



南京大學

NANJING UNIVERSITY

网络层：数据平面

殷亚凤

智能软件与工程学院

苏州校区南雍楼东区225

yafeng@nju.edu.cn , <https://yafengnju.github.io/>



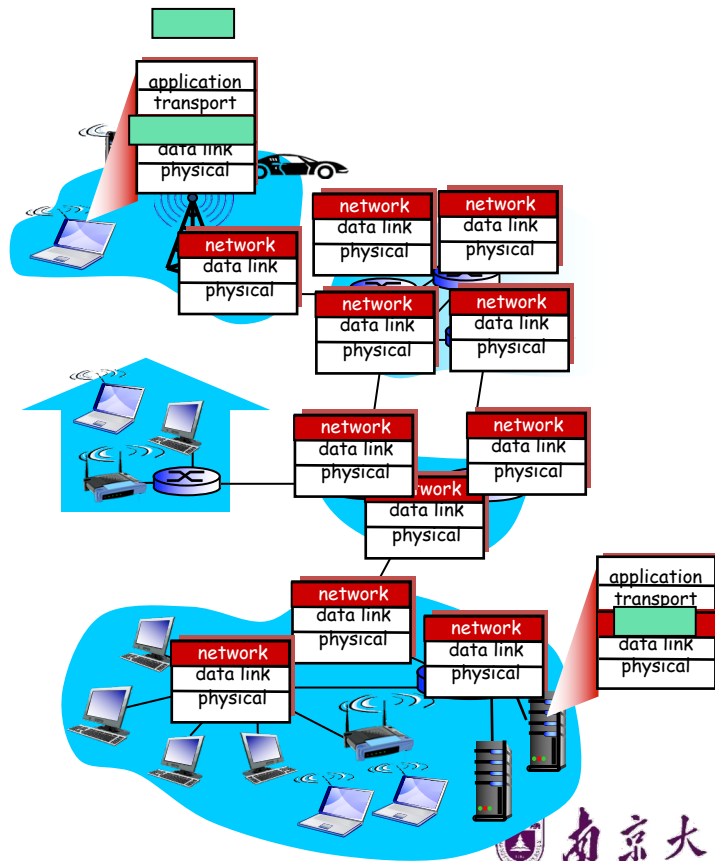
Outline

- Network Layer Functions
- Routers
- IP Packet Structure



Network Layer

- transport segment from sending to receiving host
- on **sending side** encapsulates segments into **datagrams**
- on **receiving side**, delivers segments to transport layer
- network layer protocols in **every host, router**
- router examines header fields in all IP datagrams passing through it





Two Key Network-layer Functions

- OSI network-layer functions:
- **Forwarding (Data plane)**
 - Move packets from input to designated output determined by switching (single node)
 - Error handling, queuing and scheduling
- **Switching / Routing (Control plane)**
 - Determine route taken by packets from source to destination (multiple nodes)
 - Shortest path from source to destination
 - Routing algorithms

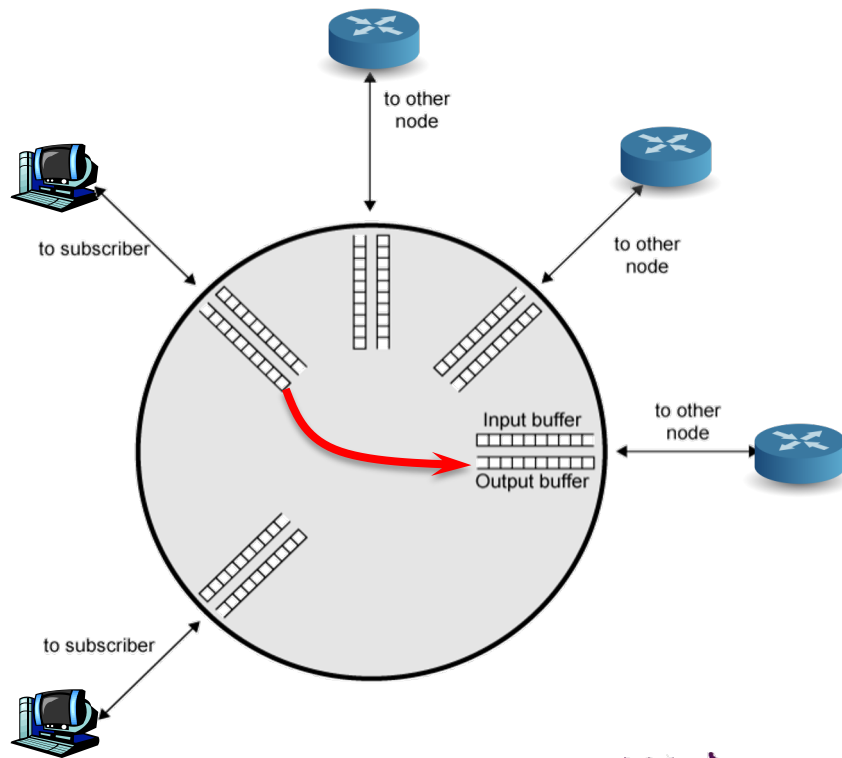
analogy: Trip Planning

- **forwarding:** getting through single city (e.g., entering and leaving Suzhou Station)
- **routing:** planning the route from Nanjing to Shanghai (e.g., Nanjing-Wuxi-Suzhou-Shanghai)



Forwarding Functions

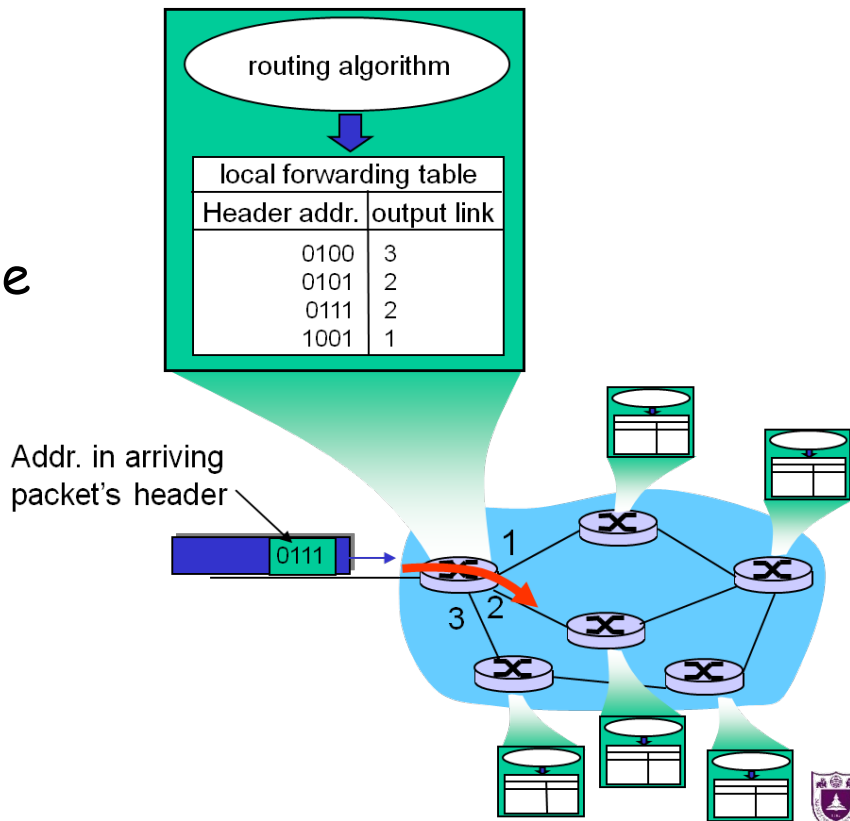
- Queuing and scheduling
 - Host to Switch
 - Switch to Host
 - Switch to Switch





Switch Functions

- Routing determines the forwarding table



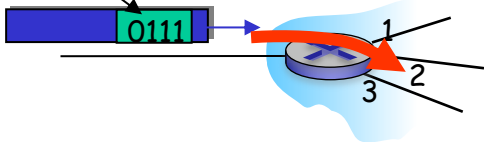


Network layer: data plane, control plane

Data plane:

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

values in arriving packet header



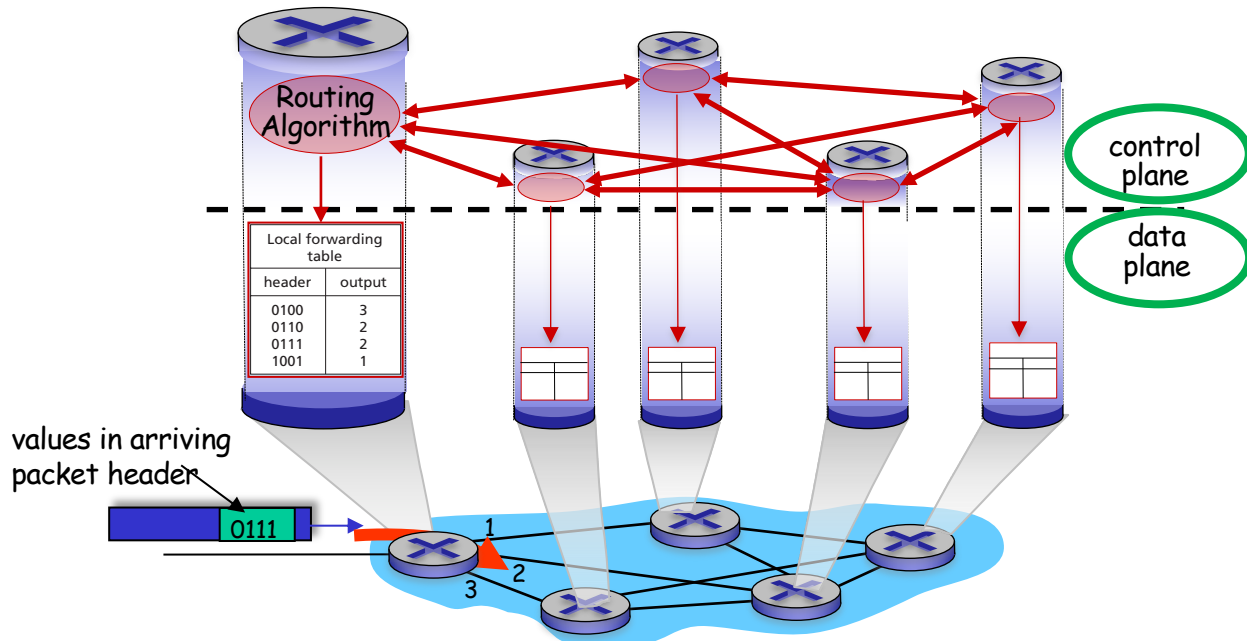
Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
 - traditional routing algorithms: implemented in routers
 - software-defined networking (SDN): implemented in (remote) servers



Per-router control plane

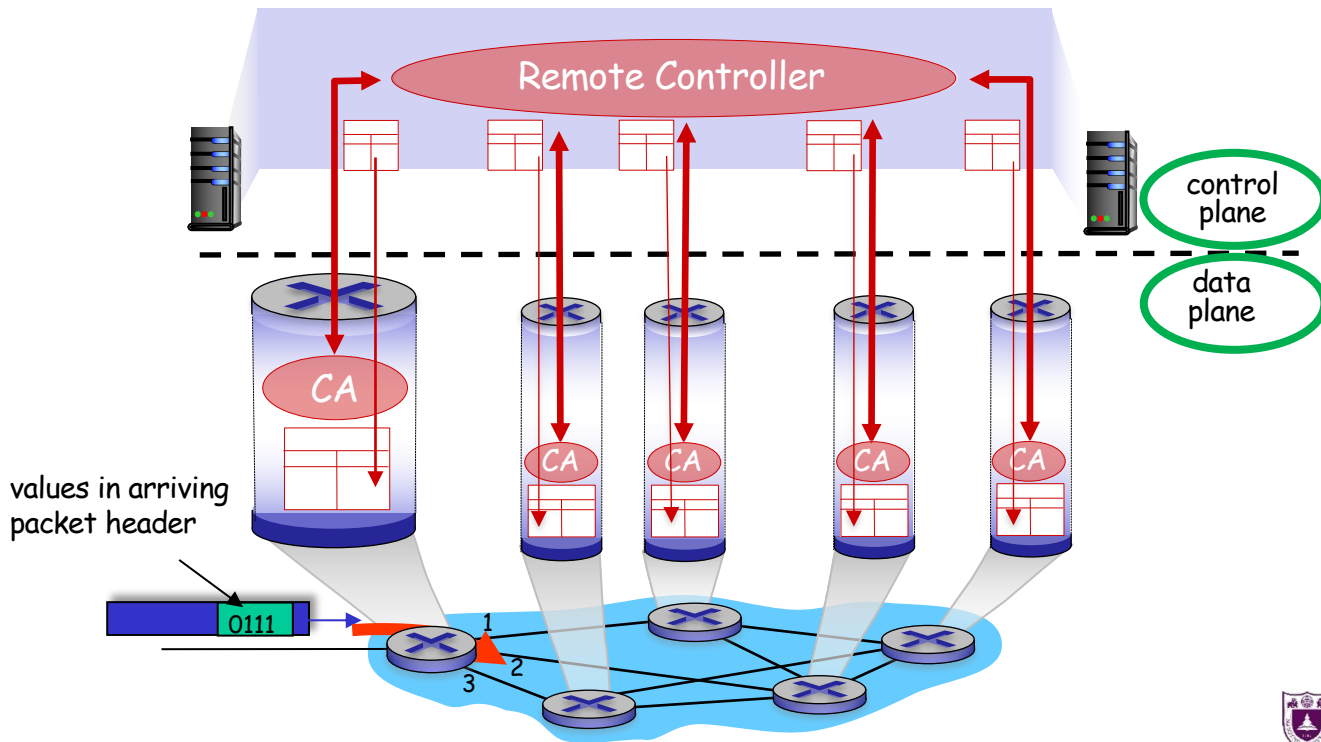
- Individual routing algorithm components in each and every router interact in the control plane





Software-Defined Networking (SDN) control plane

- Remote controller computes, installs forwarding tables in routers





Network Service Model

Q: What **service model** for “channel” transporting datagrams from sender to receiver?

- Network service model
 - **Service model** for “channel” transporting packets from sender to receiver
 - Called **Quality of Service** from host perspective

Example services for individual packets

- Guaranteed delivery
- Guaranteed delivery with less than 40 msec delay

Example services for a flow of packets

- In-order packet delivery
- Guaranteed minimum bandwidth to flow
- Restrictions on changes in inter-packet spacing



Example : Network Service Model of IP

- Best effort

Network Architecture	Service Model	Bandwidth Guarantee	No-Loss Guarantee	Ordering	Timing	Congestion Indication
Internet	Best Effort	None	None	Any order possible	Not maintained	None
ATM	CBR	Guaranteed constant rate	Yes	In order	Maintained	Congestion will not occur
ATM	ABR	Guaranteed minimum	None	In order	Not maintained	Congestion indication provided



Reflections on best-effort service

- **simplicity of mechanism** has allowed Internet to be widely deployed adopted
- sufficient **provisioning of bandwidth** allows performance of real-time applications (e.g., interactive voice, video) to be "good enough" for "most of the time"
- **replicated, application-layer distributed services** (datacenters, content distribution networks) connecting close to clients' networks, allow services to be provided from multiple locations
- congestion control of "elastic" services helps

It's hard to argue with success of best-effort service model.



Outline

- Network Layer Functions
- Routers
- IP Packet Structure

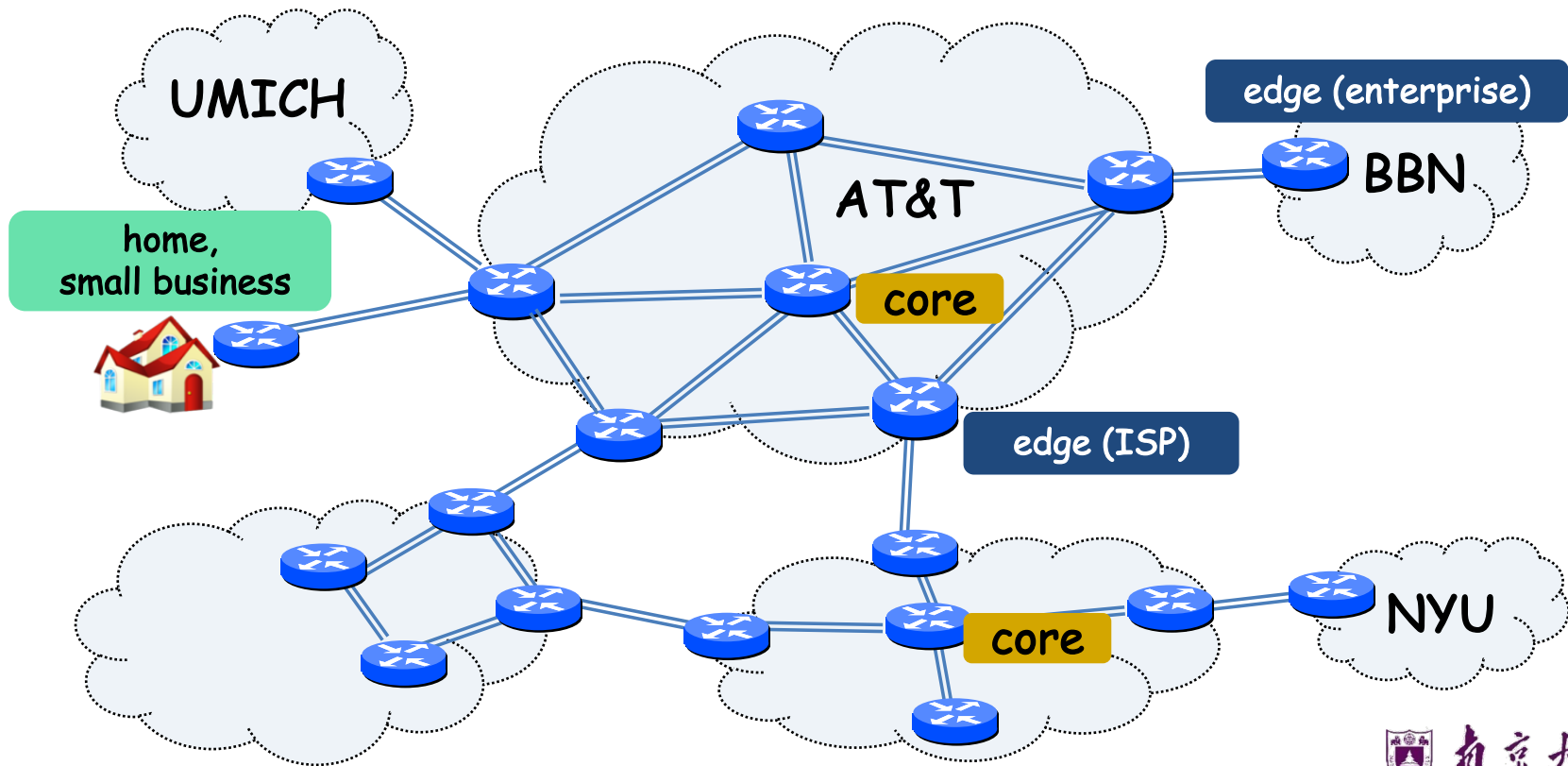


Router definitions

- Router capacity = $N \times R$
- N = Number of external router "ports"
- R = Speed ("line rate") of a port



Networks and routers





Many types of routers

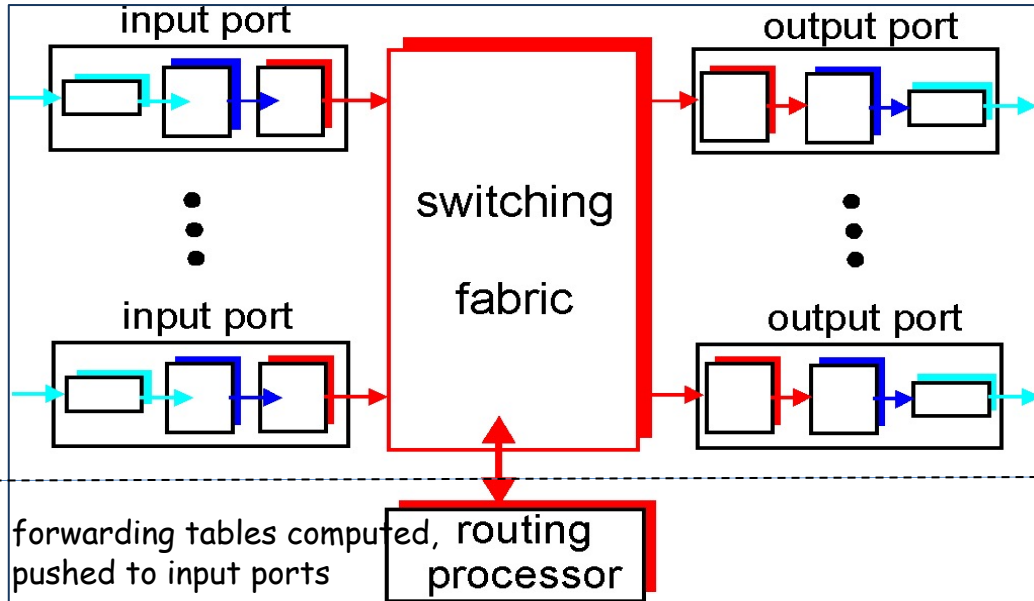
- Core
 - R = 10/40/100/200/400 Gbps
 - NR = O(100) Tbps (Aggregated)
- Edge
 - R = 1/10/40/100 Gbps
 - NR = O(100) Gbps
- Small business
 - R = 1 Gbps
 - NR < 10 Gbps



Inside a Router: Architecture Overview

Two key **switch** functions:

- Run **routing** algorithms/protocol
- **Forwarding** packets from incoming to outgoing link

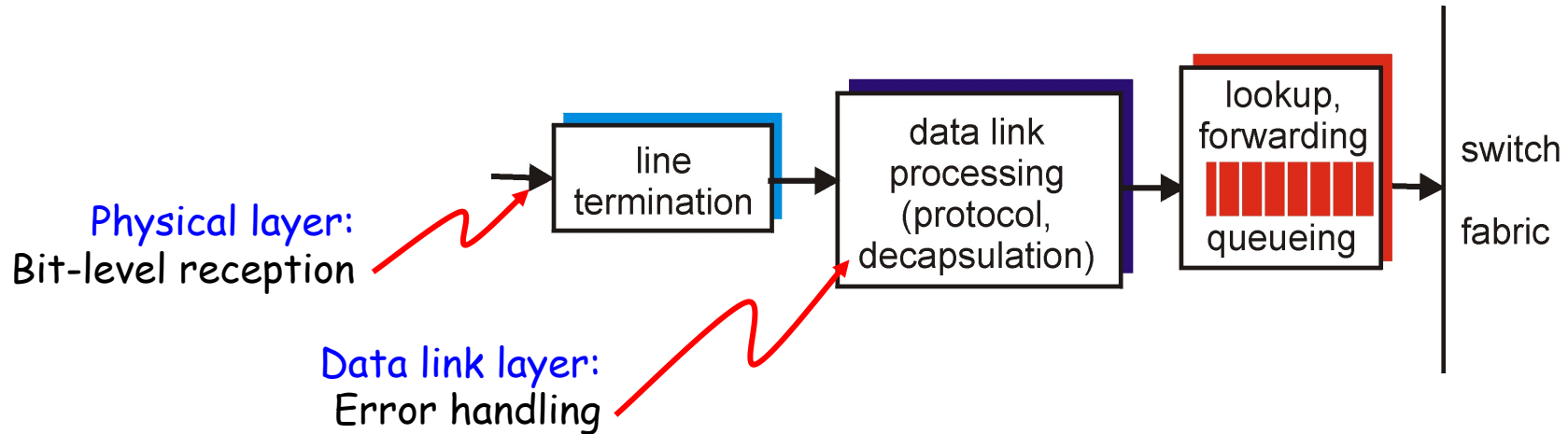


Forwarding,
Data plane
(Hardware)

Routing, Management
Control plane
(hardware&software)



Input Port Functions



Tasks

- Receive incoming packets (physical layer stuff)
- Update the IP header
 - TTL, Checksum, Options and Fragment (maybe)
- Lookup the output port for the destination IP address
- **Queuing**: if packets arrive faster than forwarding rate into switch fabric



Input Port

- Challenge: **speed!**
 - 100B packets @ 40Gbps → new packet every 20 nano secs!
 - Typically implemented with specialized ASICs (network processors)



Looking up the output port

- One entry for each **address** → 4 billion entries!
- For scalability, addresses are aggregated



Example

- Router with 4 ports
- Destination address range mapping
 - 11 00 00 00 to 11 00 00 11: Port 1
 - 11 00 01 00 to 11 00 01 11: Port 2
 - 11 00 10 00 to 11 00 11 11: Port 3
 - 11 01 00 00 to 11 01 11 11: Port 4





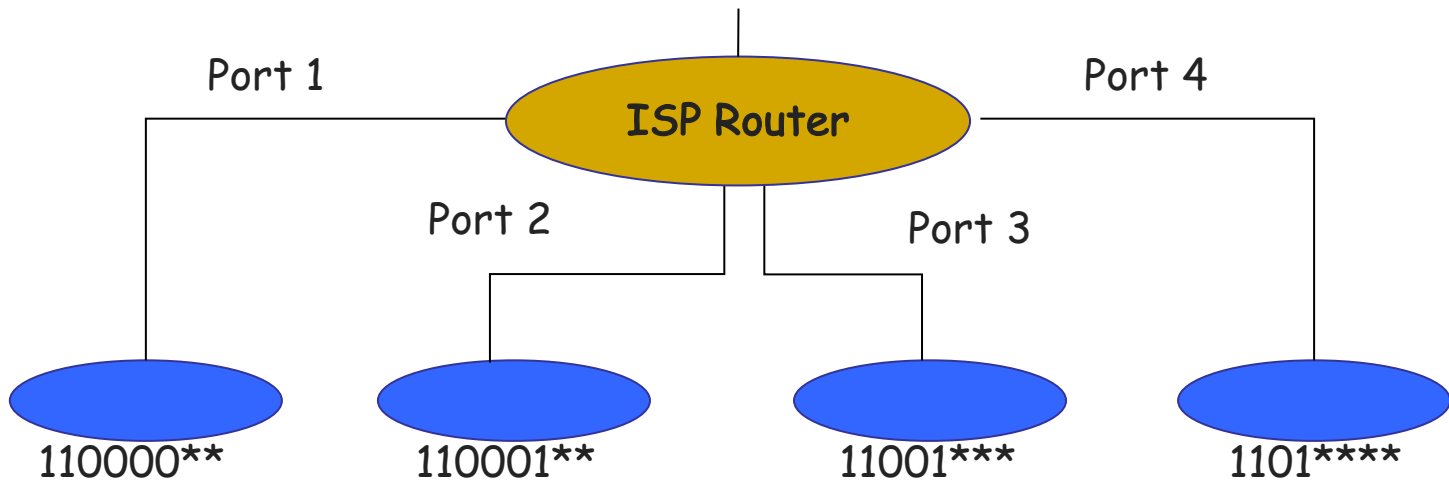
Example

- Router with 4 ports
- Destination address range mapping
 - 11 00 00 00 to 11 00 00 11: Port 1
 - 11 00 01 00 to 11 00 01 11: Port 2
 - 11 00 10 00 to 11 00 11 11: Port 3
 - 11 01 00 00 to 11 01 11 11: Port 4

Longest prefix matching rule: when looking for forwarding table entry for given destination address, use longest address prefix that matches destination address.



Longest prefix matching



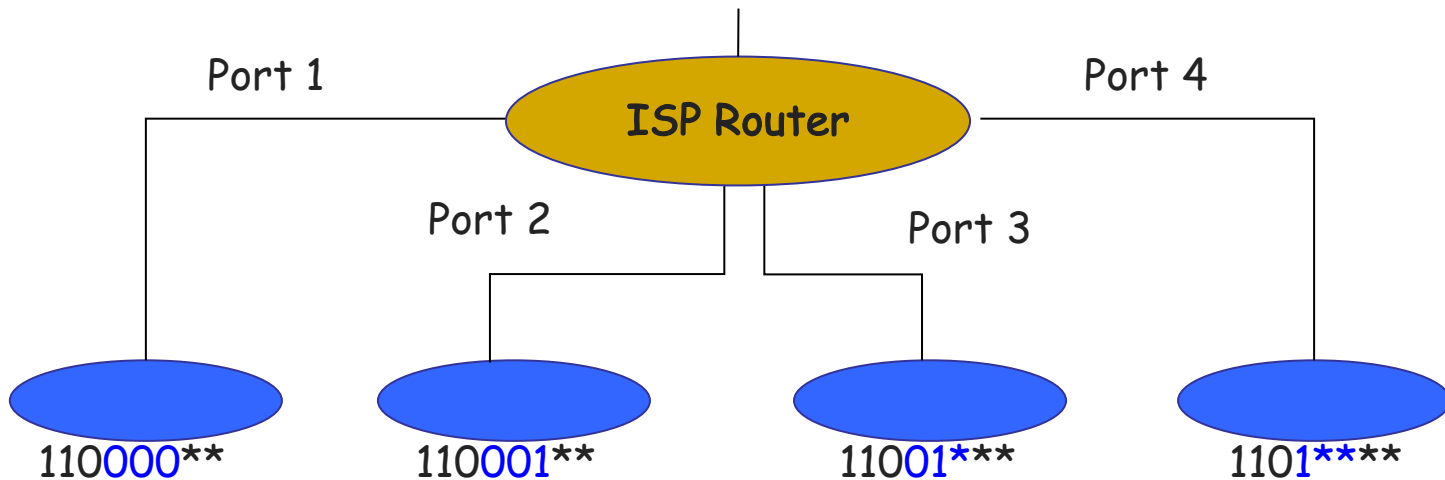


Finding match efficiently

- Testing each entry to find a match scales poorly
 - On average: $O(\text{number of entries})$
- Leverage **tree structure** of binary strings
 - Set up tree-like data structure

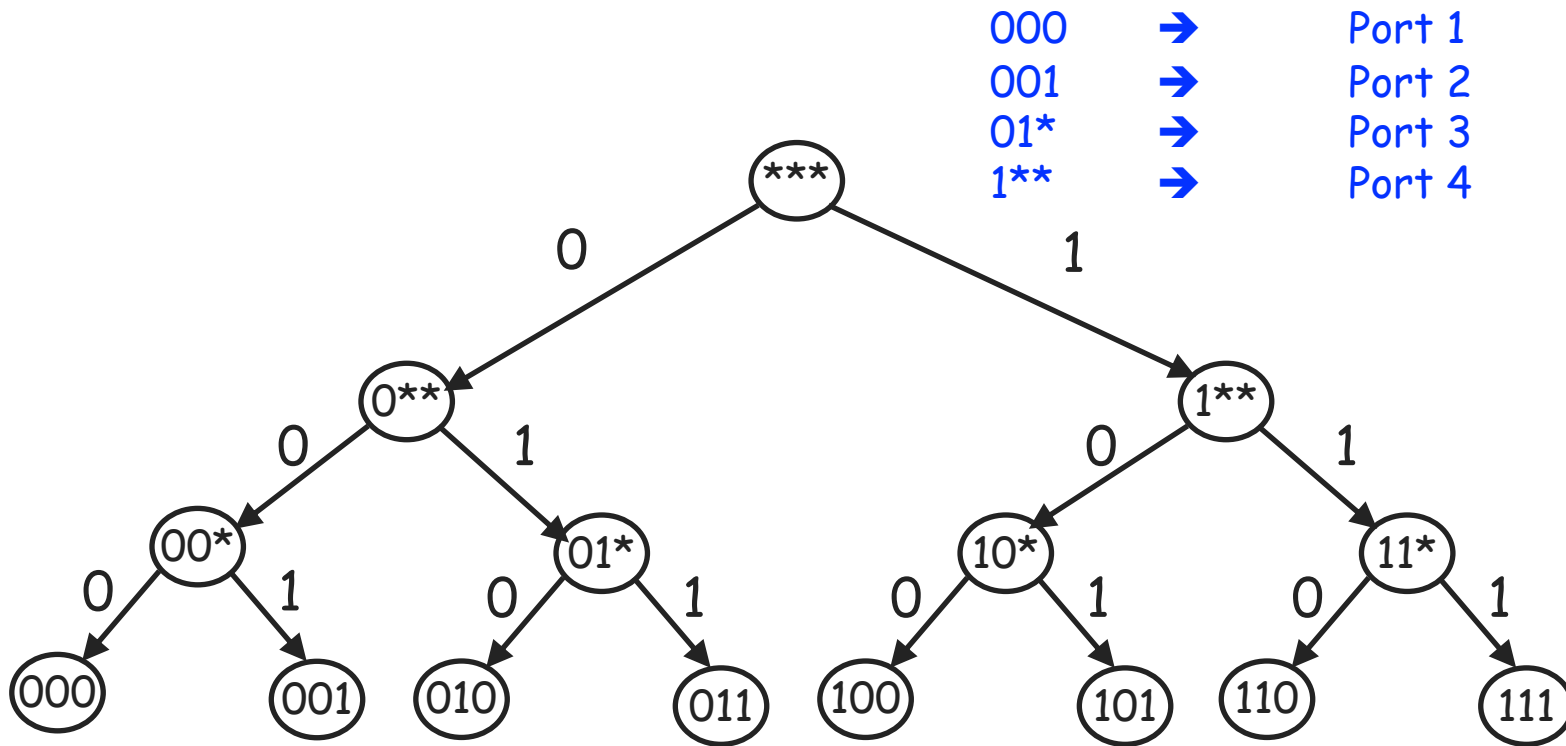


Longest prefix matching





Tree structure

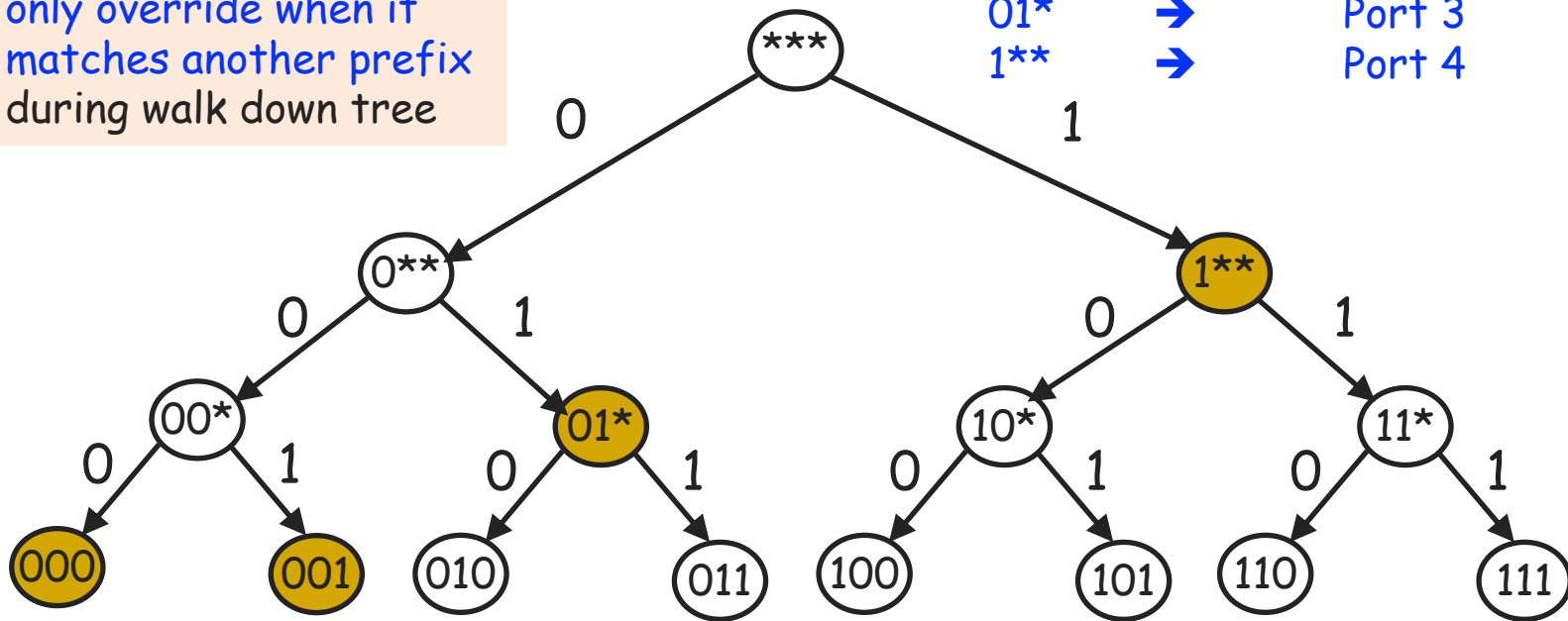




Tree structure

Record port associated with latest match, and only override when it matches another prefix during walk down tree

000 → Port 1
001 → Port 2
01* → Port 3
1** → Port 4





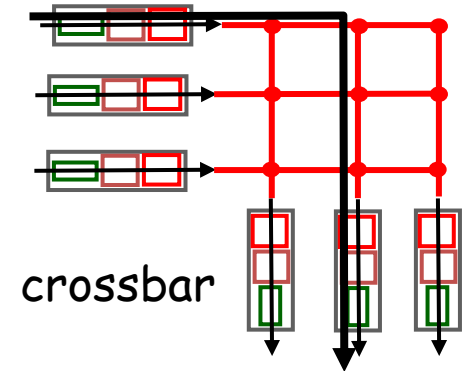
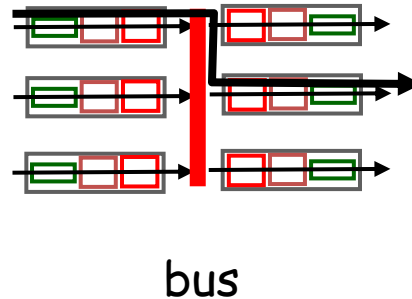
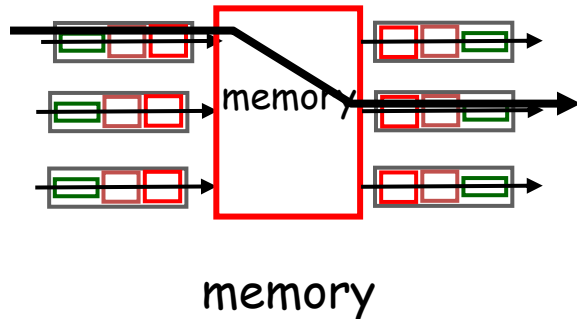
Input Port

- Main challenge is **processing speeds**
- Tasks involved:
 - Update packet header (easy)
 - LPM lookup on destination address (harder)
- Mostly implemented with specialized hardware



Connecting inputs to outputs: Switching fabric

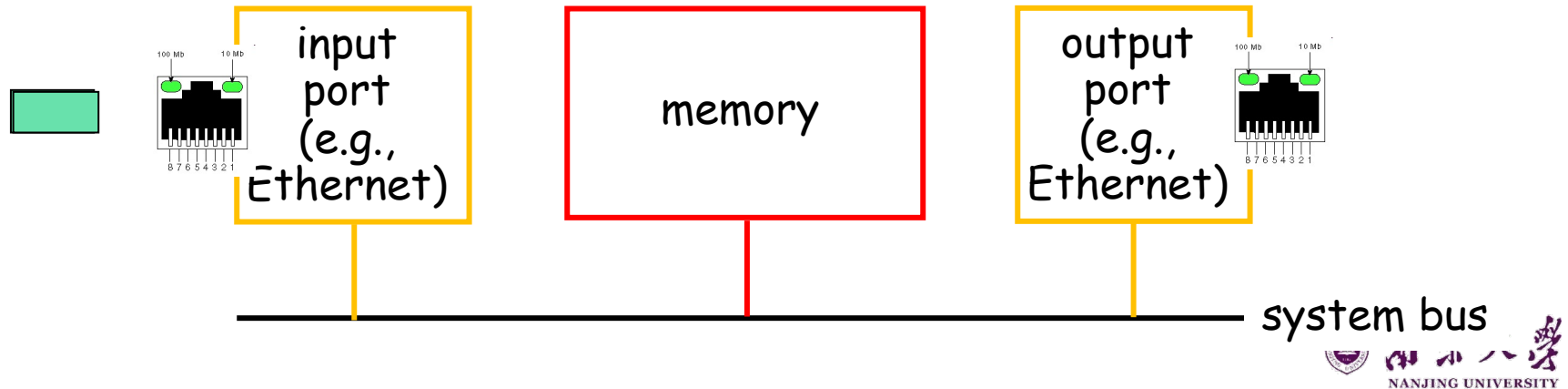
- Connecting inputs to outputs: **Switching fabric**
- Transfer packet from input buffer to appropriate output buffer
- **Switching rate**: rate at which packets can be transfer from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- **Three types of switching fabrics**





Switching via Memory

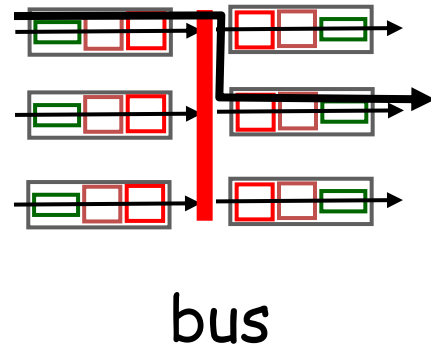
- First generation routers:
- Traditional computers with **switching under direct control of CPU**
- Packet copied to system's memory
- **Speed limited by memory bandwidth** (2 bus crossings per datagram)





Switching via a Bus

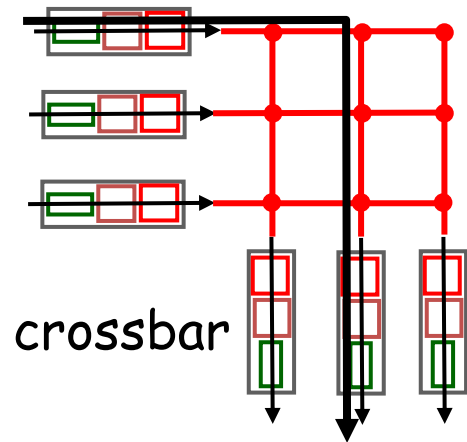
- Datagram from **input port memory** to **output port memory** via a **shared bus**
- **Bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers





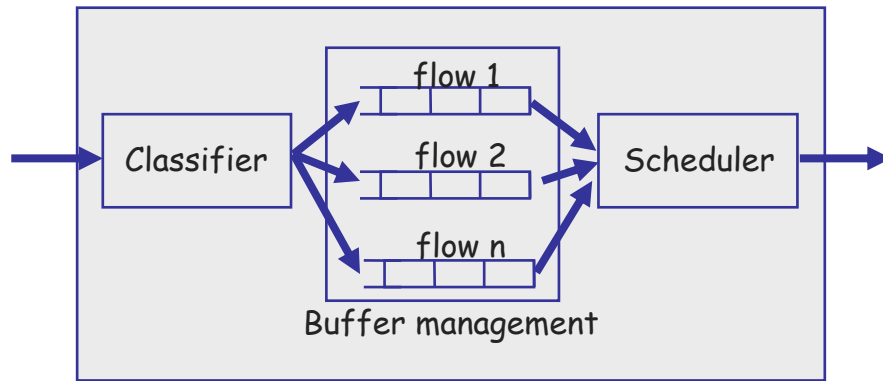
Switching via a Mesh

- Overcome bus bandwidth limitations
- Banyan networks, crossbar, other interconnection nets initially developed to **connect processors in multiprocessor**
- Advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network





Output Port Functions



- **Packet classification**: map packets to flows
- **Buffer management**: decide when and which packet to drop
- **Scheduler**: decide when and which packet to transmit
 - Chooses among queued **packets for transmission**
 - Select **packets to drop** when buffer saturates



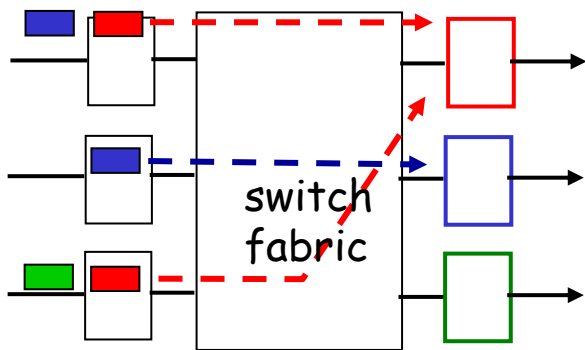
Packet classification

- **Classify an IP packet** based on a number of fields in the packet header, e.g.,
 - Source/destination IP address (32 bits)
 - Source/destination TCP port number (16 bits)
 - Type of service (TOS) byte (8 bits)
 - Type of protocol (8 bits)
- In general **fields are specified by range**
 - Classification requires a multi-dimensional range search!

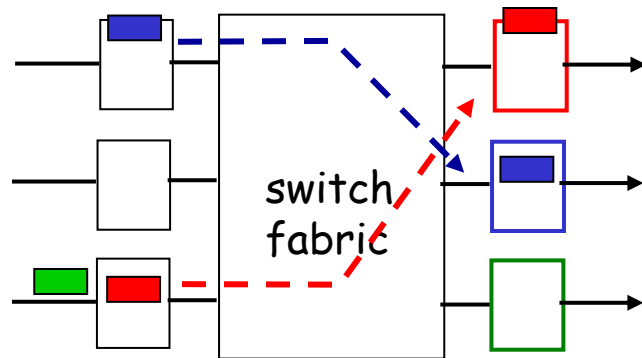


Queuing: Input port queuing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
 - queueing delay and loss due to input buffer overflow!
- **Head-of-the-Line (HOL) blocking:** queued datagram at front of queue prevents others in queue from moving forward



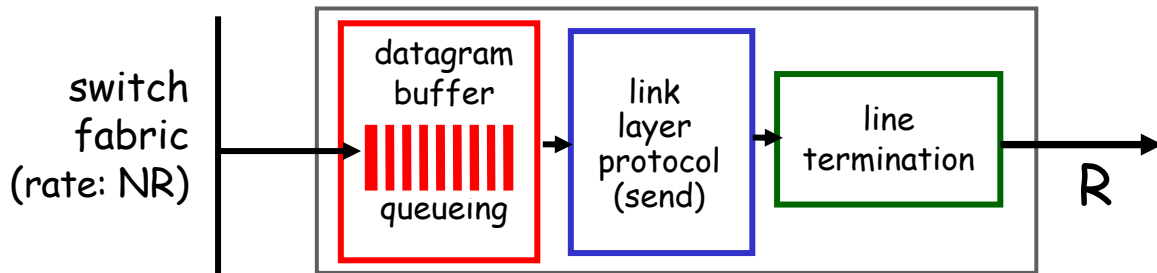
output port contention: only one red datagram can be transferred. lower red packet is **blocked**



one packet time later: green packet experiences HOL blocking



Queuing: Output port queuing



- **Buffering** required when datagrams arrive from fabric faster than link transmission rate. **Drop policy:** which datagrams to drop if no free buffers?



Datagrams can be lost due to congestion, lack of buffers

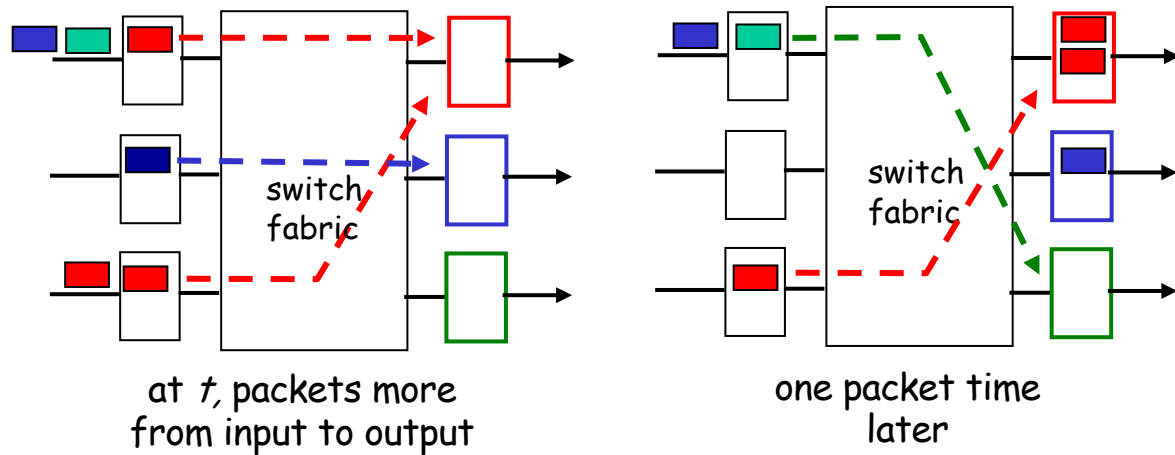
- **Scheduling discipline** chooses among queued datagrams for transmission



Priority scheduling - who gets best performance, network neutrality



Queuing: Output port queuing



- buffering when arrival rate via switch exceeds output line speed
- queuing (delay) and loss due to output port buffer overflow!



How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
 - e.g., $C = 10$ Gbps link: 2.5 Gbit buffer

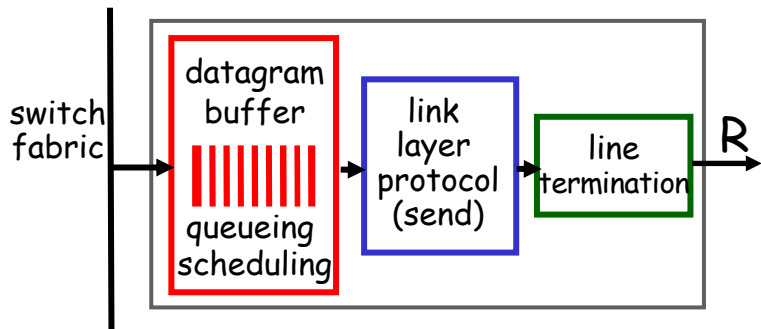
- more recent recommendation: with N flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

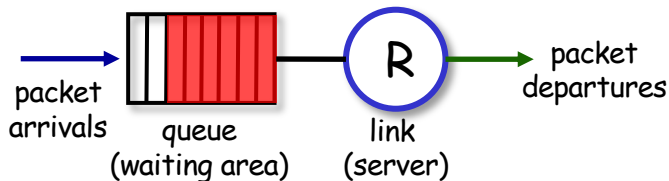
- but too much buffering can increase delays (particularly in home routers)
 - long RTTs: poor performance for real-time apps, sluggish TCP response
 - recall delay-based congestion control: “keep bottleneck link just full enough (busy) but no fuller”



Buffer Management



Abstraction: queue



buffer management:

- **drop:** which packet to add, drop when buffers are full
 - **tail drop:** drop arriving packet
 - **priority:** drop/remove on priority basis
- **marking:** which packets to mark to signal congestion (ECN, RED)



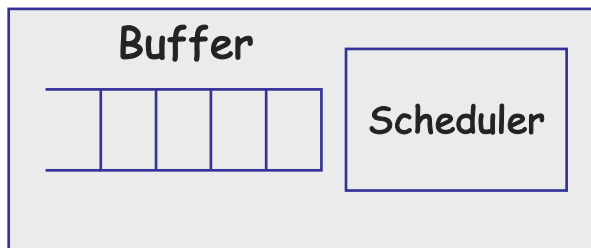
Scheduler

- One queue per “flow”
- Scheduler decides **when and from which queue to send a packet**
- Goals of a scheduling algorithm
 - **Fast!**
 - Depends on the policy being implemented (fairness, priority, etc.)



Simplest: FIFO router

- No classification
- **Drop-tail buffer management**: when buffer is full drop the incoming packet
- **First-In-First-Out (FIFO) Scheduling**: schedule packets in the same order they arrive

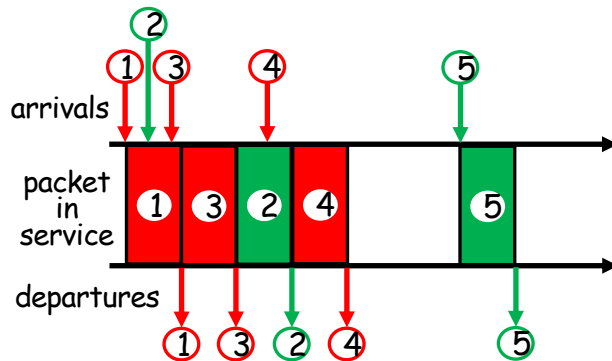
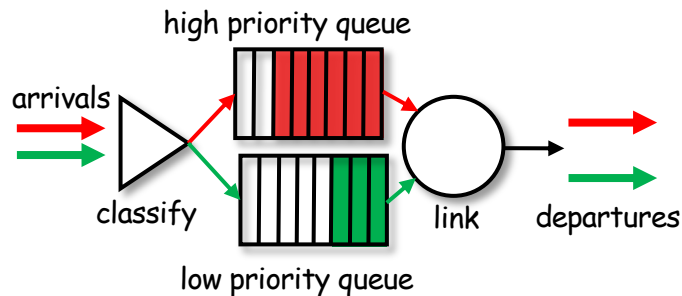




Scheduling policies: priority

Priority scheduling:

- arriving traffic classified, **queued by class**
 - any header fields can be used for classification
- **send packet from highest priority queue** that has buffered packets
 - FCFS within priority class

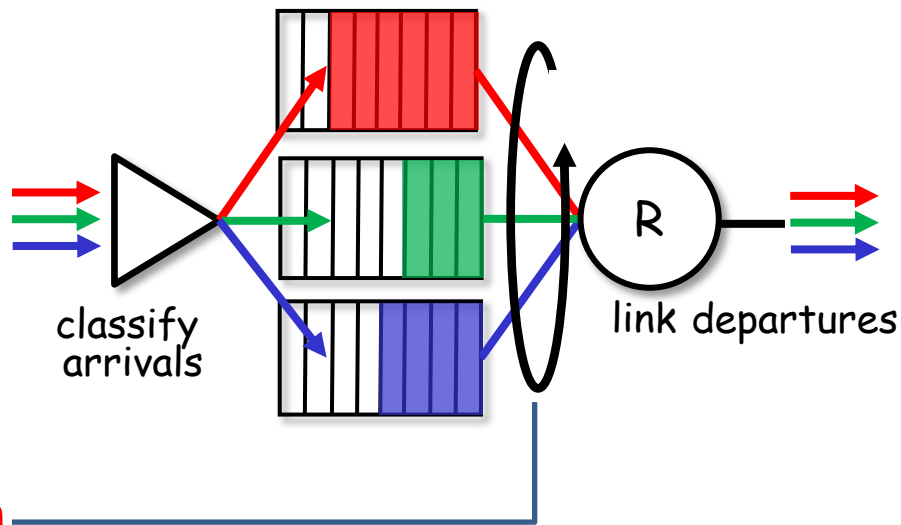




Scheduling policies: round robin

Round Robin (RR) scheduling:

- arriving traffic **classified, queued by class**
 - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, **sending one complete packet from each class (if available) in turn**





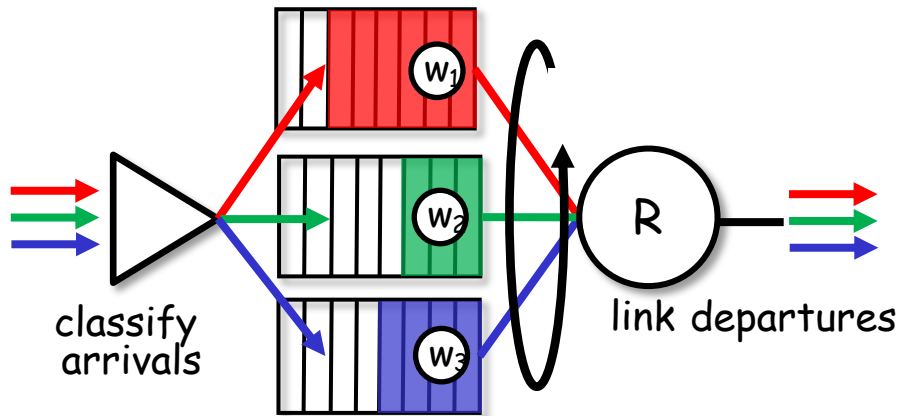
Scheduling policies: weighted fair queuing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class, i , has weight, w_i , and gets weighted amount of service in each cycle:

$$\frac{w_i}{\sum_j w_j}$$

- minimum bandwidth guarantee (per-traffic-class)



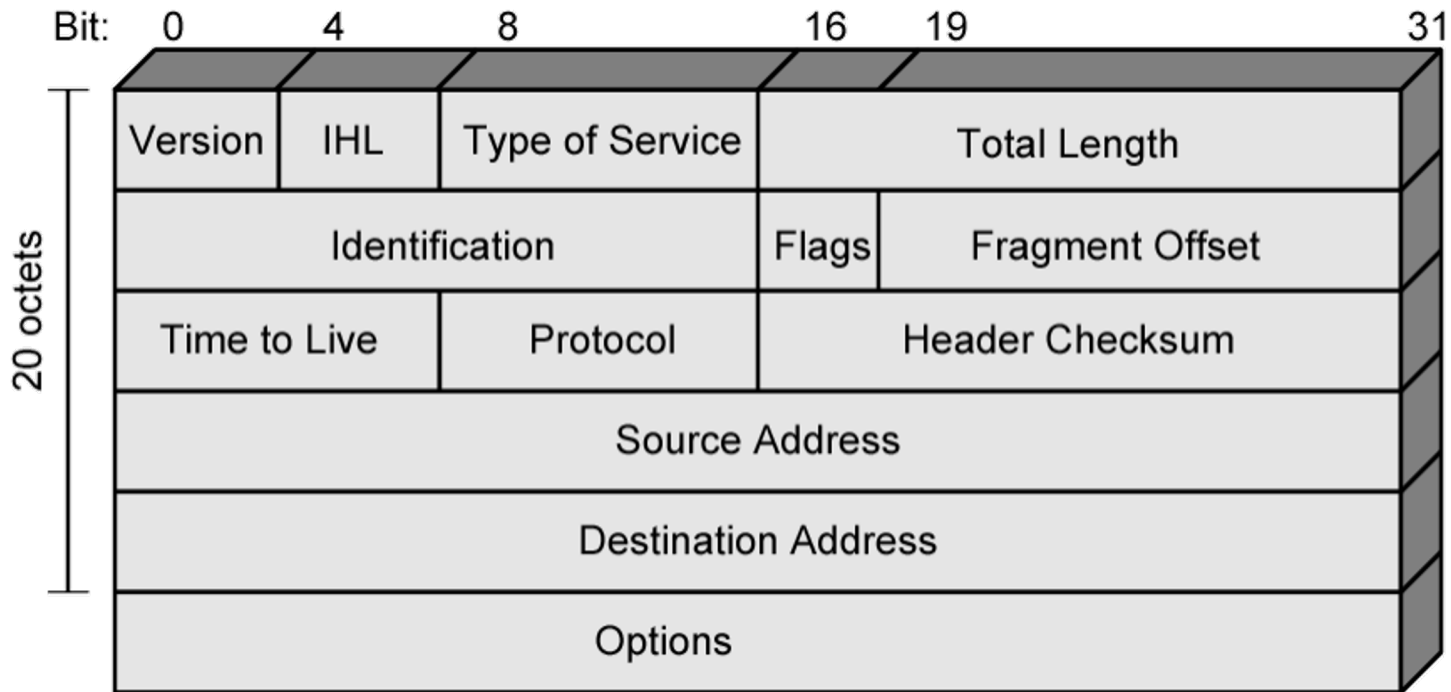


Outline

- Network Layer Functions
- Routers
- IP Packet Structure



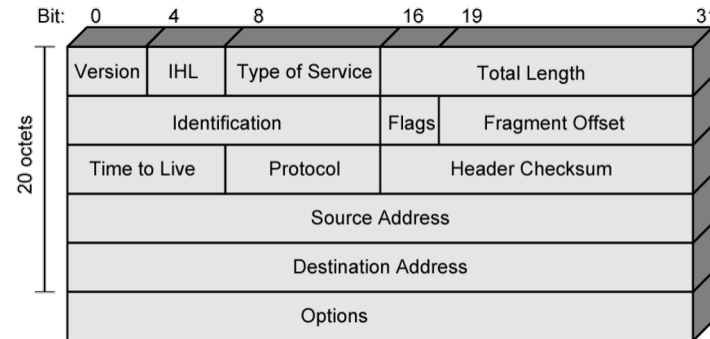
IPv4 Header





Header Fields (1)

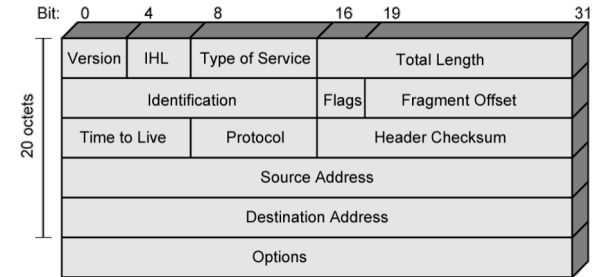
- **Version** (4 bits)
 - Currently 4
 - IPv6 - see later
- **Internet header length (IHL)** (4 bits)
 - In 32 bit words (4 octets)
 - Minimum fixed header (20 octets) + options
- **Type of service** (8 bits)
 - **Precedence**: 3 bits, 8 levels defined
 - **Reliability**: 1 bit, Normal or high
 - **Delay**: 1 bit, Normal or low
 - **Throughput**: 1 bit, Normal or high





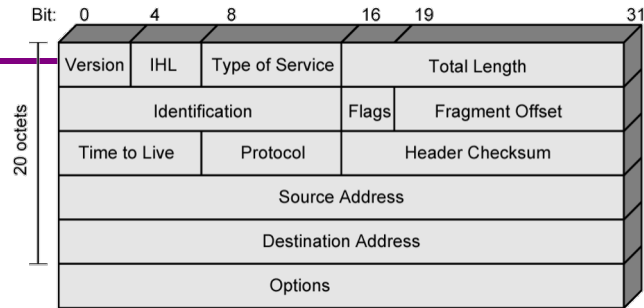
Header Fields (2)

- **Total length** (16 bits)
 - Of datagram, in octets
- **Identification** (16 bits)
 - Sequence number
 - Used with addresses and user protocol to identify datagram uniquely
- **Flags** (3 bits)
 - More flag, Don't fragment
- **Fragmentation offset** (13 bits)
- **Time to live** (8 bits)
- **Protocol** (8 bits)
 - Next higher layer to receive data field at destination





Header Fields (3)



- **Header checksum** (16 bits)
 - Complement sum of all 16 bit words in header
 - If not correct, router discards packets
 - Reverified and recomputed at each router, set to 0 during calculation.
(Why?)
- **Source address** (32 bits)
- **Destination address** (32 bits)
- **Options** (variable ≤ 40 octets)



Data Field

- Carries user data from next layer up
- Multiple of 8 bits long (i.e. octet)
- **Max length** of datagram (header + data) 65,535 octets



实验1——截止日期：4月22日晚23:59

- 实验1：可靠通信
- 提交方式：<https://selearning.nju.edu.cn/>（教学支持系统）

<p>教学支持系统</p> <ul style="list-style-type: none">▾ 2025 Spring<ul style="list-style-type: none">▸ 本科生一年级▸ 本科生二年级▸ 本科生三年级▸ 本科生四年级▸ 研究生一年级▸ 智能软件与工程学院	<p>互联网计算-智软院</p> <p>教师: 殷亚凤</p>	<p>实验1-可靠通信</p> <p>请将代码和实验报告打包提交!</p> <p>实验报告内容（以A4纸计算，不少于3页）：</p> <ol style="list-style-type: none">1. 实验名称2. 实验目的3. 实验内容4. 实验结果5. 核心代码6. 实验总结
	<p>实验作业</p> <p>实验1-可靠通信</p>	

- 命名：学号+姓名+实验*。
- 若提交遇到问题请及时发邮件或在下一次上课时反馈。



实验1——截止日期：4月22日晚23:59

Lab 1: Reliable Communication

Overview

In the assignment you are going to build a reliable communication library in Switchyard that will consist of 3 agents. At a high level, a **blaster** will send data packets to a **blastee** through a **middlebox**. As you should all know by now, IP only offers a best-effort service of delivering packets between hosts. This means all sorts of bad things can happen to your packets once they are in the network: they can get lost, arbitrarily delayed or duplicated. Your communication library will provide additional delivery guarantees by implementing some basic mechanisms at the blaster and blastee. Let's move on to the details.



实验1——截止日期：4月22日晚23:59


Lab 1: Reliable Communication

Your Tasks

In the source directory for this exercise, you can find the starter files: `middlebox.py`, `blastee.py` and `blaster.py`.

Your reliable communication library will implement the following features to provide additional guarantees: 1. ACK mechanism on `blastee` for each successfully received packet 2. A fixed-size sliding window on `blaster` 3. Coarse timeouts on `blaster` to resend non-ACK'd packets

Further details will be discussed in each Task.

The sentences marked with  are related to the content of your report. Please pay attention.



实验1——截止日期：4月22日晚23:59

Lab 1: Reliable Communication

Task 1: Preparation

Initiate your project with our template. [Start the task here](#)

Task 2: Middlebox

Implement the features of middlebox. [Start the task here](#)

Task 3: Blastee

Implement the features of blastee. [Start the task here](#)

Task 4: Blaster

Implement the features of blaster. [Start the task here](#)

Task 5: Running your code

Make sure that your blaster, blastee and middlebox function correctly. [Start the task here](#)

Please carefully read the [FAQ](#) section, for more specific details regarding the implementations.



提问

Q & A

殷亚凤

智能软件与工程学院

苏州校区南雍楼东区225

yafeng@nju.edu.cn , <https://yafengnju.github.io/>



南京大學
NANJING UNIVERSITY