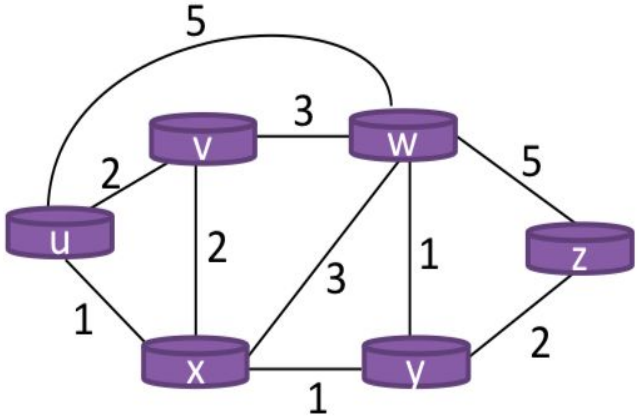
- Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y

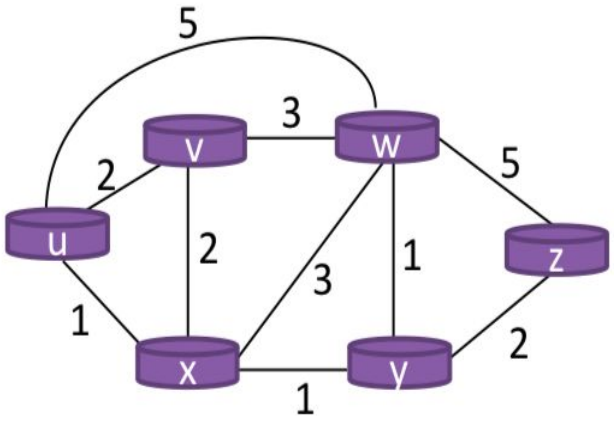
Dijkstra's algorithm: Example

- Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxvy		3,v			4,y

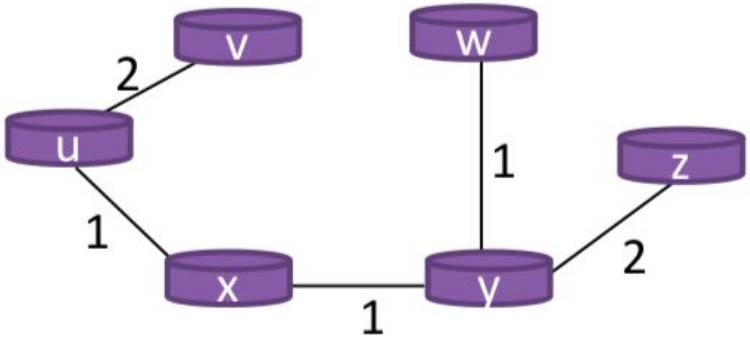
- Source is node u.



Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Dijkstra's algorithm: Example

Resulting shortest-path tree from u:



Resulting forwarding table in u:

Destination	Link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

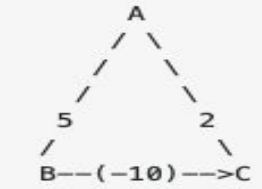
Dijkstra's algorithm

- Global routing algorithm:
  - It takes the connectivity between all nodes and all link costs as inputs.
  - Source u needs to have global knowledge of the network in order to determine its forwarding table.

Why doesn't Dijkstra's algorithm work for negative weight edges?

Recall that in Dijkstra's algorithm, **once a vertex is marked as "closed" (and out of the open set) - the algorithm found the shortest path to it**, and will never have to develop this node again - it assumes the path developed to this path is the shortest.

But with negative weights - it might not be true. For example:



V={A,B,C} ; E = {(A,C,2), (A,B,5), (B,C,-10)}

Dijkstra from A will first develop C, and will later fail to find `A->B->C`

EDIT a bit deeper explanation:

Note that this is important, because in each relaxation step, the algorithm assumes the "cost" to the "closed" nodes is indeed minimal, and thus the node that will next be selected is also minimal.

The idea of it is: If we have a vertex in open such that its cost is minimal - by adding any *positive number* to any vertex - the minimality will never change.
Without the constraint on positive numbers - the above assumption is not true.

Bellman-ford algorithm

- The **Bellman–Ford algorithm** is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph.
- It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers.
- The algorithm was first proposed by Alfonso Shimbel (1955), but is instead named after Richard Bellman and Lester Ford Jr., who published it in 1958 and 1956, respectively.
- Bellman–Ford runs in $O(|V||E|)$ time, where $|V|$ and $|E|$ are the number of vertices and edges respectively.

Bellman-ford algorithm

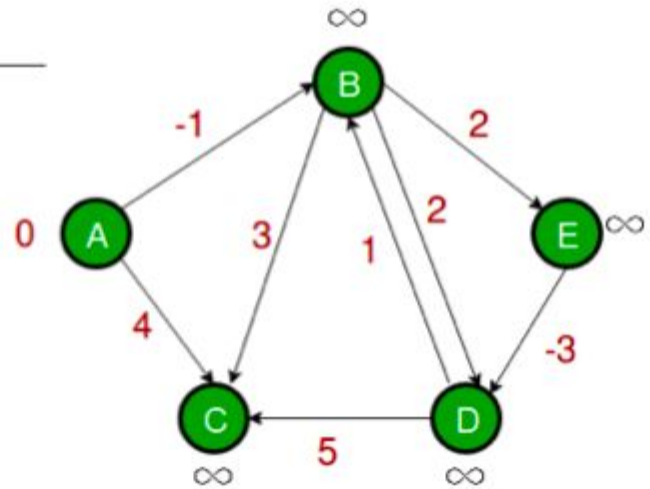
- Negative edge weights are found in various applications of graphs, hence the usefulness of this algorithm.

- If a graph contains a "negative cycle" (i.e. a cycle whose edges sum to a negative value) that is reachable from the source, then there is no *cheapest* path: any path that has a point on the negative cycle can be made cheaper by one more walk around the negative cycle.

- In such a case, the Bellman–Ford algorithm can detect and report the negative cycle.

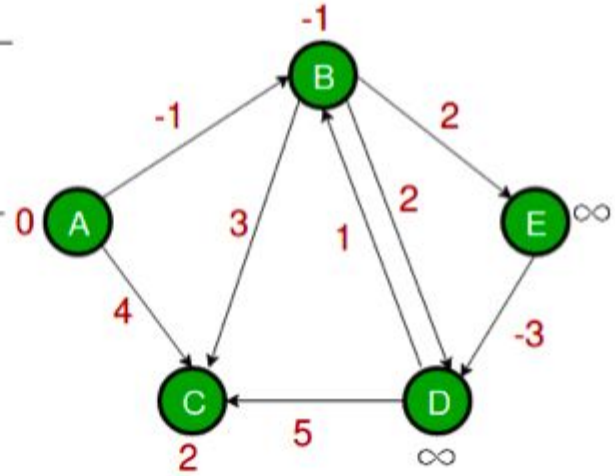
Bellman-ford algorithm

0	∞	∞	∞	∞



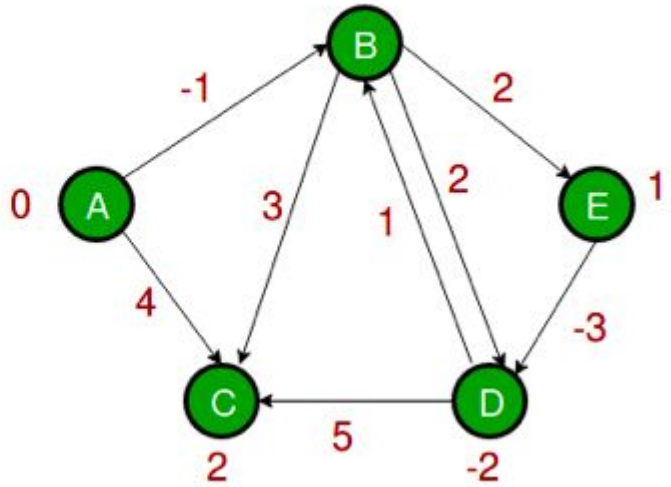
Bellman-ford algorithm

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



Bellman-ford algorithm

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



Comparison between these two algorithms

- Like Dijkstra's algorithm, Bellman–Ford proceeds by relaxation, in which approximations to the correct distance are replaced by better ones until they eventually reach the solution.

- In both algorithms, the approximate distance to each vertex is always an overestimate of the true distance, and is replaced by the minimum of its old value and the length of a newly found path.

- However, Dijkstra's algorithm takes O(|V|LogV) time; by contrast, the Bellman–Ford algorithm simply takes O(|V| |E|) .

Comparison between these two algorithms

- The performance of each algorithm varies network to network and depends on the topology and size of a particular network.

- The comparison of these two algorithms, therefore, depends on the speed of each to achieve its objective at its corresponding step given a network under routing.

OUTLINE OF LECTURE 21

- **Routing Protocols**
  - Hierarchical Routing
  - Broadcast Routing
  - Multicast Routing
- **ICMP**
- **Network Topology**