**Network Layer: Data Plane**

**Networking Layer**

**Internetworking**: Routers forward packets from source to destination, crossing several networks along the way

The Network Layer transports segments from sending to receiving hosts.

* Network layer protocols are in every host, every router.
* Routers examine the header fields in all IP datagrams passing through it.

Two key network-layer functions:

1. **Forwarding**: Move packets from router’s input to appropriate router output.  
   (Getting through an interchange station)
2. **Routing**: Determine the route taken by packets from source to destination.  
   (Planning your trip from source 🡪 multiple stations 🡪 destination)

**Routing Algorithm** determines the end-to-end path through the network

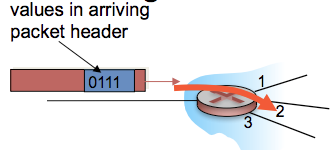
**Forwarding Table** then determines local forwarding at this router

* Packet arrives at Router
* Router uses forwarding table to determine which output link to forward the packet to **{ K=header val , V=output link }**

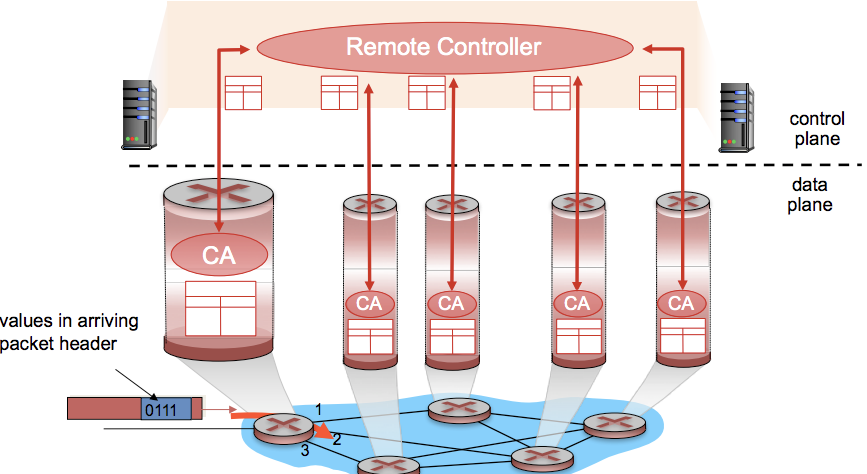
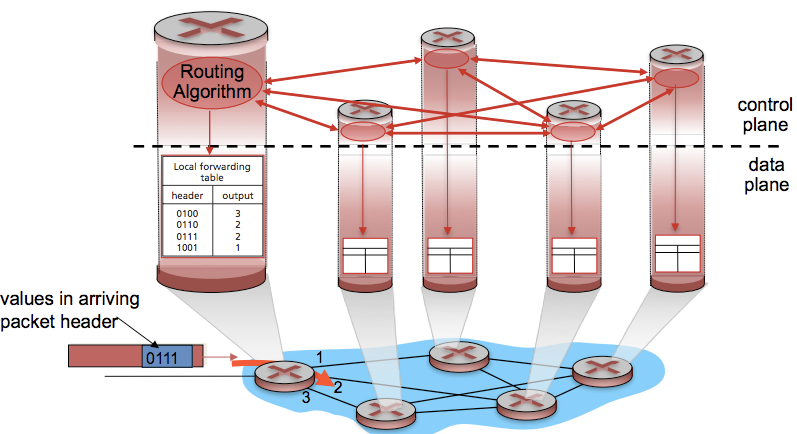
**Network Layer: Data vs Control Plane**

The **Control Plane** refers to the functions that determine how a packet is routed among routers in the end-to-end path.

The **Data Plane** refers to the functions that determine how packets are forwarded from a router input to its output port.



There are two Control Plane approaches:



**Per-Router Control Plane Logically Centralised Control Plane (SDN)**

**SDN: Software-Defined Networking**

Centralised servers.

A distinct controller interacts with local control agents (CA’s)

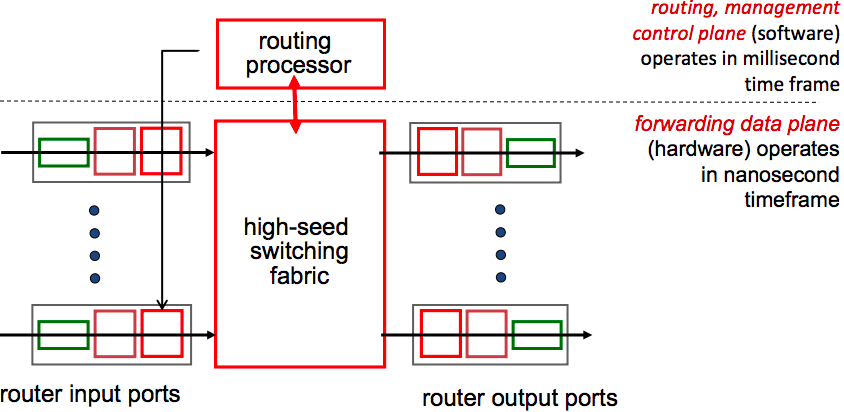
Individual Routing Algorithms in each and every router interact in the control plane.

**Network Layer: Service Models**

The N.L service model defines characteristics of the transport of data between one “edge” of the network to the other.

Example services include: *Guaranteed Delivery, Guaranteed Minimum Bandwidth to Flow, In-Order Deliveries etc.*

**Router Architecture Overview**



**Input Port Functions**

**Line Termination**

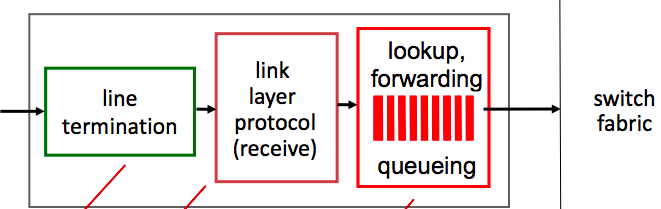
* Physical layer: bit-level reception

**Link Layer Protocol (Receive)**

* Data Link Layer e.g. Ethernet

**Lookup, Forwarding, Queuing**

* Decentralised Switching



**Decentralised Switching**

* Lookup the output port using header field values and forwarding table in input port memory. (*Match Plus Action*)
* Goal: Finish input port processing at line speed
* Queuing: if datagrams arrive faster than forwarding rate into switch fabric
* Two types of forwarding: Destination-Based (based on ONLY IP address) vs. Generalised (based on header fields)

**(1) Destination-based forwarding**: forward based only on destination IP address (traditional)

Destination-Based Forwarding Table

|  |  |
| --- | --- |
| **Destination IP Address Range** | Link Interface |
| 11001000 00010111 00010000 00000000 TO  11001000 00010111 00010111 11111111 | 0 |
| 11001000 00010111 00011000 00000000 TO  11001000 00010111 00011000 11111111 | 1 |
| 11001000 00010111 00011001 00000000 TO  11001000 00010111 00011111 11111111 | 2 |
| Otherwise | 3 |

**(2) Generalised forwarding**: forward base on any set of header field value

* Use **Longest Prefix Matching**: For a given DA, use the longest address prefix that matches the destination address.
* STEP 1: Find the IP ranges / entries in the forwarding table which matches with the Destination Address IP.
* STEP 2: Choose the IP range / entry with the longest prefix (more specific matching IP address)

Longest Prefix Matching

|  |  |
| --- | --- |
| **Destination IP Address Range** | Link Interface |
| 192.168.32.0 / 26 | 0 |
| 192.168.32.0 / 24 | 1 |
| 192.120.32.0 / 19 | 2 |
| Otherwise | 3 |

**Question: Which Link Interface would a packet with destination IP 192.168.32.1 go to? [ IMPORTANT TO STUDY ]**

STEP 1: Convert each Destination IP Address Range to binary

|  |  |
| --- | --- |
| **Destination IP Address Range** | Link Interface |
| 192 168 32 0  11000000 10101000 00100000 00000000 / 26 | 0 |
| 192 168 32 0  11000000 10101000 00100000 00000000 / 24 | 1 |
| 192 120 32 0  11000000 01111000 00100000 00000000 / 19 | 2 |
| Otherwise | 3 |

STEP 2: Get the prefix

|  |  |
| --- | --- |
| 11000000 10101000 00100000 00\*\*\*\*\*\* 26 binary digits | 0 |
| 11000000 10101000 00100000 \*\*\*\*\*\*\*\* 24 binary digits | 1 |
| 11000000 01111000 001\*\*\*\*\* \*\*\*\*\*\*\*\* 19 binary digits | 2 |
| Otherwise | 3 |

STEP 3: Convert Destination IP Address to binary

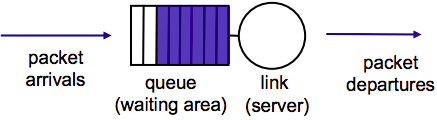
192.168.32.1 = 11000000 10101000 00100000 00000001

STEP 4: Find matching entries.

11000000 10101000 00100000 00000001 matches with Link Interfaces [0] and [1]

STEP 5: If more than one match, choose one with the longest prefix (more specific matching IP address)

Link Interface [0] has the longest prefix of 26. Forward the packet to Link Interface [0].

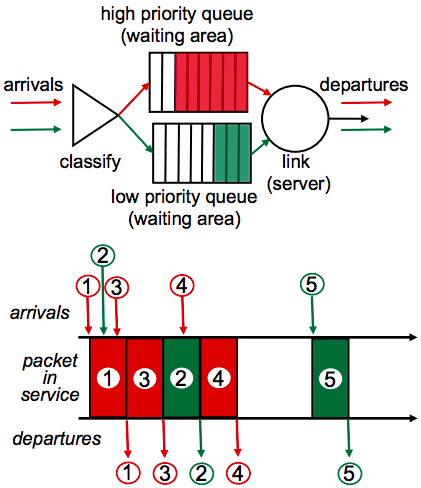
**Scheduling Mechanisms**

**Scheduling**: choose the next packet to send on the link

**FIFO (First In First Out) scheduling**: send in order of arrival to the queue.

**Discard Policy**: if a packet arrives to a full queue, which packet do we discard?

**Priority Scheduling**

**Scheduling Policy: PRIORITY**

**Priority Scheduling** is sending the highest priority queued packet.

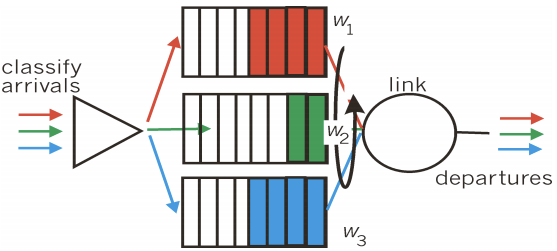
* Multiple classes, with different priorities.
* Class may depend on marking or other header info.  
  E.g. IP source / destination, port numbers etc.

**Scheduling Policy: ROUND ROBIN**

**Round Robin Scheduling (RR)**

* Multiple classes
* Cyclically scan class queues, sending one complete packet from  
  each class if available.

**Weighted Fair Queuing**

**Scheduling Policy: Weighted Fair Queuing (WFQ)**

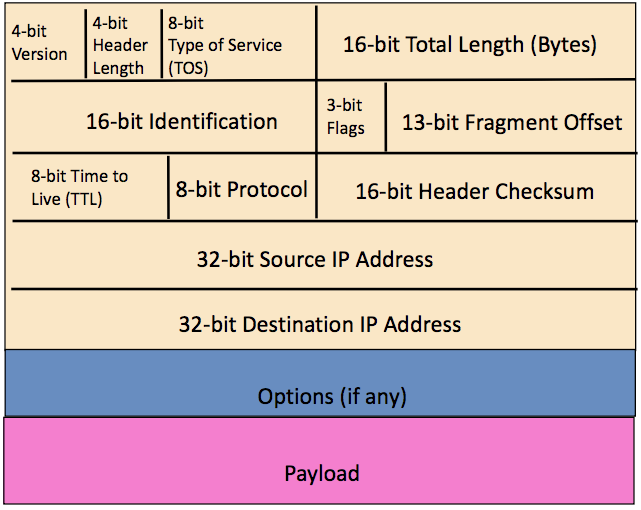
* Generalised version of Round Robin
* Each class gets a weighted amount of service in each cycle

**The Internet Network Layer**

The Network Layer consists of:

* **Routing Protocols**: Path selection, RIP, OSPF, BGR 🡪 **Forwarding Table**
* **IP Protocol**: Addressing conventions, datagram format, packet handling conventions
* **ICMP Protocol**: Error reporting, router signalling

IP Packet Structure: 20 bytes of Standard Header, then Options



Fields for reaching dest and back

Fields for reading packet correctly

Telling end-host how to handle packet

Checksum, TTL and Fragment Fields

**Version Number** (4 bits)

* Indicates version of the IP protocol
* “4” = IPv4 | “6” = IPv6

**Header Length** (4 bits)

* Number of 32-bit words in the header
* Typically “5” for 20-byte IPv4 header
* Can be more with IP options

**Total Datagram Length** (16 bits)

* # bytes in the packet
* Max size = 65,535 bytes (216 – 1)

**Protocol** (8 bits)

* Identifies the upper-layer protocols
* Important for demultiplexing at receiving host

E.g. “6” = TCP

E.g. “17” = UDP

<---------------------------------- 32 bits ------------------------------------->

**IP Packet Structure: Checksum, TTL, Fragmentation Fields**

**Header Checksum** (16 bits)

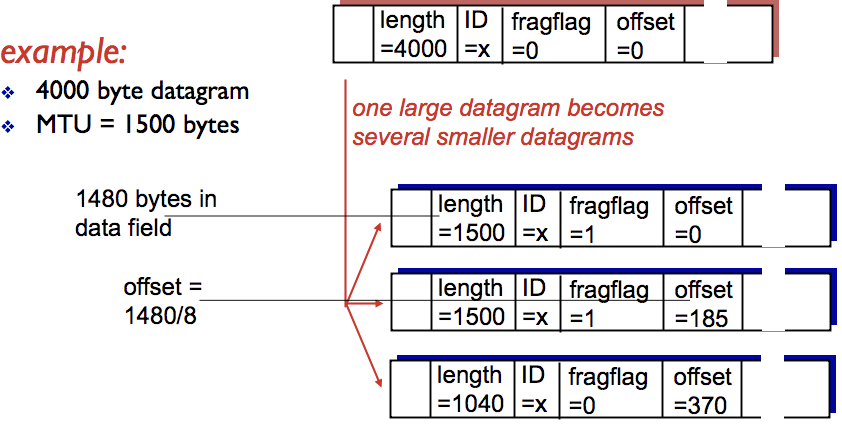
* If not correct, router discards packets. Checksum is recalculated every router.

**Time To Live Field** (8 bits)

* Max number of remaining hops.
* Decremented at each hop 🡪 packet is discarded if it reaches 0 and a “time exceed” message is sent to the source
* **Preventing Loops** **(TTL)**
  + Forwarding loops cause packets to cycle for a long long time.
  + As these accumulate, eventually they will consume all capacity.

**IP Fragmentation Reassembly** (Frag Offset – 13 bits)

* Network links have a **Maximum Transmission Unit (MTU)**
* A large IP datagram is divided / “Fragmented” within the network
  + One datagram becomes several datagrams, which are “reassembled” only at final destination
  + IP header bits are used to identify the ORDER related to the datagram fragments.



20 byte header in each packet.

**Original packet** (4000 bytes)

= 3980 payload + 20 header

**Fragmentation pkt #1** (1500 bytes)

= 1480 payload + 20 header

**Fragmentation pkt #2** (1500 bytes)

= 1480 payload + 20 header

**Fragmentation pkt #3** (1040 bytes)

= 1020 payload + 20 header

**original 3980 bytes = 1480 + 1480 + 1020**

* Offset = 1480/8 because every bit in the header is precious space.
* Dividing the offset by 8 allows it to fit in the 13 bit Fragmentation Offset space.
* This means that every packet but the last must contain a number of bytes that is a multiple of 8.

**NOTE: The offset is the address or the locator from where the data starts in the original payload. The system/router takes the payload and divides it into smaller parts, keeping track of this offset so that reassembly can be done later.**

**IP Fragmentation Exploits**

**IP fragmentation overlapped**: Two fragments within the same IP datagram have offsets that mean they overlap.

* A or B might overlap each other, causing undesirable results or exceptions.
* This is the basis for **Teardrop Denial of Service** attacks, where overlapped fragments cause bugs and may eventually crash a network.

**IP fragmentation buffer full**: Excessive amount of incomplete fragmented datagrams, a large number of fragments in each datagram or a mix of both.

* Typically used to bypass security measures or intrusion detection systems by intentional fragmentation.

**IP fragment overrun**: Reassembled datagram exceeds the declared IP data length or max datagram length (65,535 bytes)

* Systems that try to process these oversized datagrams may crash, causing a Denial of Service.

**IP fragment overwrite**: Sending in fragments along with additional random data.

* Future fragments may overwrite the random data with the re

**IP fragment “Too Many Datagrams”**: Too Many Datagrams exploit = excessive number of incomplete datagrams detected in the network.

* Indicates Denial of Service or attempts to bypass security measures.

**IP fragment incomplete datagram**: Datagram can not be fully reassembled due to missing data.

* Indicates a Denial of Service or attempt to defeat packet filtering security policies.

**IP fragment too small**: Any fragment other than the final fragment is < 400 bytes.

* Indicates that the fragment is likely intentionally crafted.
* Small fragments may be used in Denial of Service attacks or to bypass security measures or detection.

**Fields for special handling**

**Type of Service / Differentiated Services Code Point (DSCP)** (8 bits)

* Allows packets to be treated differently based on needs
* E.g. low delay for audio, high bandwidth for bulk transfer etc.

**Options**: not often used

* Timestamp, record route taken, specify list of routers to visit.