CS4450

Computer Networks:
Architecture and Protocols

Lecture 16
BGP—unsolved problems
THE Internet Protocol
Switch Architecture

Rachit Agarwal

Announcements

- **Problem set 4 released**

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

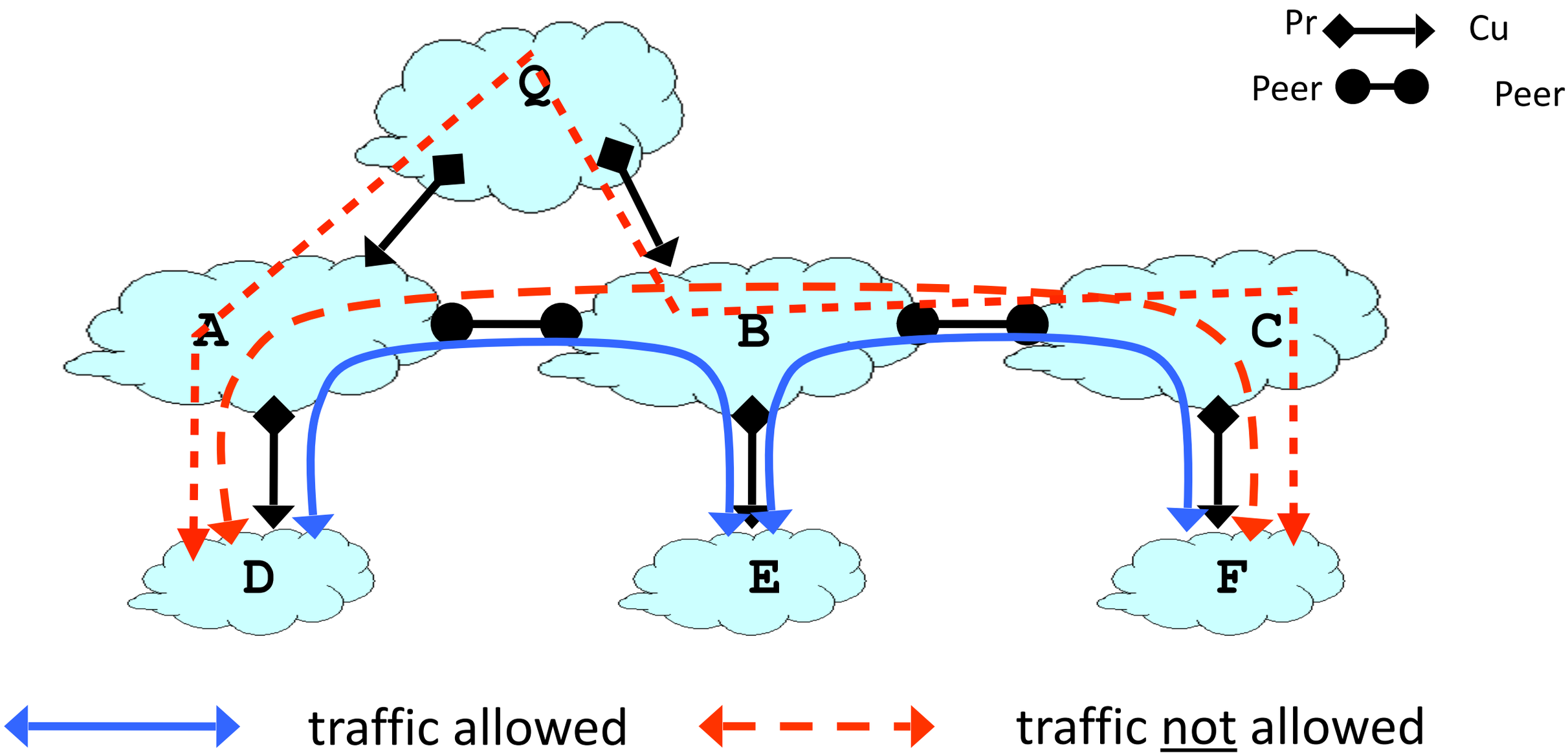
- **Next lecture is very very very ….**
    - **very very very very ….**
    - **important**
    - Please attend

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

Goals for Today's Lecture

- Wrap up BGP
- Understand IP (the Internet Protocol)
    - Packet Header as a network "interface"
- Understand switch architecture

Recap: Inter-domain Routing Follows the Money



- ASes provide "transit" between their customers
- Peers do not provide transit between other peers

Recap: BGP Inspired by Distance Vector

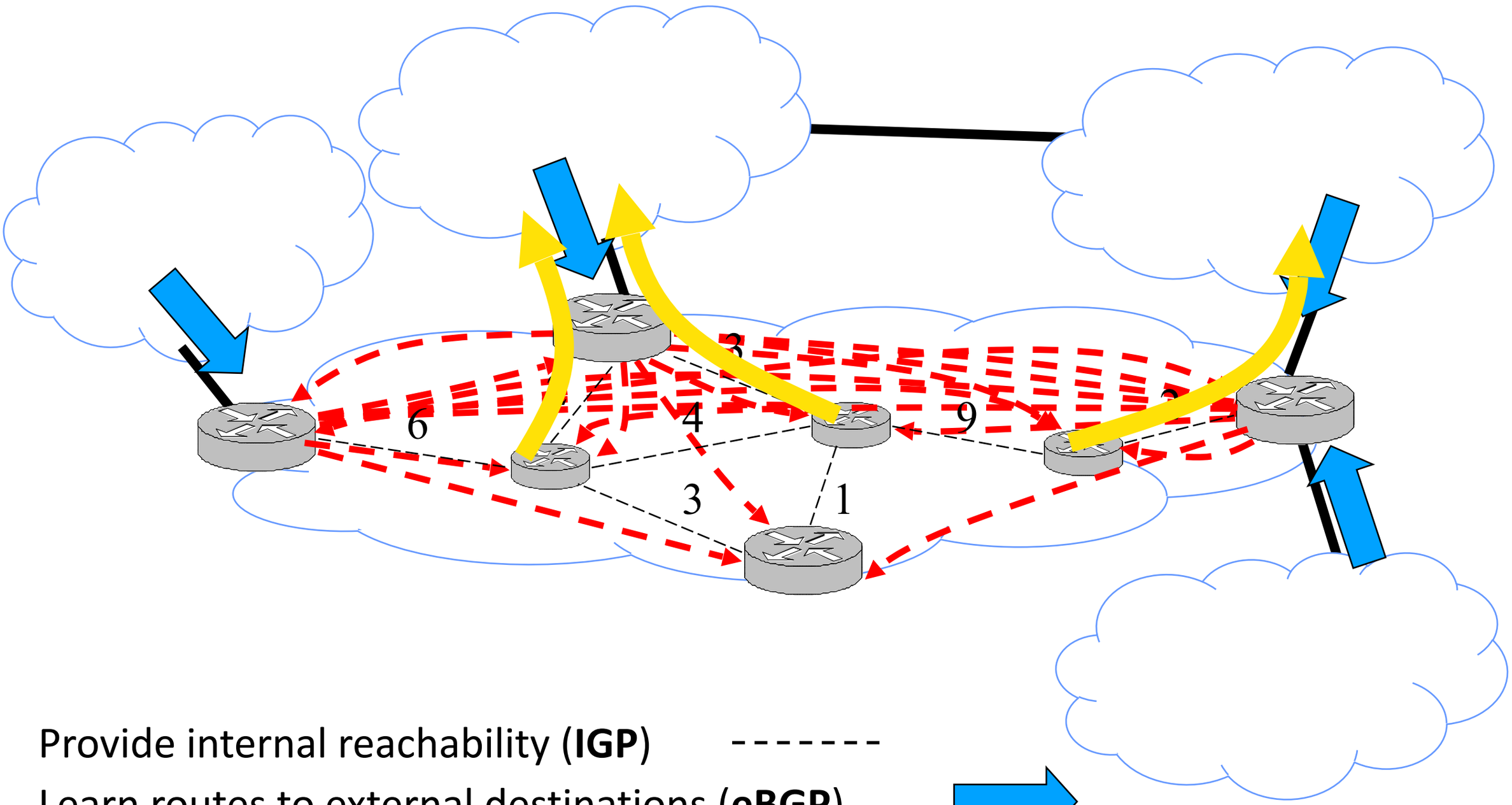
- Per-destination route advertisements
- No global sharing of network topology
- Iterative and distributed convergence on paths
- But, **four key differences**
  - **BGP does not pick shortest paths**
  - **Path-vector rather than distance vector**
    - Each announcement contains the path for each destination
  - **Selective route advertisement**
    - If I select a path, I don't *have to* advertise it to others
  - **Route aggregation**
    - Rather than storing a.b.*.*/16 and a.c.*.*/16, store a.*.*.*/8

Recap: Policy on how routes are selected and exported

- **Selection**: Which path to use
  - Controls whether / how traffic leaves the network
  - Typical policy:
    1. Make or save money (send to customer > peer > provider)
    2. Maximize performance (smallest AS path length)
    3. Minimize use of my network bandwidth ("hot potato")
    4. …
- **Export**: Which path to advertise
  - Controls whether / how traffic enters the network

Destination prefix advertised by…	Export route to…
Customer	Everyone (providers, peers, other customers)
Peer	Customers
Provider	Customers

Recap: Putting the Pieces Together



1. Provide internal reachability (**IGP**) - - - - - - -
2. Learn routes to external destinations (**eBGP**)
3. Distribute externally learned routes internally (**iBGP**)
4. Travel shortest path to egress (IGP)

Recap: Route Updates

- Format: *<IP prefix: route attributes>*
- Two kinds of updates:
  - Announcements: new routes or changes to existing routes
  - Withdrawals: remove routes that no longer exist

- Route Attributes
  - Describe routes, used in selection/export decisions
    - ASPATH
    - Local PREF
    - MED
    - IGP Cost

Questions?

BGP Details

- BGP Policy
  - Typical policies and implementation

- BGP protocol details

- **Why am I not blindly in love with BGP?**

- Why do I love BGP?

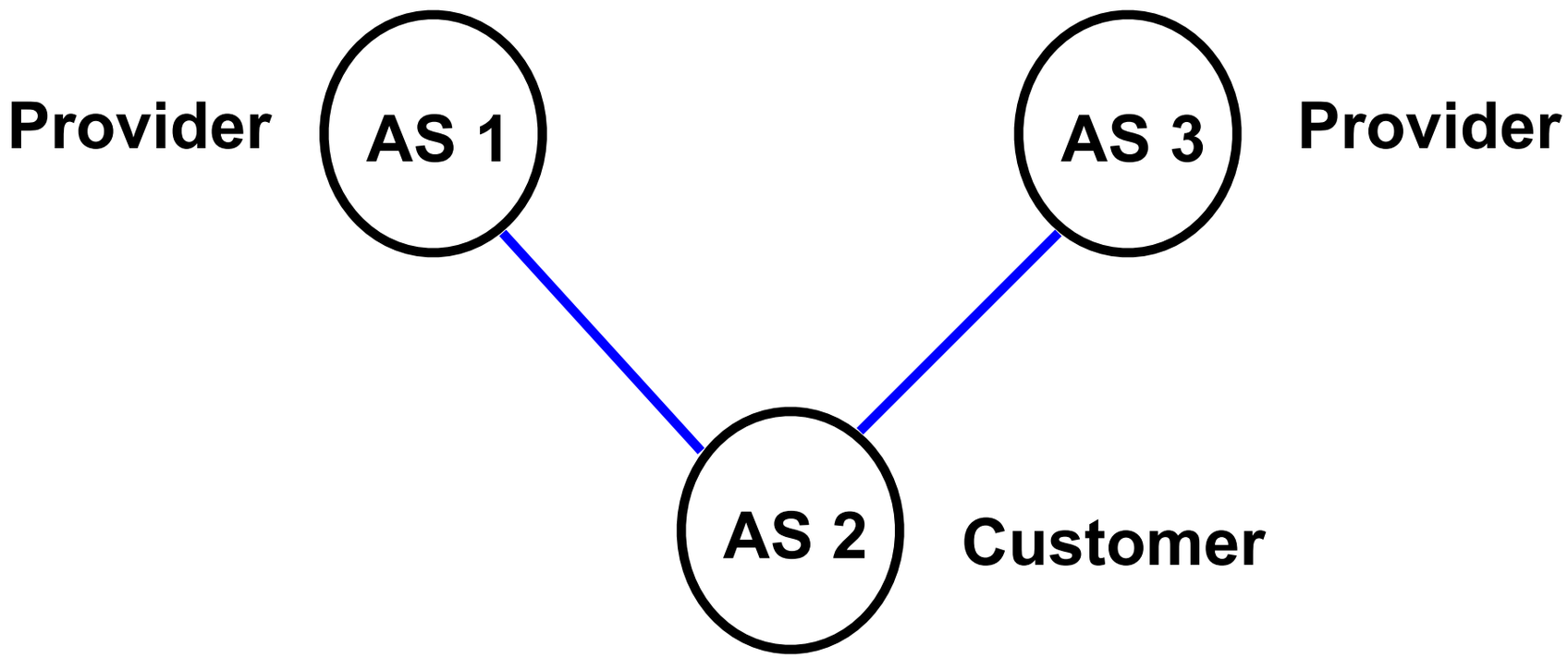
Why am I not blindly in love with BGP?

- Reachability
- Convergence
- Performance
- Anomalies
- Security

Reachability

- In shortest-path routing, if graph is connected then reachability is assured

- With policy routing, this doesn't always hold



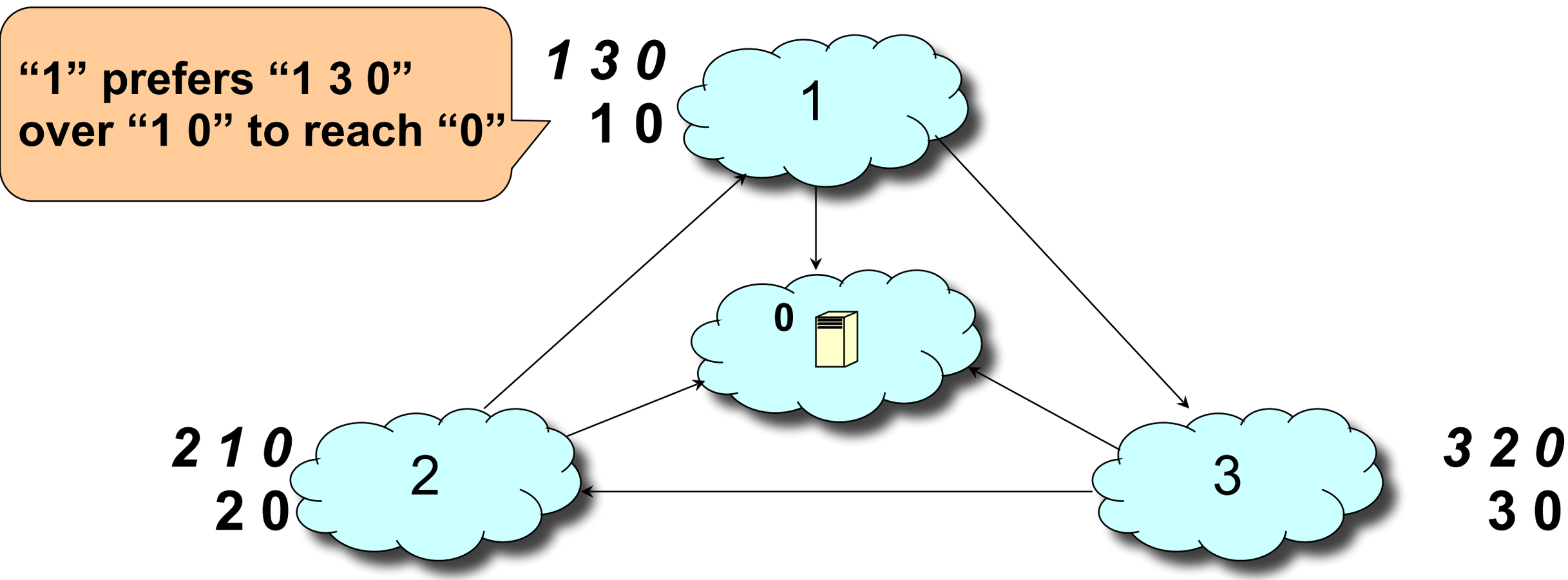
Convergence

- If all AS policies follow Gao-Rexford rules,
  - Then BGP is guaranteed to converge (safety)

- For arbitrary policies, BGP may fail to converge!

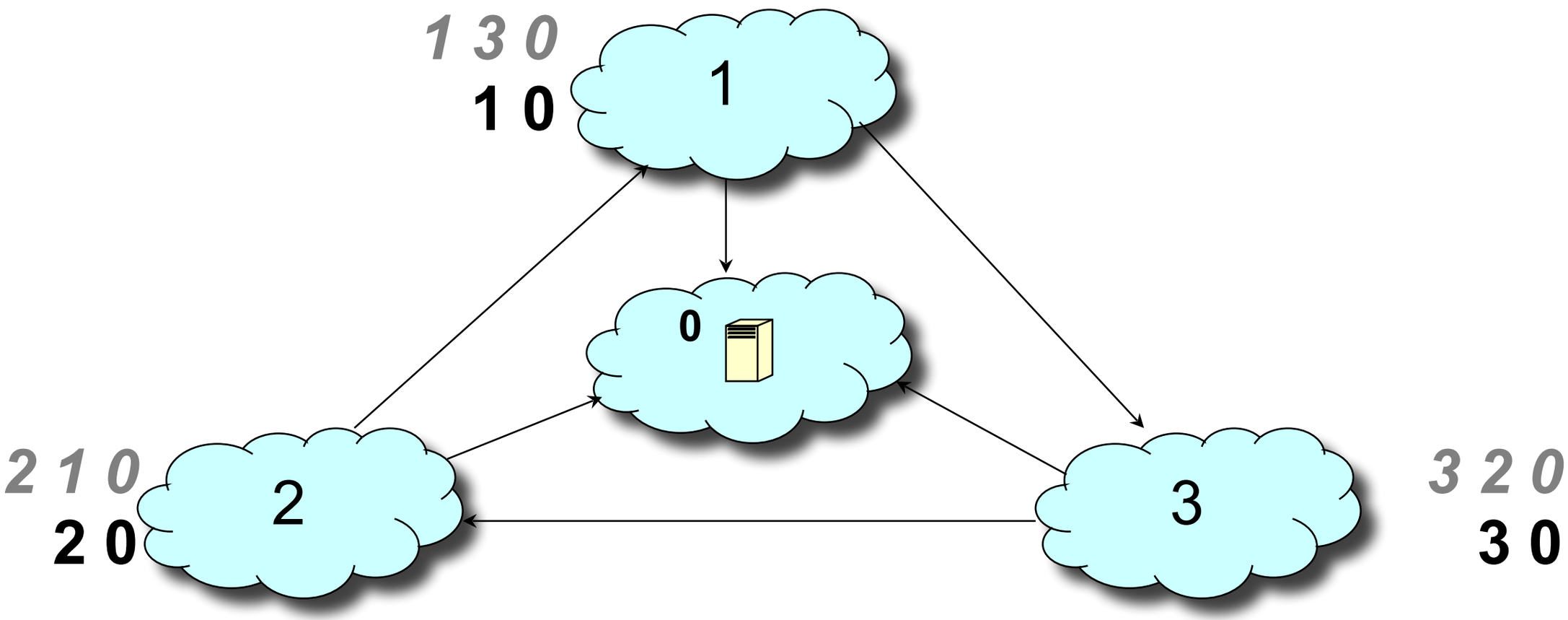
- **Do you appreciate the beauty of the result (and Gao-Rexford policies?)**

Example of Policy Oscillation



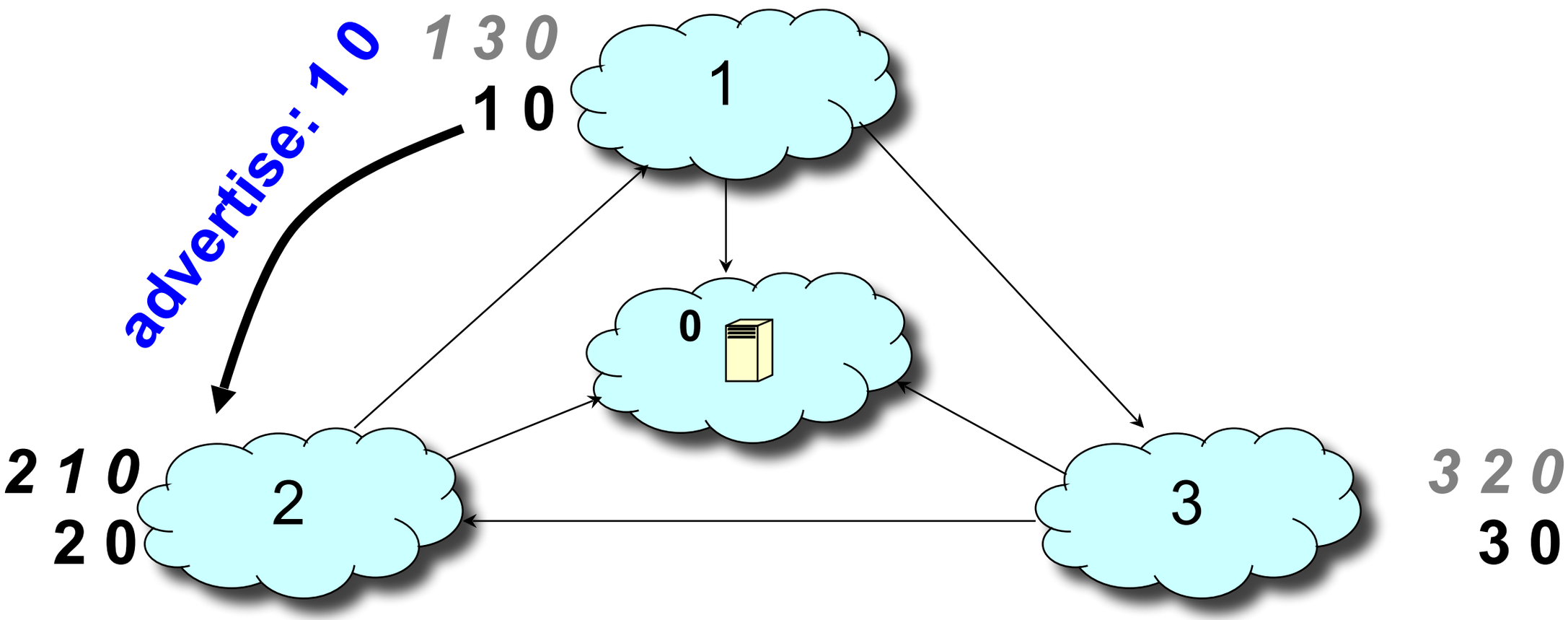
Step-by-step Policy Oscillation

Initially: nodes 1, 2, 3 know only shortest path to 0

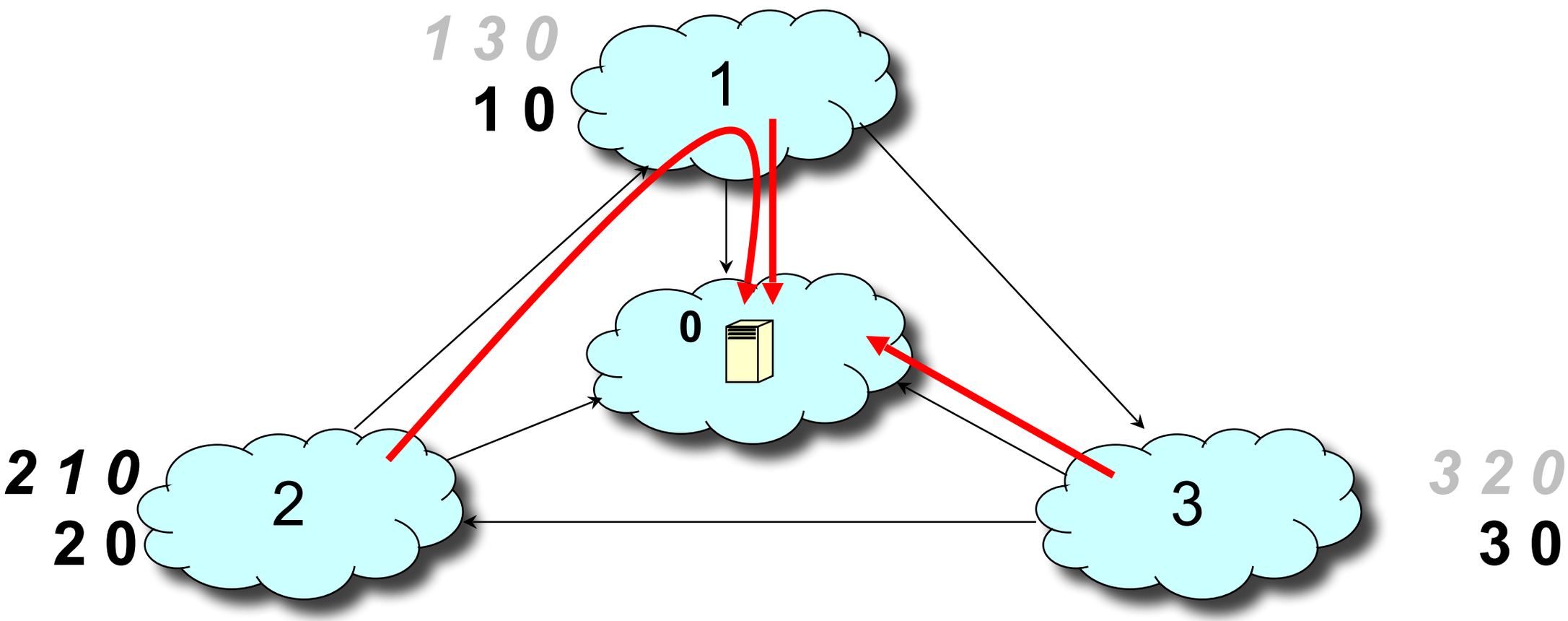


Step-by-step Policy Oscillation

1 **advertises** its path 1 0 to 2

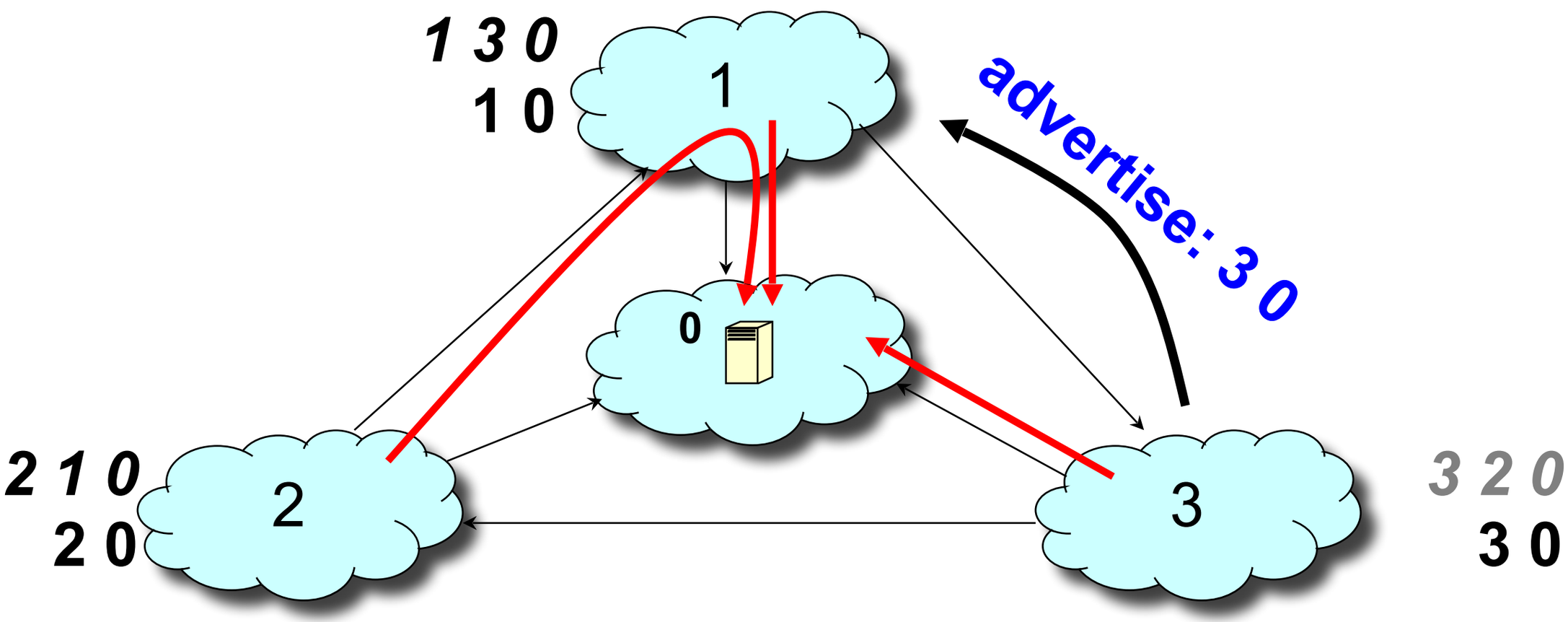


Step-by-step Policy Oscillation

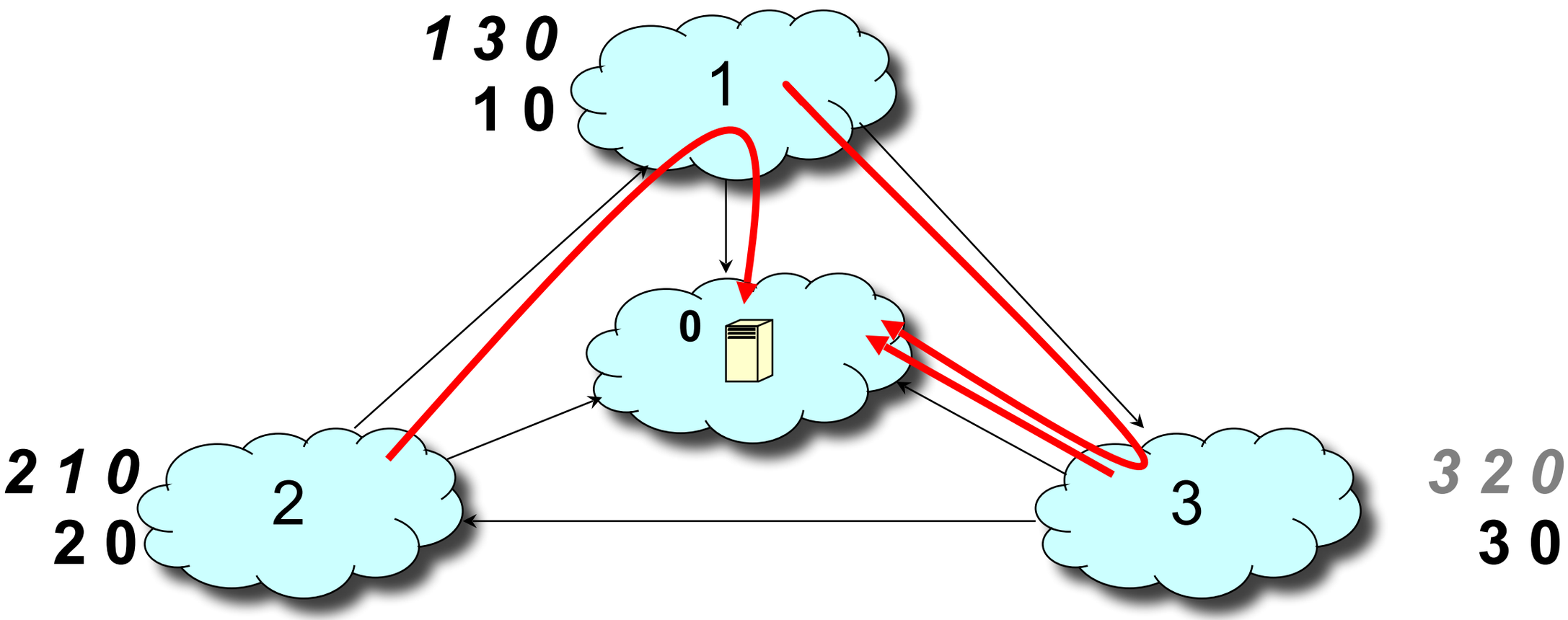


Step-by-step Policy Oscillation

3 **advertises** its path 3 0 to 1

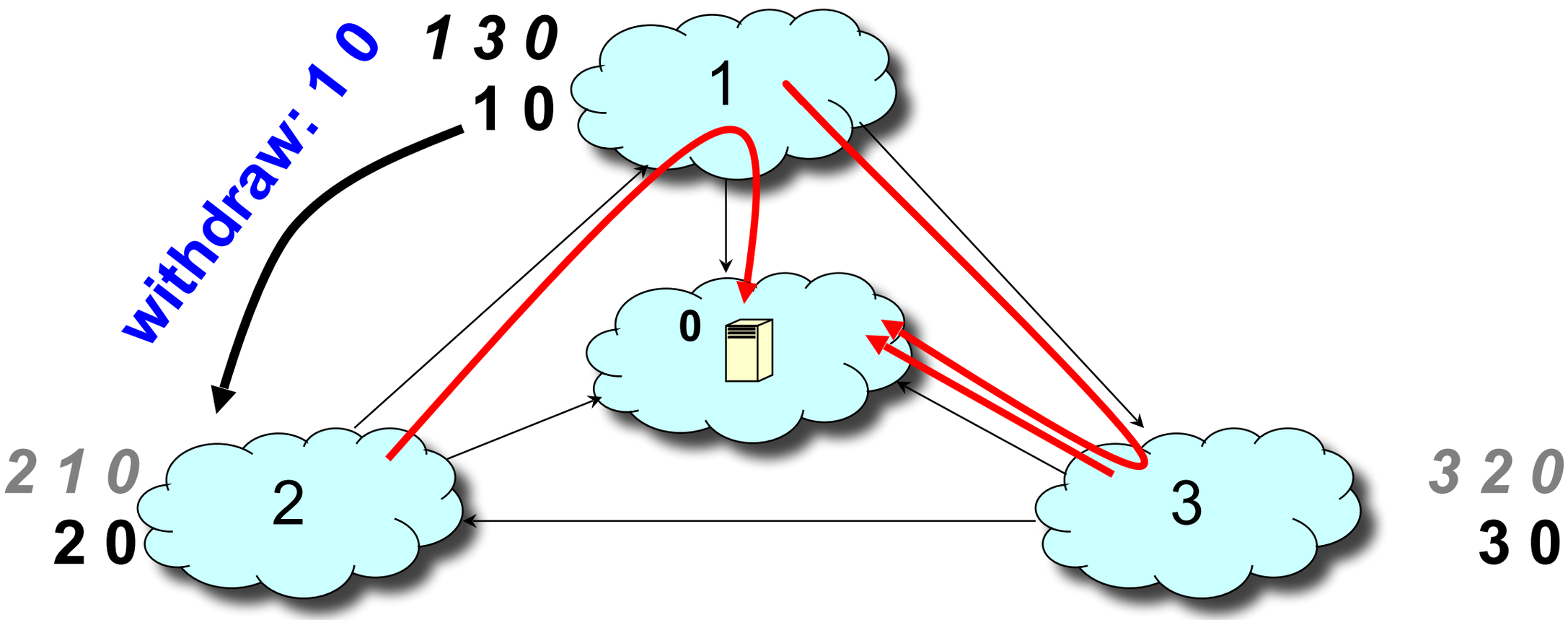


Step-by-step Policy Oscillation

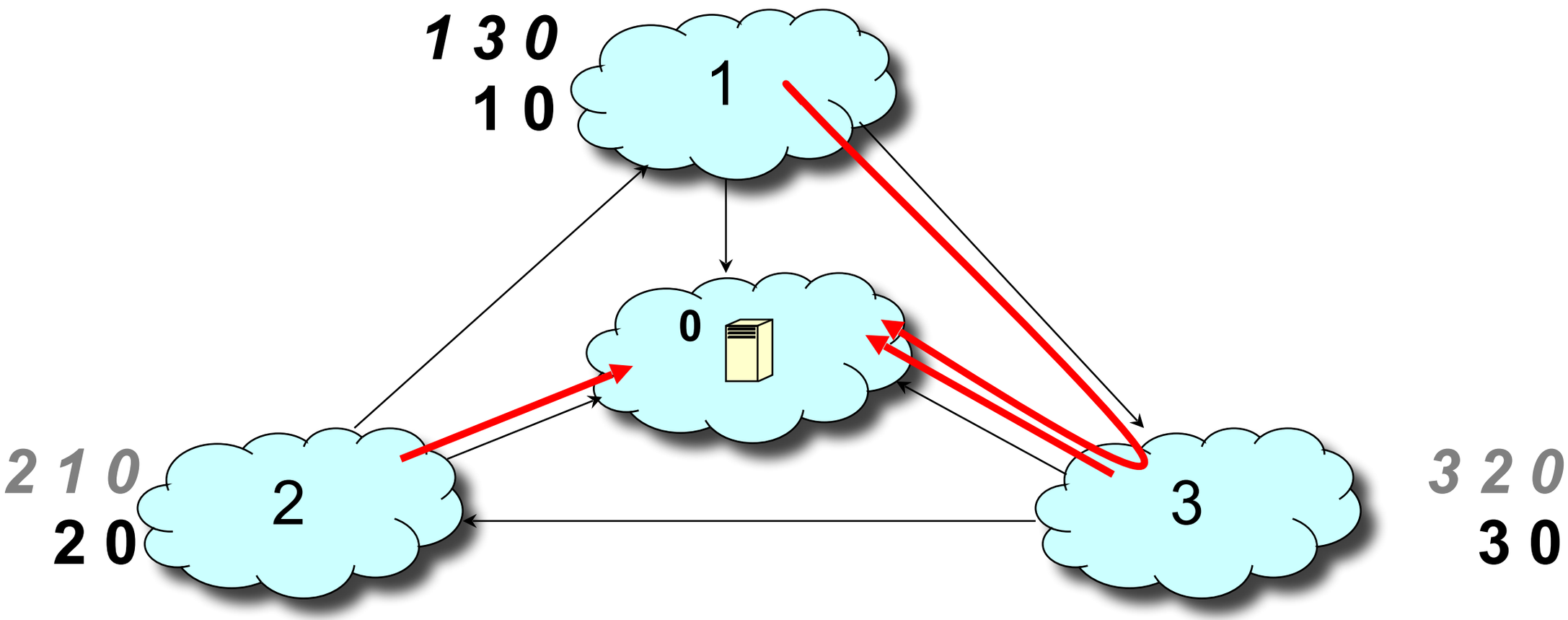


Step-by-step Policy Oscillation

1 **withdraws** its path 1 0 from 2

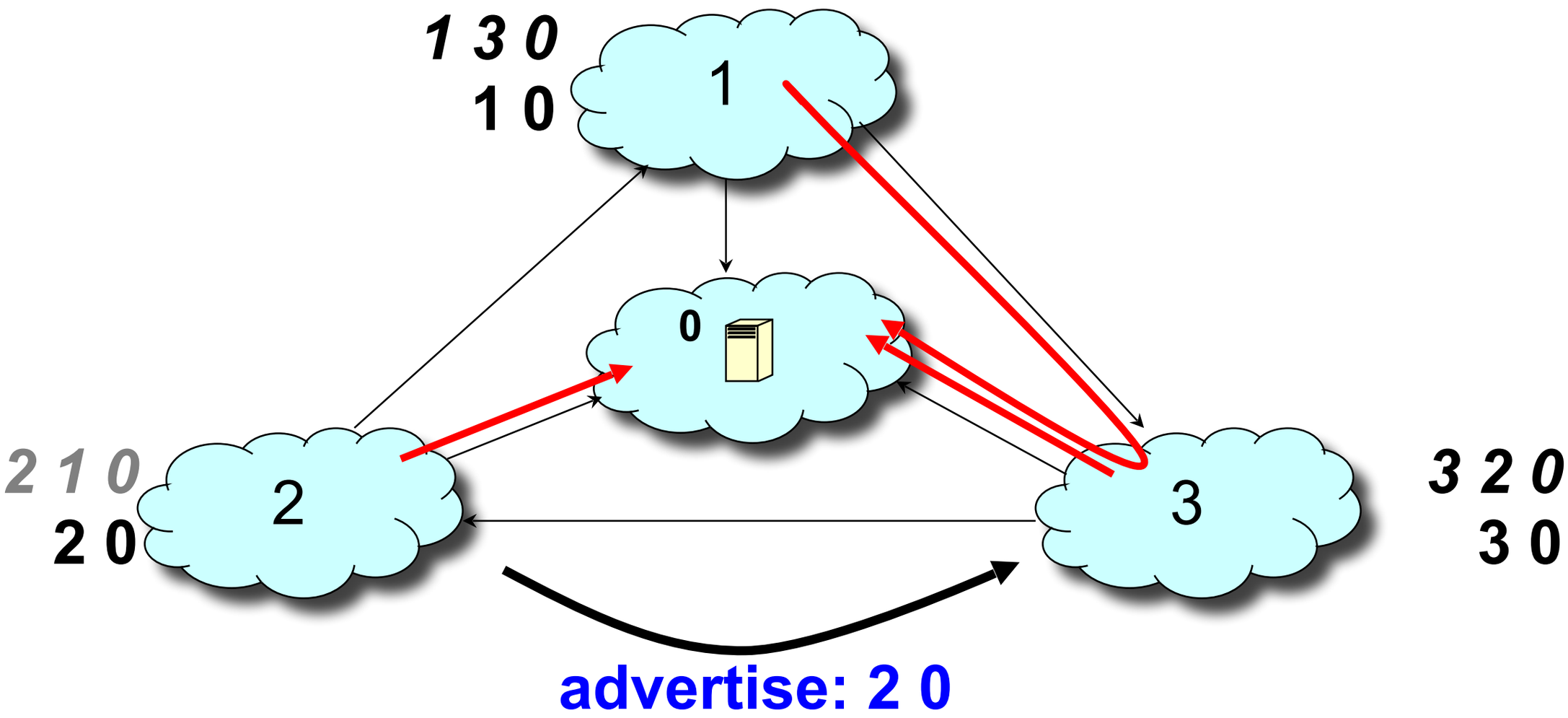


Step-by-step Policy Oscillation

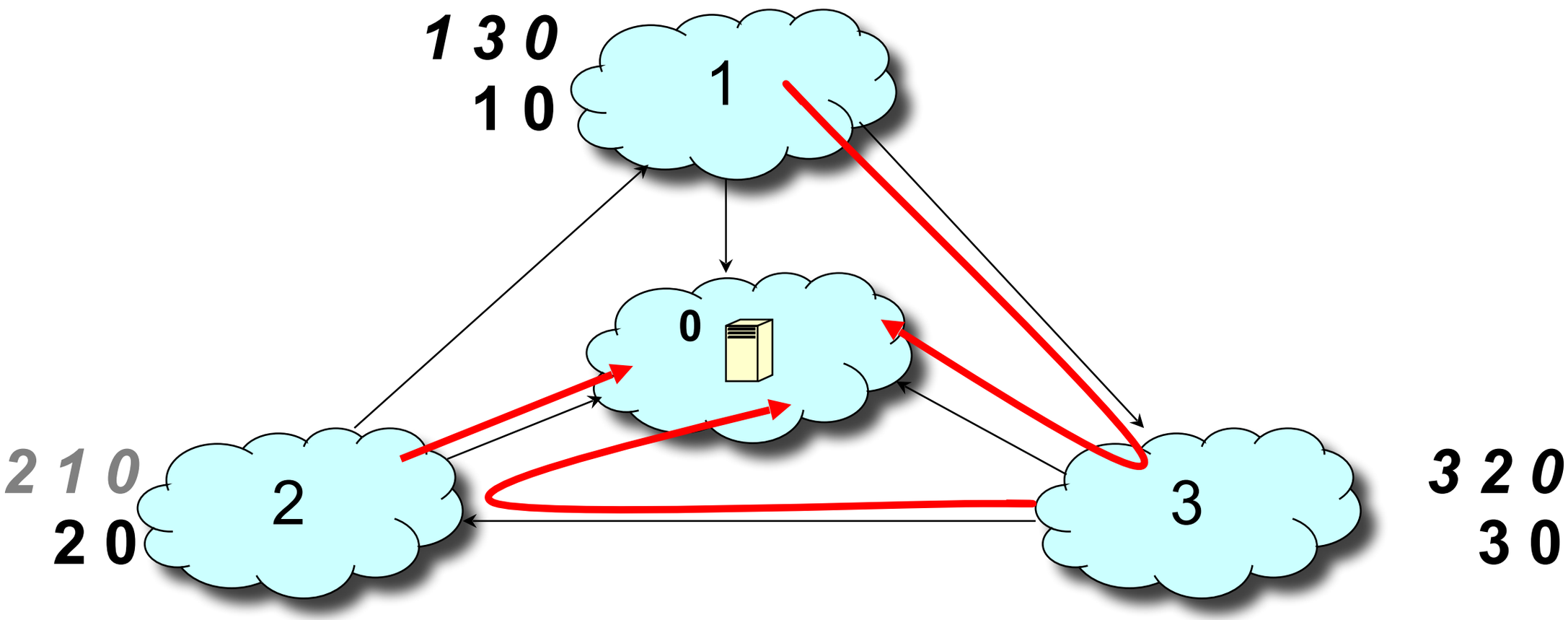


Step-by-step Policy Oscillation

2 **advertises** its path 2 0 to 3

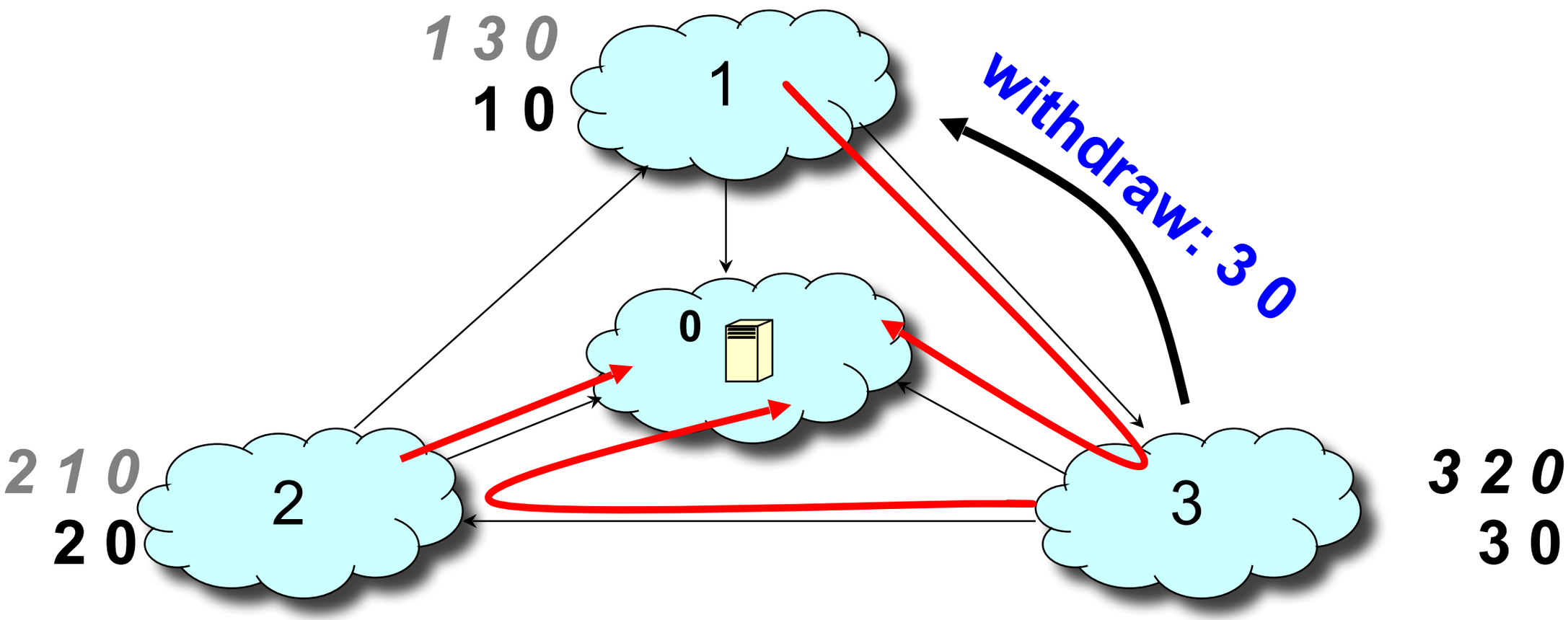


Step-by-step Policy Oscillation

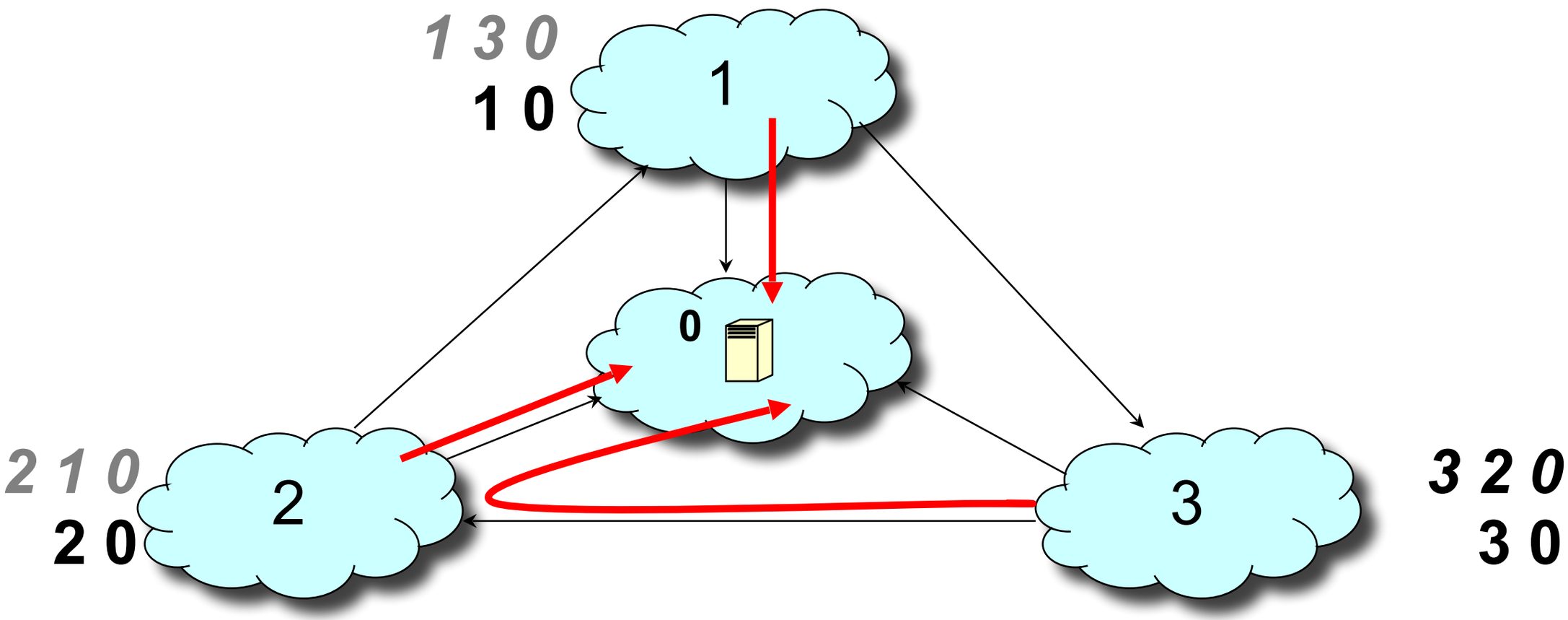


Step-by-step Policy Oscillation

3 **withdraws** its path 3 0 from 1

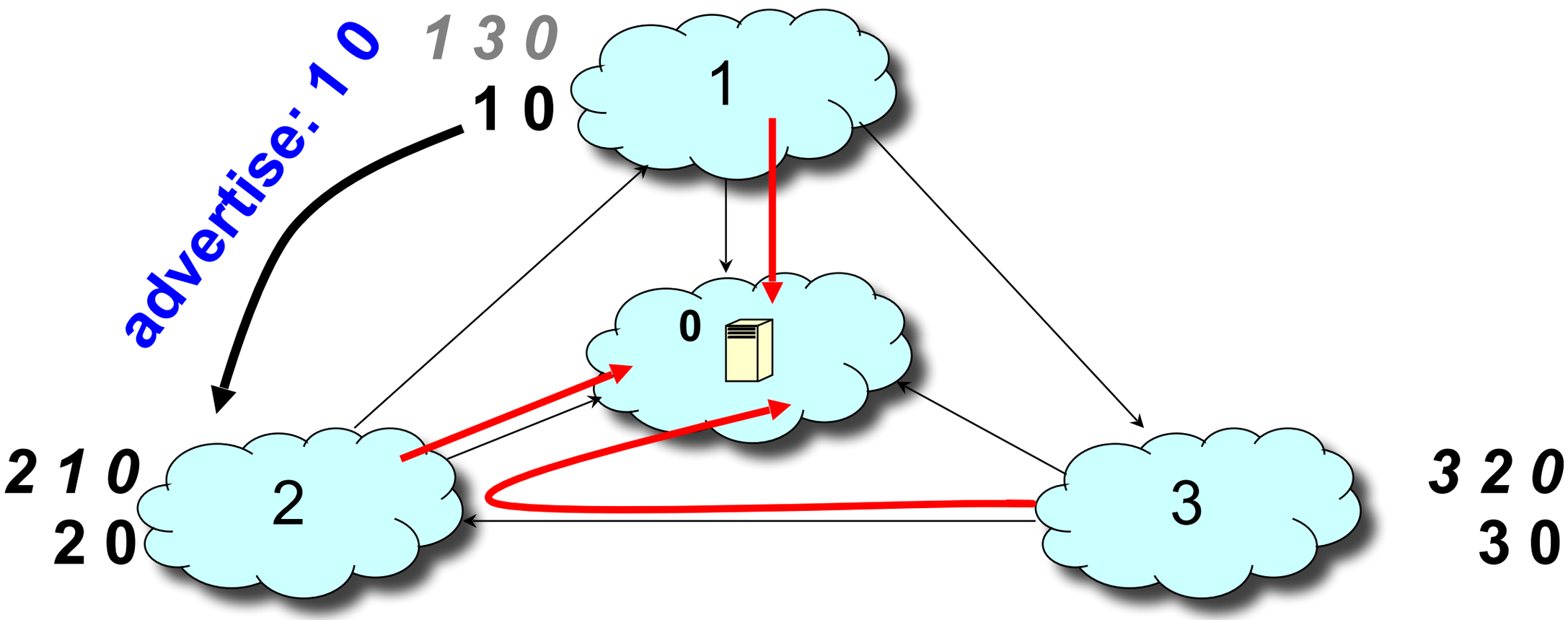


Step-by-step Policy Oscillation

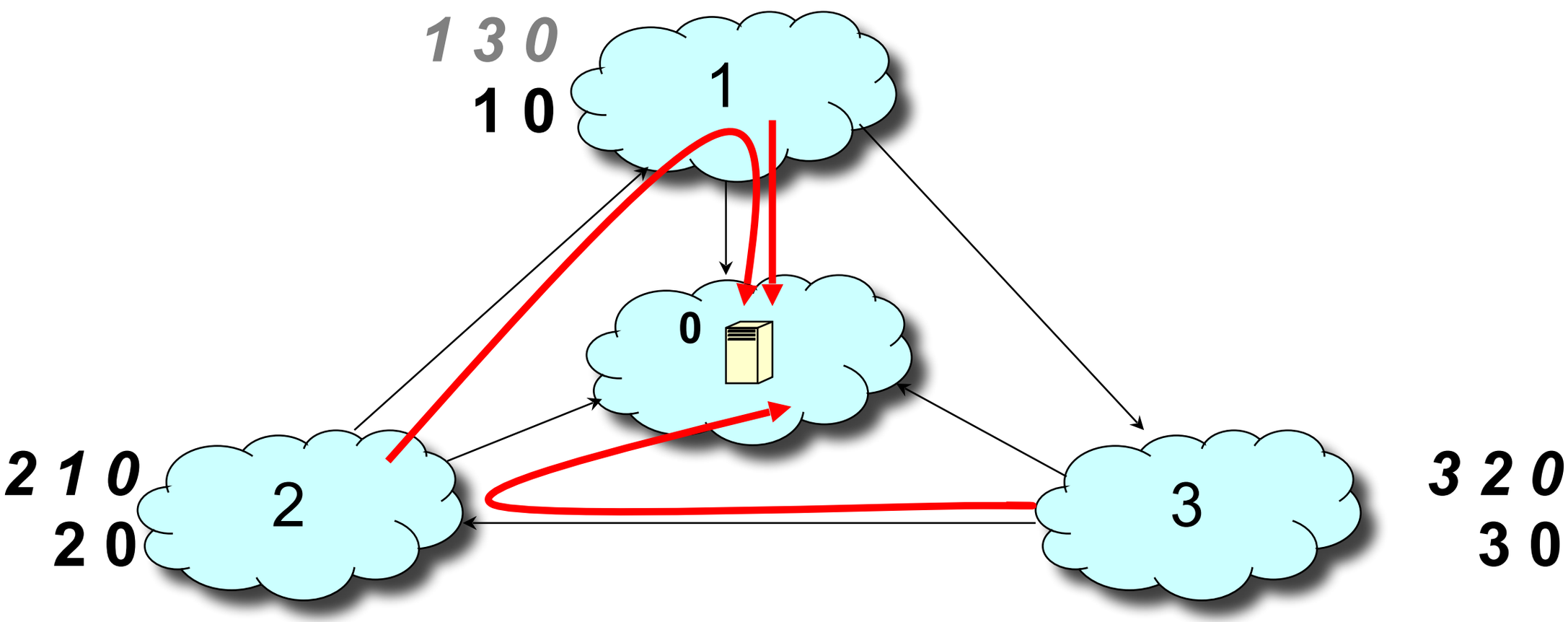


Step-by-step Policy Oscillation

1 **advertises** its path 1 0 to 2

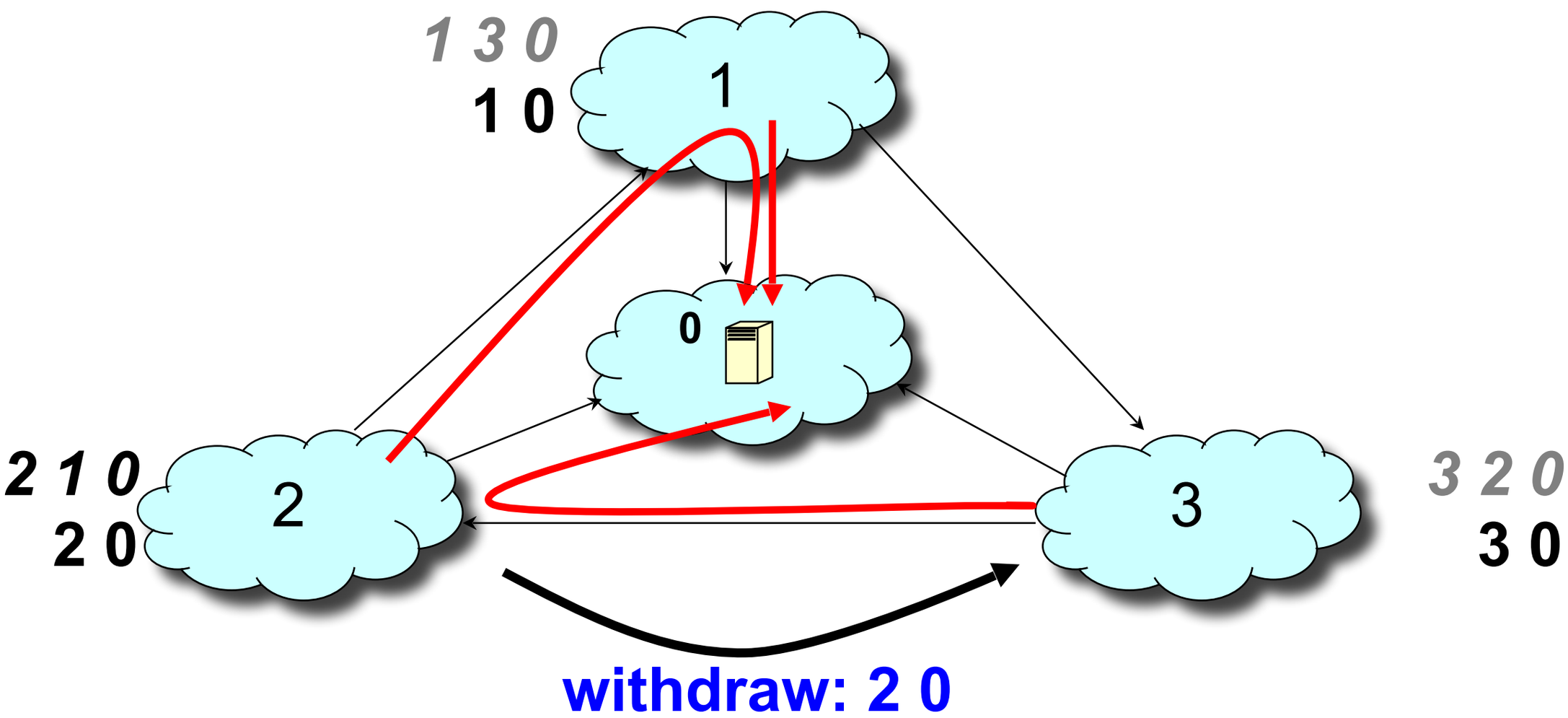


Step-by-step Policy Oscillation

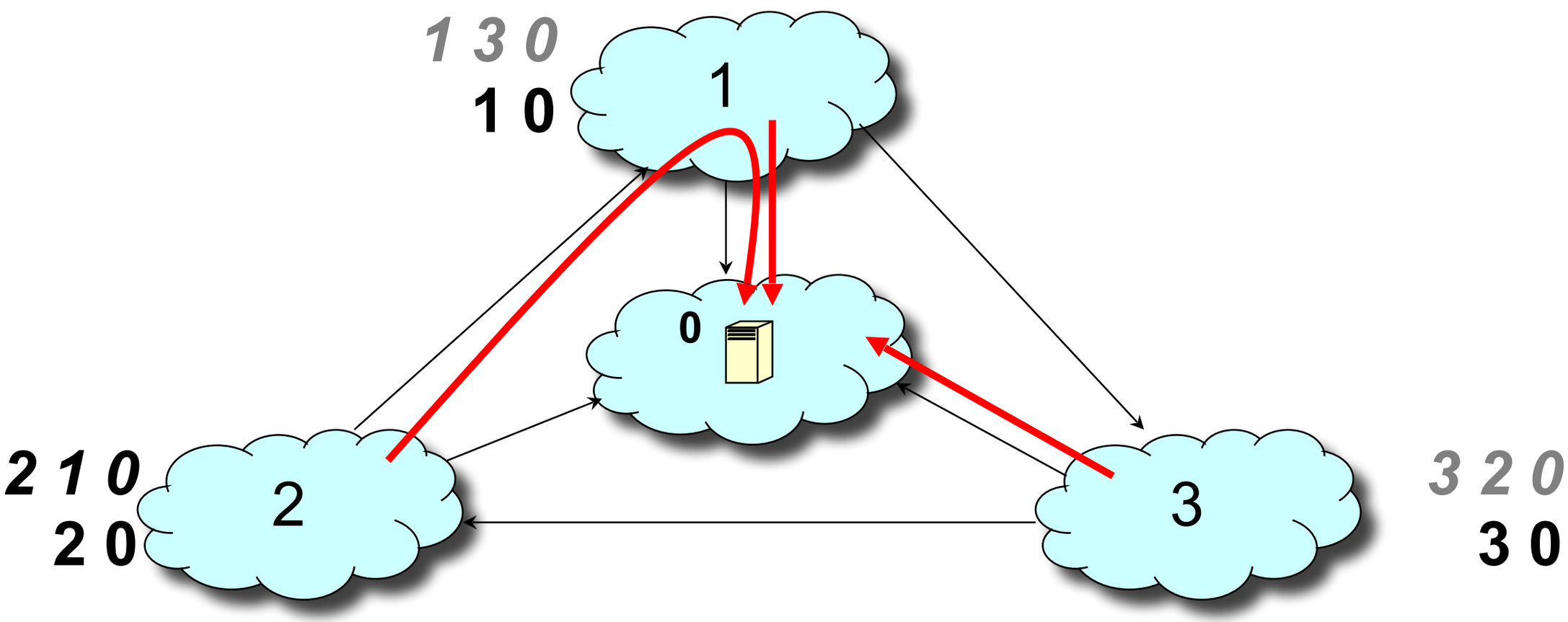


Step-by-step Policy Oscillation

2 **withdraws** its path 2 0 from 3

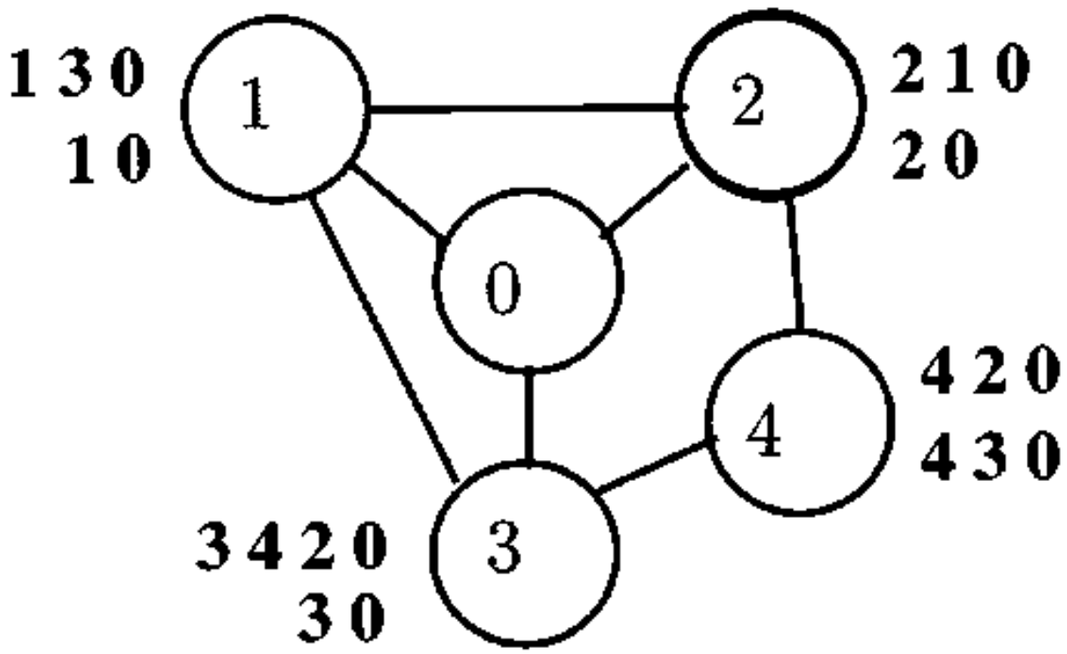


Step-by-step Policy Oscillation



We are back to where we started!

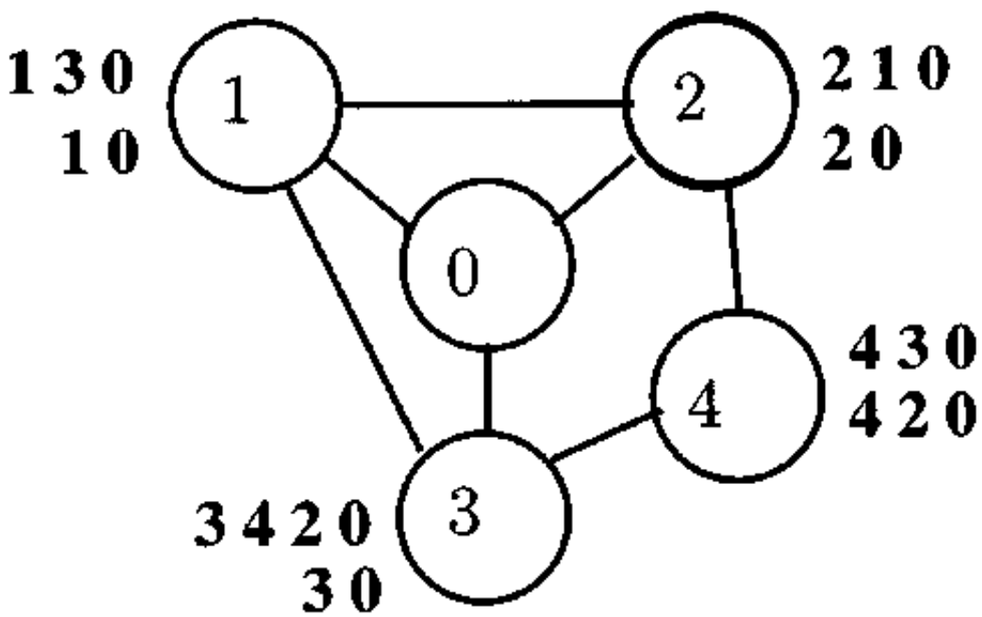
BGP Example (Persistent Loops)



BAD GADGET

	1	2	3	4
R1	10	**20**	30	-
R2	10	20	30	**420**
R3	**10**	20	3420	420
R4	10	**210**	3420	420
R5	10	210	3420	**-**
R6	10	210	**30**	-
R7	**130**	210	30	-
R8	130	**20**	30	-
R9	130	20	30	**420**
R10	130	20	**3420**	420
R11	10	20	3420	420

BGP Example (Bad bad bad)



NAUGHTY GADGET

	1	2	3	4
R1	10	20	**30**	-
R2	10	20	**30**	430
R3	130	20	30	430

	1	2	3	4
R1	10	**20**	30	-
R2	10	20	30	**420**
R3	**10**	20	3420	420
R4	10	**210**	3420	420
R5	10	210	3420	**-**
R6	10	210	**30**	-
R7	**130**	210	30	-
R8	130	**20**	30	-
R9	130	20	30	**420**
R10	130	20	**3420**	420
R11	10	20	3420	420

Performance Non-Issues

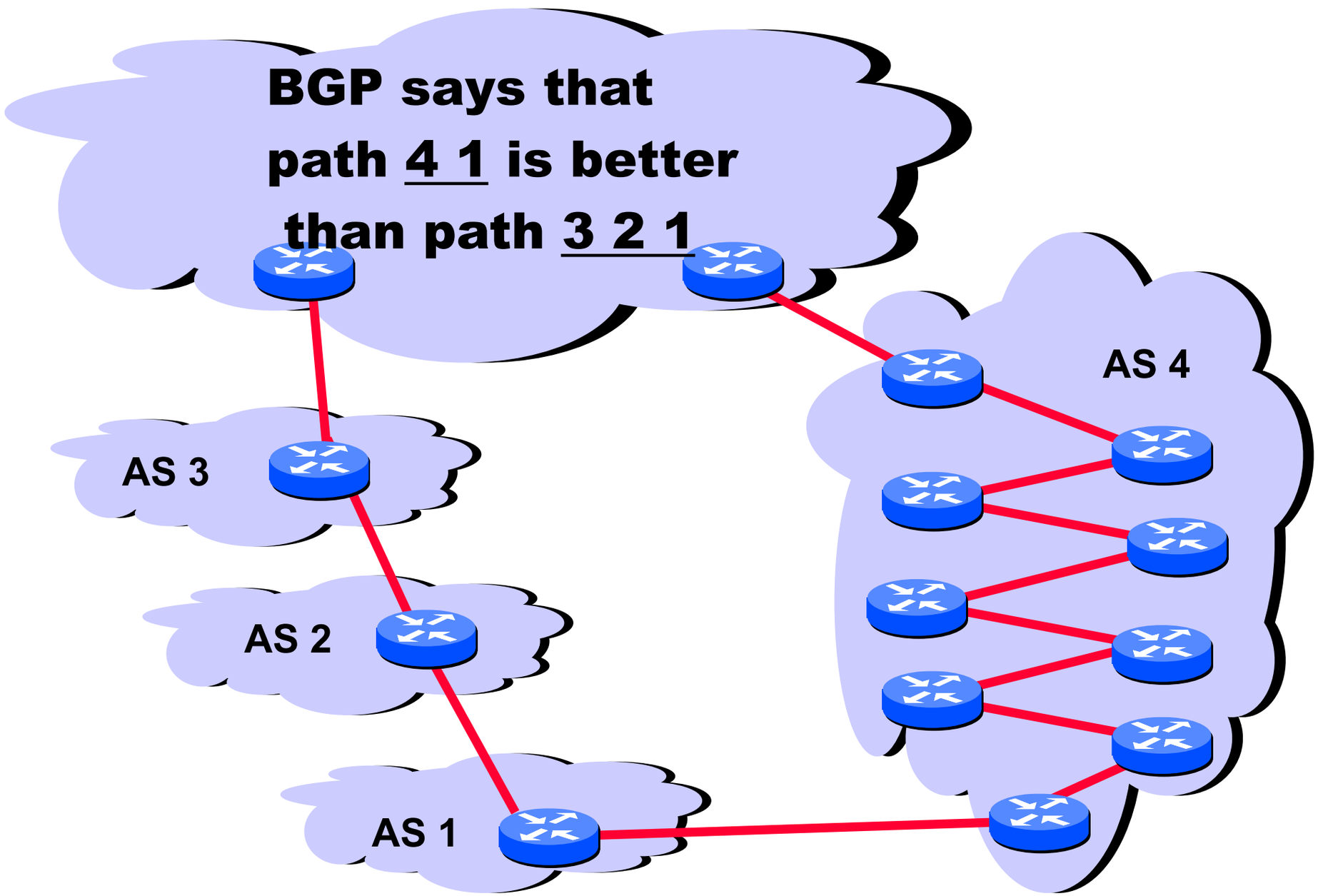
- Internal Routing
  - Domains typically use "hot potato" routing
  - Not always optimal, but economically expedient

- Policy not about performance
  - So policy-chosen paths aren't shortest

- AS path length can be misleading
  - 20% of paths inflated by at least 5 router hops

Performance (example)

- AS path length can be misleading
  - An AS may have many router-level hops



Performance: Real Issue

Slow Convergence

- BGP outages are biggest source of Internet problems
- Labovitz et al. *SIGCOMM'97*
  - 10% of routes available less than 95% of the time
  - Less than 35% of routes available 99.99% of the time
- Labovitz et al. *SIGCOMM 2000*
  - 40% of path outages take 30+ minutes to repair
- But most popular paths are very stable

Security

- An AS can claim to serve a prefix that they actually don't have a route to (blackholing traffic)
  - Problem not specific to policy or path vector
  - Important because of AS autonomy
  - *Fixable: make ASes prove they have a path*

- But…
  - AS may forward packets along a route different from what is advertised
    - Tell customers about a fictitious short path…
    - Much harder to fix!

BGP Outline

- BGP Policy
  - Typical policies and implementation
- BGP protocol details
- Why am I not blindly in love with BGP?
- **Why do I love BGP?**

Think about what BGP is able to achieve….

- **Every ISP acts completely independently and selfishly….**
  - Policies, path selection, path advertisement, complete privacy
  - Creates relationships independently
  - Cares only about its own interest—money earned
  - Operates in a distributed manner
  - **And yet ….**
- **How frequently are we disconnected from the Internet?**
  - Almost rare ….
  - **It just works …. And …**

BGP is one of those examples where ….

- **Real-world protocol that was born in an organic manner …**
  - **… has led to advances in computing**
  - A recent result:
    - BGP is an example of the “hardest” Nash equilibrium instance
    - Many recent beautiful results in theoretical computer science
  - **But …**

- **My real love?**
  - **It just works …. And nobody knows why!!**

- **Has been working since 1991(5)**
  - With almost no changes …
  - Perfect in inception, but perhaps poor in execution

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

One last piece—Packet headers

Network Layer

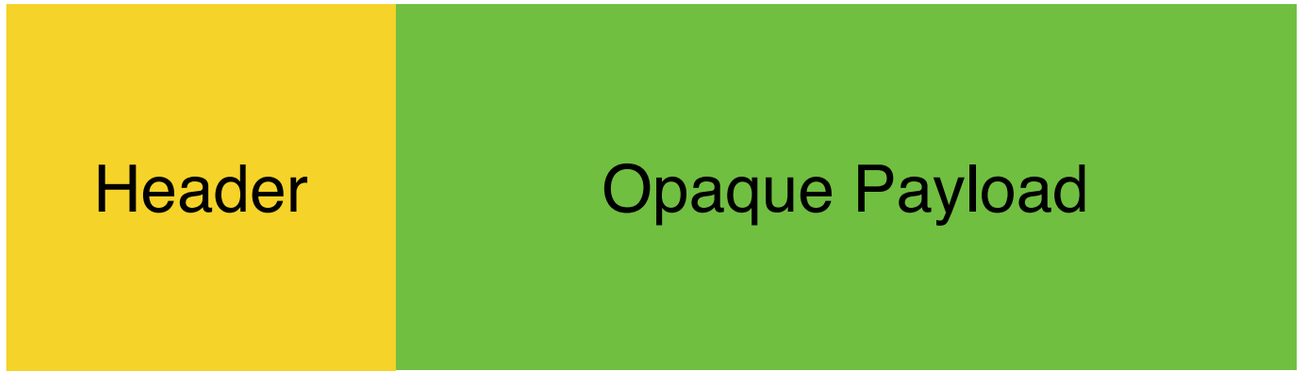
- THE functionality: **delivering the data**

- **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)

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

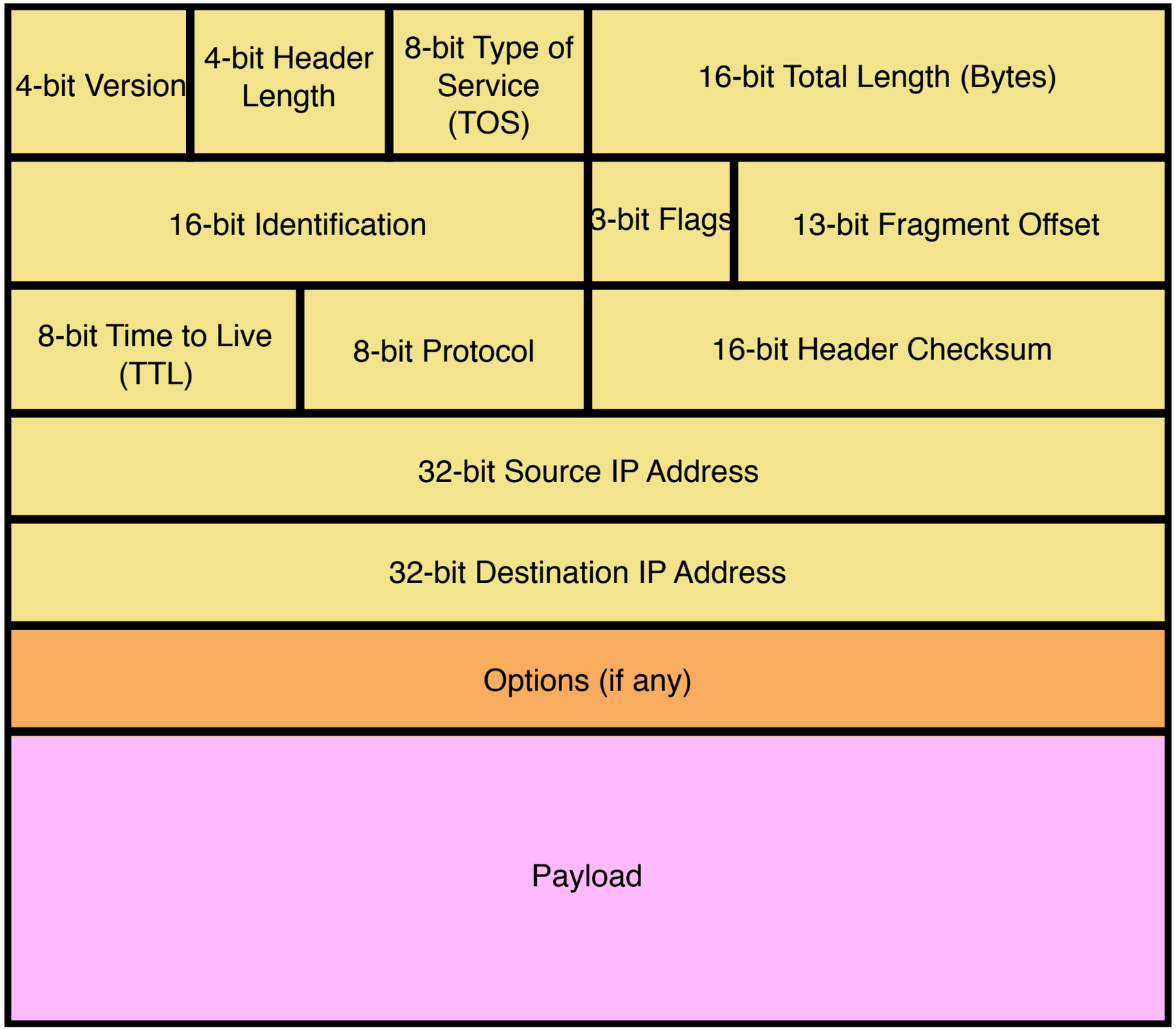
Are We Missing Anything?

- 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

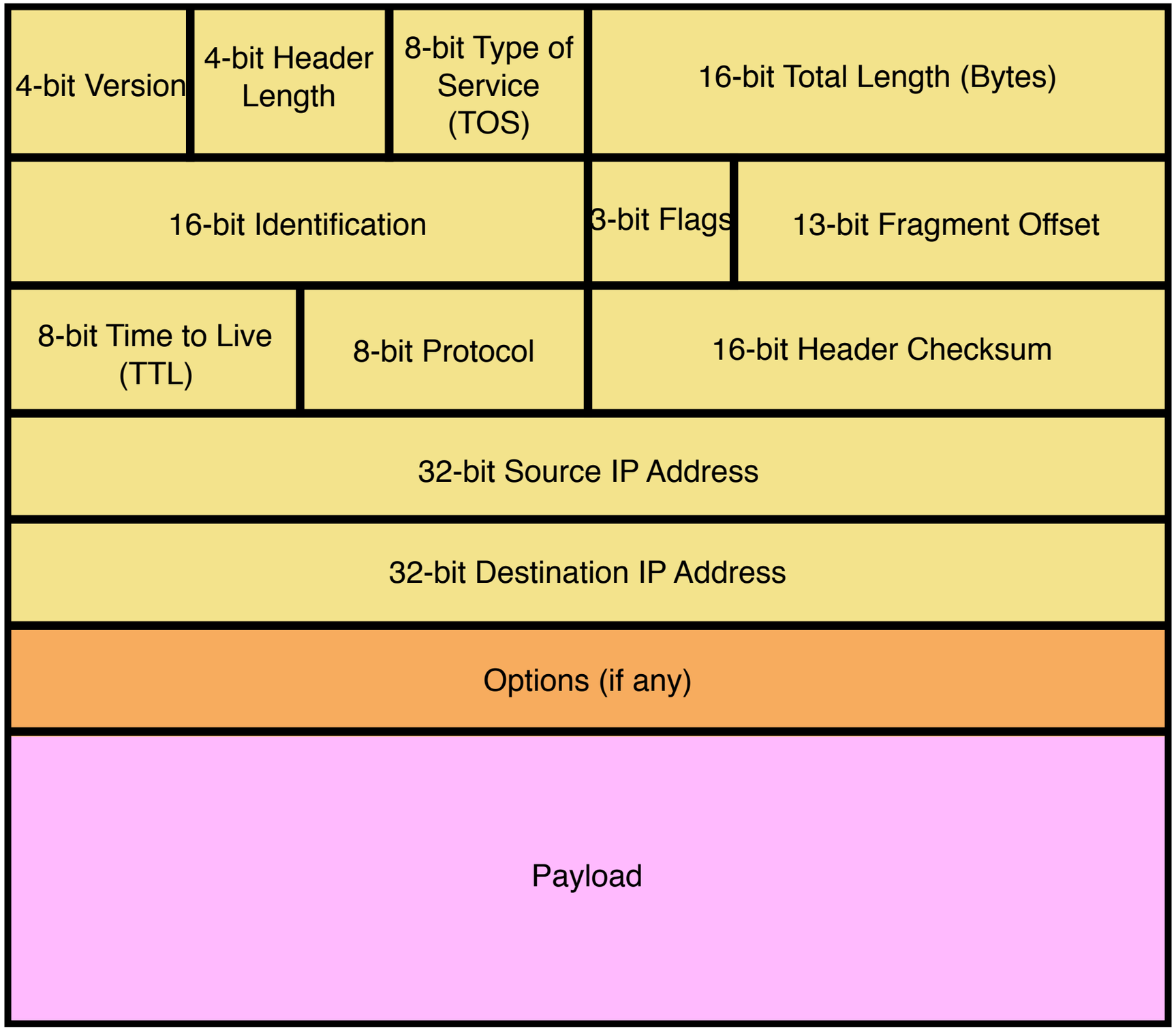
From Semantics to Syntax

- 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…

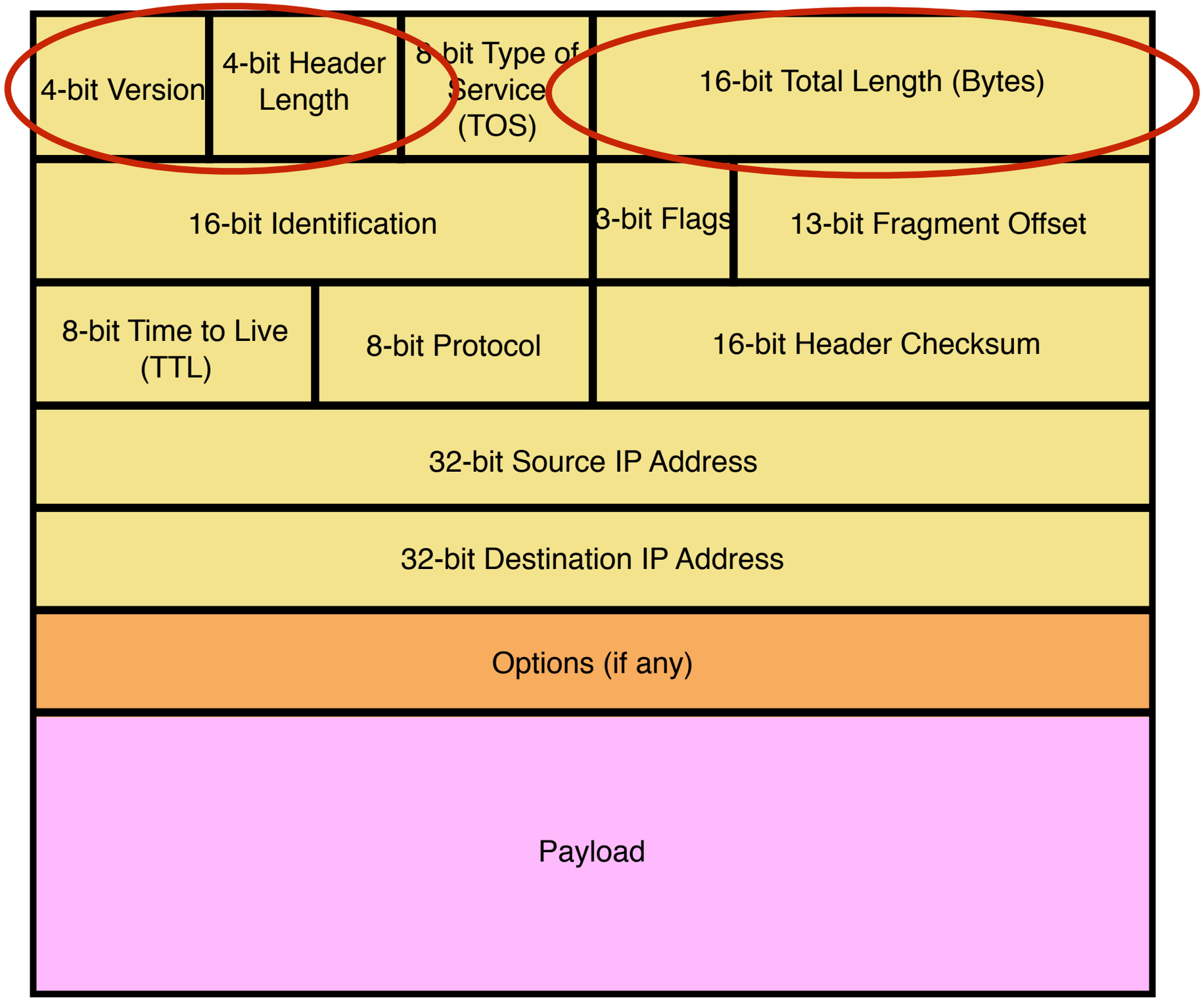
Go Through Tasks One-by-One

- 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

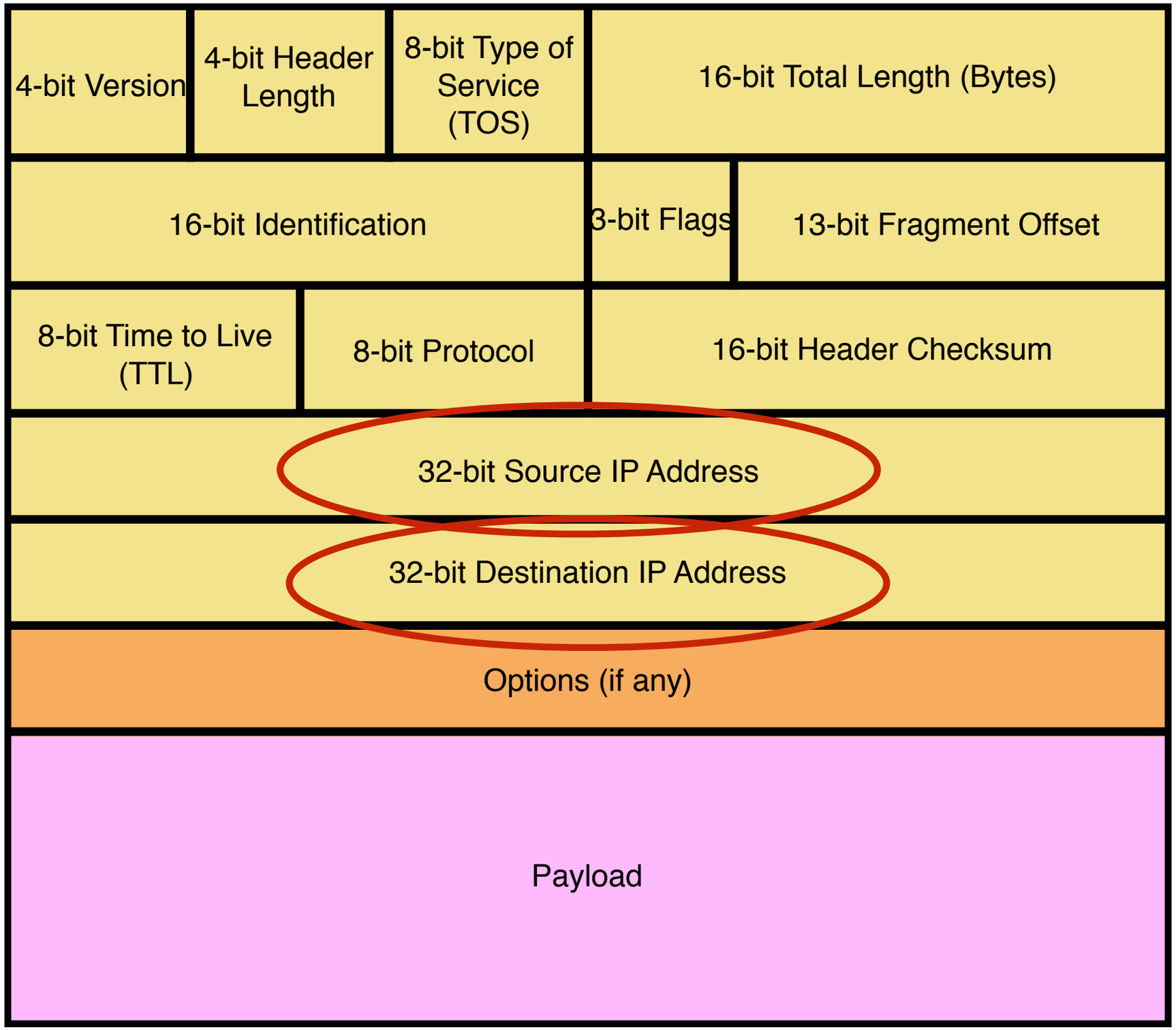
Fields for Reading Packet Correctly



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



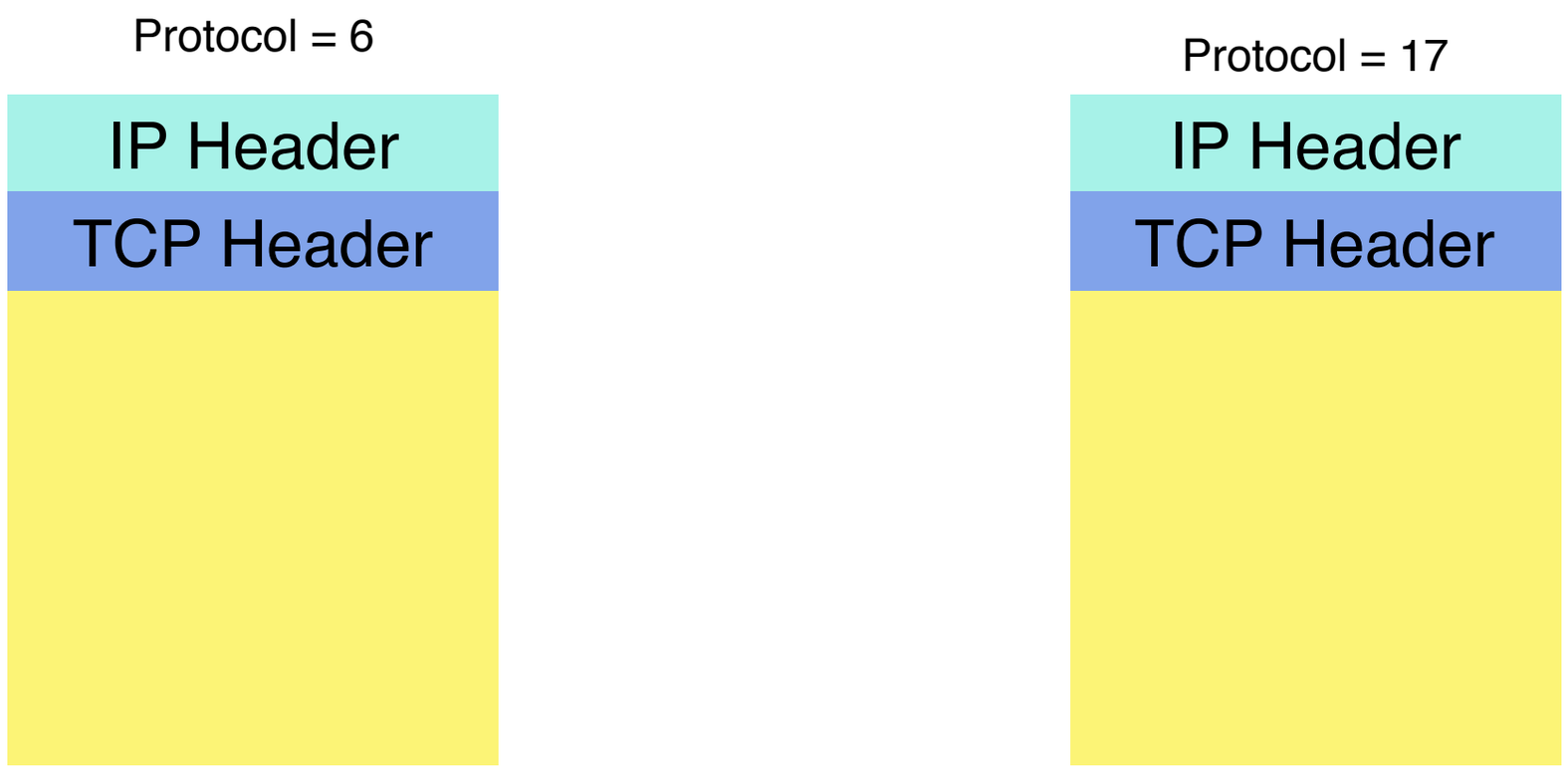
Questions?

List of Tasks

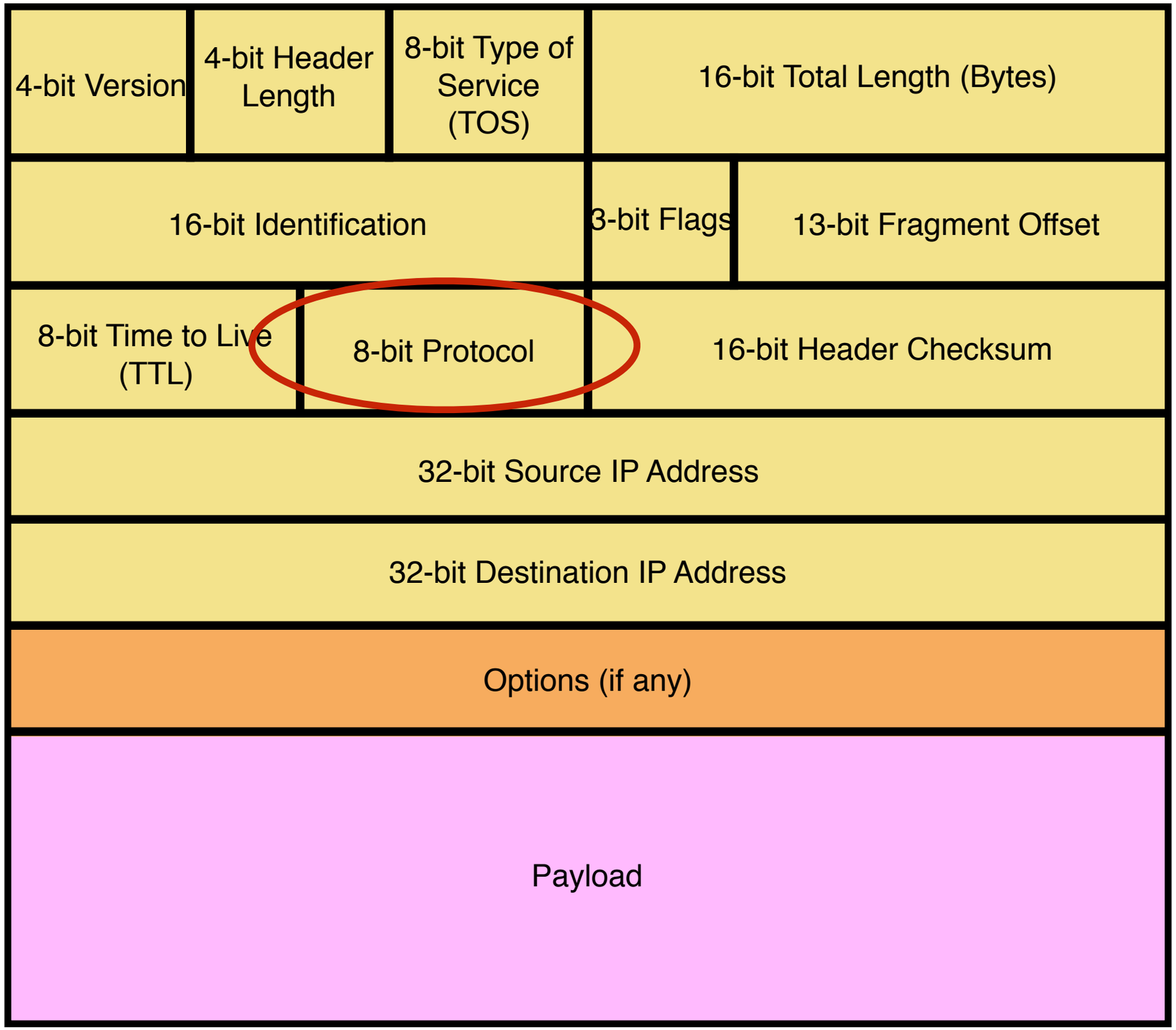
- Read packet correctly
- Get the packet to the destination
- Get responses to the packet back to source
- Carry data
- **Tell host what to do with packet once arrived**
- Specify any special network handling of the packet
- Deal with problems that arise along the path

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

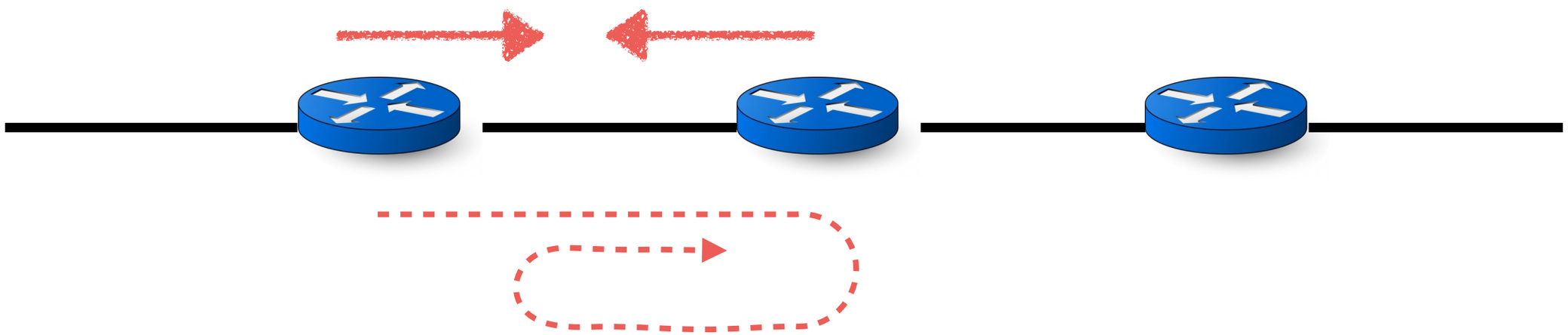
- Record Route
- Strict Source Route
- Loose Source Route
- Timestamp
- Traceroute
- Router Alert
- …

Potential Problems

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

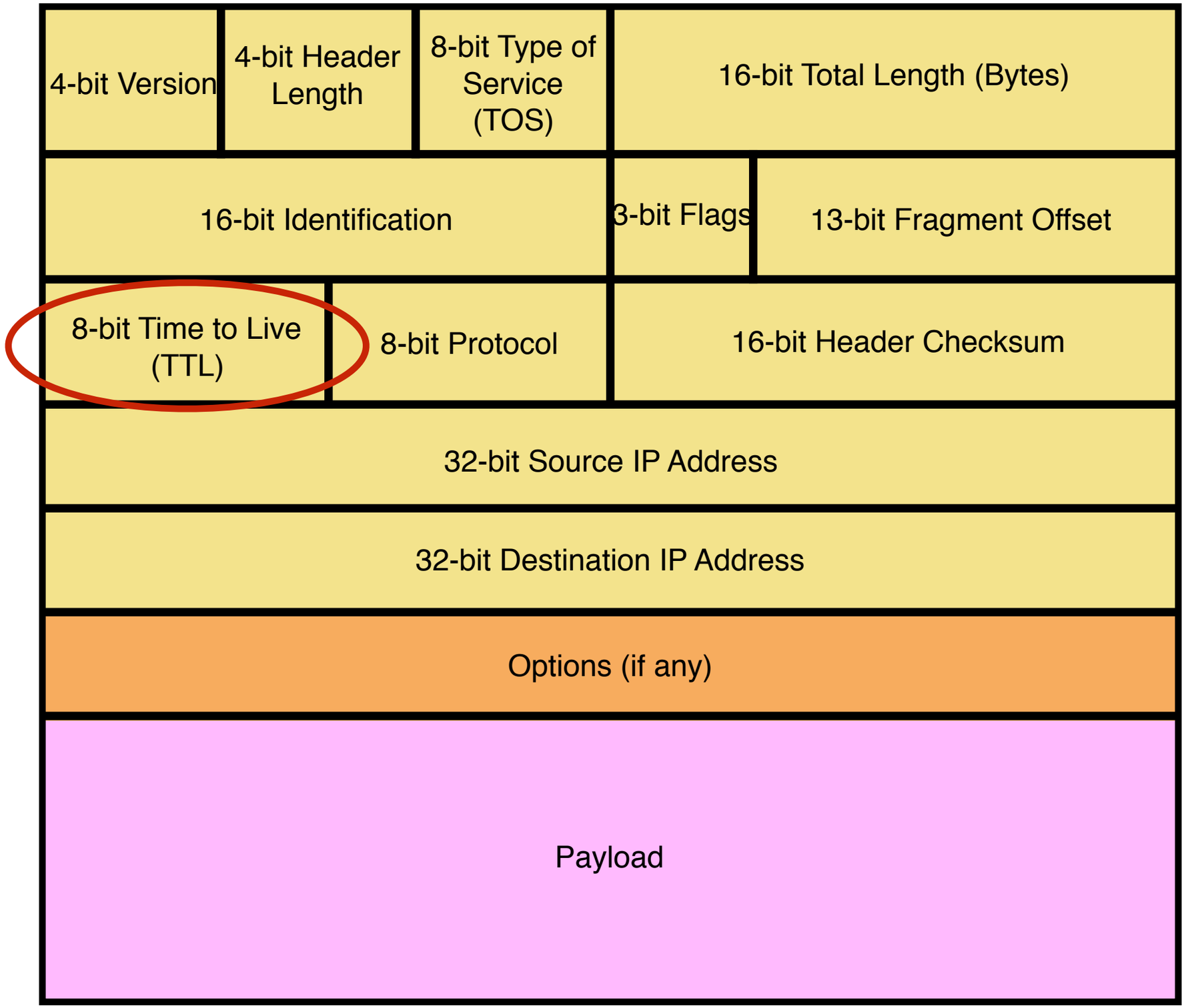
Preventing Loops

- 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

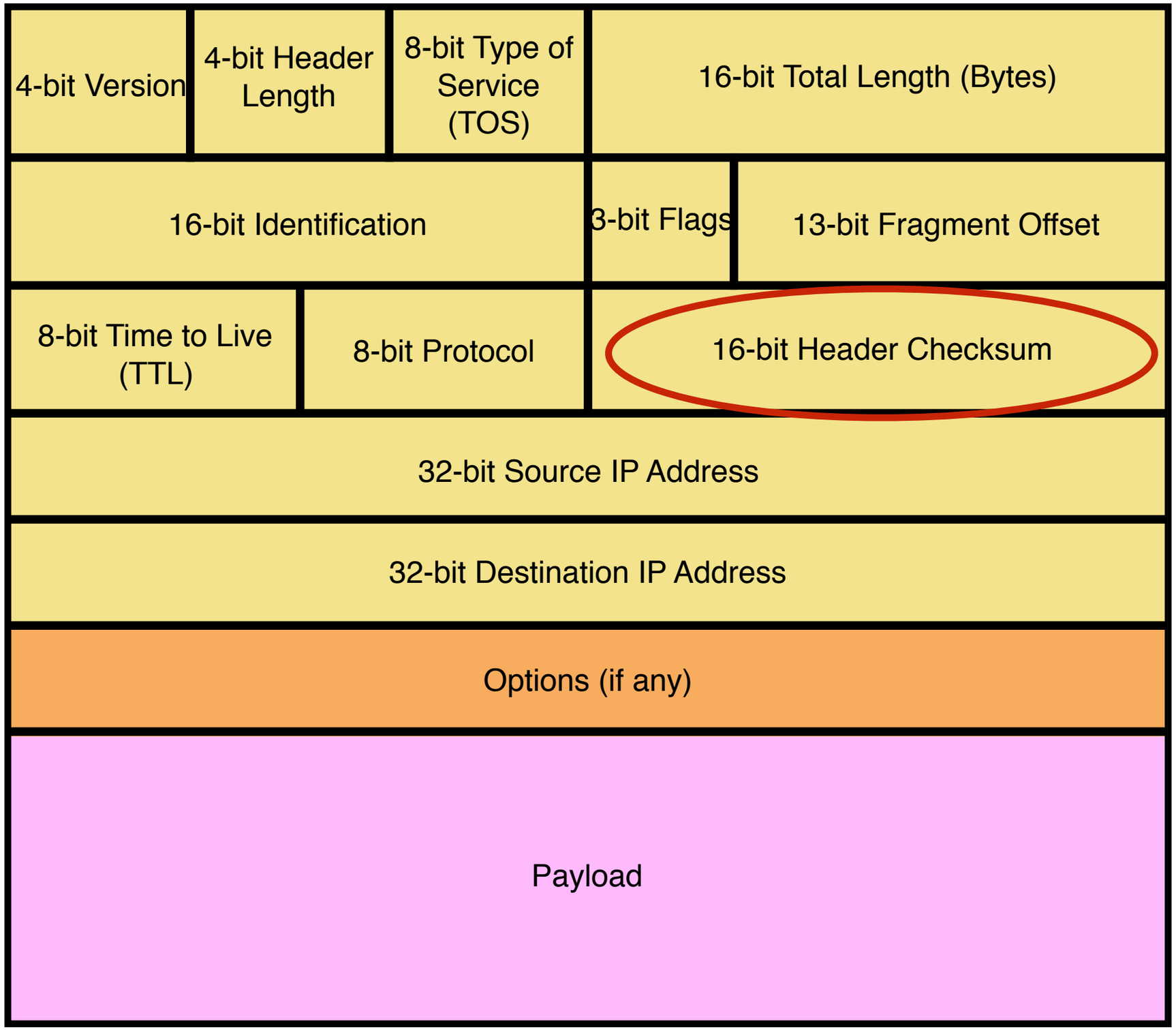
TTL Field


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

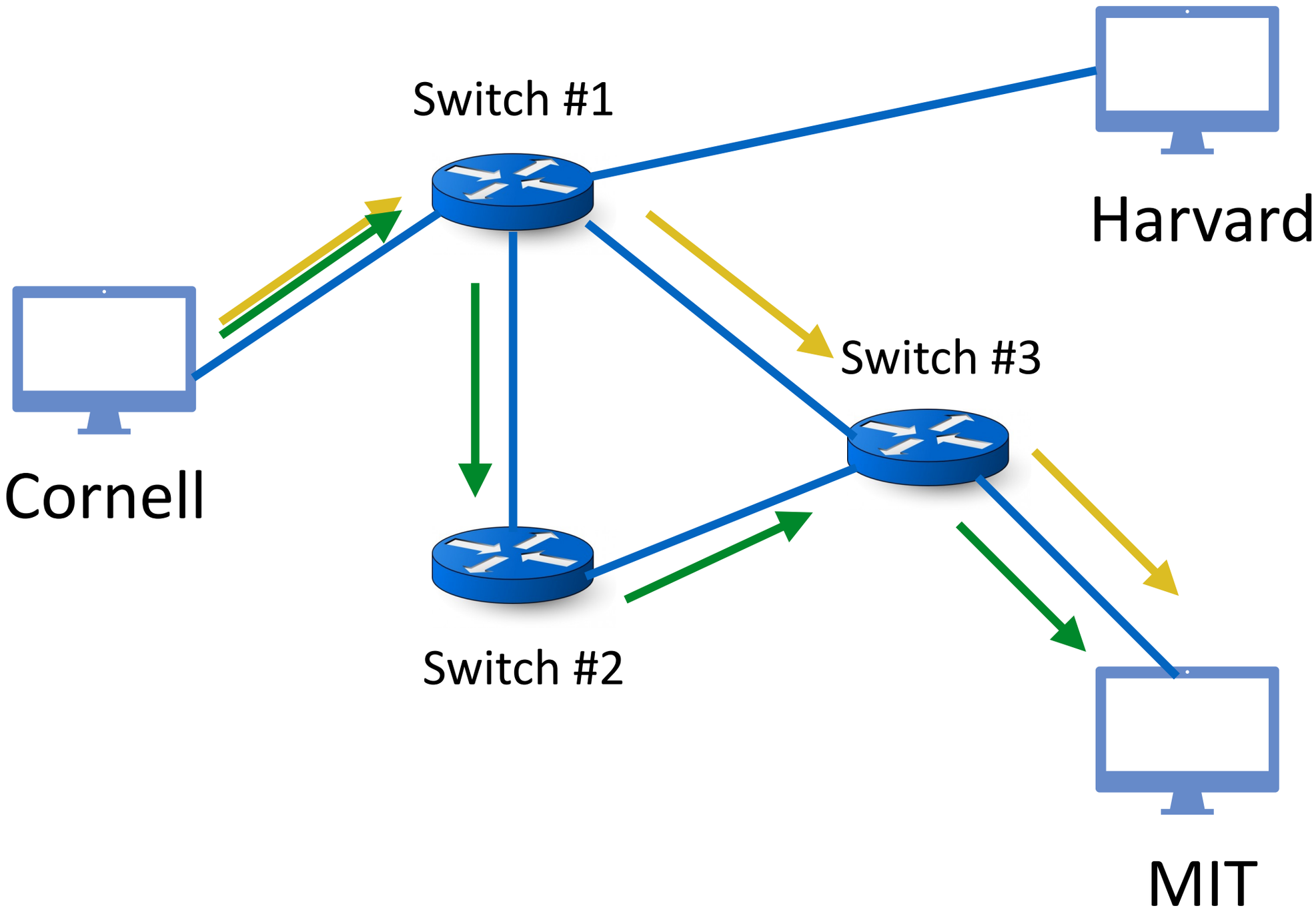
- **Useless to learn the header format by heart**
  - If you remember the tasks that need to be performed …
  - Understanding **why** header format is what it is …
  - In general: if you understand the problem, solution is easy
  - As the problem evolves, you will know where to look for a solution
-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

Switch/Router Architecture

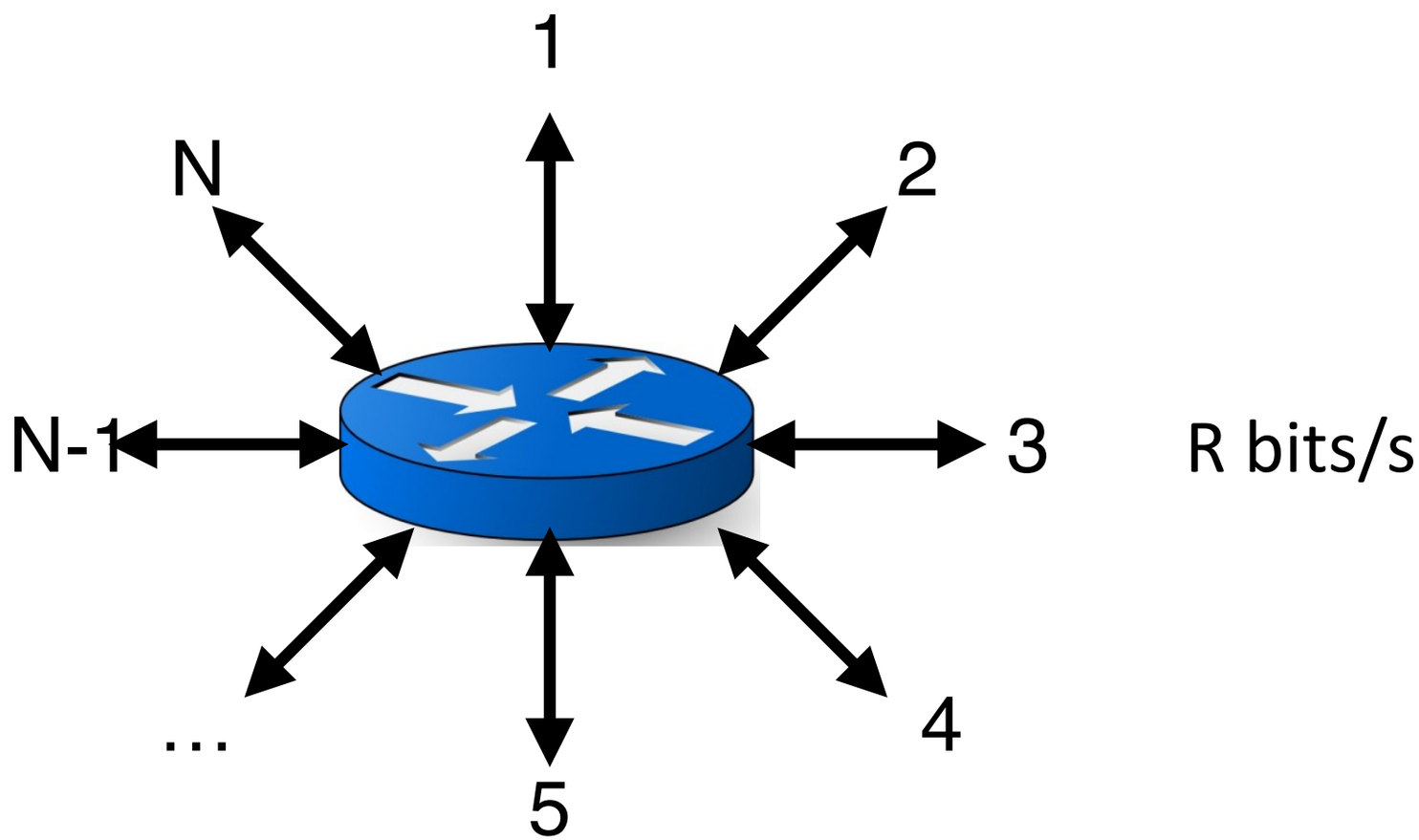
IP Routers and Switches (used interchangeably today)

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

Recap: Routers Forward Packets

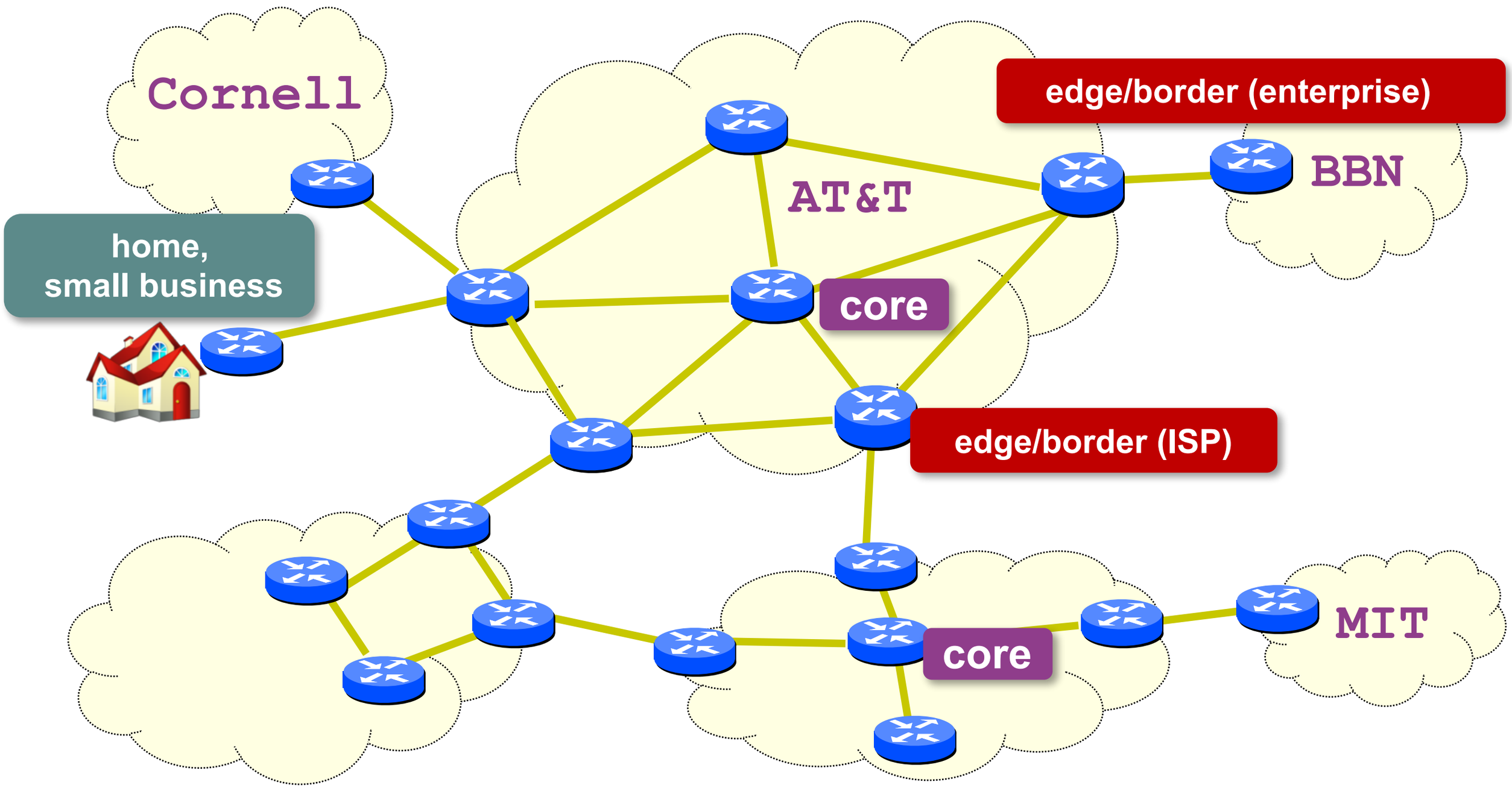


Router Definitions



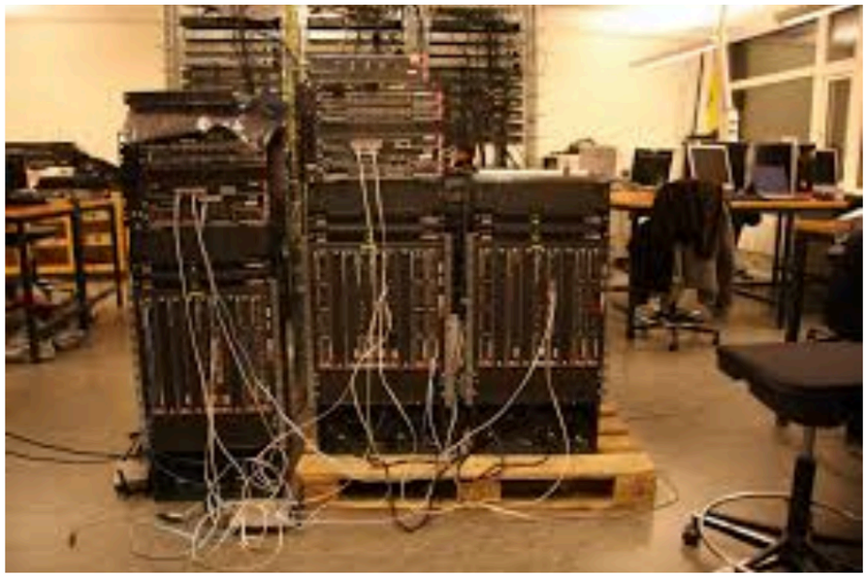
- N = No. Of external router ports
- R = bandwidth (“line rate”) of a port
- Router capacity = NxR

Networks and Routers



Examples of Routers (core)

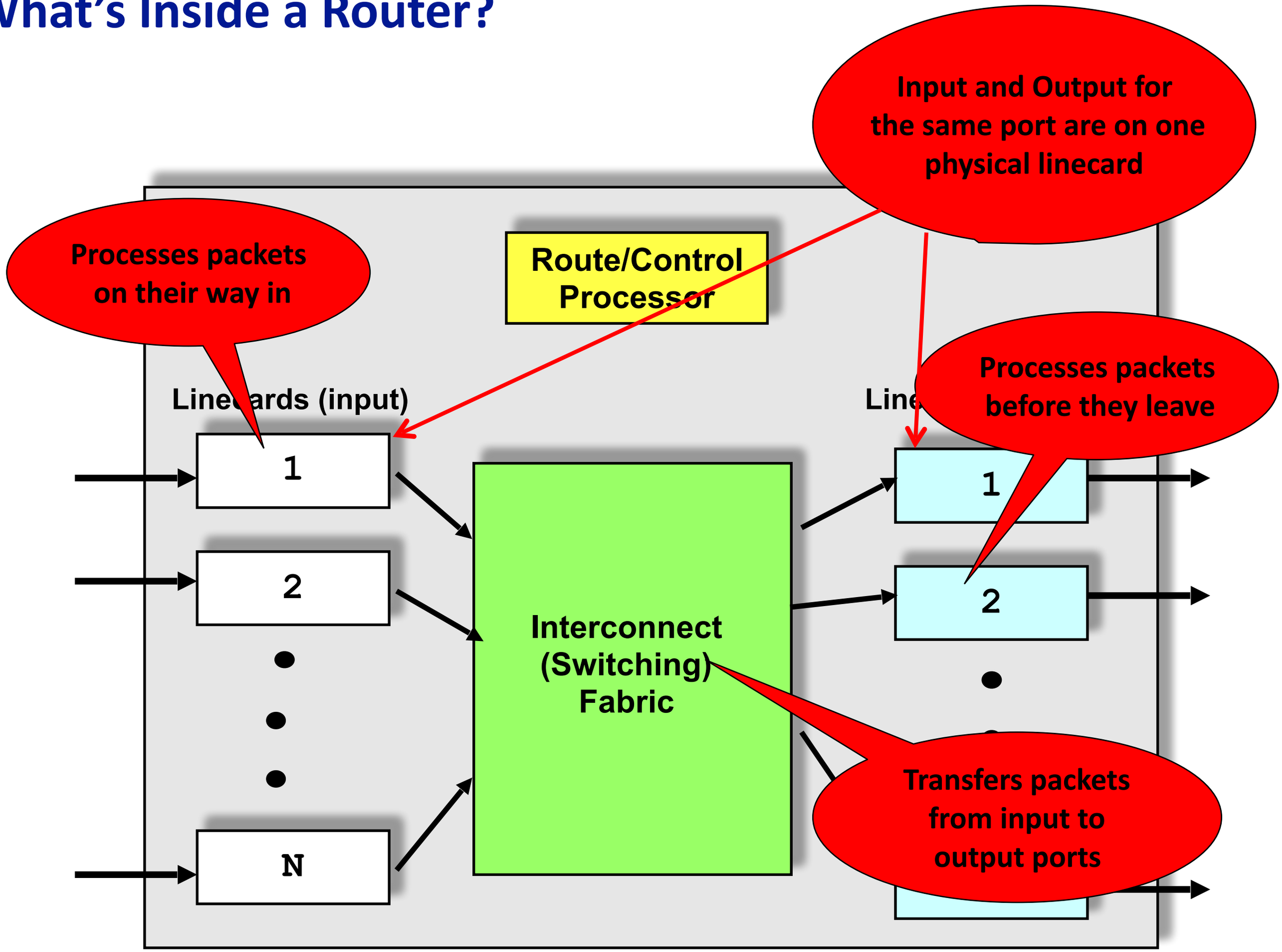
- **Core:** Cisco CRS
  - R = 10/40/100 Gbps
  - NR = 922 Tbps
  - Netflix: 0.7 GB/hr (1.5Mb/s)
  - ~600 million concurrent Netflix users



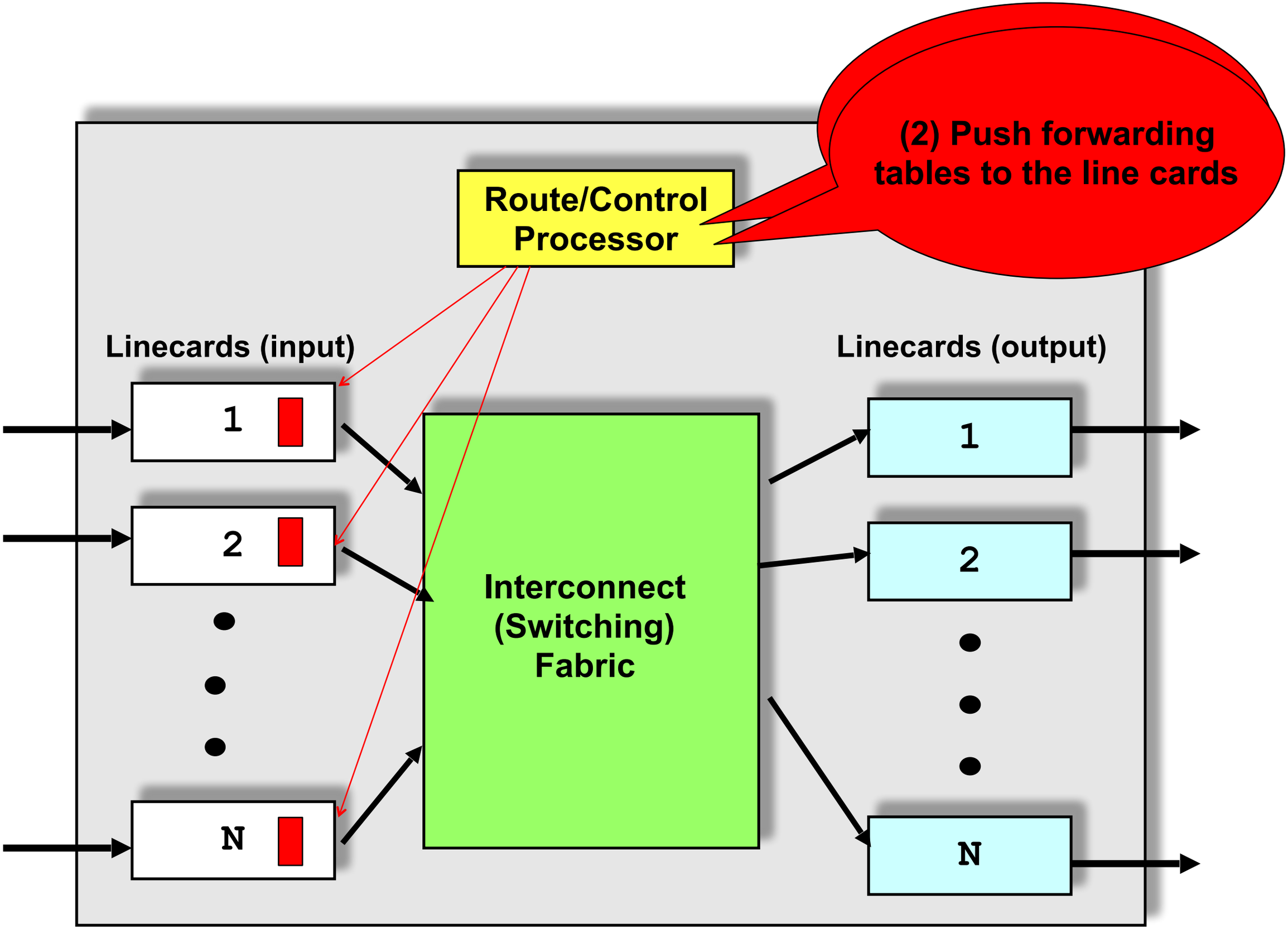
- **Edge (ISP):** Cisco ASR
  - R = 1/10/40 Gbps
  - NR = 120 Gbps

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

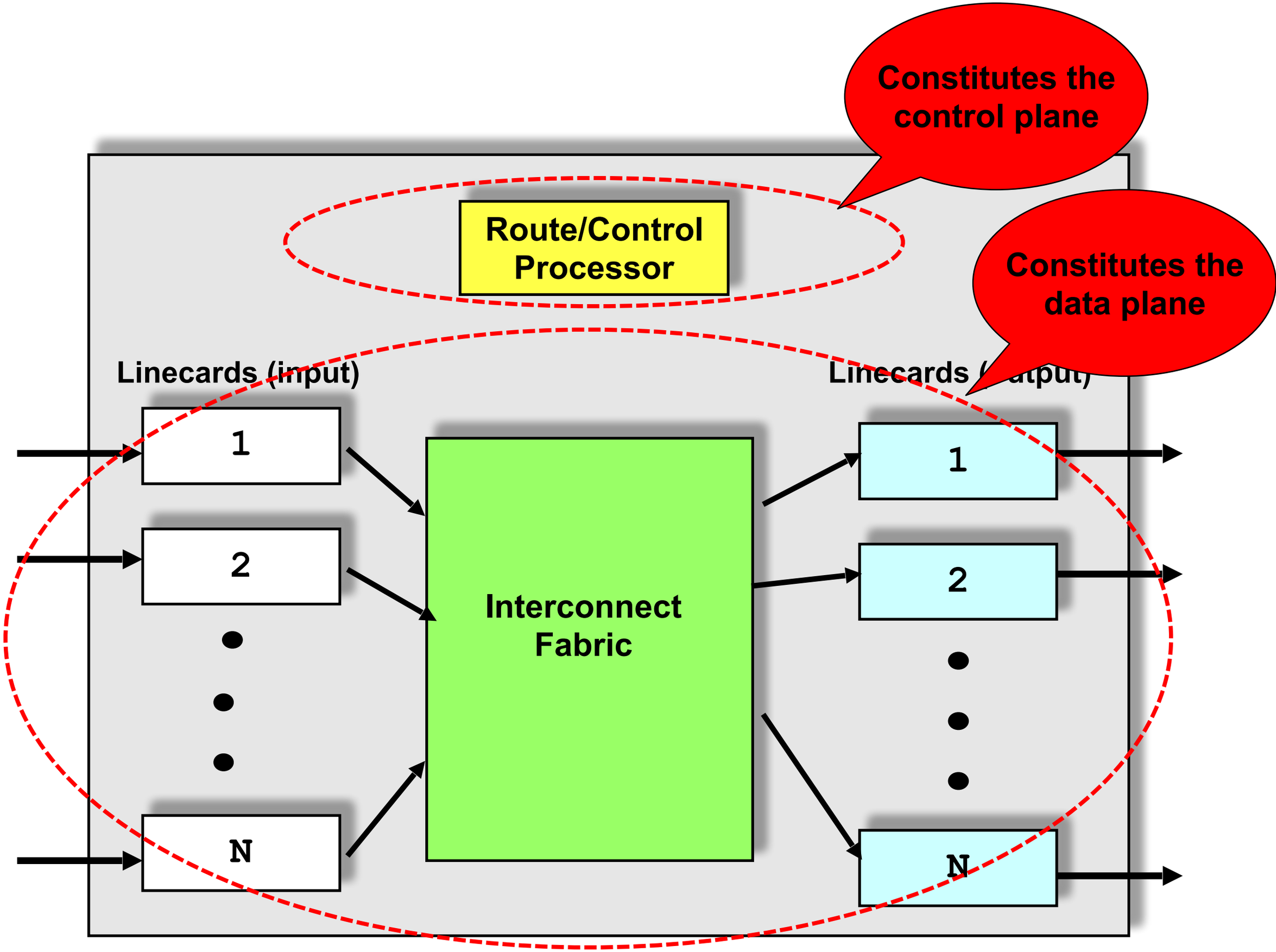
What's Inside a Router?



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”

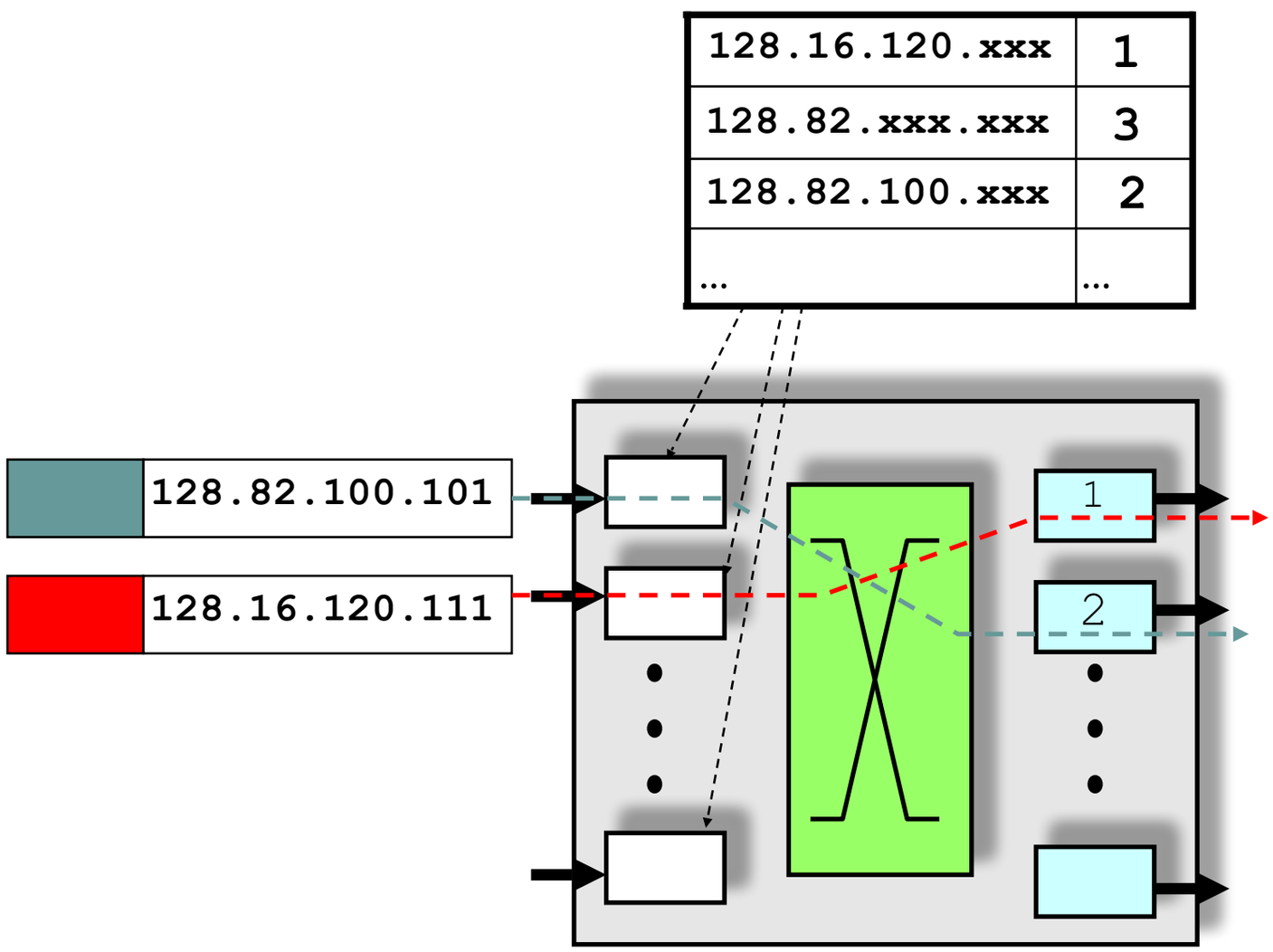
Looking up the Output Port

- 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	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	000001 1-	- - - - - - - -

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)

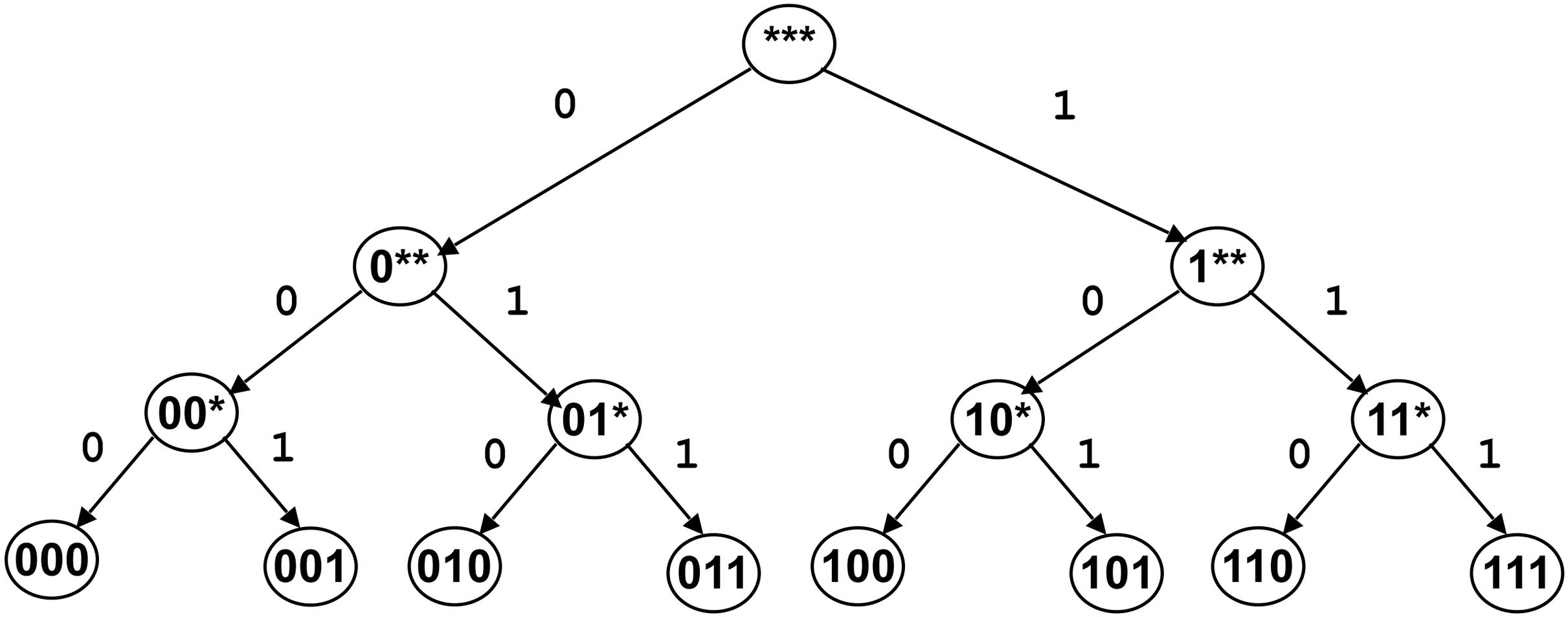
- Must leverage tree structure of binary strings
  - Set up tree-like data structure
  - Called a **TRIE**
    - We will briefly discuss it; more details in text
    - In case you are interested ….

Consider Four 3-Bit Prefixes

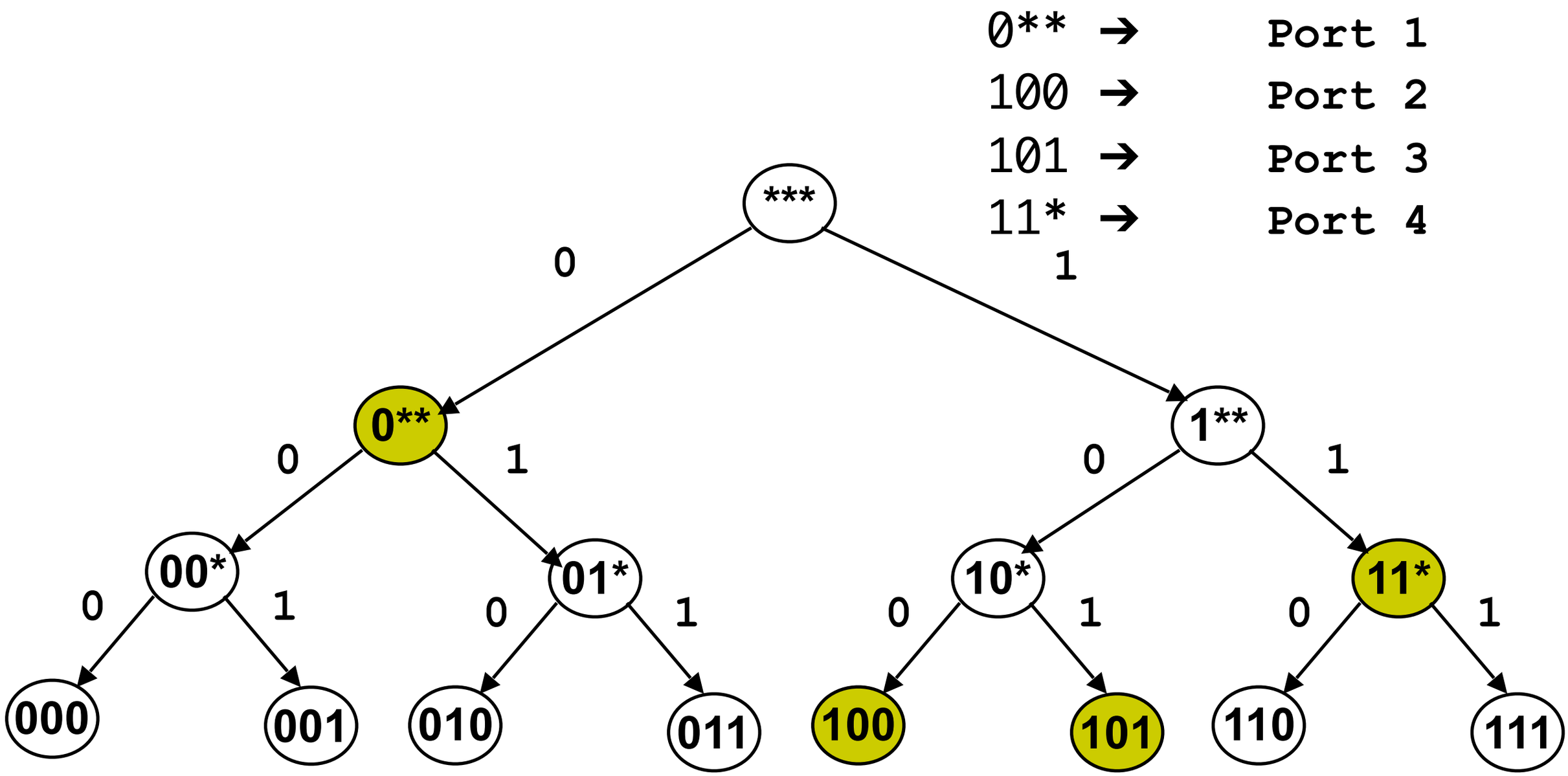
- Just focusing on the bits where all the action is….

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

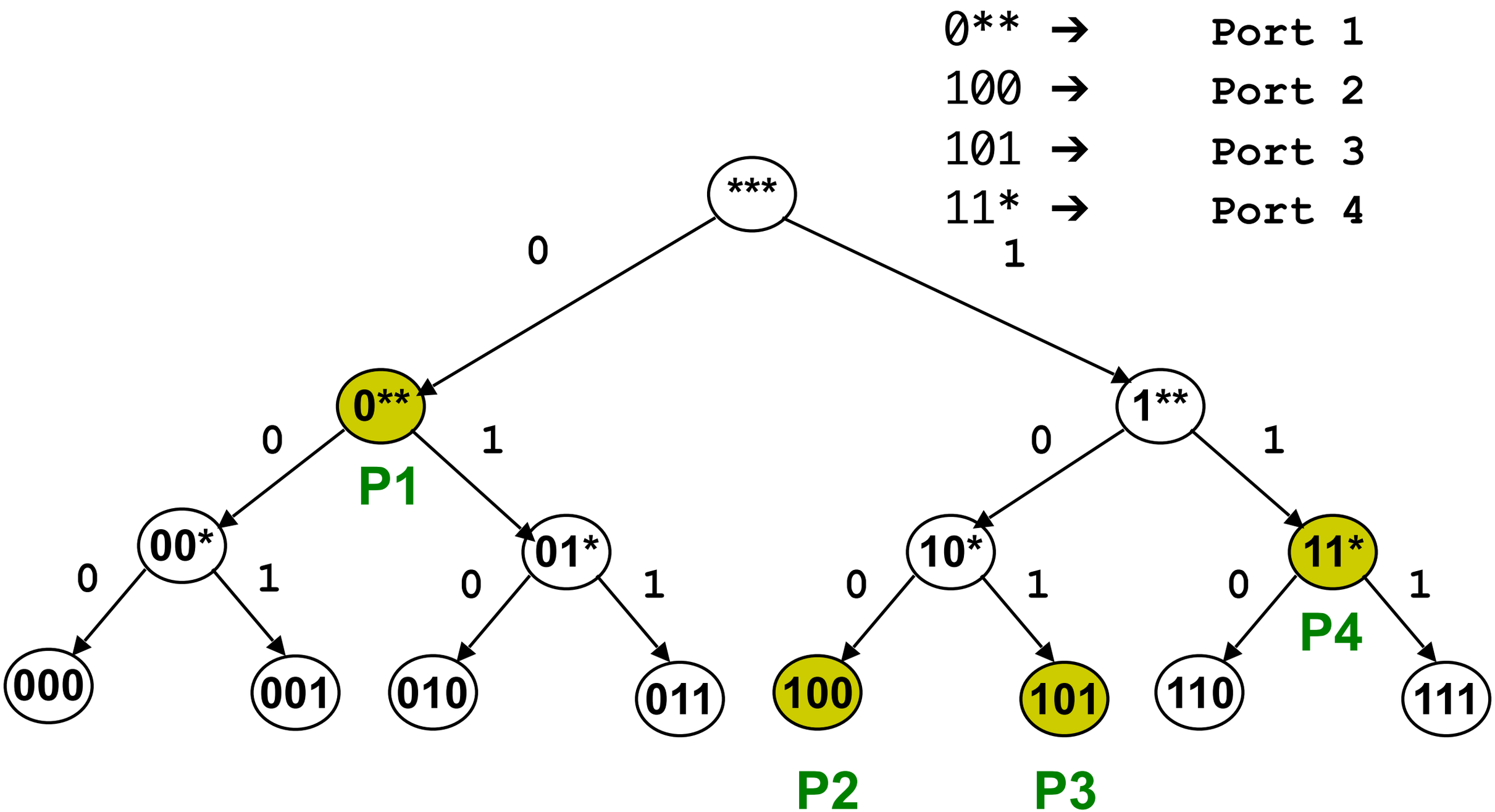
Tree Structure



Walk Tree: Stop at Prefix Entries



Walk Tree: Stop at Prefix Entries



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

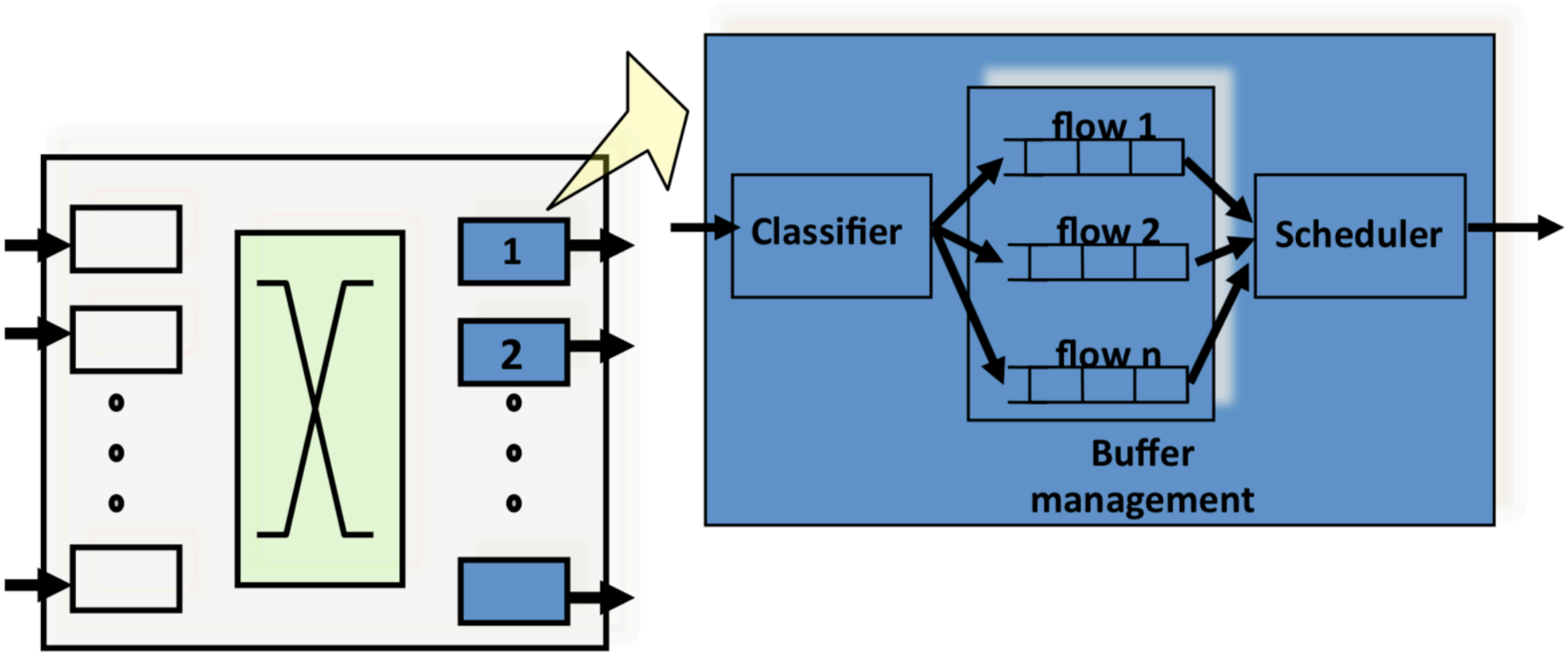
Recap: Input Linecards

- 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



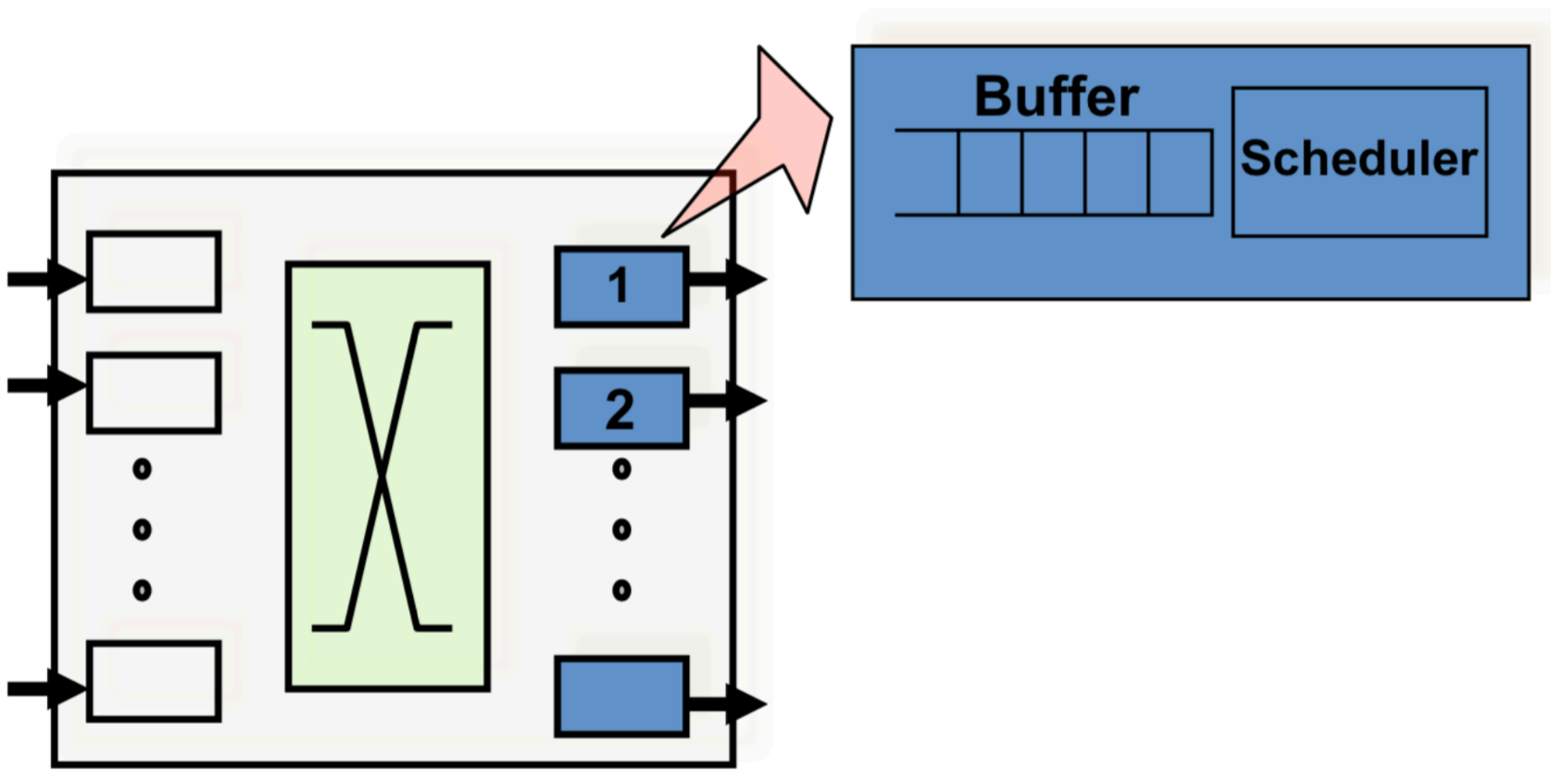
- **Packet Classification**: map each packet to a "flow"
    - Flow (for now): set of packets between two particular endpoints
- **Buffer Management**: decide when and which packet to drop
- **Scheduler**: decide when and which packet to transmit

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

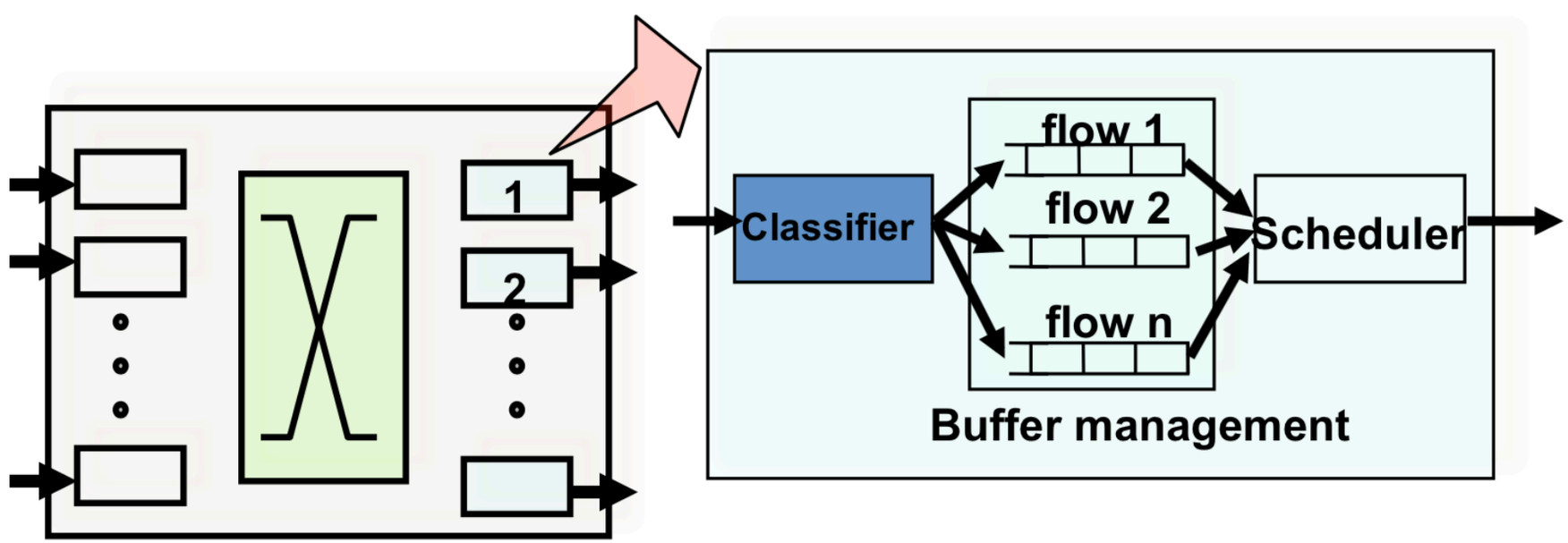
- **No classification**
- **Drop tail buffer management:** when buffer is full drop incoming packet
- **First In First Out (FIFO) Scheduling:** schedule packets in order of arrival



Packet Classification

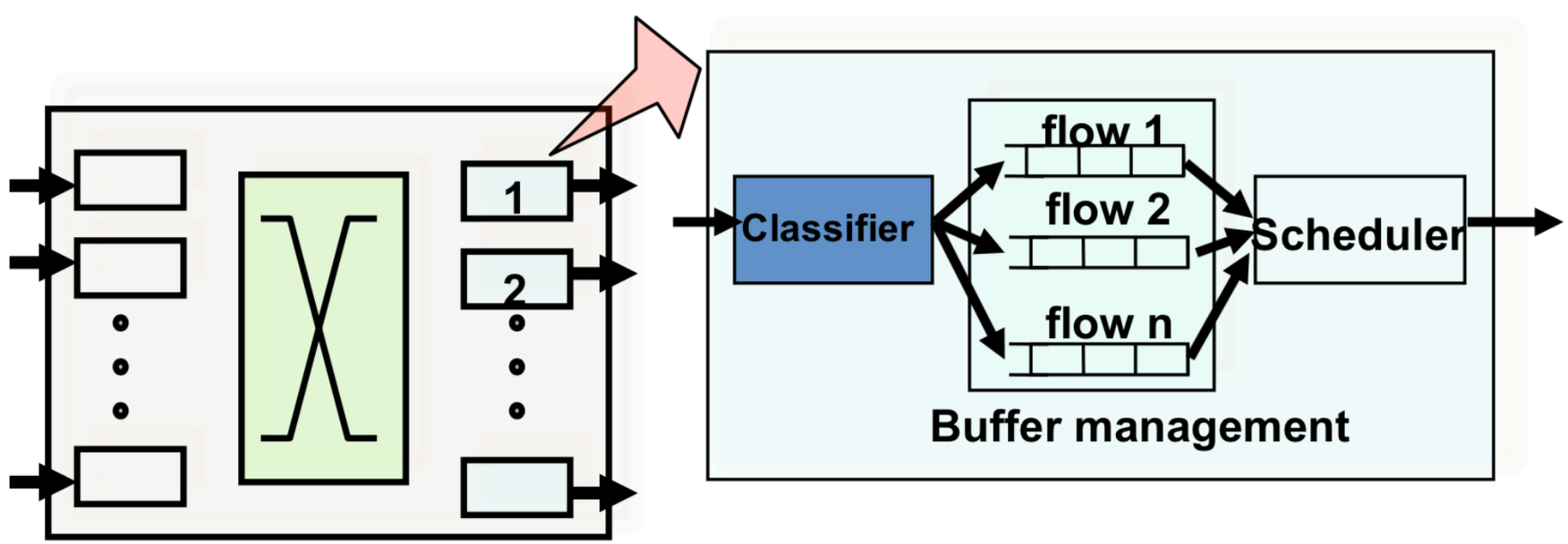
- Classify an IP packet based on the number of fields in the packet header
  - Source/destination IP address (32 bits)
  - Source/destination TCP port number (16 bits)
  - Type of Service (TOS) byte (8 bits)
  - Type of Protocol (8 bits)

- In general fields are specified by range
  - Classification requires a multi-dimensional range search



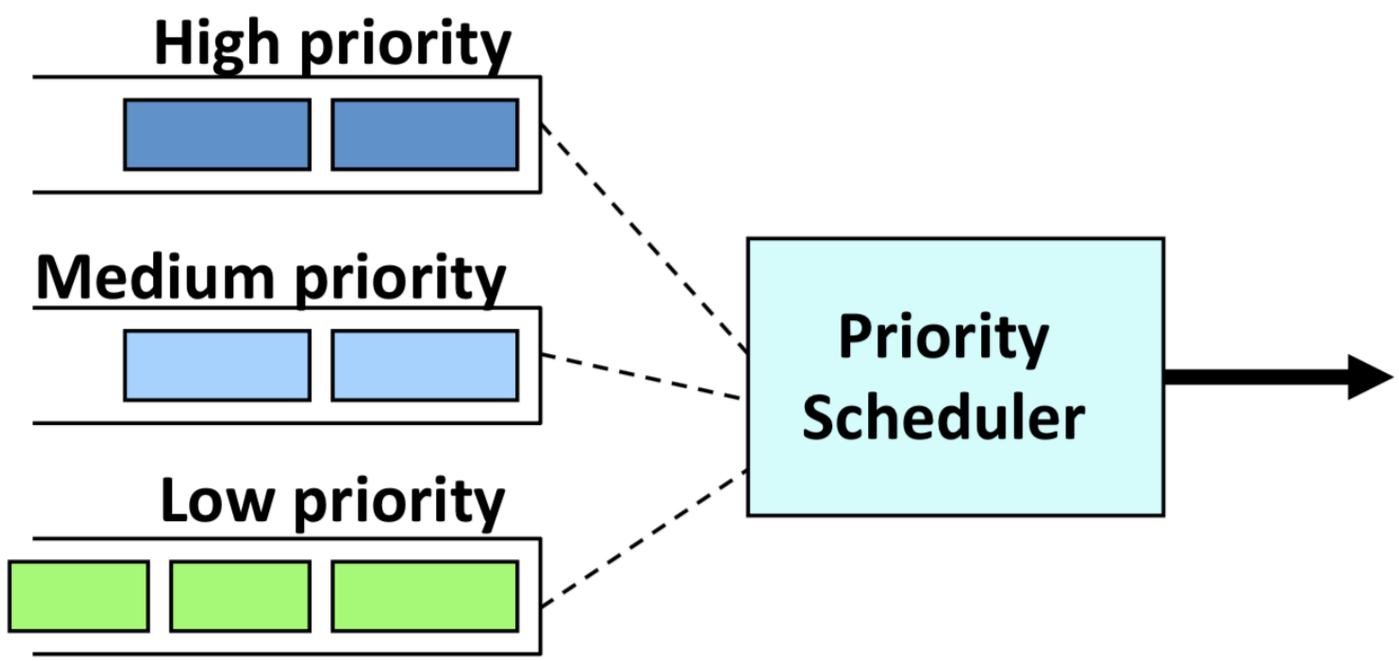
Scheduler

- 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.)



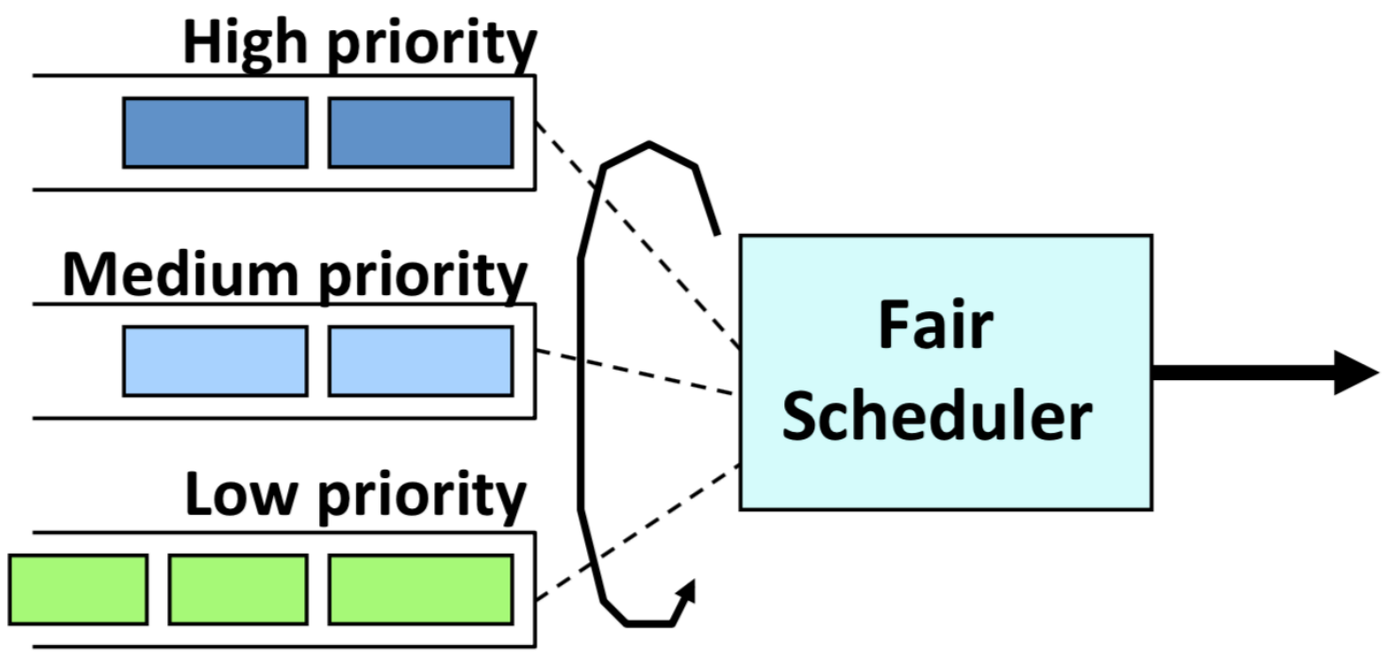
Example: Priority Scheduler

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



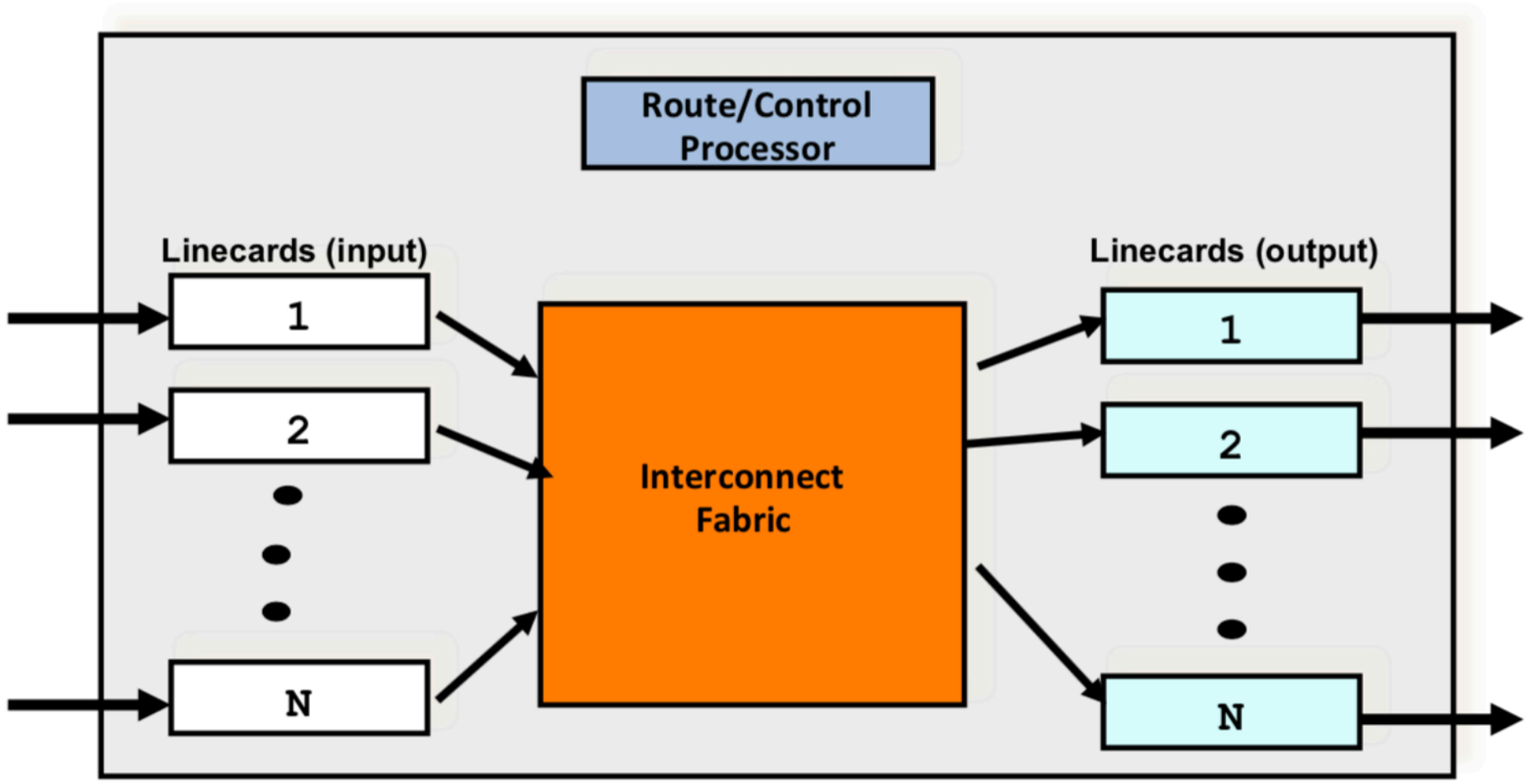
Example: Round Robin Scheduler

- Packets are served from each queue in turn

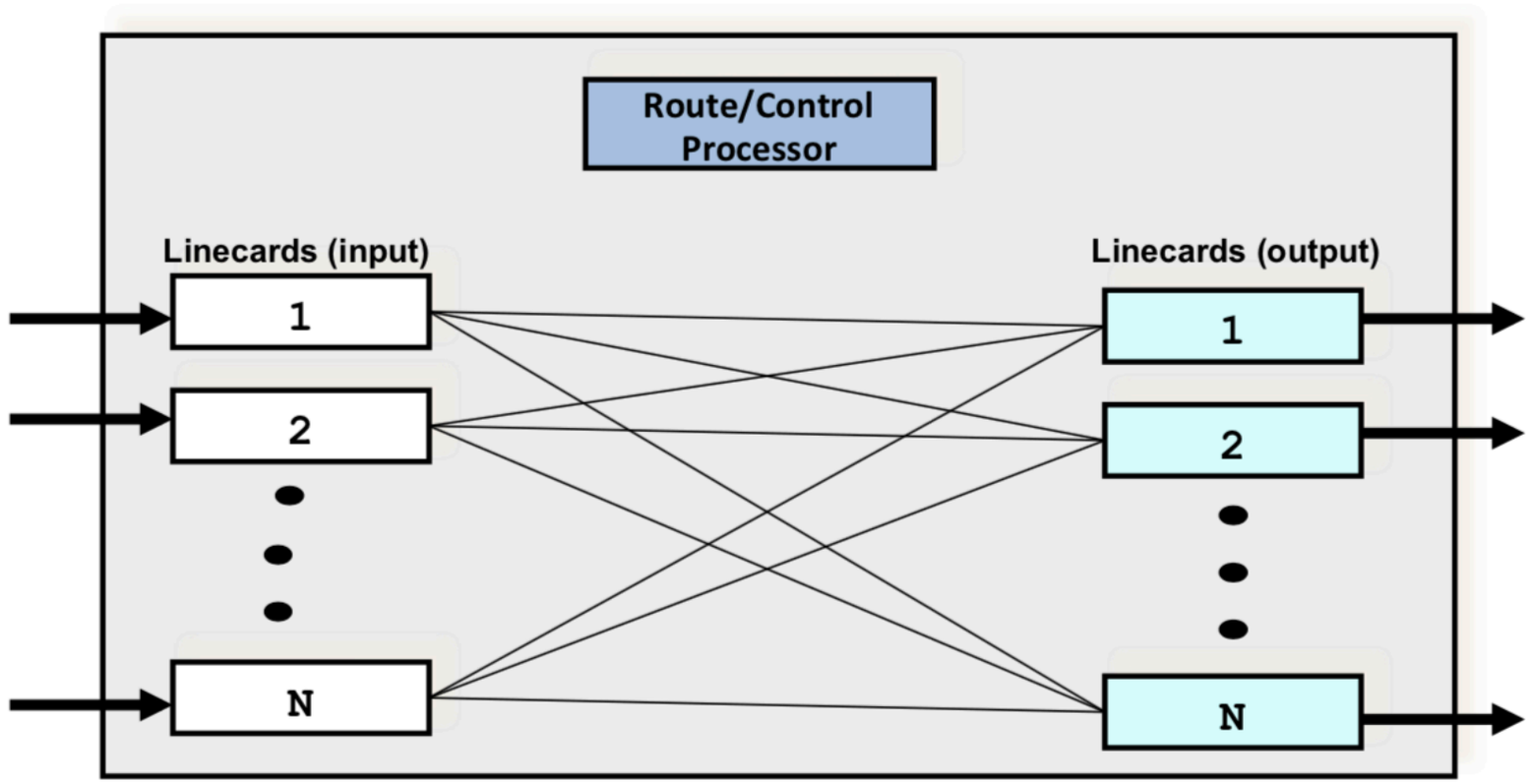


Connecting Input to Output: Switch Fabric

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



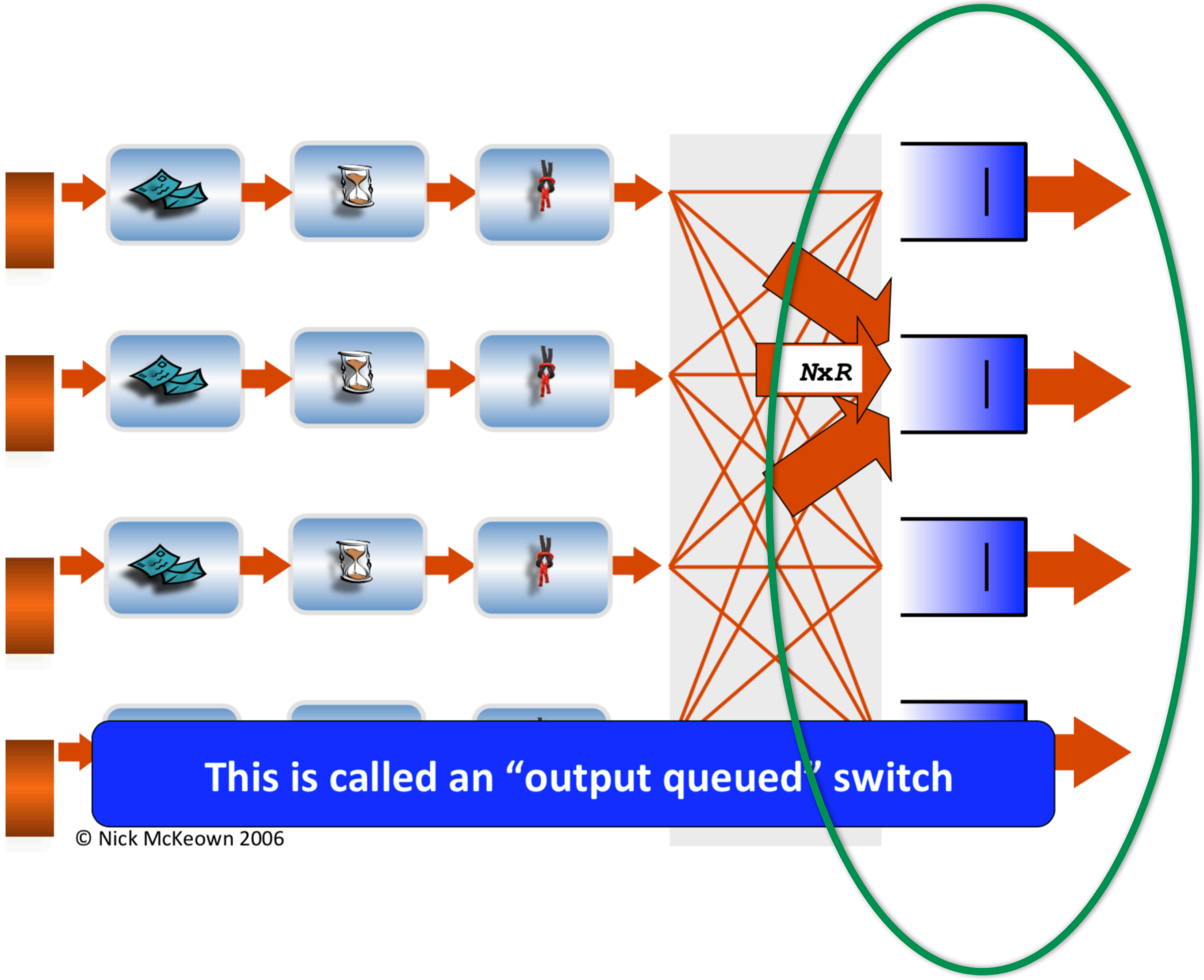
Today's Switch Fabrics: Mini Network!



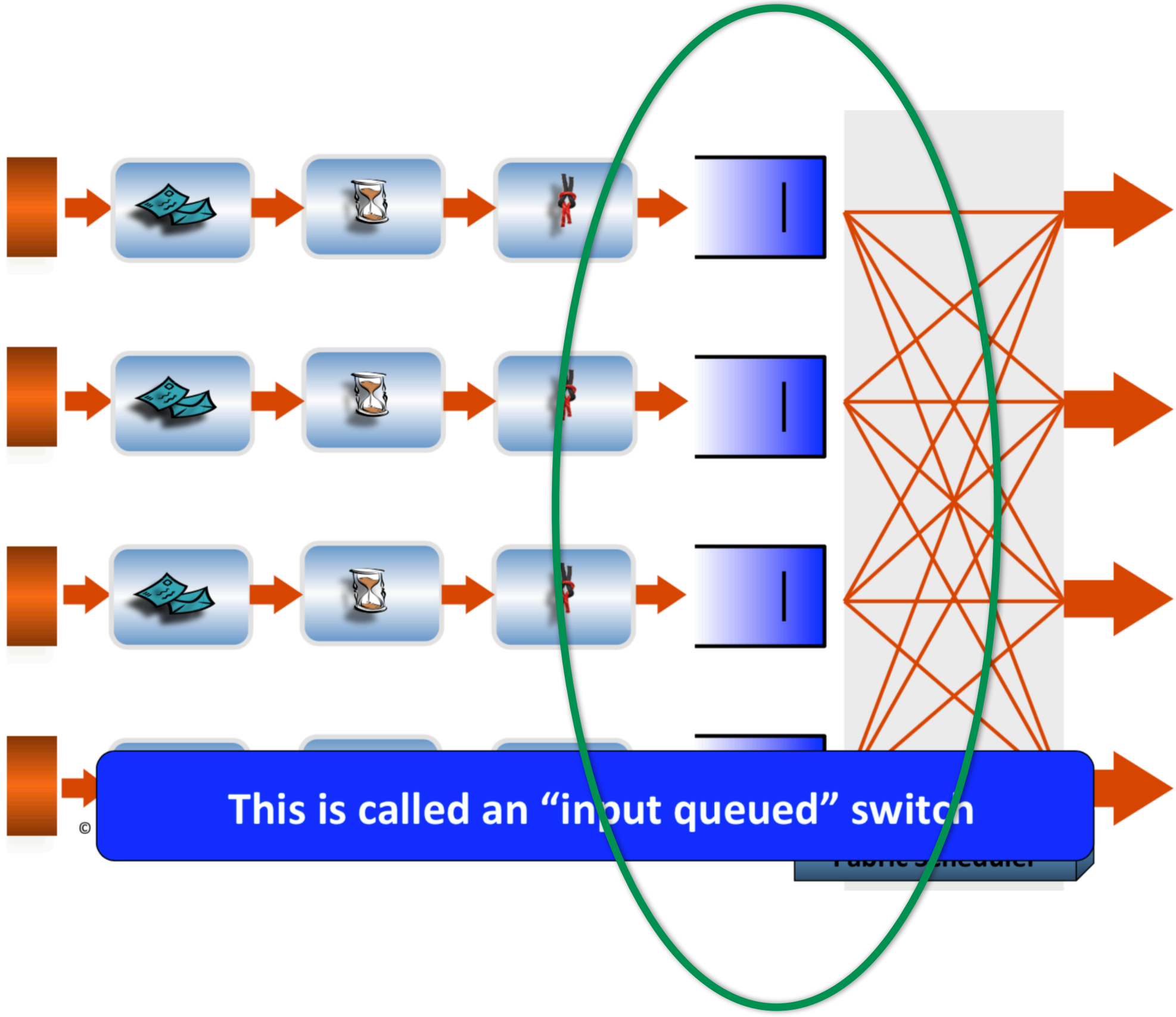
What’s Hard About the Switch Fabric?

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

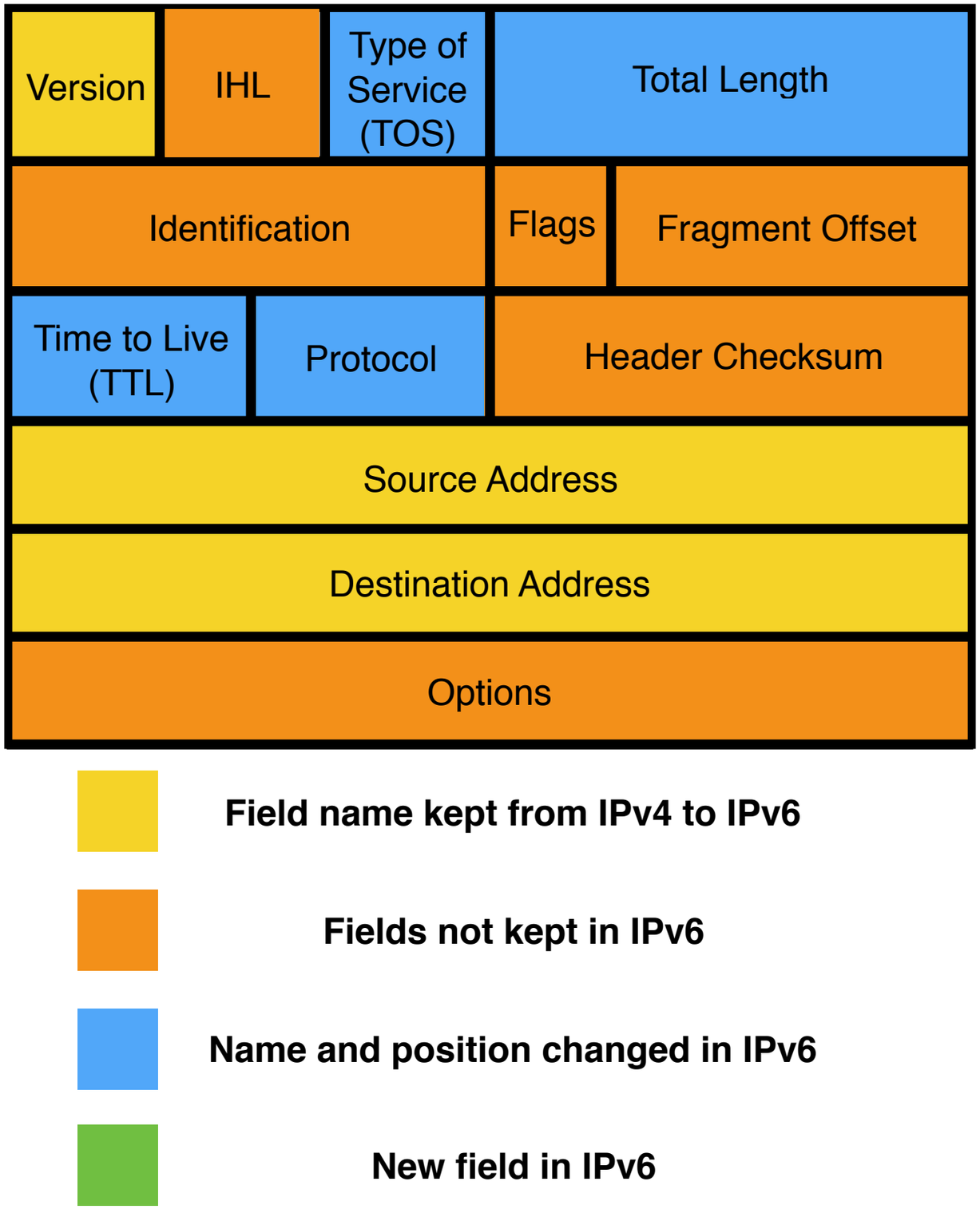
Now you know as much as my PhD students :-)

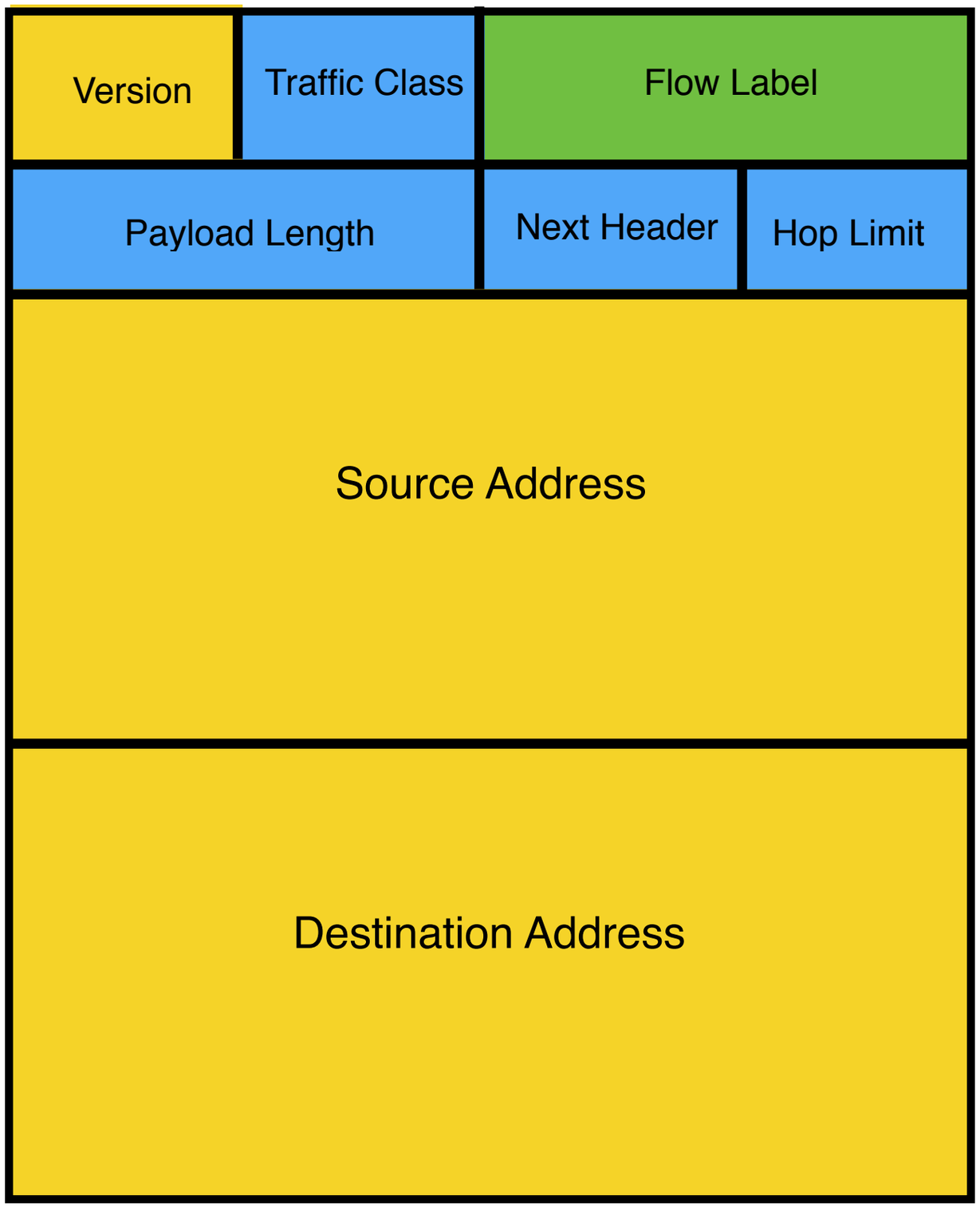
IPv6

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

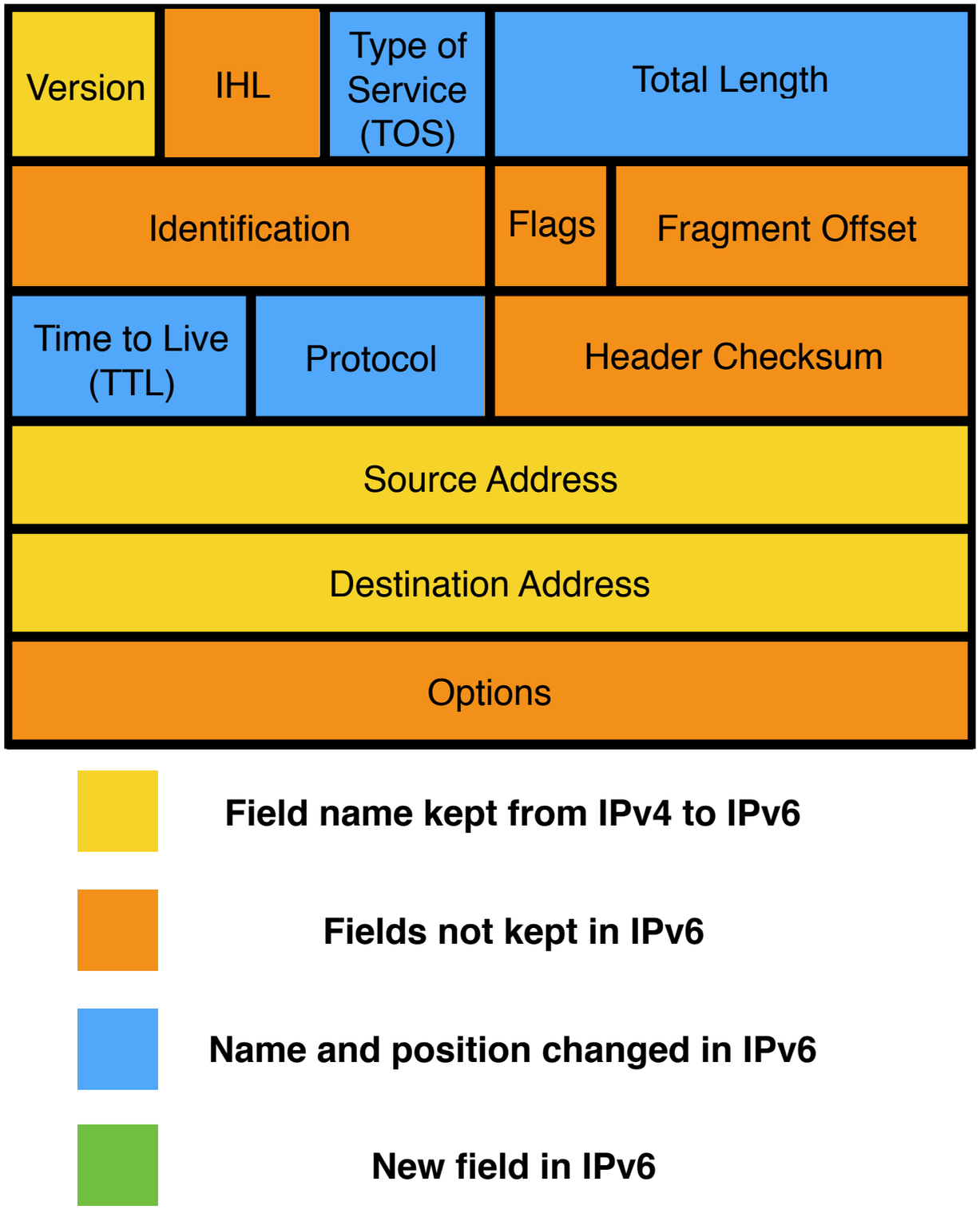


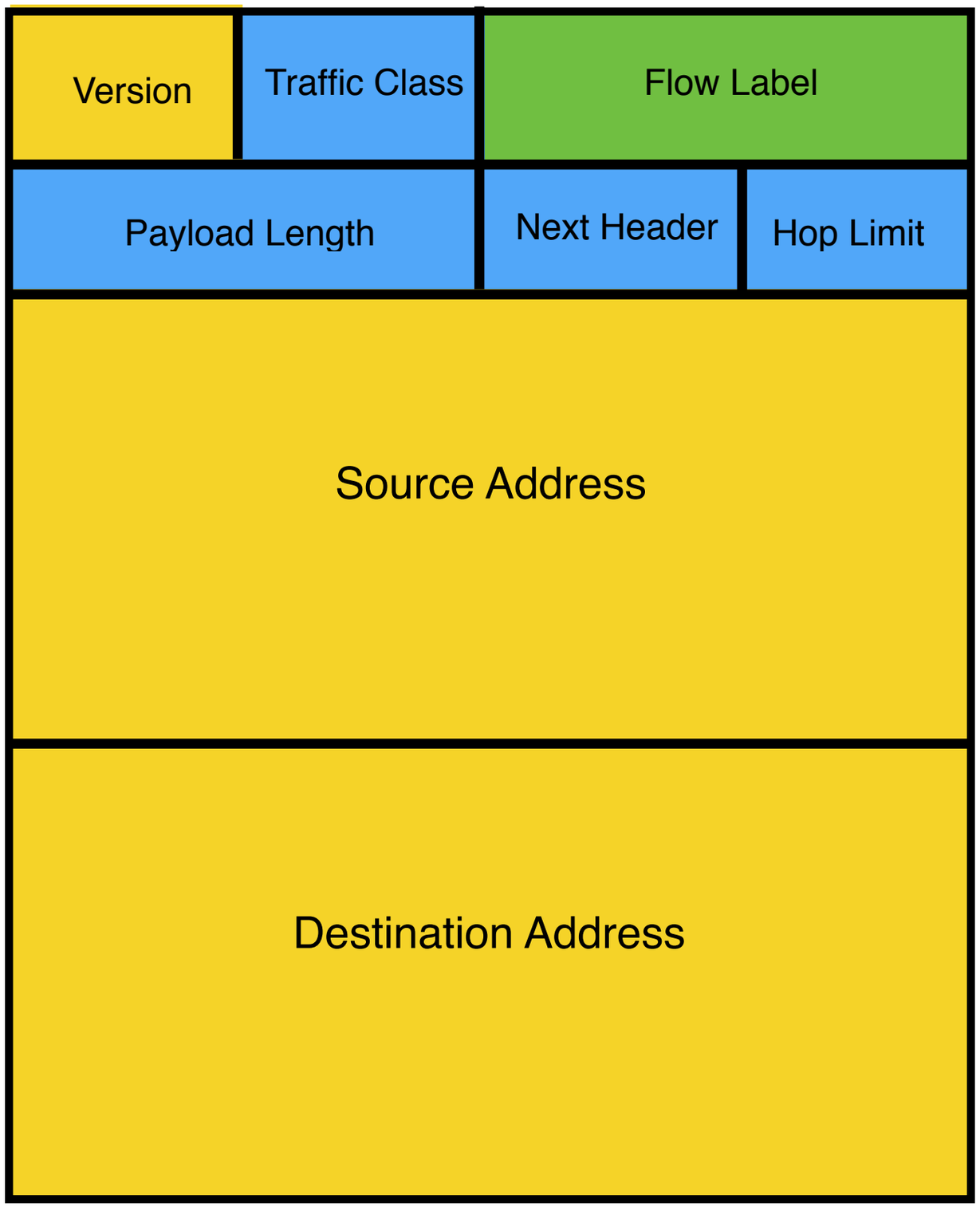


Summary of Changes

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

IPv4 and IPv6 Header Comparison





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