Module 4

(Lecture – 5)

(Network Layer: Router architecture; Internet Protocol (IP) - Forwarding and Addressing in the Internet; Routing algorithms - Link-state routing, Distance vector routing, Hierarchical routing; Routing in the Internet - RIP, OSPF, BGP; Broadcast & multicast routing; ICMP; Next Generation IP - IPv6)

Dr. Nirnay Ghosh

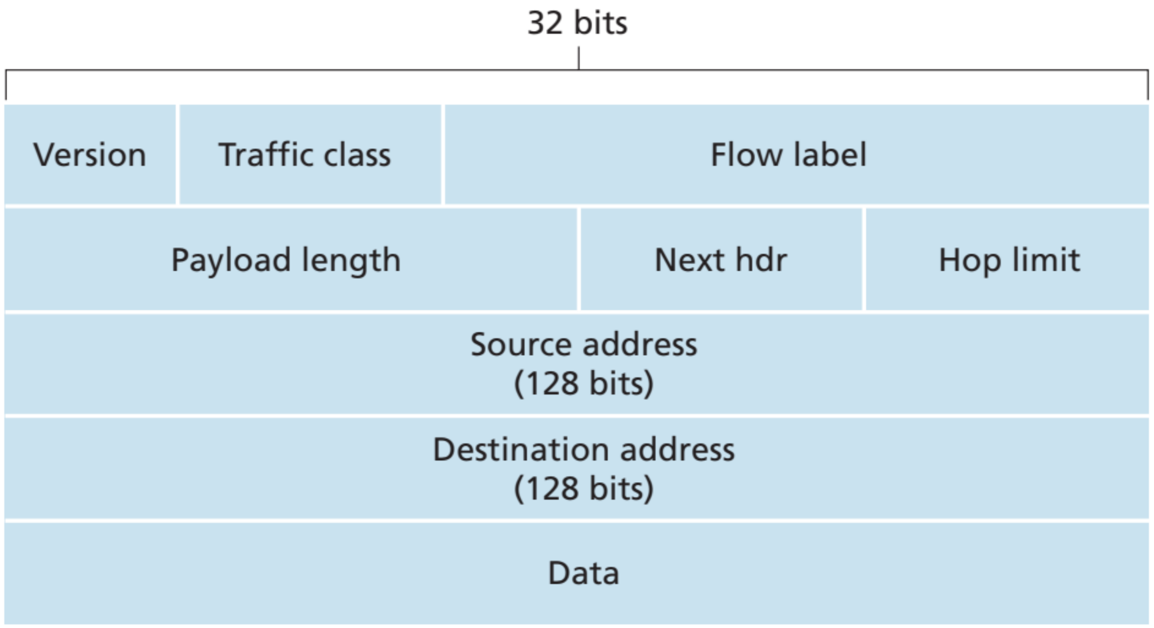
Assistant Professor

Department of Computer Science & Technology

IIEST, Shibpur

Internet Protocol Version 6 (IPv6)

- Huge increase in the number of new subnets and IP nodes getting attached to the Internet
  - 32-bit address space of IPv4 was rapidly used up
- IPv6: addresses this need for large IP address space
- ICMPv6 uses additional types and codes such as “Packet Too Big” and “Unrecognized IPv6 Options”
- Significant changes in IPv6 datagram format are
  - Expanded addressing capability:
    - Increases the size of IP address from 32 to 128 bits
    - Introduces a new type of address – anycast address – allows a datagram to be delivered to any one of a group of hosts
  - A Streamlined 40-bytes header
    - Number of IPv4 fields have been dropped or made optional
    - Allows for faster processing of the IP datagram

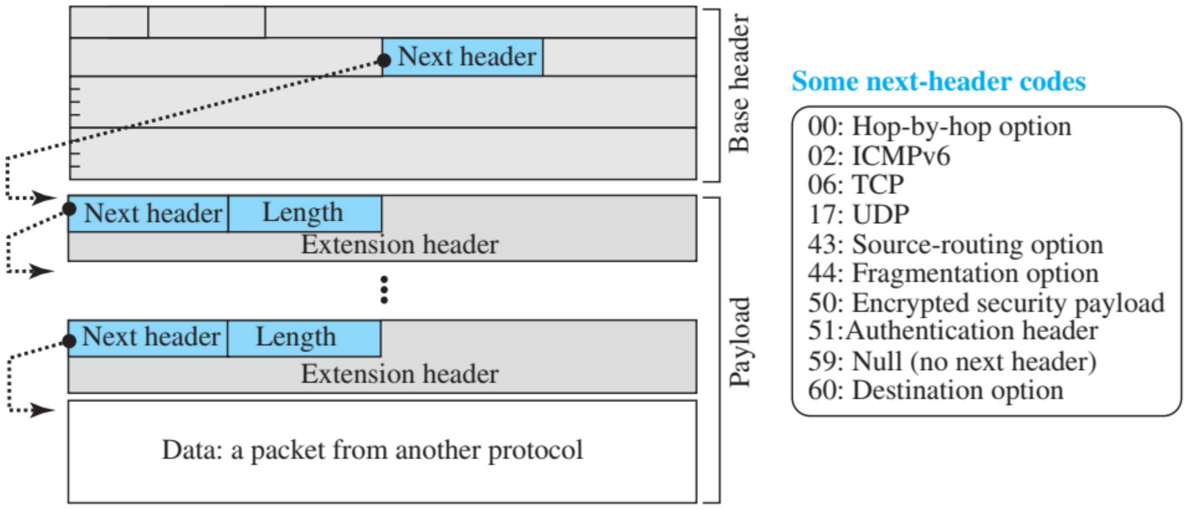


IPv6 Datagram Format

- Flow labelling and priority:
  - Introduces an elusive concept of “flow” – differentiates among traffic and assigns priority
  - Enables labelling of packets belonging to a particular flow for which the sender requests special handling (e.g., non-default quality of service or real-time service)
    - Example: audio and video transmission
  - Traditional applications such as file transfer and email might not be treated as flow

IPv6: Datagram Fields

- Version (4-bit): identifies the IP version number – carries a value of 6 in this field
- Traffic class (8-bit): gives priority to certain datagrams within a flow – similar to Type of Service field in IPv4
- Flow label (20-bit): used to identify a flow of datagrams
- Payload length (16-bit): unsigned integer giving the number of bytes following the fixed length header (40 bytes) in the IPv6 datagram
- Next header (8-bit): defining the type of the first extension header (if present) or the type of the data that follows the base header in the datagram – uses same values as the protocol field in IPv4
- Hop limit (8-bit): contents of this field are decremented by one at each forwarding router – discarded if hop limit count reaches zero



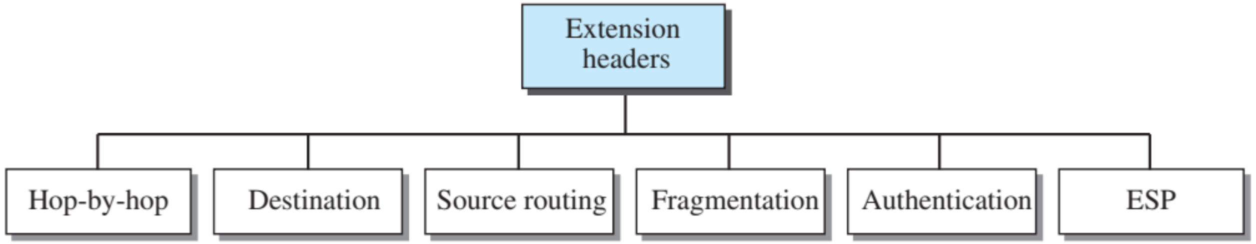
Payload in an IPv6 Datagram

- Source & destination addresses (128-bit each): contains the 128-bit IPv6 addresses
- Data: payload portion of IPv6 datagram – removed from the IP datagram at the destination – passed onto the protocol specified in the Next header field

IPv4 vs. IPv6

- Fields present in IPv4 but not in IPv6:
  - Fragmentation/Reassembly
    - IPv6 does not allow fragmentation/reassembly at intermediate routers
      - Speeds up forwarding within the network
    - Source and destination are responsible for these operations
    - ICMP error message "Packet Too Big" is sent if a router receives over-sized packet
    - Sender resends the data using smaller IP datagram size
  - Header checksum
    - IPv6 objective: faster processing of IP packets
    - Checksum is performed at both transport layer (e.g., TCP and UDP) and link-layer (e.g., Ethernet) protocols
      - Redundant in the network layer
    - Removed from the IPv6 datagram
  - Options
    - No longer part of standard IPv6 header
    - Replaced as the Next headers pointed to from within the IPv6 header

IPv6: Extension Header

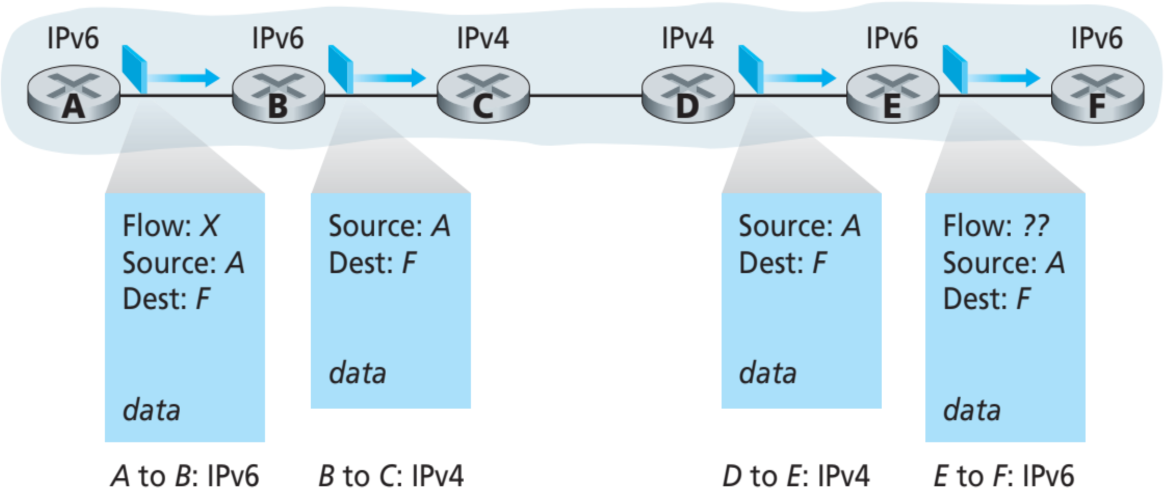


Extension Header Types

- Hop-by-hop
  - Used if the source needs to pass information (such as management, debugging, control functions) to all routers visited by the datagram
- Destination
  - Used if the source needs to pass information to the destination only
  - Intermediate routers are not permitted access to this information.
- Source routing
  - Provides extended routing, similar to IPv4 source routing
- Fragmentation
  - Fragmentation done at source
  - Use a Path MTU Discovery technique to find the smallest MTU supported by any network on the path.
- Authentication
  - Provides packet integrity and authentication
- Encrypted Security Payload (ESP)
  - Provides confidentiality and guards against eavesdropping

Interoperation between IPv4 to IPv6

- Dual-stack approach
  - IPv6 nodes also have complete IPv4 implementation
  - Nodes should have the ability to send and receive both IPv4 and IPv6 datagrams
  - Uses IPv4/IPv6 datagram while interoperating with an IPv4/IPv6 node
  - Must have both IPv4 and IPv6 addresses and able to determine whether another node is IPv6-capable or IPv4-only

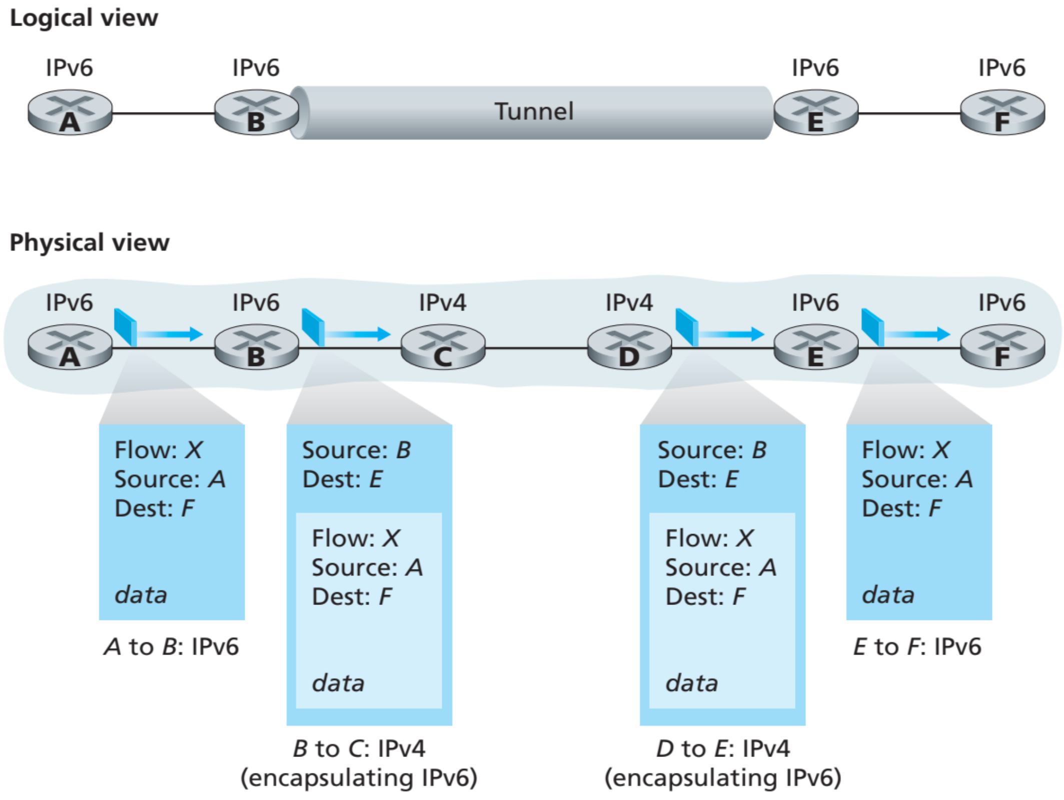


Dual Stack Approach

- Drawback:
  - Two IPv6 nodes can end up sending IPv4 datagrams owing to the presence of intermediate IPv4 nodes
  - Information specific to IPv6 datagram is lost due to conversion to the IPv4 datagram

Interoperation between IPv4 to IPv6

- Tunneling
  - An alternative to dual-stack approach
  - Allows two IPv6 nodes to interoperate using IPv6 datagrams inspite of being connected to each other by intervening IPv4 routers
  - Tunnel: intervening set of IPv4 routers between two IPv6 routers
  - IPv6 node on the sending side of the tunnel: encapsulates the entire IPv6 datagram in the data (payload) field of an IPv4 datagram
  - Intervening IPv4 routers in the tunnel: route this datagram among themselves – unaware that this datagram itself contains the entire IPv6 datagram

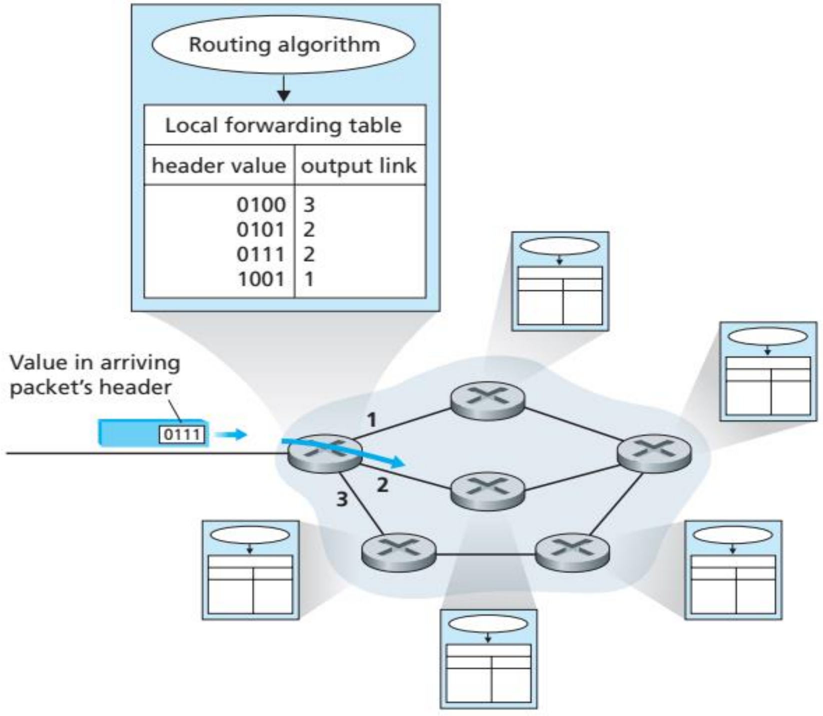


Tunneling

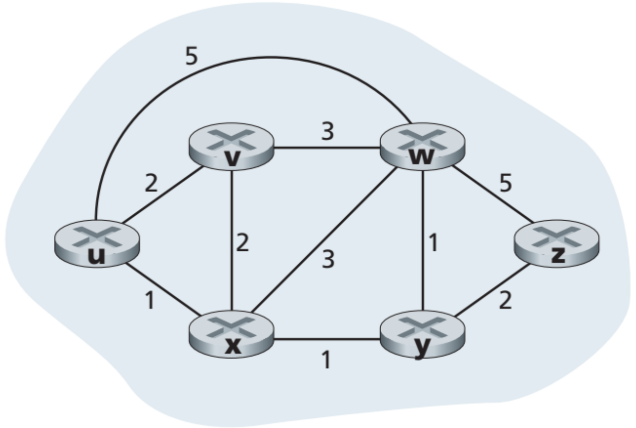
- IPv6 node on the receiving side of the tunnel:
  - Receives the IPv4 datagram
  - Extracts the IPv6 datagram and routes exactly as it would if it has received the IPv6 datagram from a directly connected IPv6 neighbor

Routing Algorithms

- Network layer – offers datagram service or virtual-circuit service
- Needs to determine the path packets take from source to destination
- Routing algorithms
  - Operates in the network routers
  - Exchange and compute the information to configure forwarding tables
  - Objective of routing: finding "good" (least cost) path from source (router) to destination (router)
  - Routing problem – formulated by the graph abstraction model
    - Set of routers (vertices)
    - Weighted links connecting the routers (edges)
- Classification of routing algorithms:
  - Global routing algorithm
  - Decentralized routing algorithm



Routing Algorithms configuring Forwarding Tables



Abstract Graph Model of a Computer Network

Routing Algorithms

- Global routing algorithm
  - Uses complete global knowledge (connectivity and link cost) of the network to compute least-cost path between a source and a destination
  - Runs at one centralized site or replicated at multiple sites
  - Also referred as *Link-state (LS) algorithms*
- Decentralized routing algorithm
  - No node has complete information about connectivity and link cost
  - Begins with the knowledge of costs of neighboring nodes
  - Iterative calculation and exchange of information with neighboring nodes to calculate least-cost path
  - Each node maintains a vector of estimates of the costs (distances) to all other nodes in the network
  - Also referred as *Distance-Vector (DV) routing*

Routing Algorithm – The Link-State (LS) Routing Algorithm

- Each node broadcast link-state packets to all other nodes in the network
- The link-state packet contains the identities and costs of its attached links
- All nodes have an identical and complete view of the network
- Each node runs the LS algorithm (Dijkstra's algorithms) and compute the same set of least-cost paths as every other node
  - Initialization
  - Loop: no. of iterations = no. of the nodes in the network
  - Output: shortest paths from node *u* to every other node in the network (spanning tree)
  - Forwarding table in *u* can be constructed from this information
  - Notations:
    - *D(v)*: cost of the least-cost path from the source node to destination *v* as of this iteration of the algorithm
    - *N'*: subset of nodes visited till now; *v* is in *N'* if the least-cost path from the source to *v* is definitively known.

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of u
5        then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for each neighbor v of w and not in N':
12         D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    least path cost to w plus cost from w to v */
15 until N'= N
```

Link-State (LS) Algorithm for Node *u*

- Computational complexity:
  - First iteration: search *n* nodes to determine the node with minimum cost
  - Second iteration: search *(n-1)* nodes
  - Third iteration: search *(n-2)* nodes and so on……
  - Total number of nodes searched = *n(n+1)/2* → worst-case complexity = $O(n^2)$
- Sophisticated implementation: Search time can be logarithmic if special data structure, heap, is used in line-9