CS4450

Computer Networks: Architecture and Protocols

Lecture 16
THE Internet Protocol
Switch Architecture

Rachit Agarwal

Announcements

- **Prelim on next Thursday**
    - Will cover everything discussed in class and problem sets
- **Problem Set 3 solutions** are released
- **Problem Set 4** is out (solutions will be released Thursday)
- **Practice prelim** is released (solutions will be released Thursday)
- We will release our **first project** this week (not included in prelim)
    - Recall: not graded, but we will provide all the help
- You must have received an email for **mid-semester feedback** for the class
    - Please provide frank and constructive feedback
    - Recall: I already know I am an asshole; iterating it might not be useful
    - What you like? Where could we improve?

What do we know so far [1] …

- **Network performance metrics**
    - Transmission delay, propagation delay, queueing delay, bandwidth
- **Sharing networks**
    - Circuit switching, packet switching, and associated tradeoffs
    - **Why** is Internet packet switched?
- **Architectural principles and design goals**
    - Layering principle, End-to-end principle, Fate sharing principle
    - Many important design goals from David Clark's paper
        - And many important missing goals
- **Addressing**
    - **Link layer MAC names,** and scalability challenges at the Internet
    - **Network layer IP addresses**: three requirements, aggregation, CIDR

What do we know so far [2] …

- **Link Layer**
    - Sharing a Broadcast medium, associated challenges, CSMA/CD
    - Link layer addressing: MAC names
    - **Why** Frames? **Why** Switched Ethernet?
    - The Spanning Tree Protocol (STP)

- **Network Layer**
    - **Why Network Layer? Why not just use STP across the Internet?**
    - **Routing Tables:** A collection of spanning trees, one per destination
    - **Generating Valid Routing tables (within a domain):**
        - Global view (Link-State Protocol), and limitations
        - Local view (Distance-vector Protocol)
    - **Generating Valid Routing tables (across domains):**
        - Border Gateway Protocol, Internet structure, routing policies

Next lecture

- You may not realize this but ….
    - **We have learnt a lot of material!!!!!**

- **Next lecture is very very very ….**
    - **very very very very ….**
    - **important**
    - Please attend

- I will discuss **how everything we have covered so far FITS TOGETHER** …
    - … into an end-to-end design

- **You will feel awesome — I promise!**

Goals for Today's Lecture

- Understand IP (the Internet Protocol)
    - Packet Header as a network "interface"

- Understand switch architecture

Network Layer

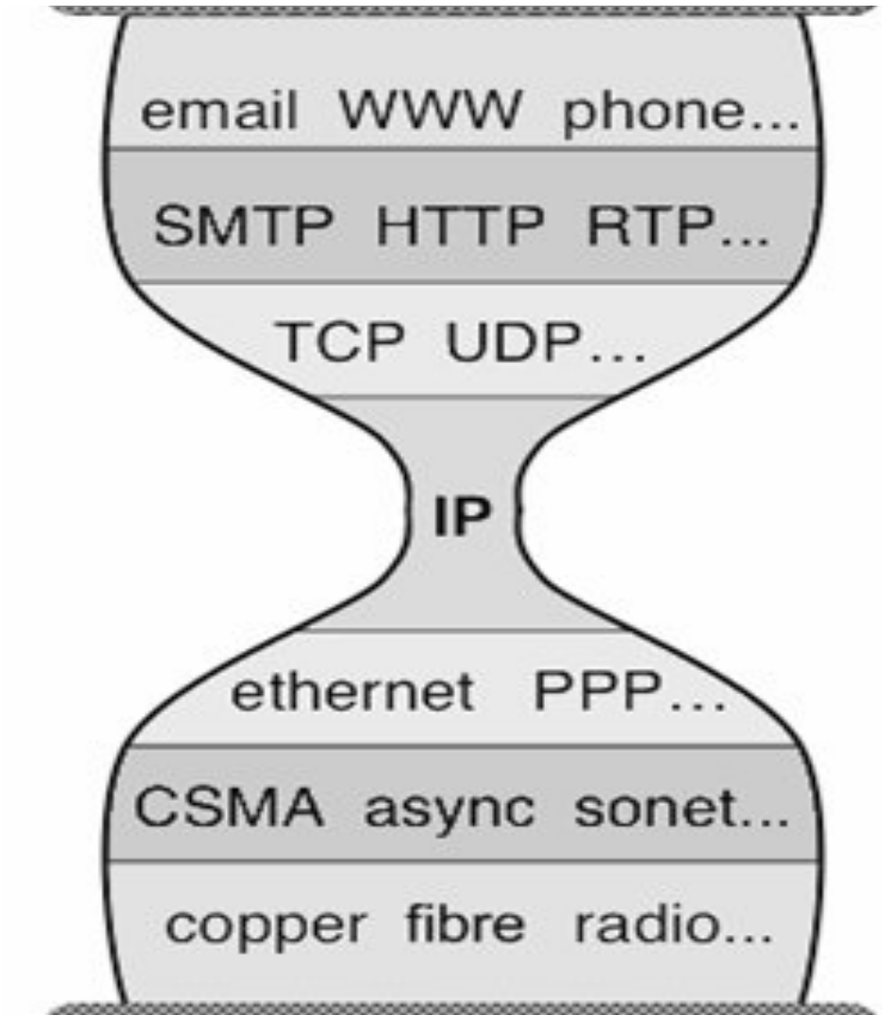
- THE functionality: **delivering the data**

- **THE protocol: Internet Protocol (IP)**

- **Achieves its functionality (delivering the data), using three ideas:**
    - **Addressing** (IP addressing)
    - **Routing** (using a variety of protocols)
    - **Packet header as an interface** (Encapsulating data into packets)

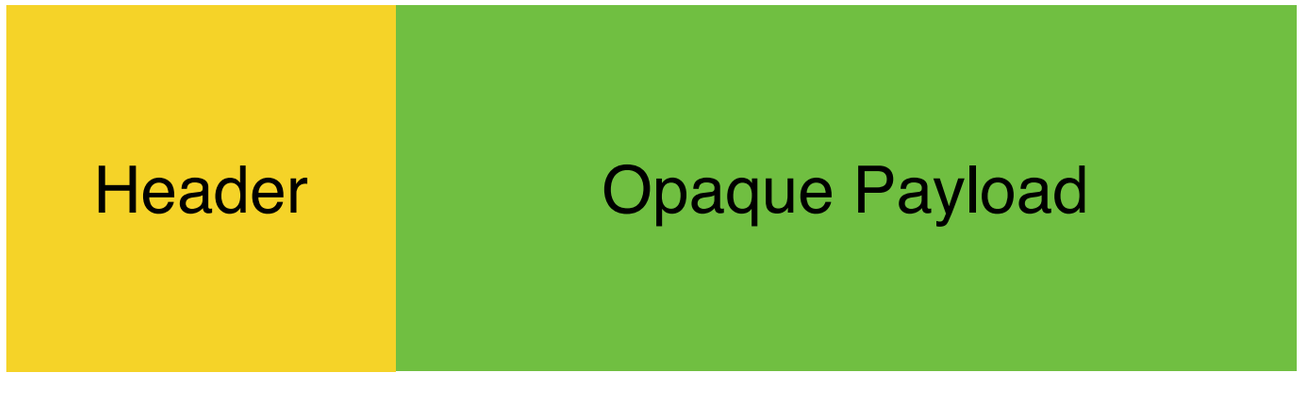
Internet Protocol

- THE functionality: **delivering the data**
- **THE protocol: Internet Protocol (IP)**
- Unifying protocol



What is Designing IP?

- Syntax: format of packet
    - Nontrivial part: packet “header”
    - Rest is opaque payload (**why opaque?**)



- Semantics: meaning of header fields
    - Required processing

Packet Header as Interface

- Think of packet header as interface
    - Only way of passing information from packet to switch
- Designing interfaces:
    - What task are you trying to perform?
    - What information do you need to accomplish it?
- Header reflects information needed for basic tasks

What Tasks Do We Need to Do?

- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with the packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

Reading Packet Correctly

- Where does the header end?
- Where the the packet end?
- What protocol are we using?
    - Why is this so important?

Getting to the Destination

- Provide destination address
- Should this be location or identifier (name)?
    - And what's the difference?
- If a host moves should its address change?
    - If not, how can you build scalable Internet?
    - If so, then what good is an address for identification?

Getting Response Back to Source

- Source address
- Necessary for routers to respond to source
    - When would they need to respond back?
        - Failures!
    - Do they really need to respond back?
        - How would the source know if the packet has reached the destination?

Carry Data

- Payload!

Questions?

List of Tasks

- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

Telling Destination How to Process Packet

- Indicate which protocols should handle packet
- What layers should this protocol be in?
- What are some options for this today?
- How does the source know what to enter here?

Special Handling

- Type of service, priority, etc.
- Options: discuss later

Dealing With Problems

- Is packet caught in loop?
    - TTL
- Header corrupted:
    - Detect with Checksum
    - What about payload checksum?
- Packet too large?
    - Deal with fragmentation
    - Split packet apart
    - Keep track of how to put together

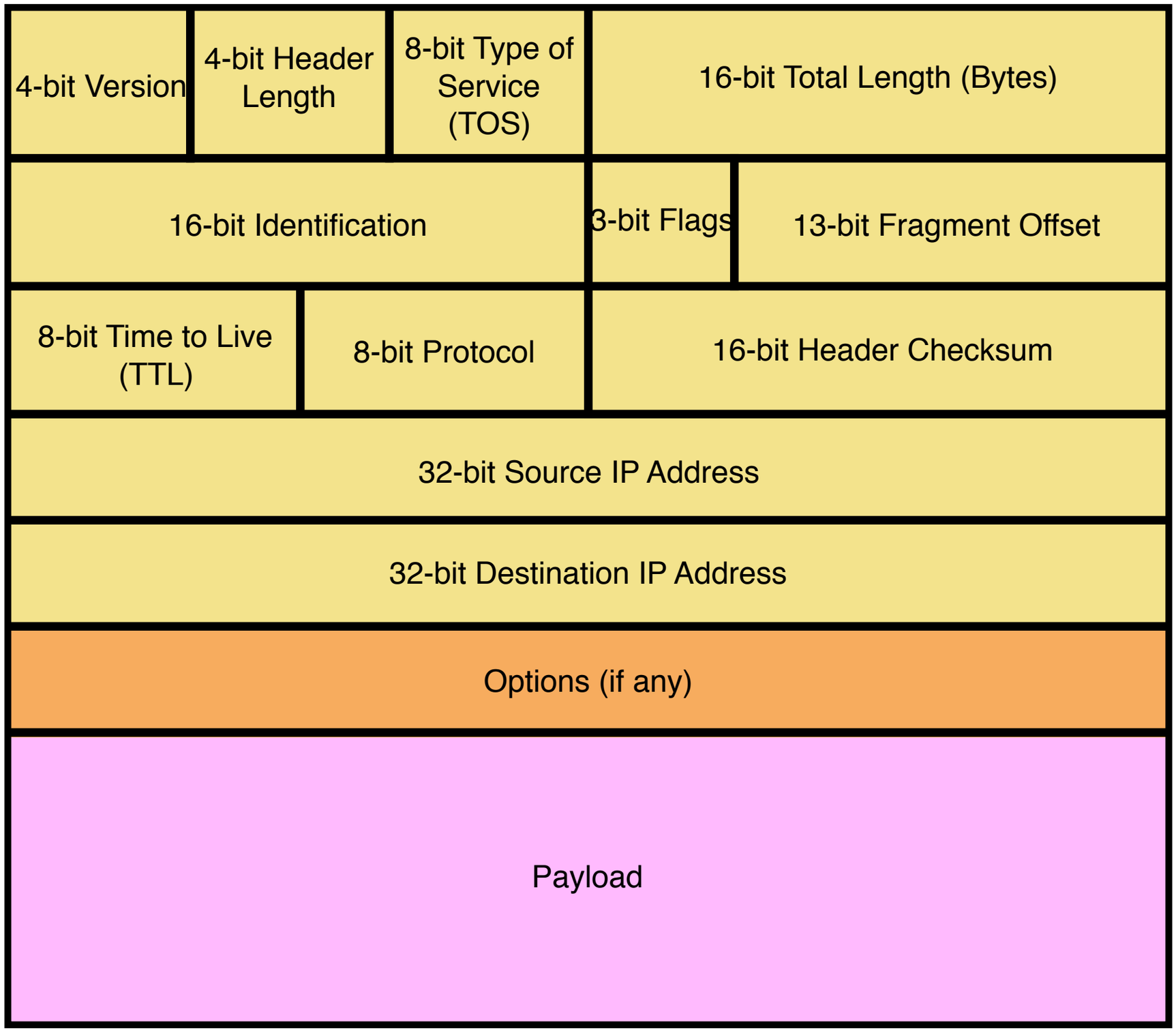
Are We Missing Anything?

- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

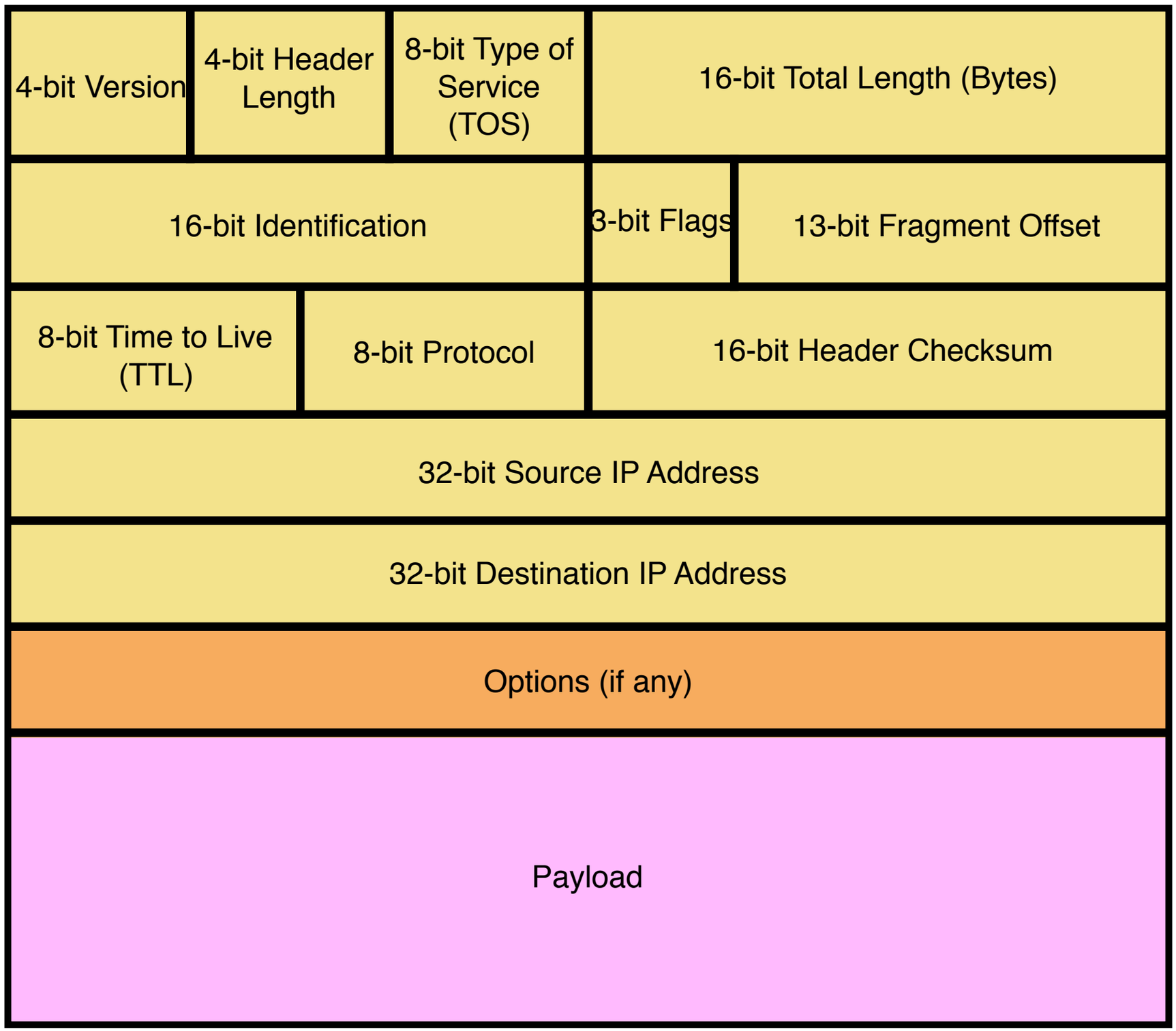
From Semantics to Syntax

- The past few slides discussed the information the header must provide
- Will now show the syntax (layout) of IPv4 header, and discuss the semantics in more detail

IP Packet Structure



20 Bytes of Standard Header, then Options



Next Set of Slides

- Mapping between tasks and header fields
- Each of these fields is devoted to a task
- Let’s find out which ones and why…

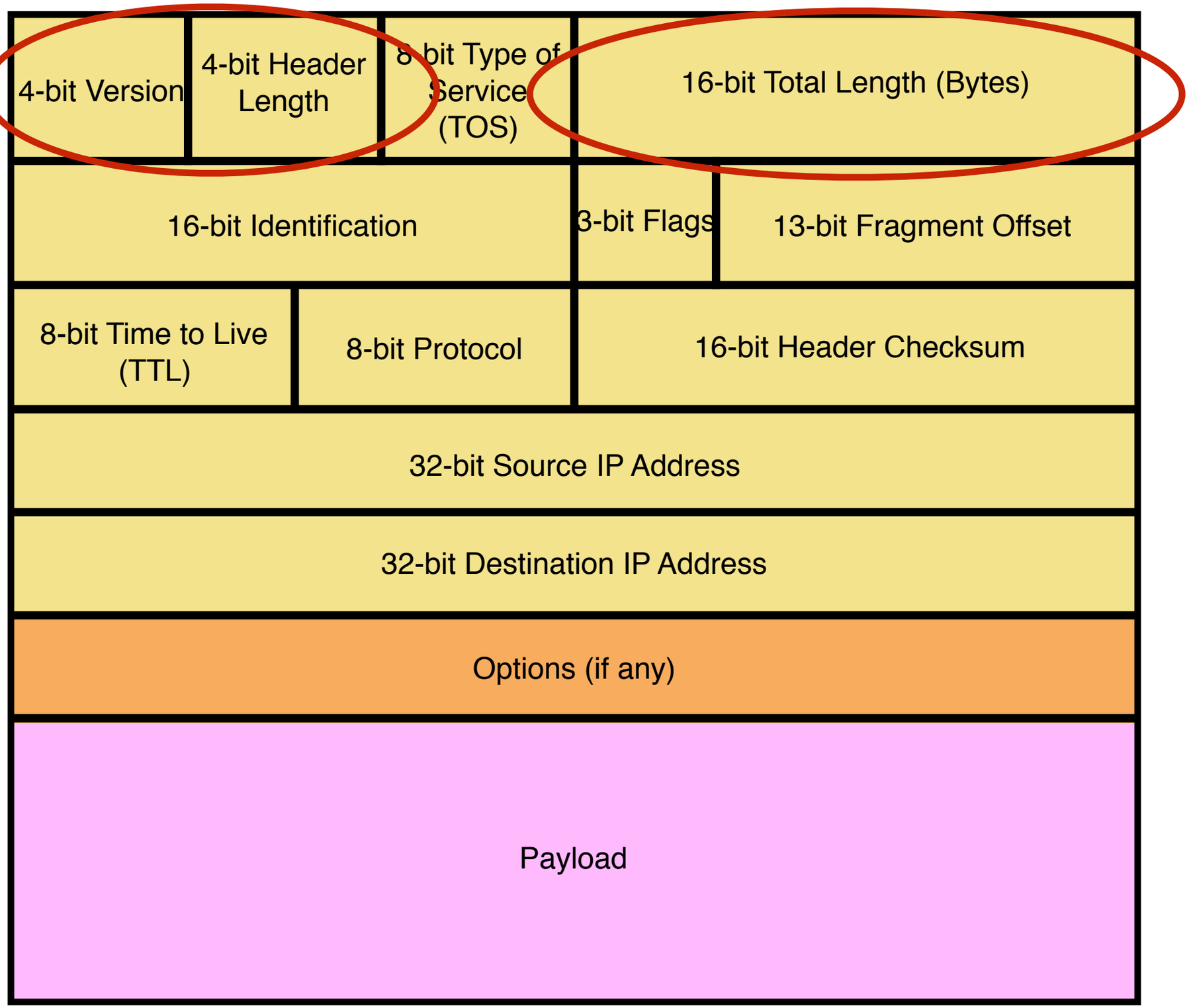
Go Through Tasks One-by-One

- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

Read Packet Correctly

- **Version number** (4 bits)
    - Indicates the version of the IP protocol
    - Necessary to know what other fields to expect
    - Typically “4” (for IPv4), and sometimes “6” (for IPv6)
- **Header length** (4 bits)
    - Number of 32-bit words in the header
    - Typically “5” (for a 20-byte IPv4 header)
    - Can be more when IP options are used
- **Total length** (16 bits)
    - Number of bytes in the packet
    - Maximum size is 65,535 bytes (2^16 -1)
    - … though underlying links may impose smaller limits

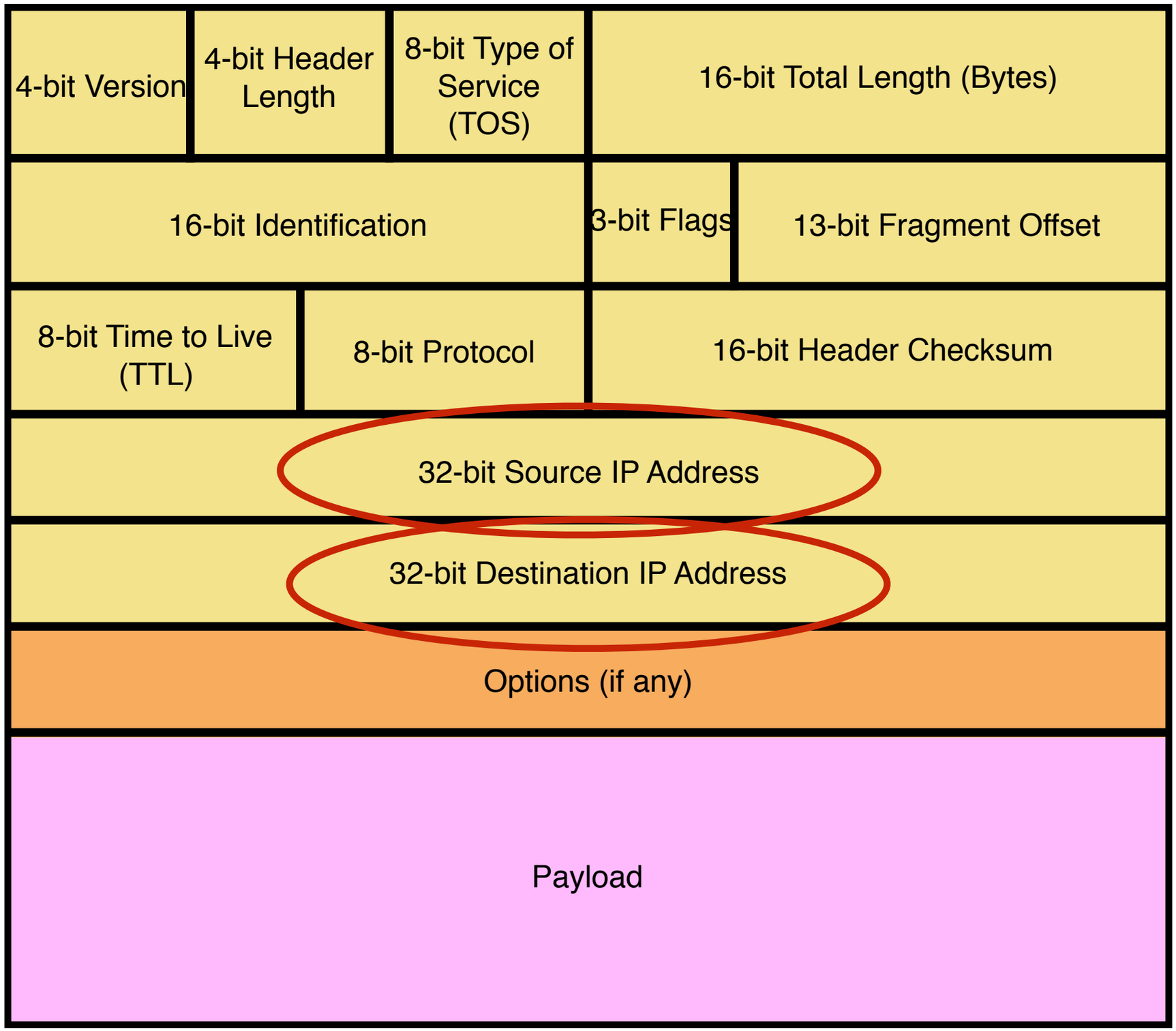
Fields for Reading Packet Correctly



Getting Packet to Destination and Back

- **Two IP addresses**
    - Source IP address (32 bits)
    - Destination IP address (32 bits)
- **Destination Address**
    - Unique locator for the receiving host
    - Allows each node to make forwarding decisions
- **Source Address**
    - Unique locator for the sending host
    - Recipient can decide whether to accept packet
    - Enables recipient to send a reply back to the source

Fields for Reading Packet Correctly



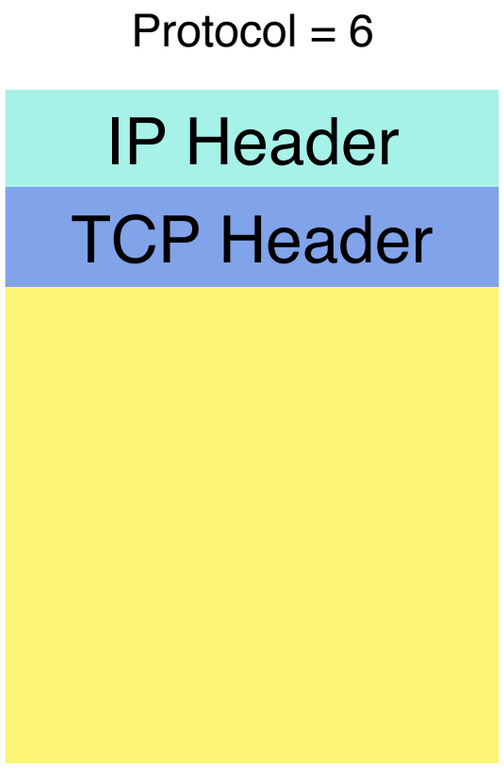
Questions?

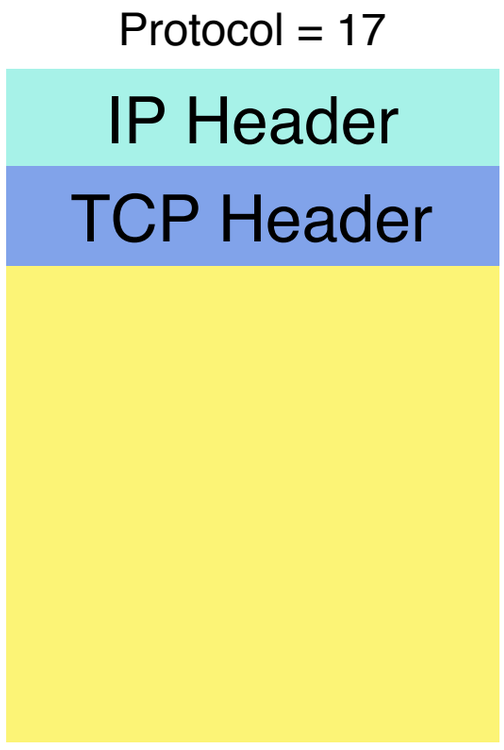
List of Tasks

- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- **Tell host what to do with packet once arrived**
- Specify any special network handling of the packet
- Deal with problems that arise along the path

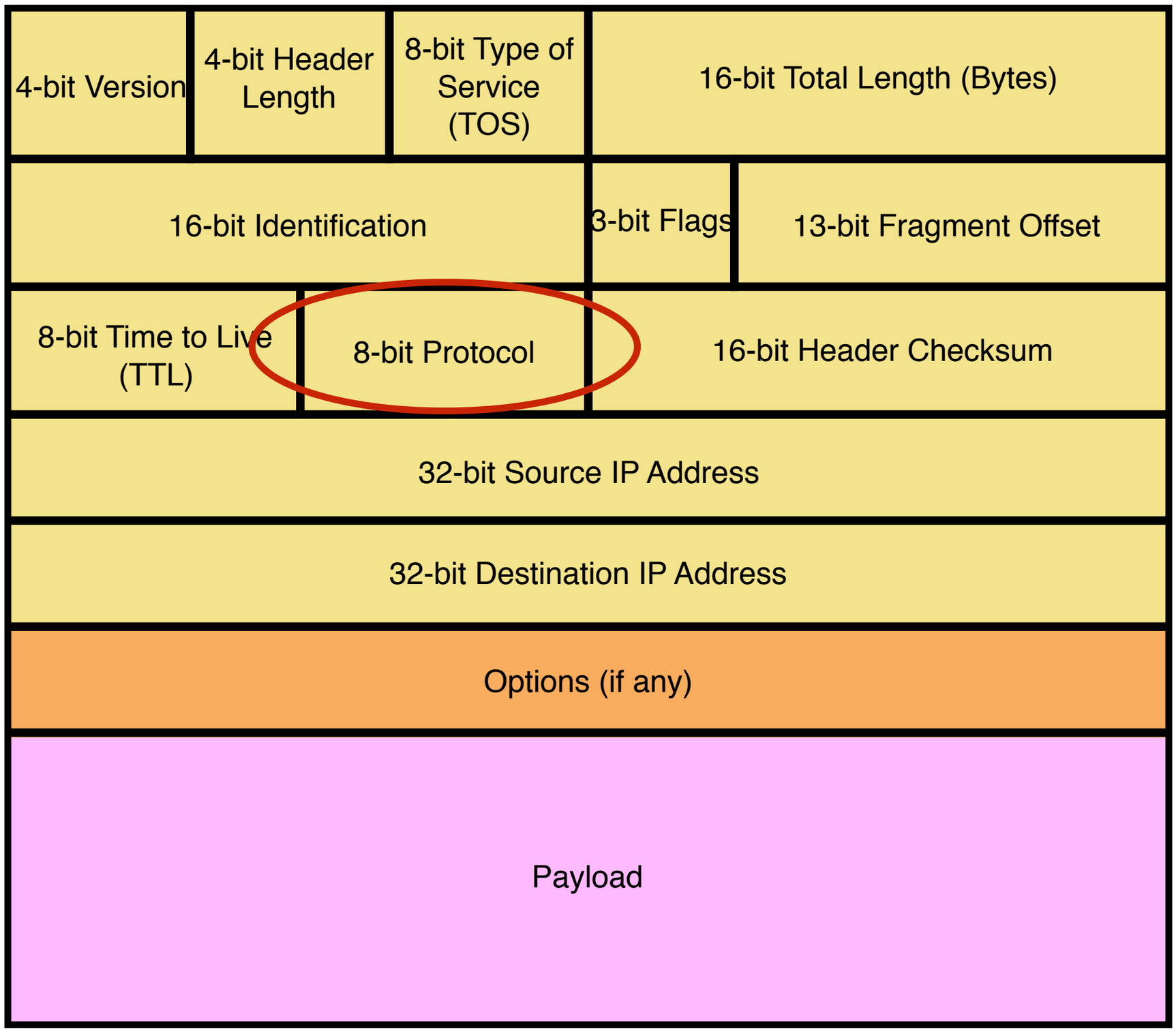
Telling Host How to Handle Packet

- **Protocol (8 bits)**
  - Identifies the higher level protocol
  - Important for demultiplexing at receiving host
- **Most common examples**
  - E.g., “6” for the Transmission Control Protocol (TCP)
  - E.g., “17” for the User Datagram Protocol





Fields for Reading Packet Correctly



Special Handling

- **Type-of-Service (8-bits)**
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer
  - Has been redefined several times, no general use
- **Options**
  - Ability to specify other functionality
  - Extensible format

Examples of Options

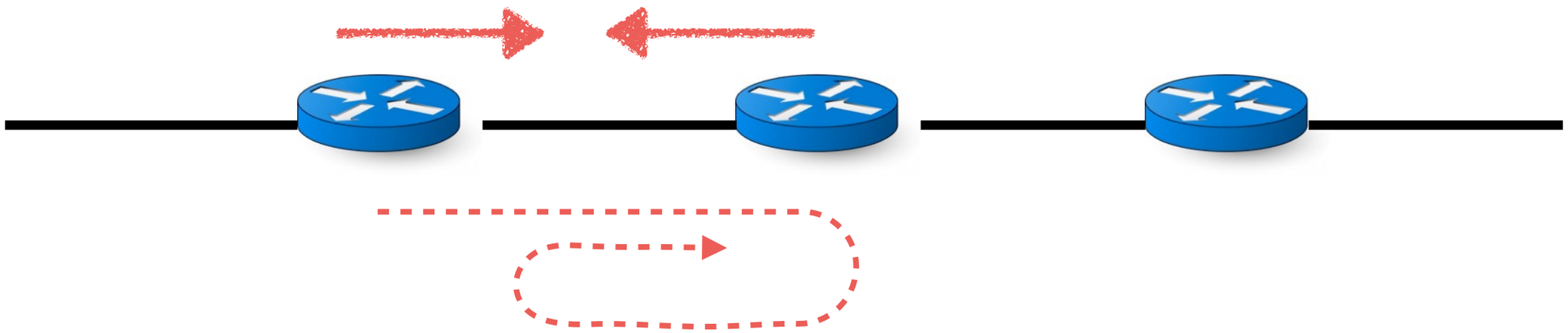
- Record Route
- Strict Source Route
- Loose Source Route
- Timestamp
- Traceroute
- Router Alert
- ...

Potential Problems

- Header Corrupted: **Checksum**
- Loop: **TTL**
- Packet too large: **Fragmentation**

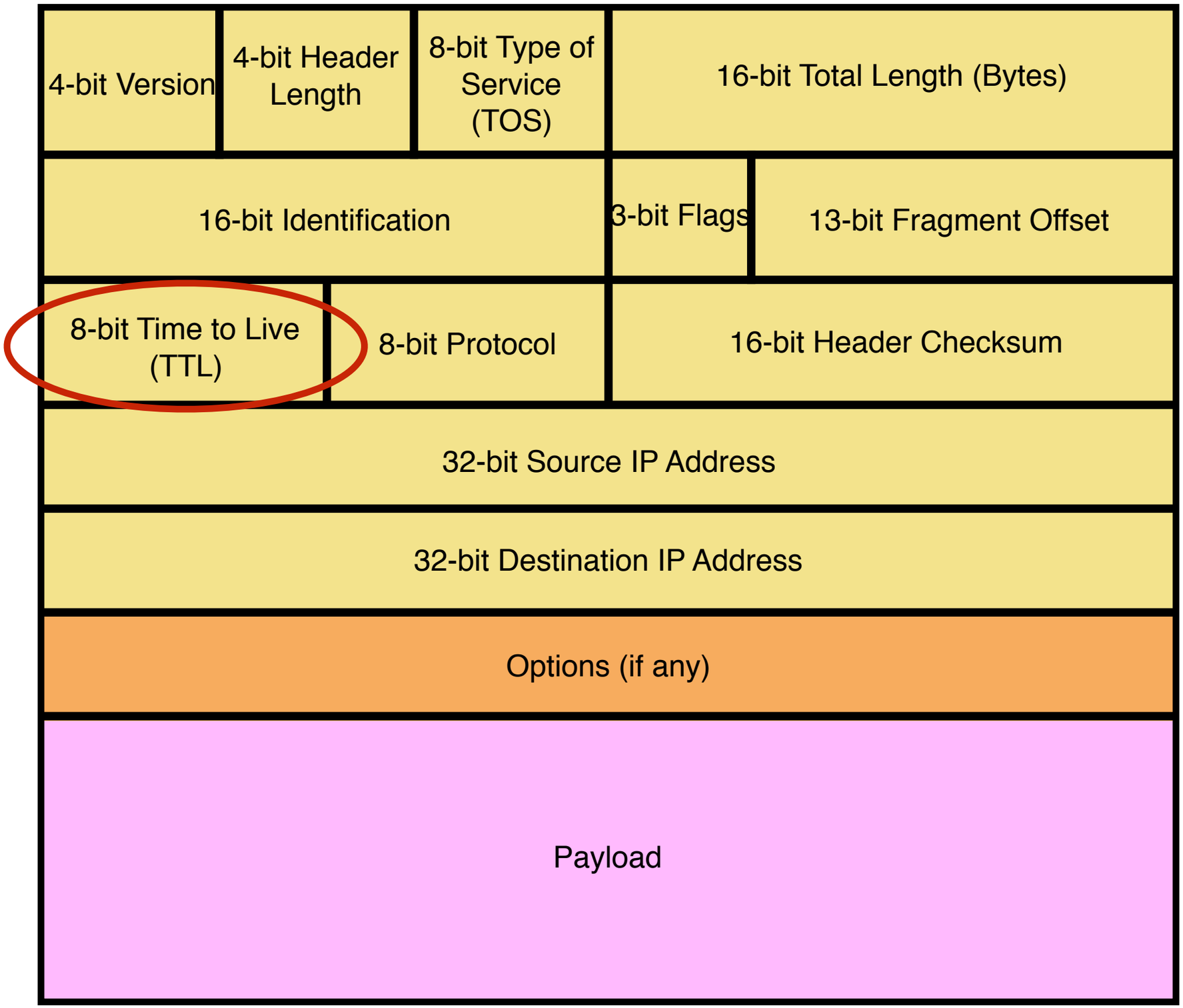
Preventing Loops

- Forwarding loops cause packets to cycle forever
    - As these accumulate, eventually consume all capacity



- Time-to-live (TTL) Field (8-bits)
    - Decremented at each hop, packet discarded if reaches 0
    - … and “time exceeded” message is sent to the source
        - Using “ICMP” control message; basis for traceroute

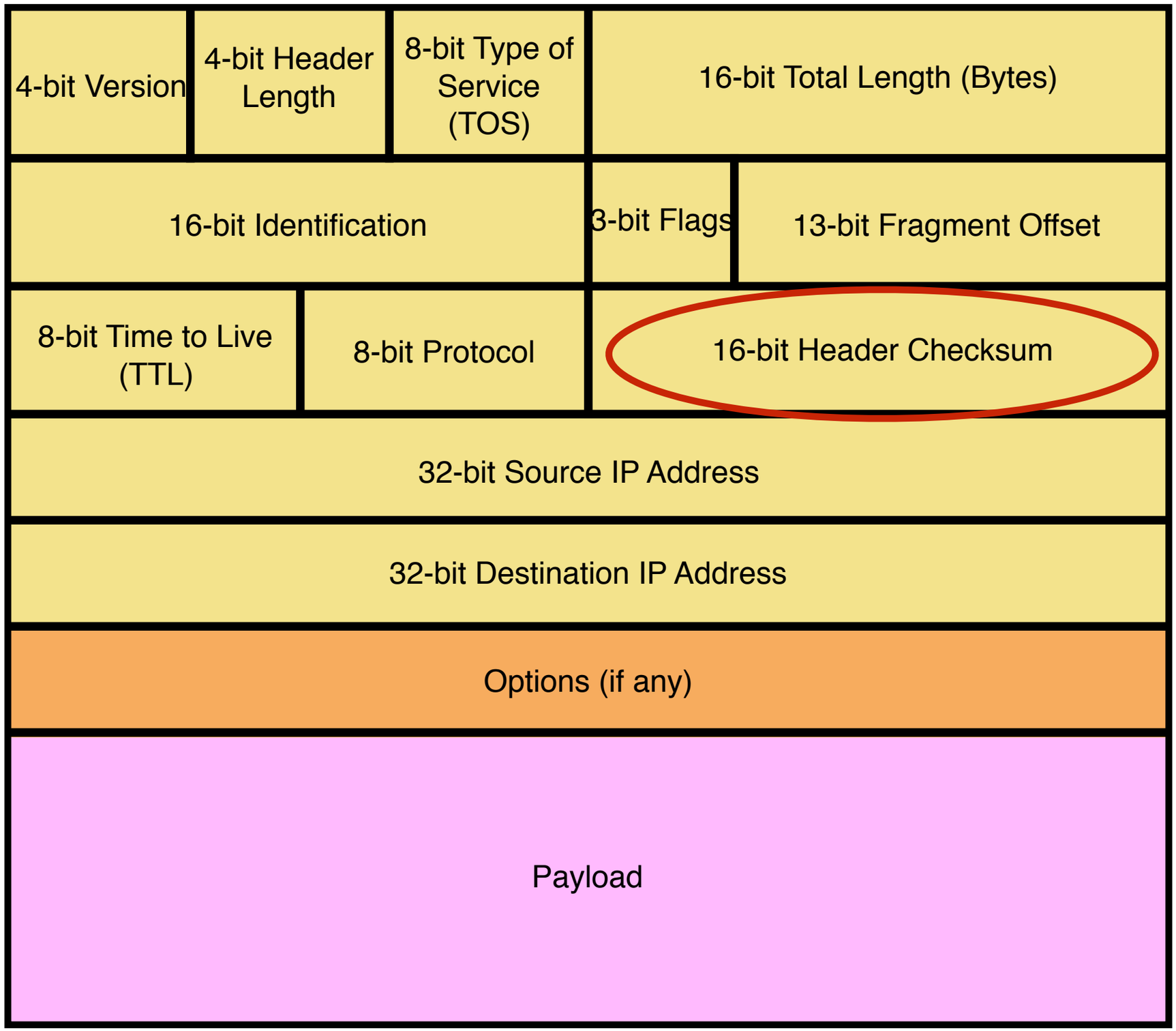
TTL Field


4-bit Version
4-bit Header Length
8-bit Type of Service (TOS)
16-bit Total Length (Bytes)
16-bit Identification
3-bit Flags
13-bit Fragment Offset
8-bit Time to Live (TTL)
8-bit Protocol
16-bit Header Checksum
32-bit Source IP Address
32-bit Destination IP Address
Options (if any)
Payload

Header Corruption

- Checksum (16 bits)
    - Particular form of checksum over packet header
- If not correct, router discards packets
    - So it doesn’t act in bogus information
- Checksum recalculated at every router
    - Why?
    - Why include TTL?
    - Why only header?

Checksum Field



Packet Header as an interface

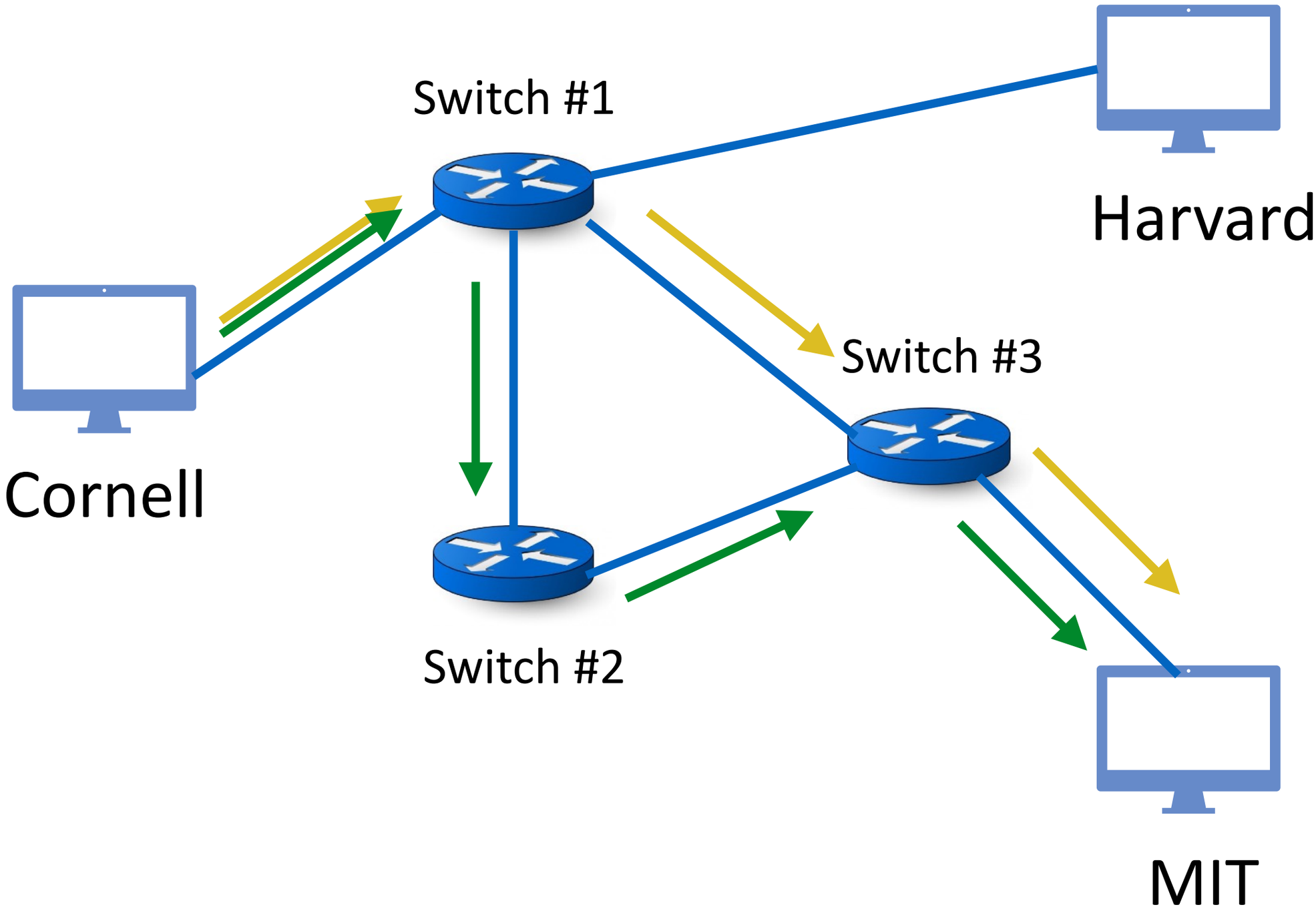
- **Useless to learn the header format by heart**
  - If you remember the tasks that need to be performed …
  - Understanding **why** header format is what it is …
  - In general: if you understand the problem, solution is easy
  - As the problem evolves, you will know where to look for a solution
- **Transition from IPv4 to IPv6**
  - Gradually happening …
  - If you want to learn a bit, see backup slides

Switch/Router Architecture

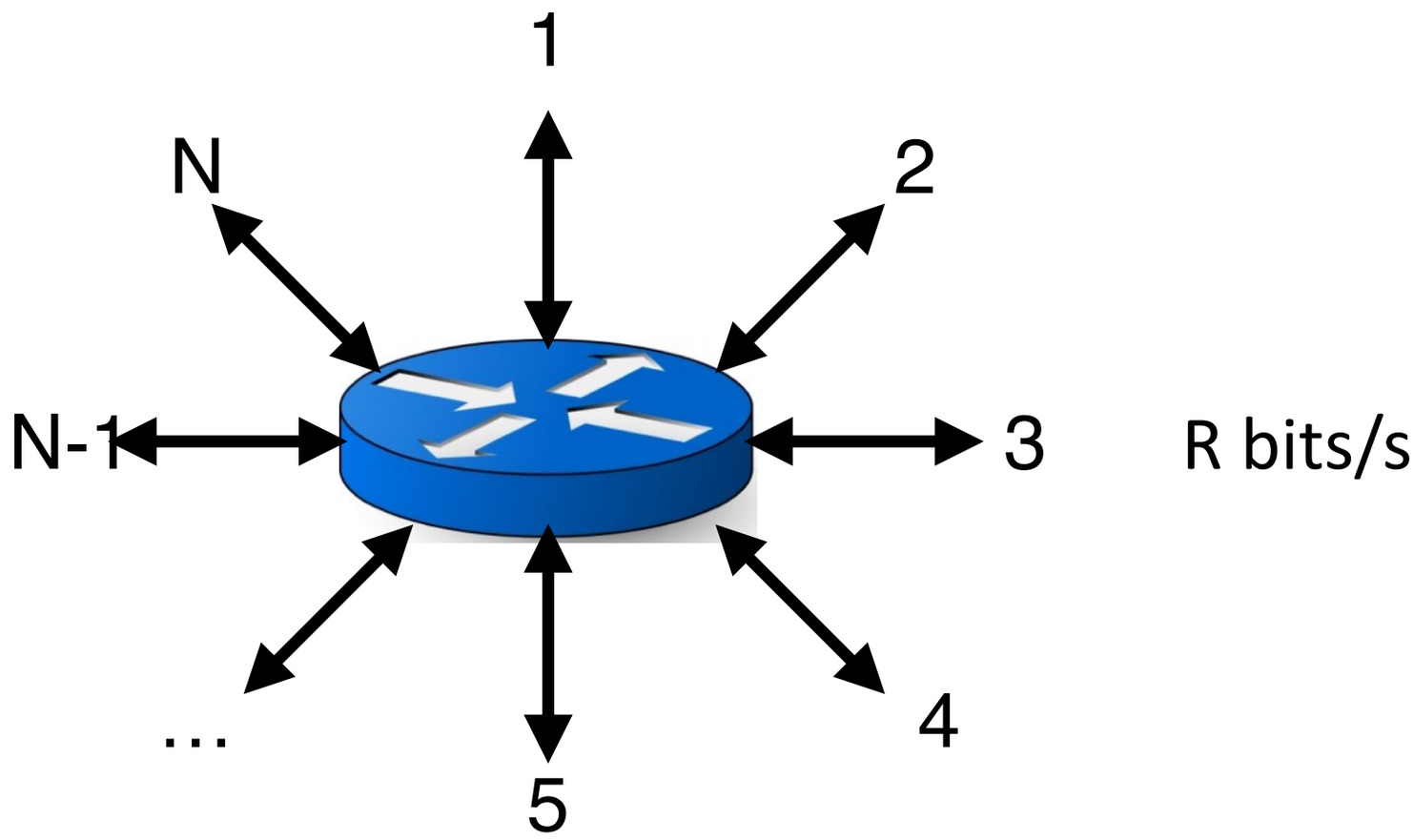
IP Routers and Switches (used interchangeably today)

- Core building block of Internet infrastructure
- $120B+ industry
- Vendors: Cisco, Huawei, Juniper, Alcatel-Lucent (account for >90%)

Recap: Routers Forward Packets

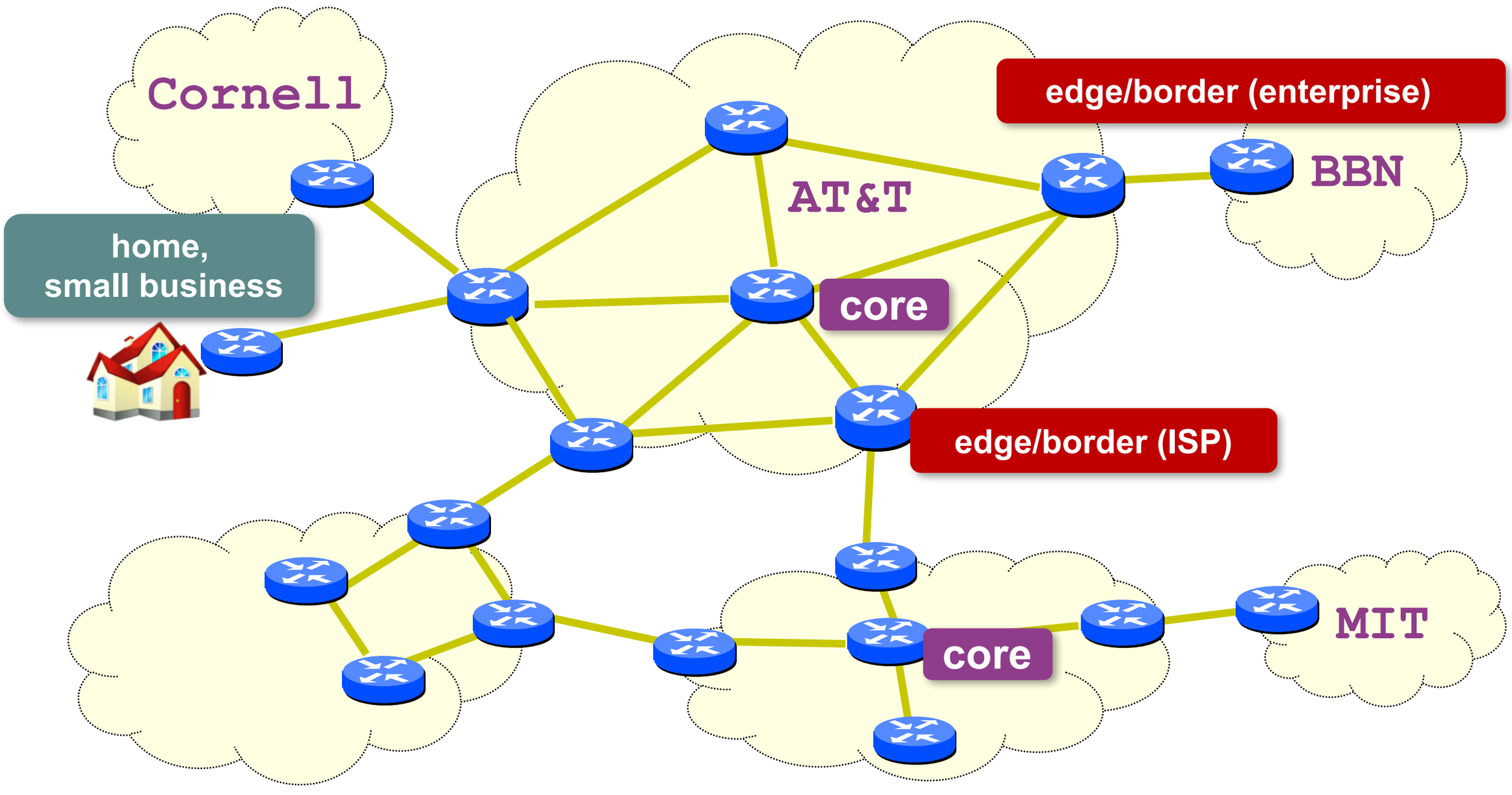


Router Definitions



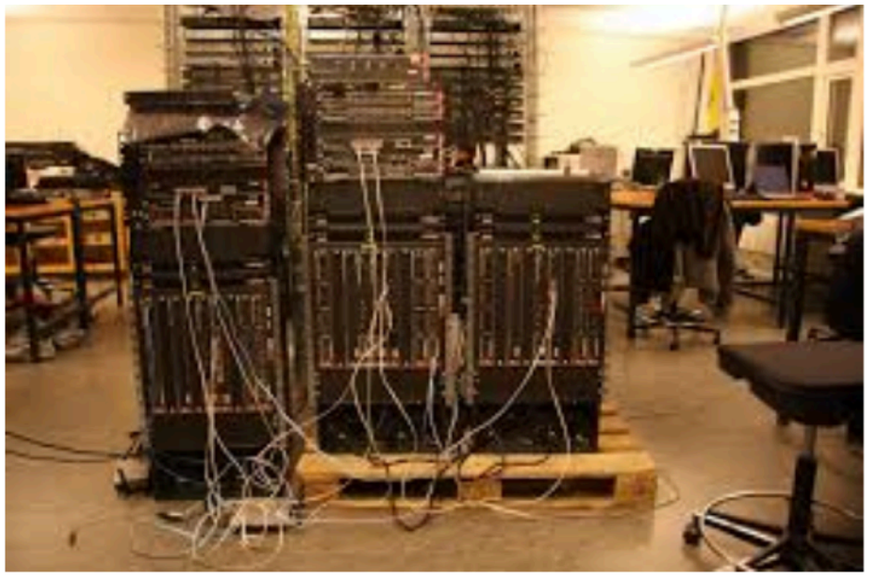
- N = No. Of external router ports
- R = bandwidth (“line rate”) of a port
- Router capacity = NxR

Networks and Routers



Examples of Routers (core)

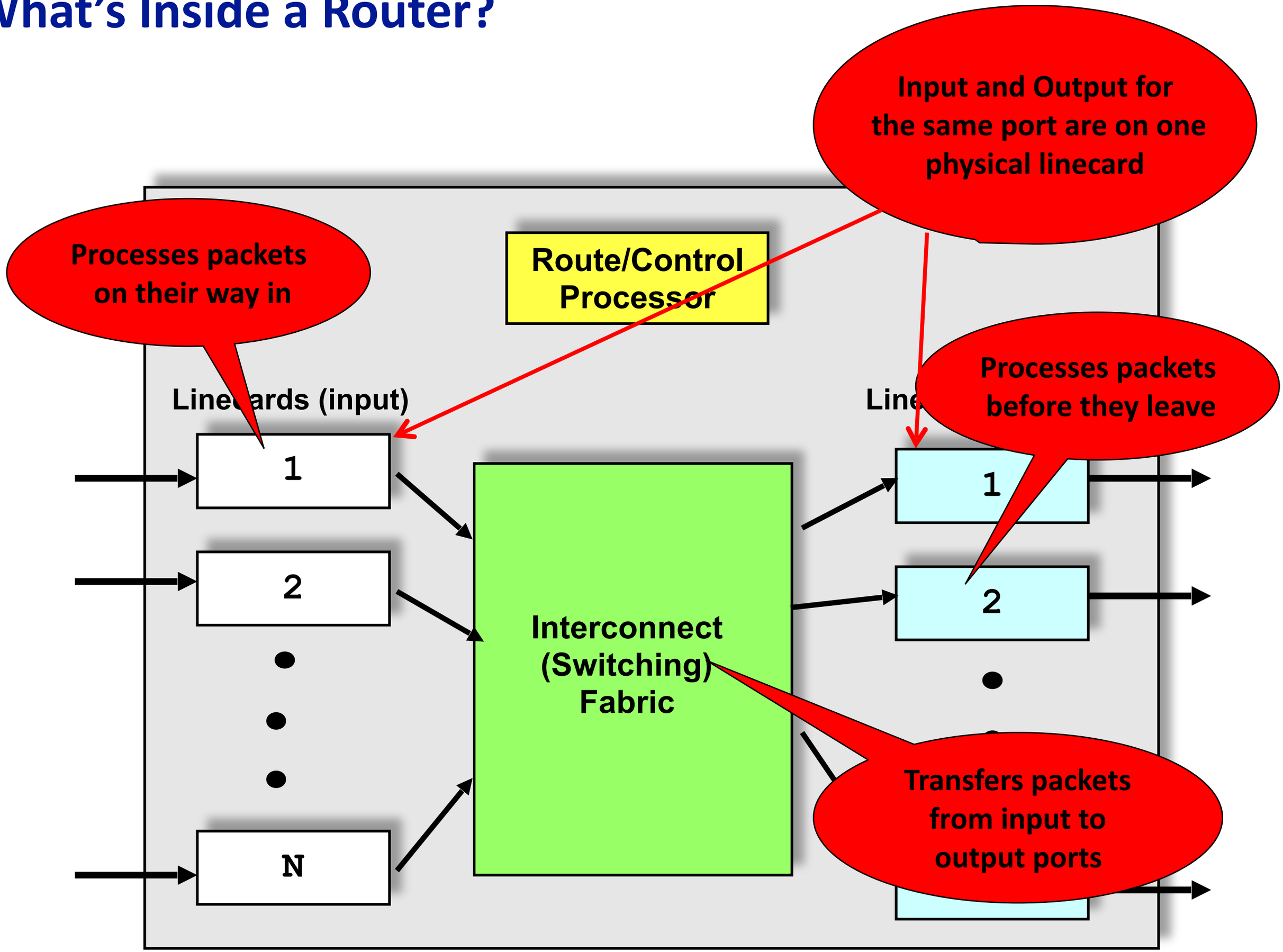
- **Core:** Cisco CRS
    - R = 10/40/100 Gbps
    - NR = 922 Tbps
    - Netflix: 0.7 GB/hr (1.5Mb/s)
    - ~600 million concurrent Netflix users



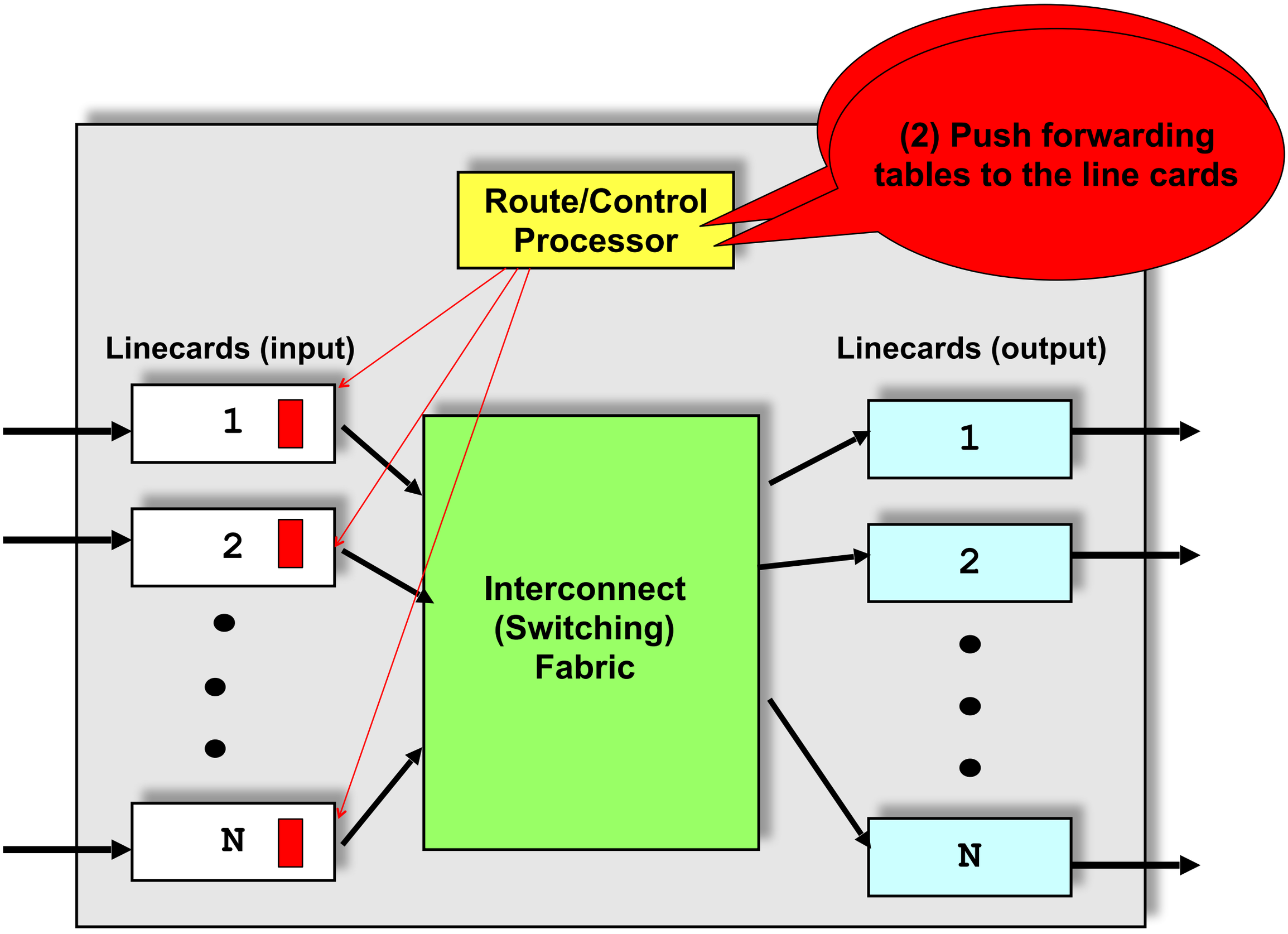
- **Edge (ISP):** Cisco ASR
    - R = 1/10/40 Gbps
    - NR = 120 Gbps

- **Edge (enterprise):** Cisco 3945E
    - R = 10/100/1000 Mbps
    - NR < 10 Gbps

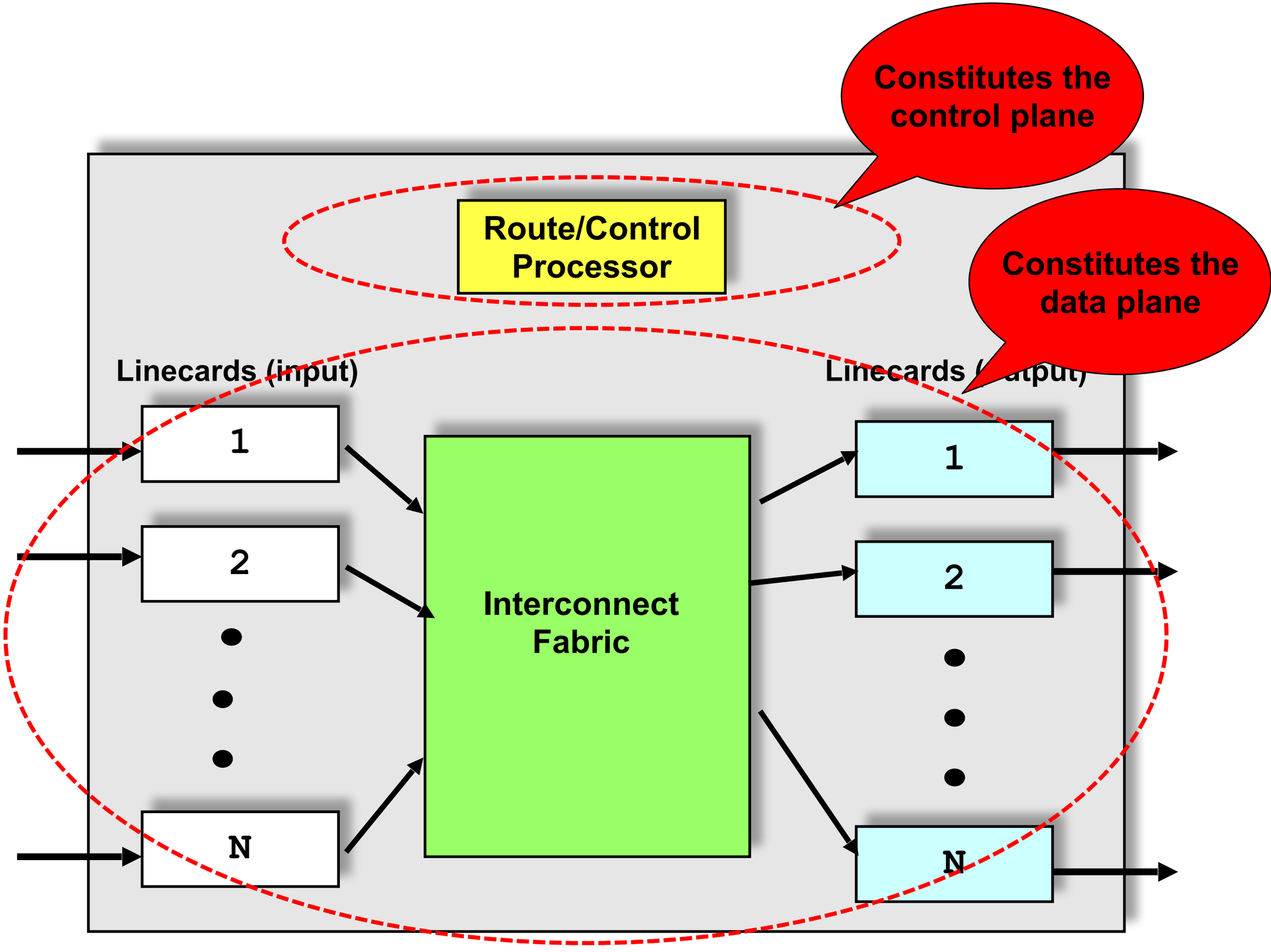
What's Inside a Router?



What's Inside a Router?



What's Inside a Router?



Input Line Cards: Tasks

- Receive incoming packets (physical layer stuff)
- Update the IP header
  - TTL, Checksum (maybe some other fields)
- Lookup the output port for the destination IP address
- Queue the packet at the switch fabric

Challenge: Speed!

- 100B packets @ 40Gbps => packet every 20 nano secs!

- Typically implemented with specialized hardware
  - ASICs, specialized “network processors”

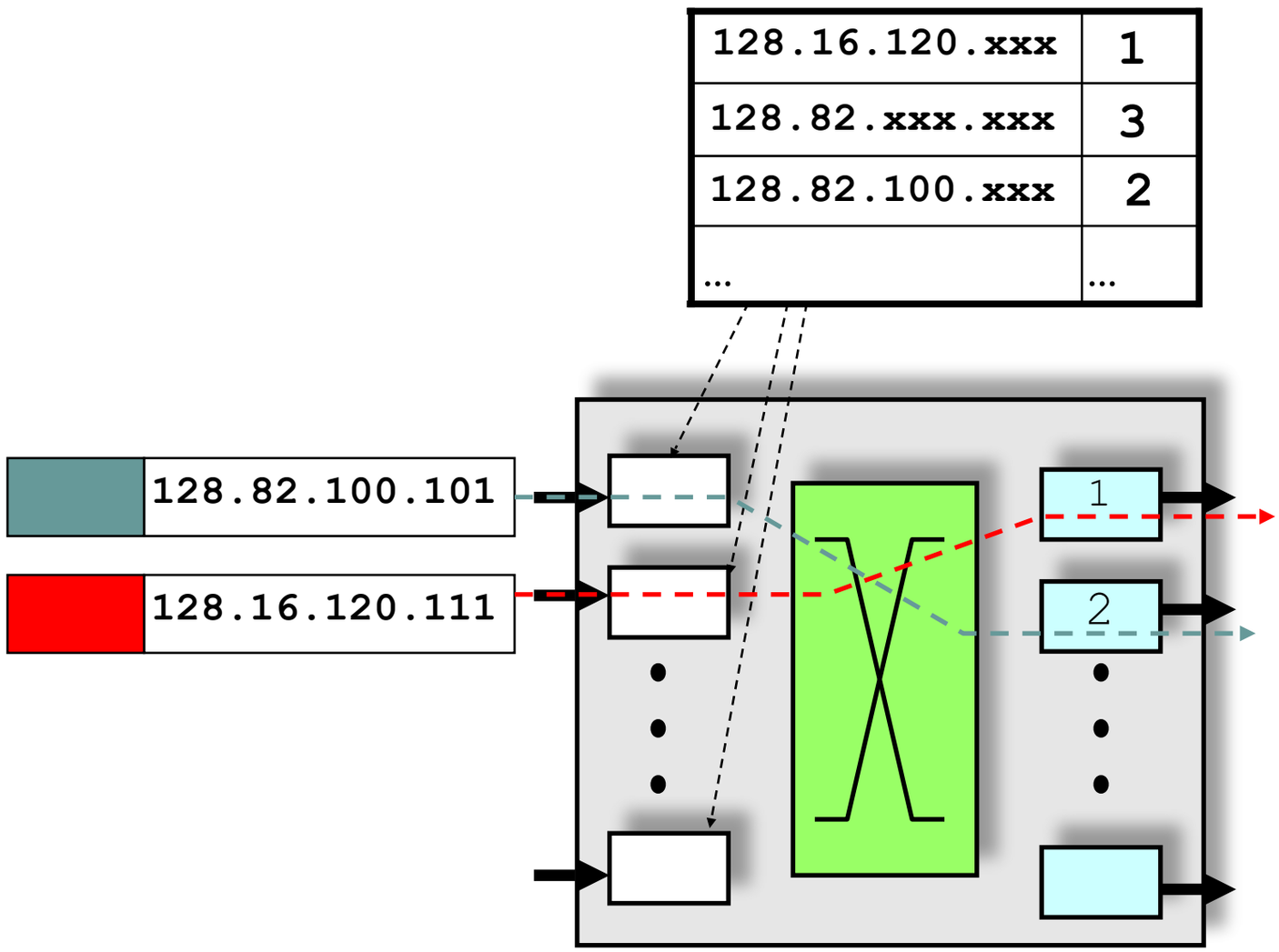
Looking up the Output Port

- Upon receiving a packet
    - Inspect the destination IP address in the header
    - Index into the routing/forwarding table
    - If no match, select the **default route**
    - Forward packet out appropriate interface

- **Default route**
    - Configured to cover cases where no matches
    - Allows small tables at edge (w/o routing algorithms)
        - **if it isn't on my subnet, send it to my ISP**

Scaling the Lookup

- Recall: For scalability, addresses are **aggregated**
- **Longest Prefix match**
  - Find the entry with matching “longest prefix” with destination address



Finding a Match

- Incoming packet destination: 201.143.7.0

Prefix	Port
201.143.0.0/22	Port 1
201.143.4.0.0/24	Port 2
201.143.5.0.0/24	Port 3
201.143.6.0/23	Port 4

Finding a Match: Covert to Binary

- Incoming packet destination: 201.143.7.0

11001001	10001111	00000111	11010010

Routing Table

201.143.0.0/22

11001001	10001111	000000 - -	- - - - - - - -

201.143.4.0/24

11001001	10001111	00000100	- - - - - - - -

201.143.5.0/24

11001001	10001111	00000101	- - - - - - - -

201.143.6.0/23

11001001	10001111	0000011-	- - - - - - - -

Finding a Match: Covert to Binary

- Incoming packet destination: 201.143.7.0

11001001	10001111	00000111	11010010

Routing Table

201.143.0.0/22

11001001	10001111	000000 - -	- - - - - - - -

201.143.4.0/24

11001001	10001111	00000100	- - - - - - - -

201.143.5.0/24

11001001	10001111	00000101	- - - - - - - -

201.143.6.0/23

11001001	10001111	00000 11-	- - - - - - - -

Finding a Match: Covert to Binary

- Incoming packet destination: 201.143.7.0

11001001	10001111	00000111	11010010

Routing Table

201.143.0.0/22

11001001	10001111	000000--	--------

201.143.4.0/24

11001001	10001111	00000100	--------

201.143.5.0/24

11001001	10001111	00000101	--------

201.143.6.0/23

11001001	10001111	0000011-	--------

Longest Prefix Match

- Incoming packet destination: 201.143.7.0

11001001	10001111	00000111	11010010

Routing Table

201.143.0.0/22

11001001	10001111	000000- -	- - - - - - - -

201.143.4.0/24

11001001	10001111	00000100	- - - - - - - -

201.143.5.0/24

11001001	10001111	00000101	- - - - - - - -

201.143.6.0/23

11001001	10001111	0000011-	- - - - - - - -

Check an address against all destination prefixes and select the prefix it matches with on the most bits

Finding the Match Efficiently

- Testing each entry to find a match scales poorly
    - Roughly (number of entries) × (number of bits)

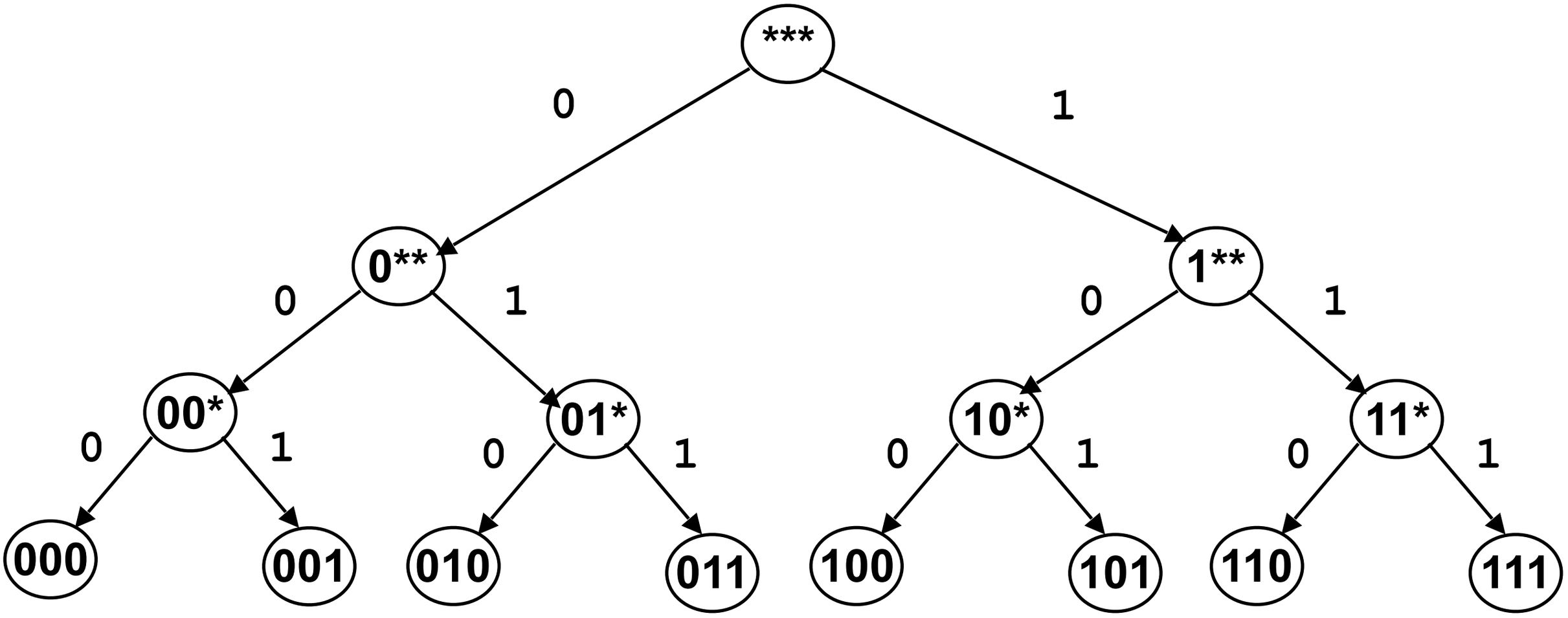
- Must leverage tree structure of binary strings
    - Set up tree-like data structure
    - Called a **TRIE**
    - We will briefly discuss it; more details in text
    - In case you are interested ….

Consider Four 3-Bit Prefixes

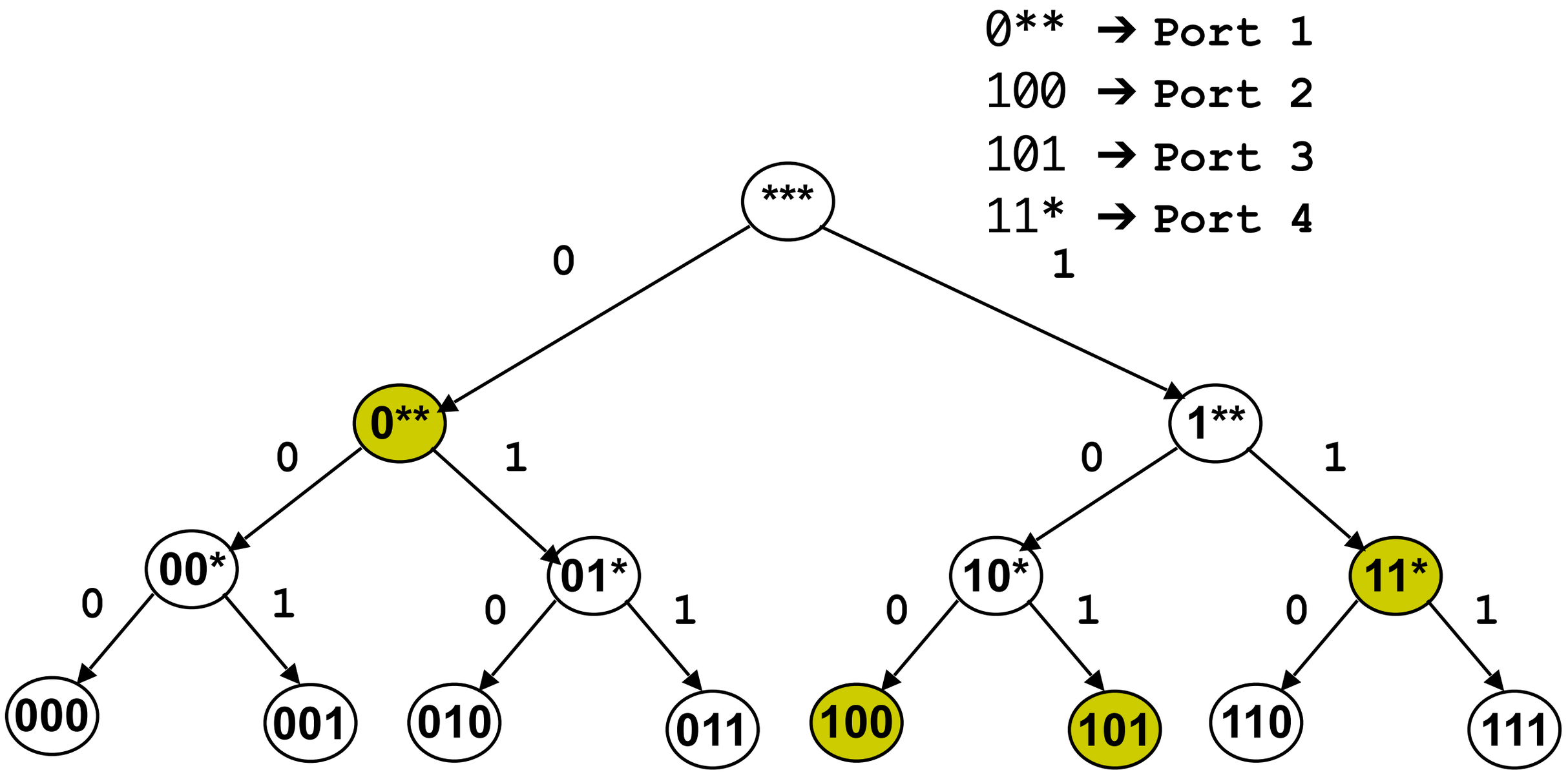
- Just focusing on the bits where all the action is….

- 0** → Port 1
- 100 → Port 2
- 101 → Port 3
- 11* → Port 4

Tree Structure

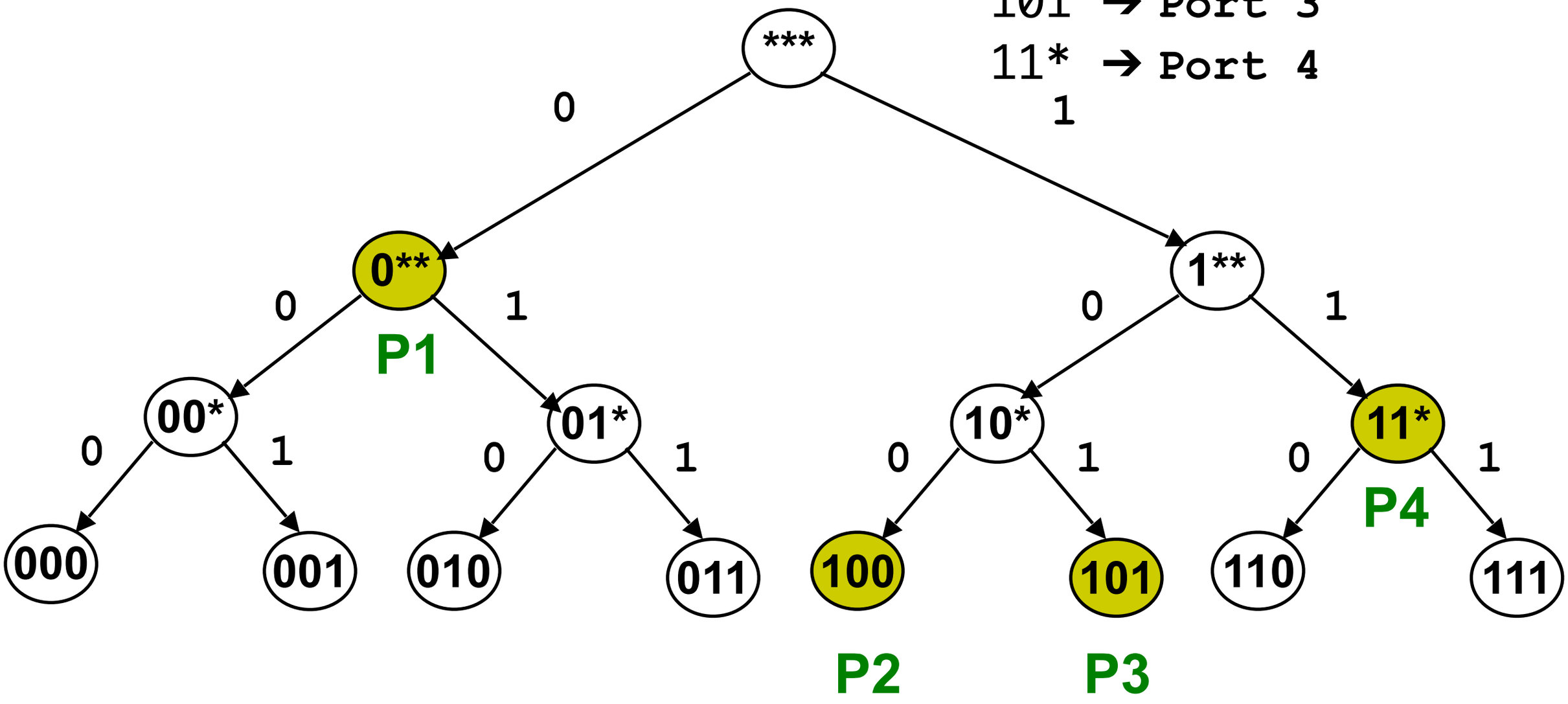


Walk Tree: Stop at Prefix Entries



Walk Tree: Stop at Prefix Entries

0** → Port 1
100 → Port 2
101 → Port 3
11* → Port 4



walking trees takes O(#bits)

Longest Prefix Match in Real Routers

- Real routers use far more advanced/complex solutions
    - But what we discussed is the starting point

- With many heuristics and optimizations that leverage real-world patterns
    - Some destinations more popular than others
    - Some ports lead to more destinations
    - Typical fix granularities

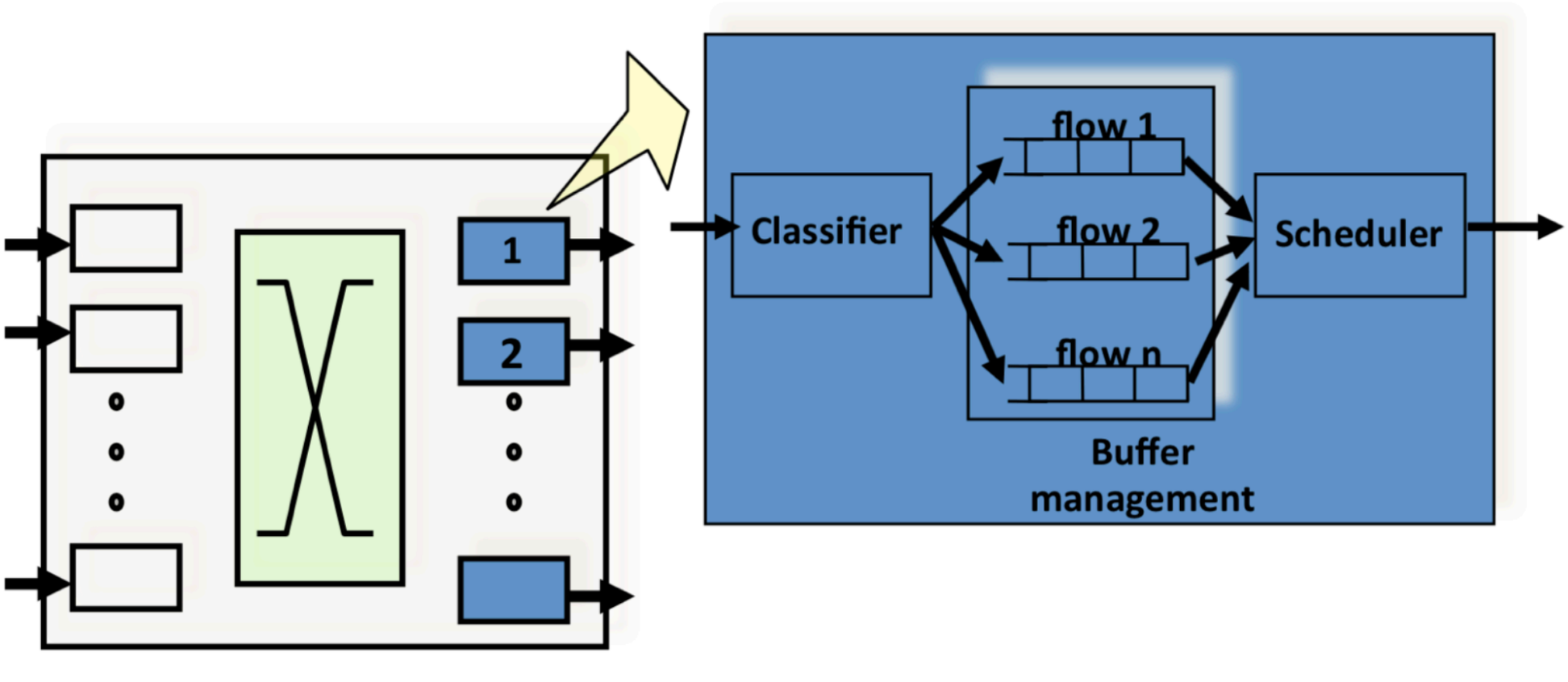
Recap: Input Linecards

- Main challenge is processing speed
    - But what we discussed is the starting point

- Tasks involved
    - Update packet header (easy)
    - Longest prefix match lookup on destinations address (harder)

- Mostly implemented with specialized hardware

Output Linecard



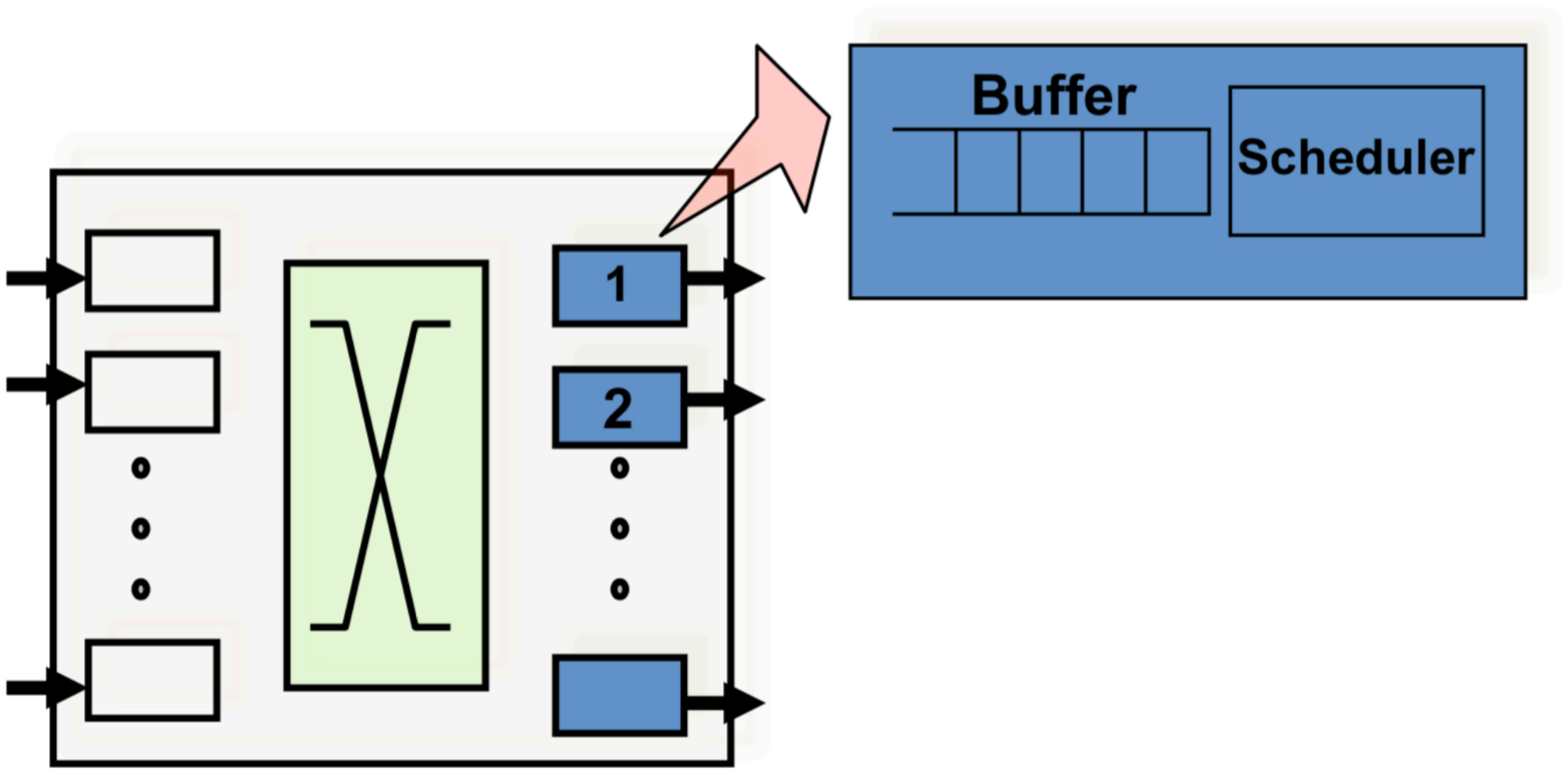
- **Packet Classification**: map each packet to a “flow”
  - Flow (for now): set of packets between two particular endpoints
- **Buffer Management**: decide when and which packet to drop
- **Scheduler**: decide when and which packet to transmit

Output Linecard

- **Packet Classification**: map each packet to a “flow”
    - Flow (for now): set of packets between two particular endpoints
- **Buffer Management**: decide when and which packet to drop
- **Scheduler**: decide when and which packet to transmit
- Used to implement various forms of policy
    - Deny all e-mail traffic from ISP X to Y (**access control**)
    - Route IP telephony traffic from X to Y via PHY_CIRCUIT (**policy**)
    - Ensure that no more than 50 Mbps are injected from ISP-X (**QoS**)

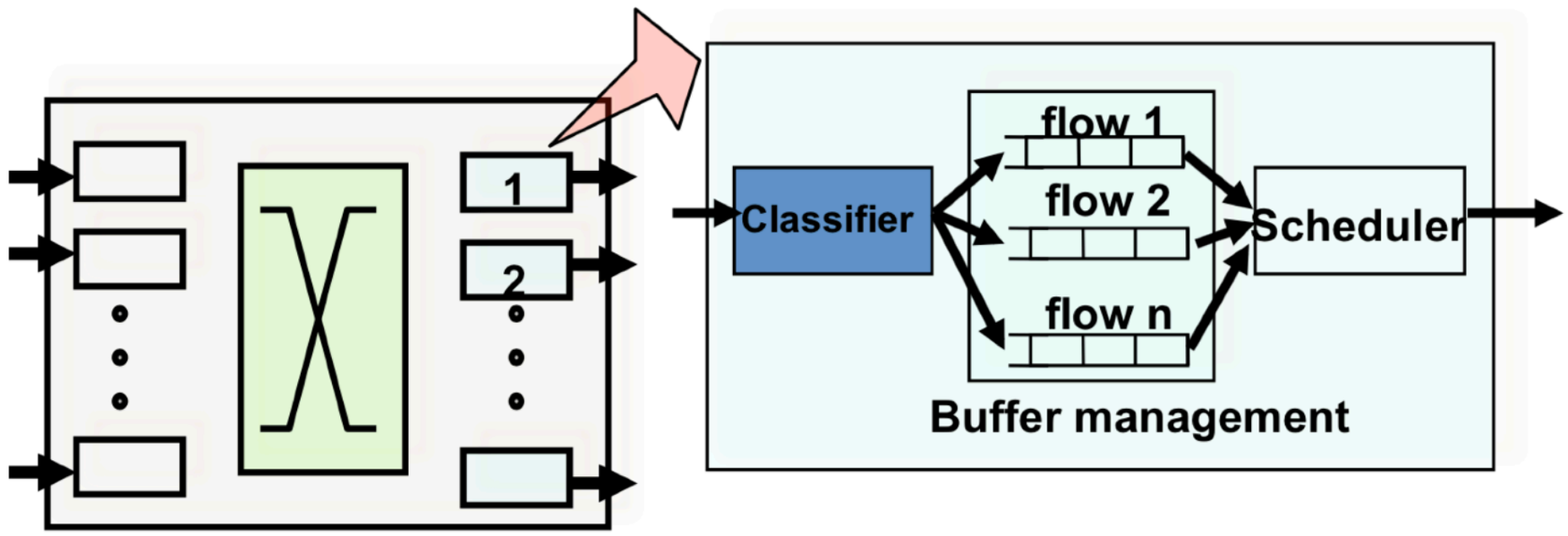
Simplest FIFO Router

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



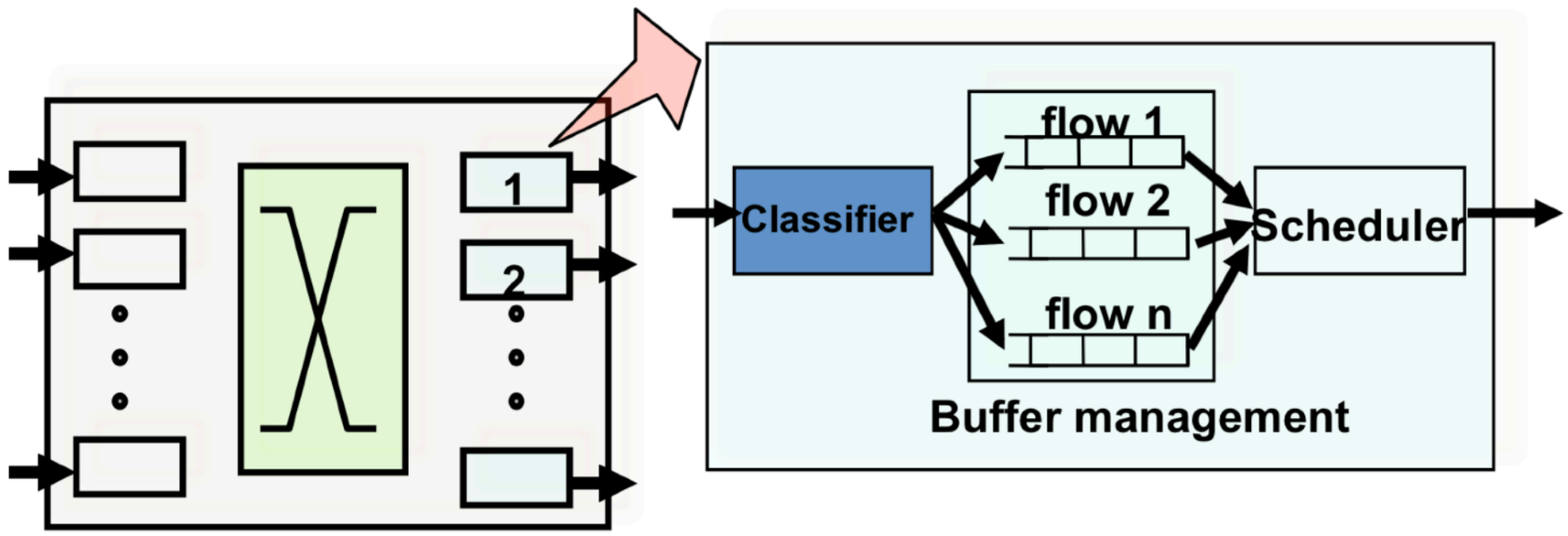
Packet Classification

- Classify an IP packet based on the number of fields in the packet header
    - 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



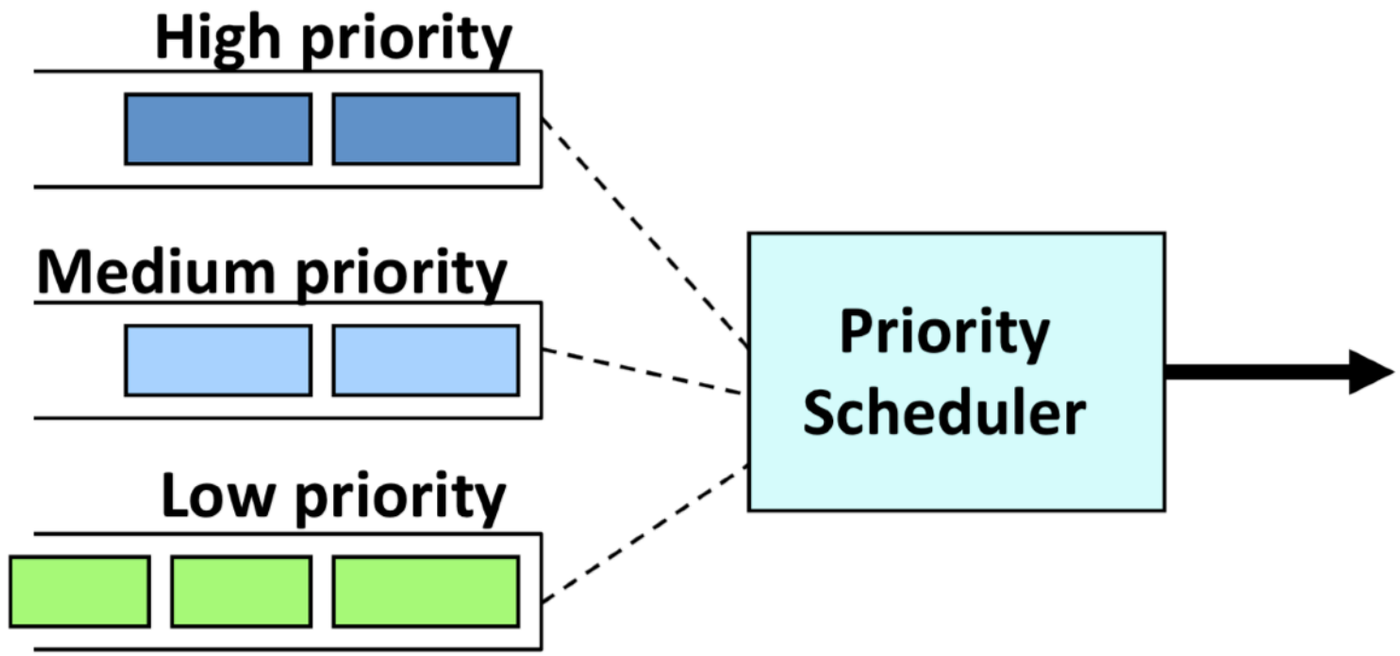
Scheduler

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



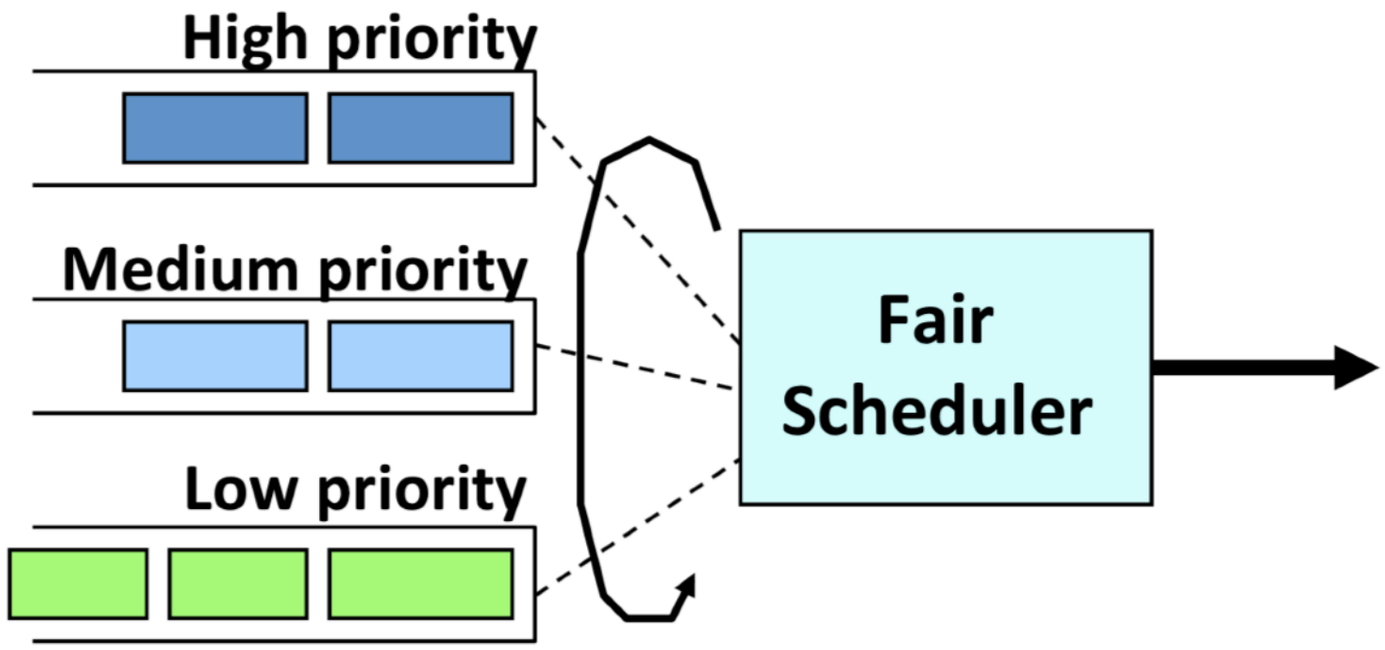
Example: Priority Scheduler

- Packets in the highest priority queue are always served before the packets in the lower priority queues



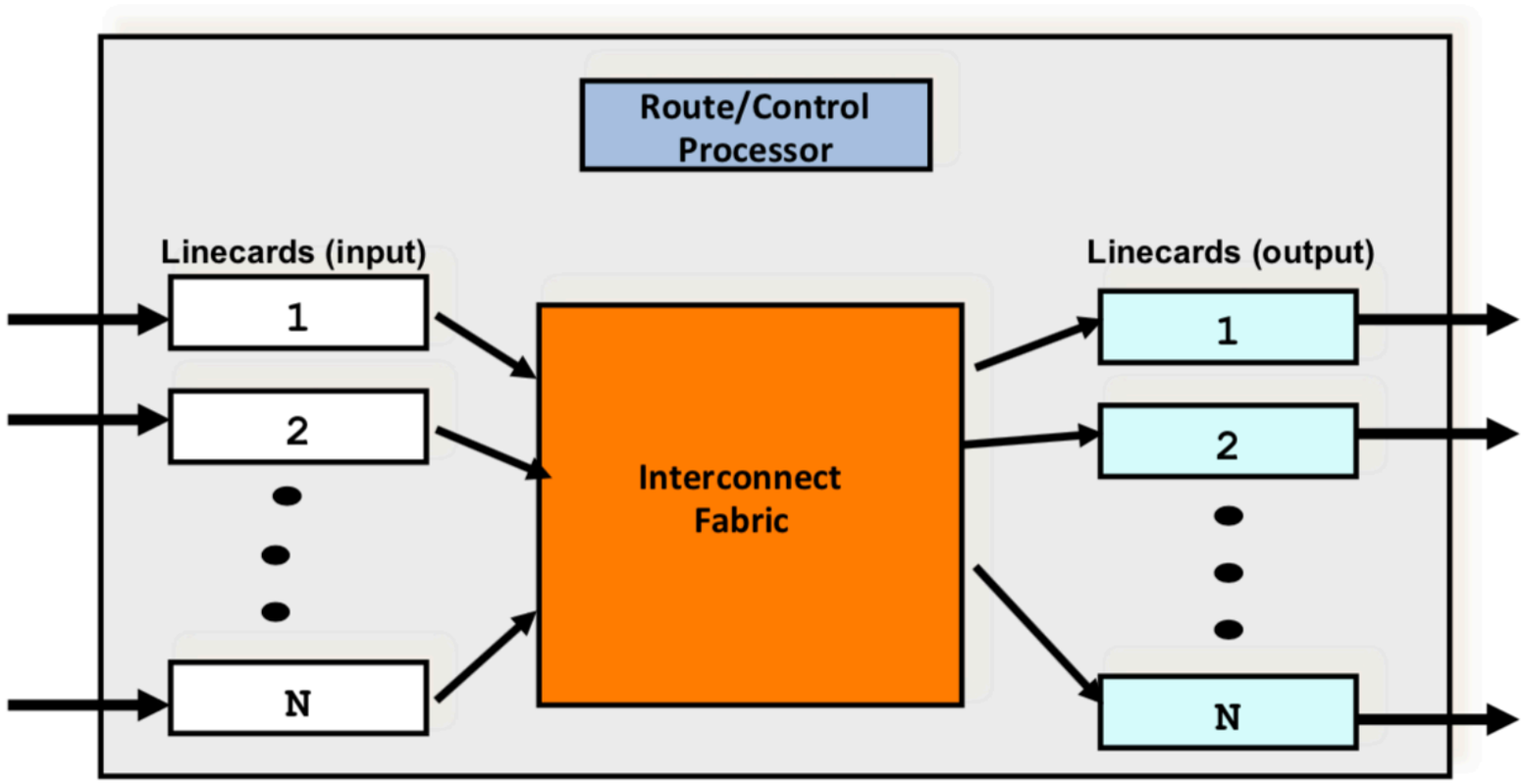
Example: Round Robin Scheduler

- Packets are served from each queue in turn

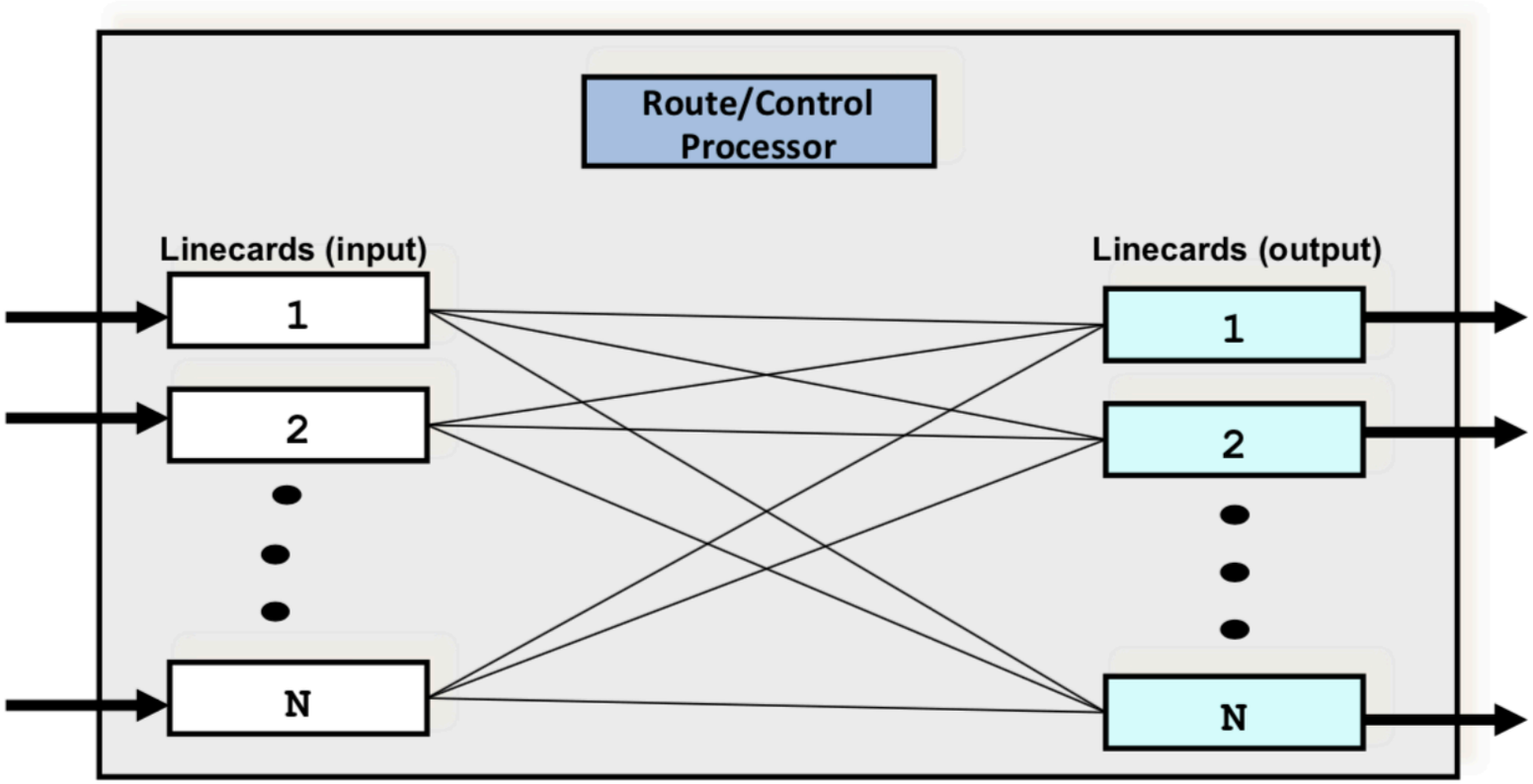


Connecting Input to Output: Switch Fabric

- Priority Scheduler: packets are served from each queue in turn



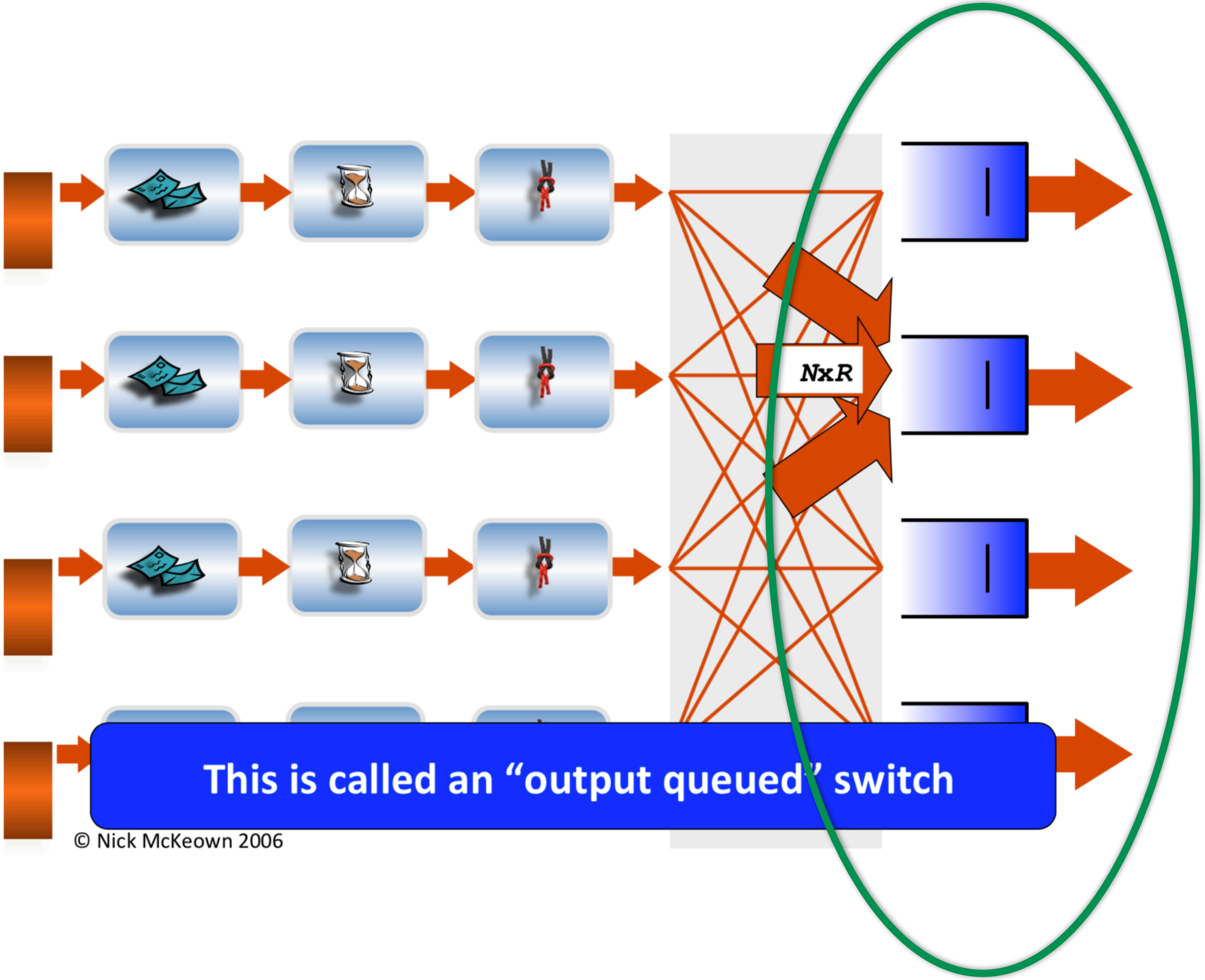
Today's Switch Fabrics: Mini Network!



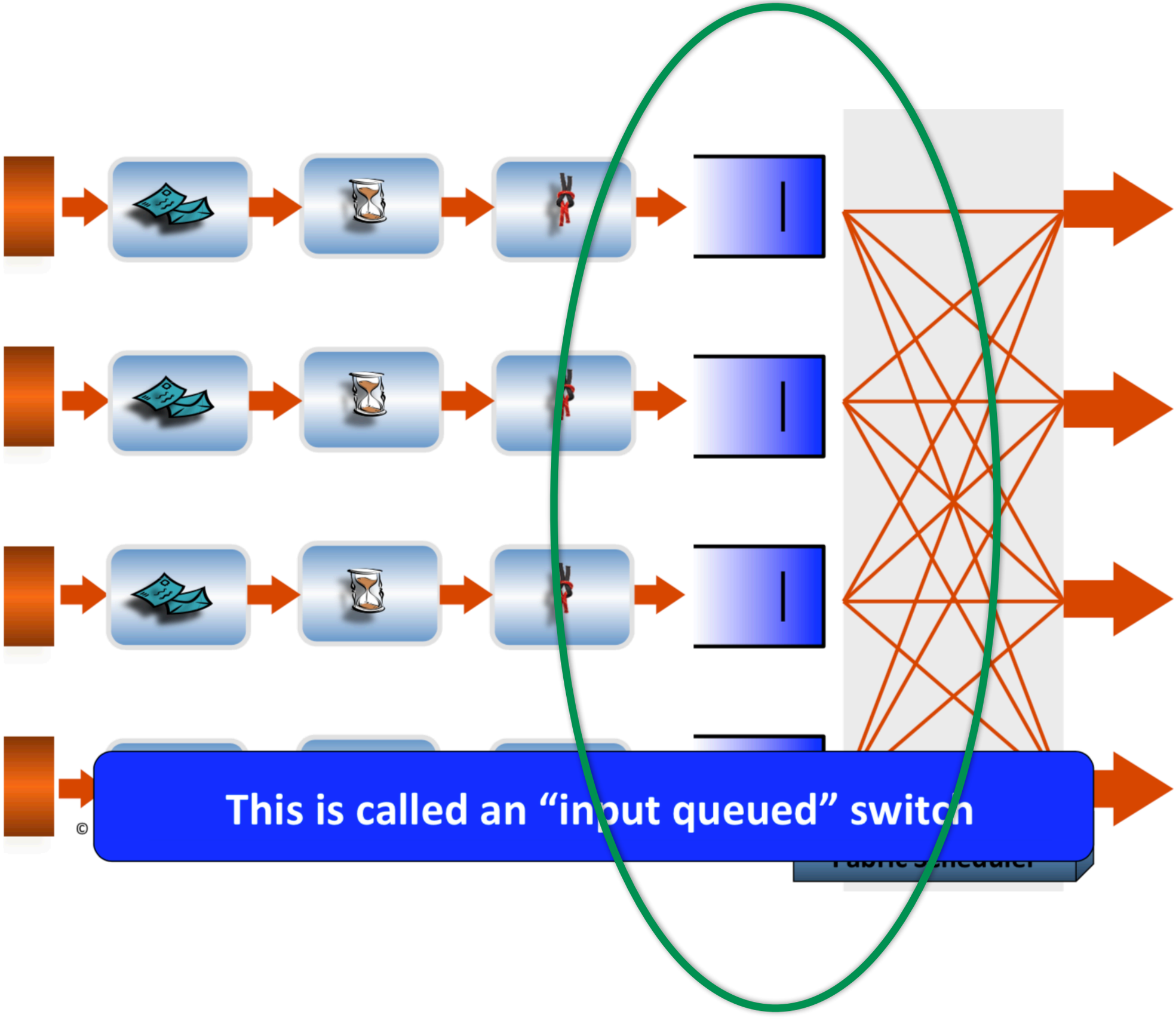
What’s Hard About the Switch Fabric?

Queueing!

Third Generation Router: Switched Interconnects



Third Generation Router: Switched Interconnects



Reality is More Complicated

- Commercial high-speed routers use
    - Combination of input and output queueing
    - Complex multi-stage “topologies”
    - Distributed multi-stage schedulers (for scalability)

IP Routers Recap

- Core building block of Internet infrastructure
- Scalable Routing -> Longest Prefix Matching
- Need fast implementations for
    - Longest prefix matching
    - Switch fabric scheduling

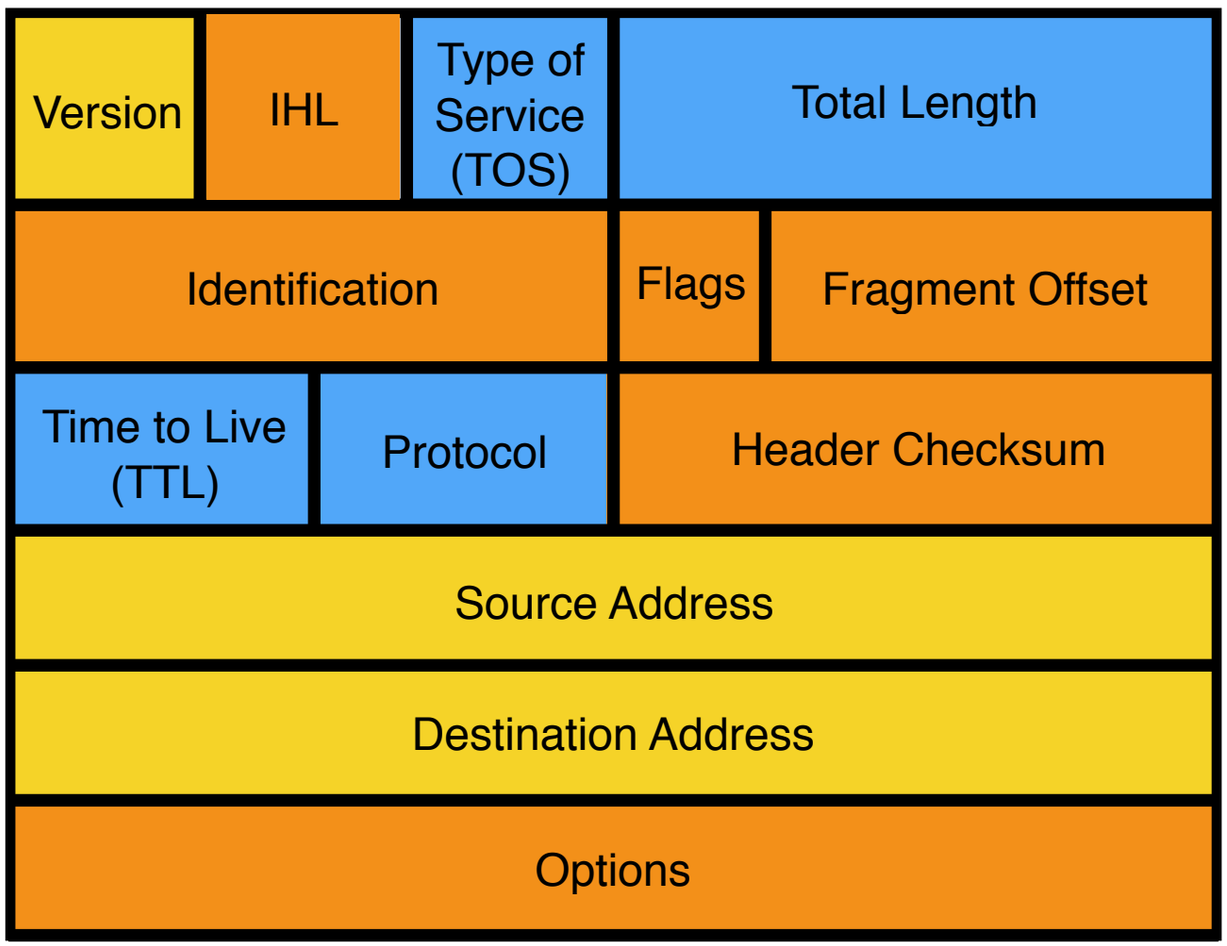
This is it for today!

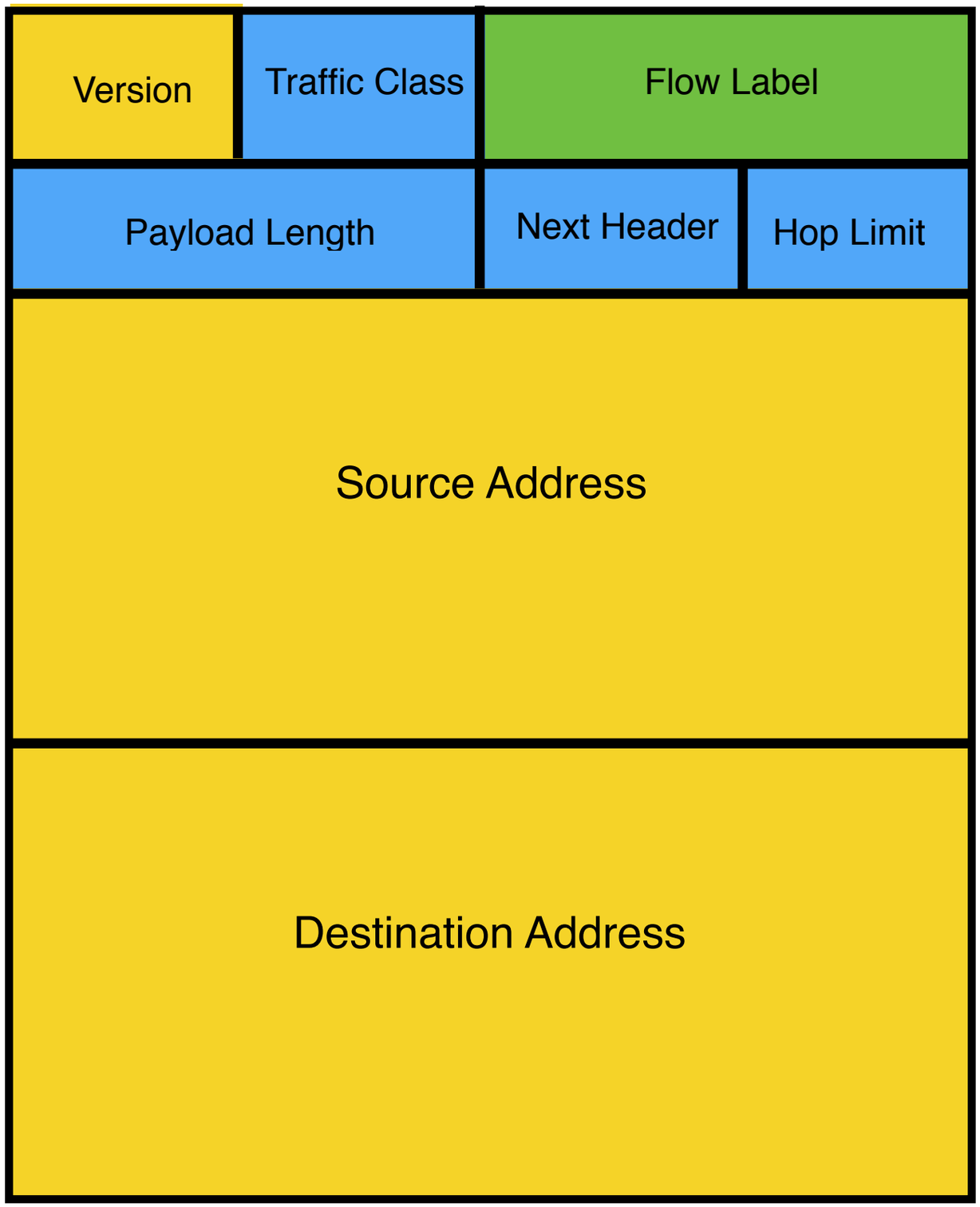
IPv6

IPv6

- Motivated (prematurely) by address exhaustion
    - Address **four** times as big
- Steve Deering focused on simplifying IP
    - Got rid of all fields that were not absolutely necessary
    - “Spring Cleaning” for IP
- Result is an elegant, if unambitious, protocol

IPv4 and IPv6 Header Comparison





Field name kept from IPv4 to IPv6

Fields not kept in IPv6

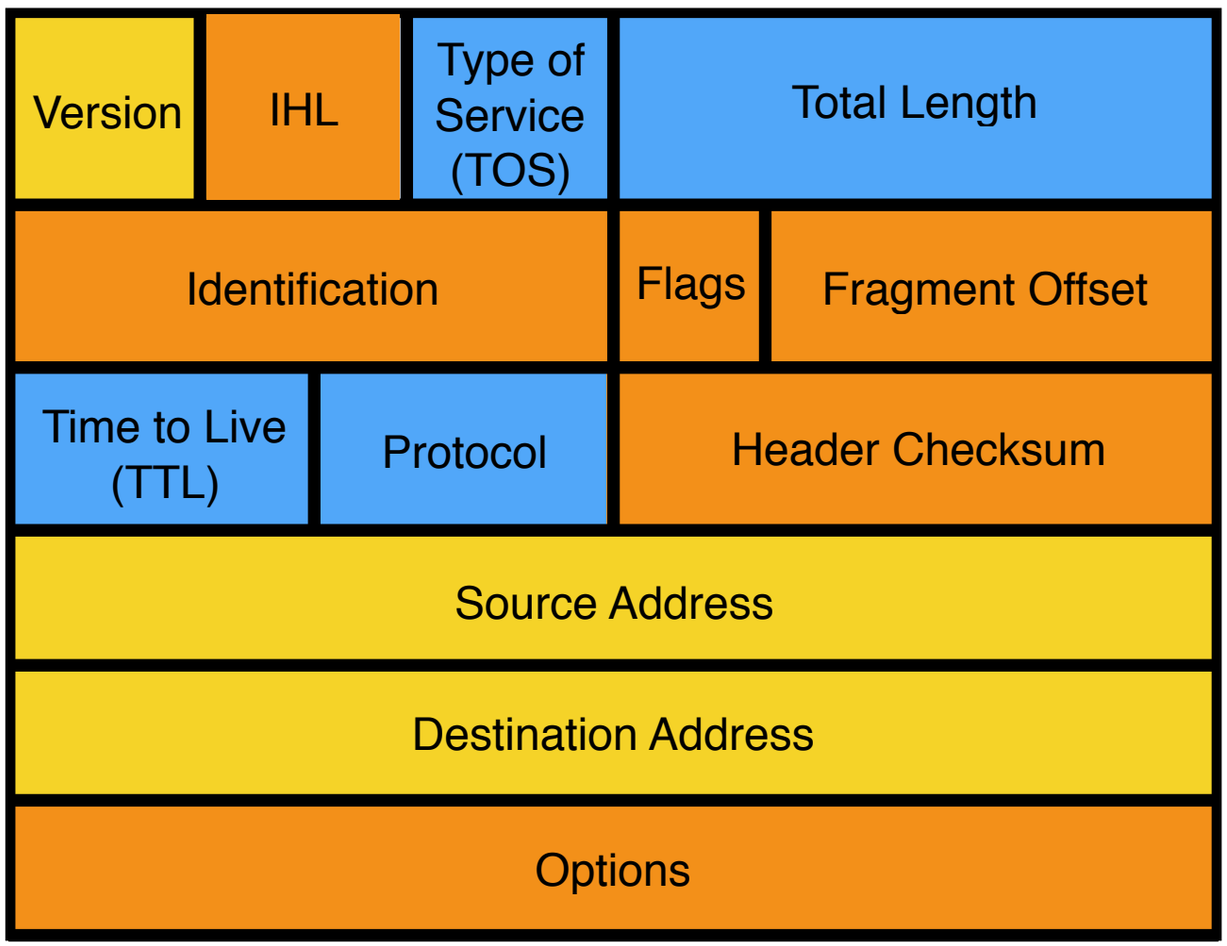
Name and position changed in IPv6

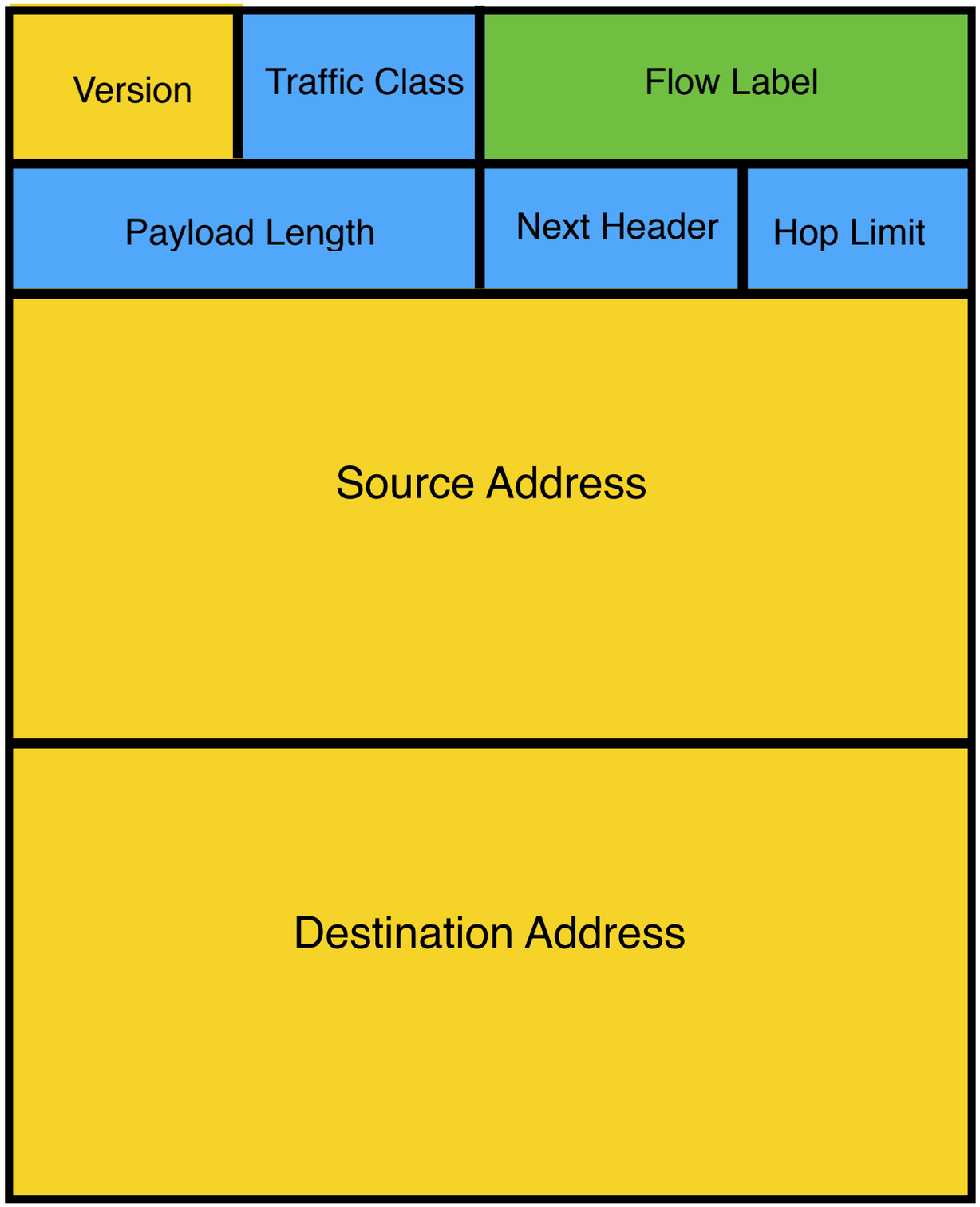
New field in IPv6

Summary of Changes

- Eliminated Fragmentation
- Eliminated header length
- Eliminated Checksum
- New options mechanism (next header)
- Expanded address
- Added Flow Label

IPv4 and IPv6 Header Comparison





Field name kept from IPv4 to IPv6

Fields not kept in IPv6

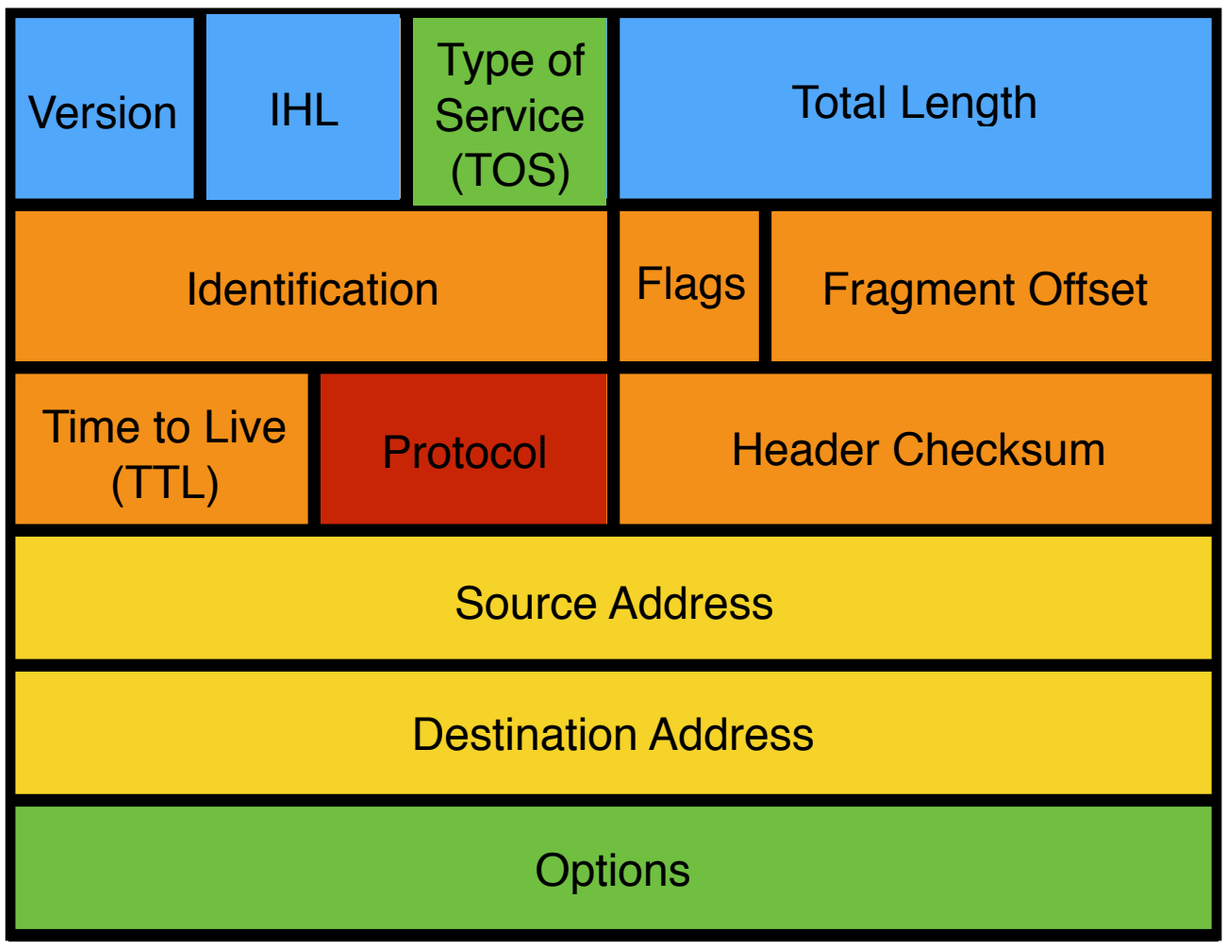
Name and position changed in IPv6

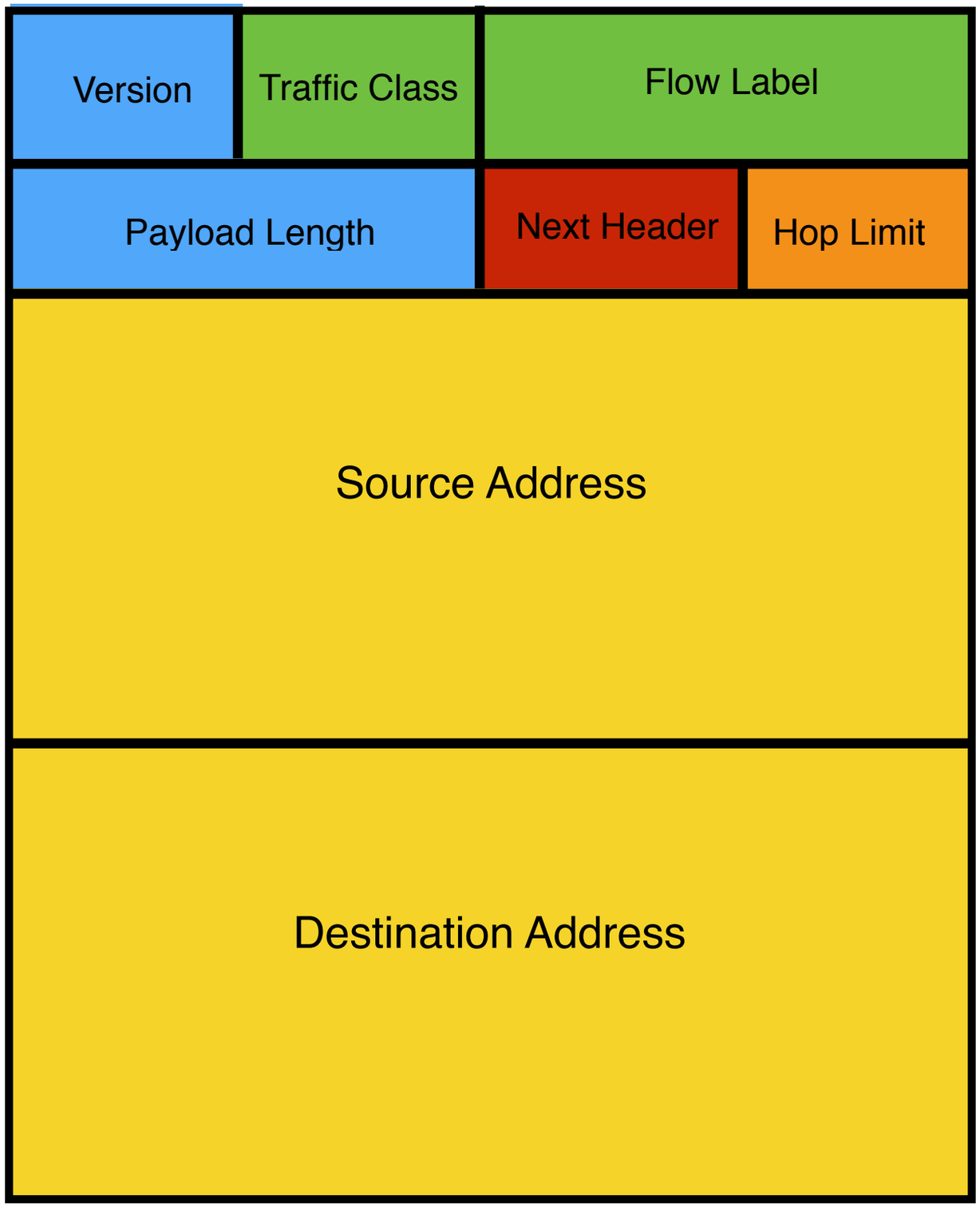
New field in IPv6

Philosophy of Changes

- Don’t deal with problems: leave to ends
    - Eliminated fragmentation
    - Eliminated checksum
    - **Why retain TTL?**
- Simplify handling
    - New options mechanism (uses next header approach)
    - Eliminated header length
        - **Why couldn’t IPv4 do this?**
- Provide general flow label for packet
    - Not tied to semantics
    - Provides great flexibility

IPv4 and IPv6 Header Comparison





To Destination and Back (expanded)

Deal with Problems (greatly reduced)

Read Correctly (reduced)

Special Handling (Similar)