Internet Protocols EBU5403 The Network Layer (Part II) C1

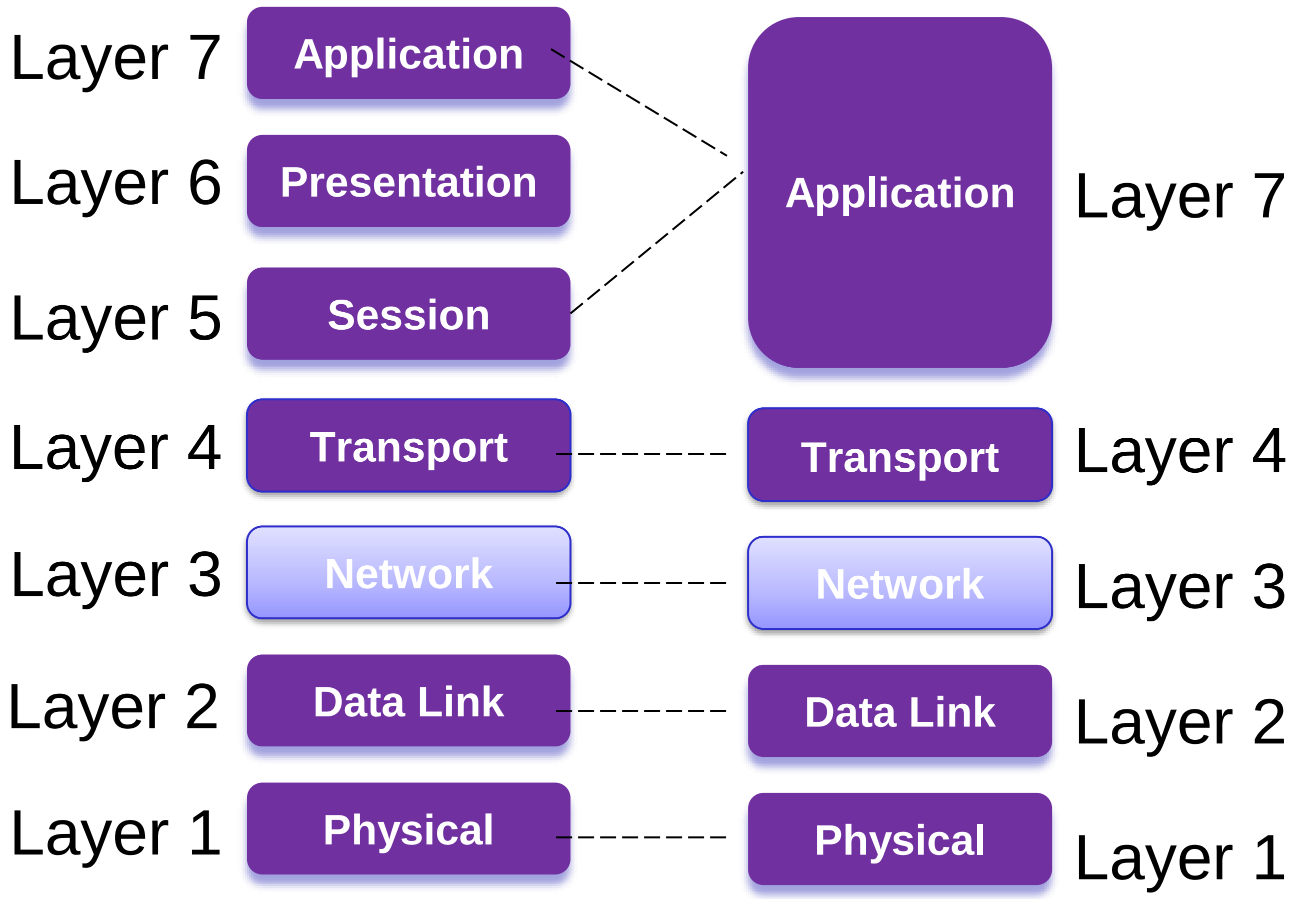
Michael Chai (michael.chai@qmul.ac.uk)
Richard Clegg (r.clegg@qmul.ac.uk)
Cunhua Pan (c.pan@qmul.ac.uk)

	Part 1	Part 2	Part 3	Part 4
Ecommerce + Telecoms 1	Richard Clegg		Cunhua Pan	
Telecoms 2	Michael Chai			

Structure of course

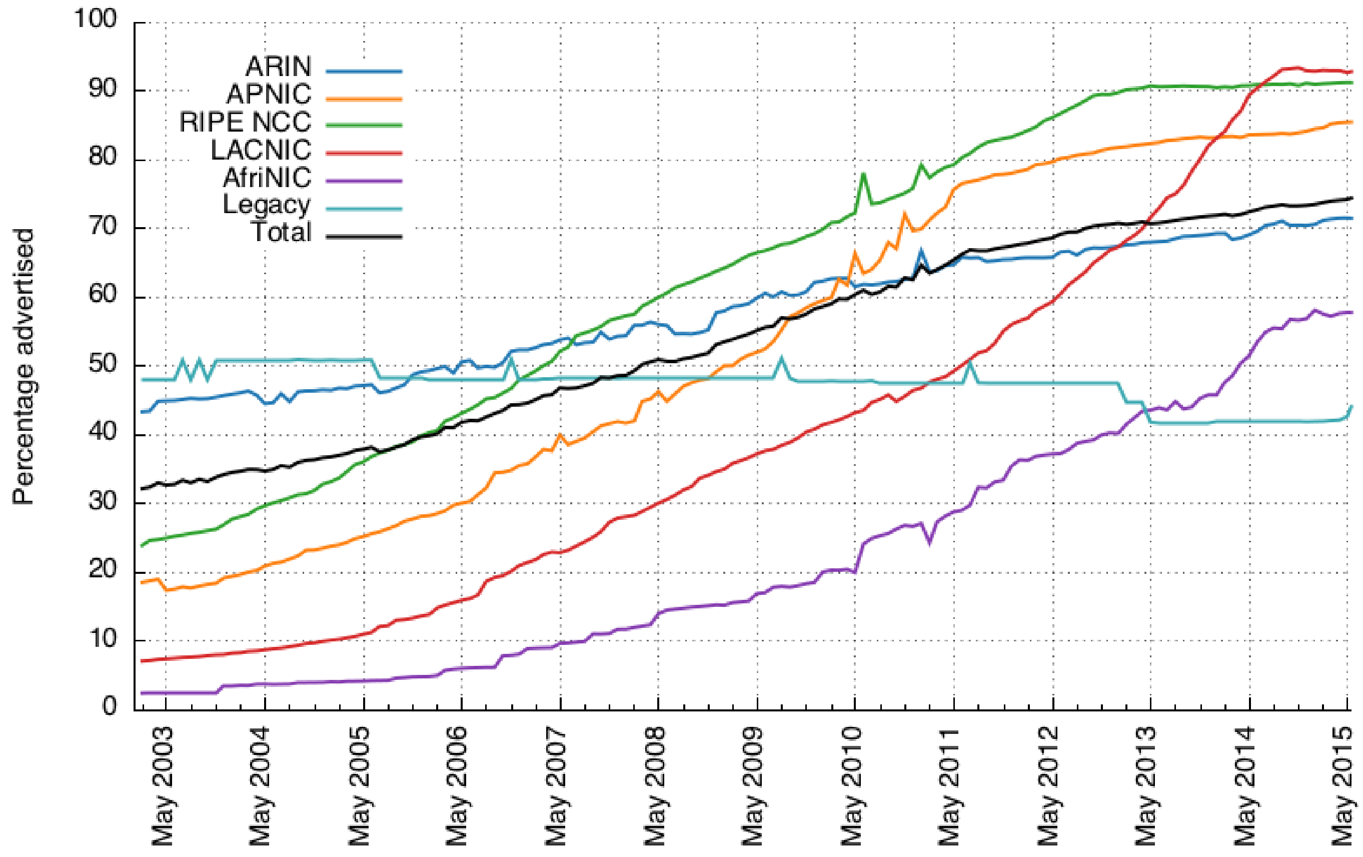
- Part 1
  - Introduction to IP Networks
  - The Transport layer (part 1)
- Part 2
  - The Transport layer (part II)
  - The Network layer (part I)
- Part 3
  - The Network layer (part II)
  - The Data link layer (part I)
  - Router lab
- Part 4
  - The Data link layer (part II)
  - Network management and security
  - Class test

Network Layer



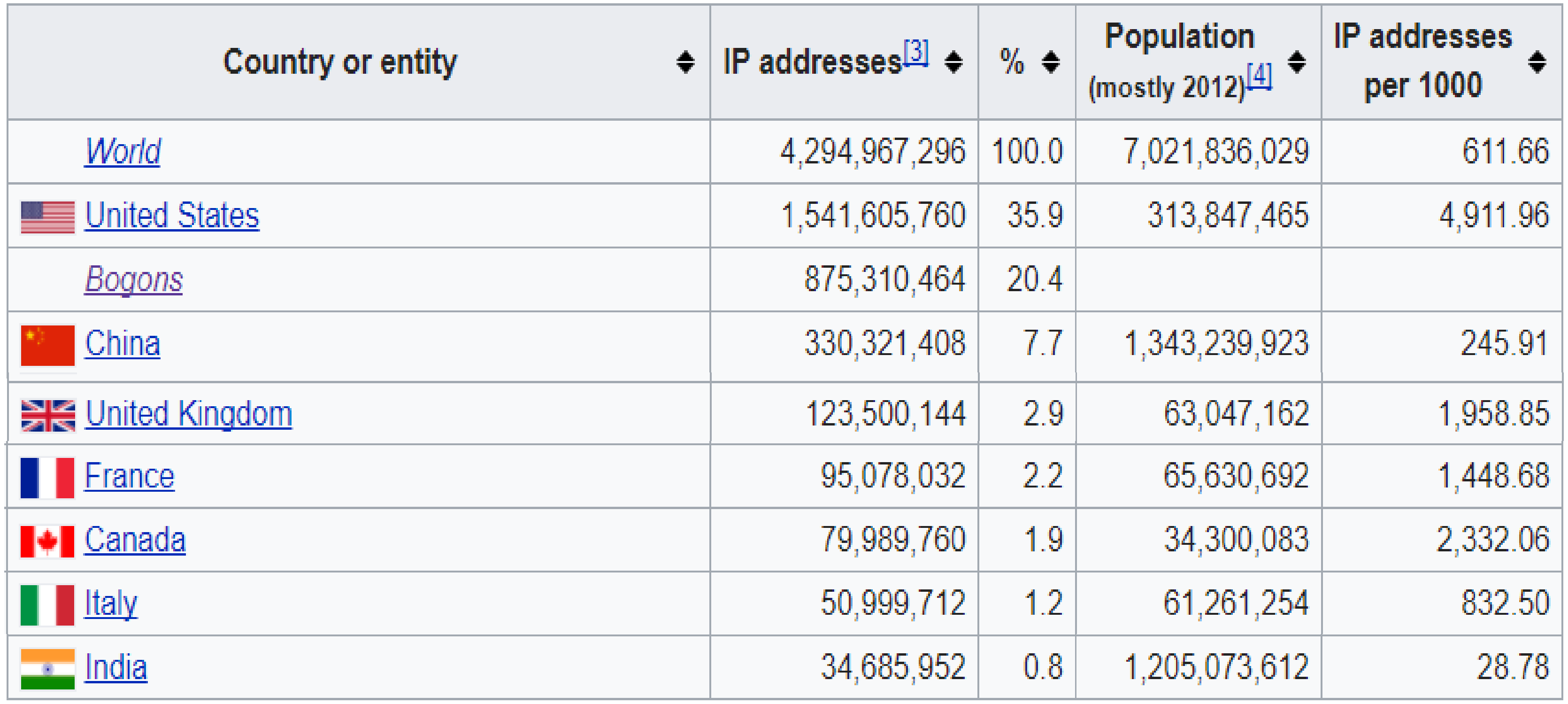
Network Data Plane: outline

IPv4 addresses are running out (RIRs allocate IPv4s to regions)



ARIN – American Registry for Internet Numbers
AfriNIC – African Network Information Centre
APNIC – Asia Pacific (most of Asia, Australia, New Zealand) – China in here
LACNIC – Latin America and Carribean
RIPE NCC – Reseaux IP Europeens (Europe Russian, Middle East and Central Asia

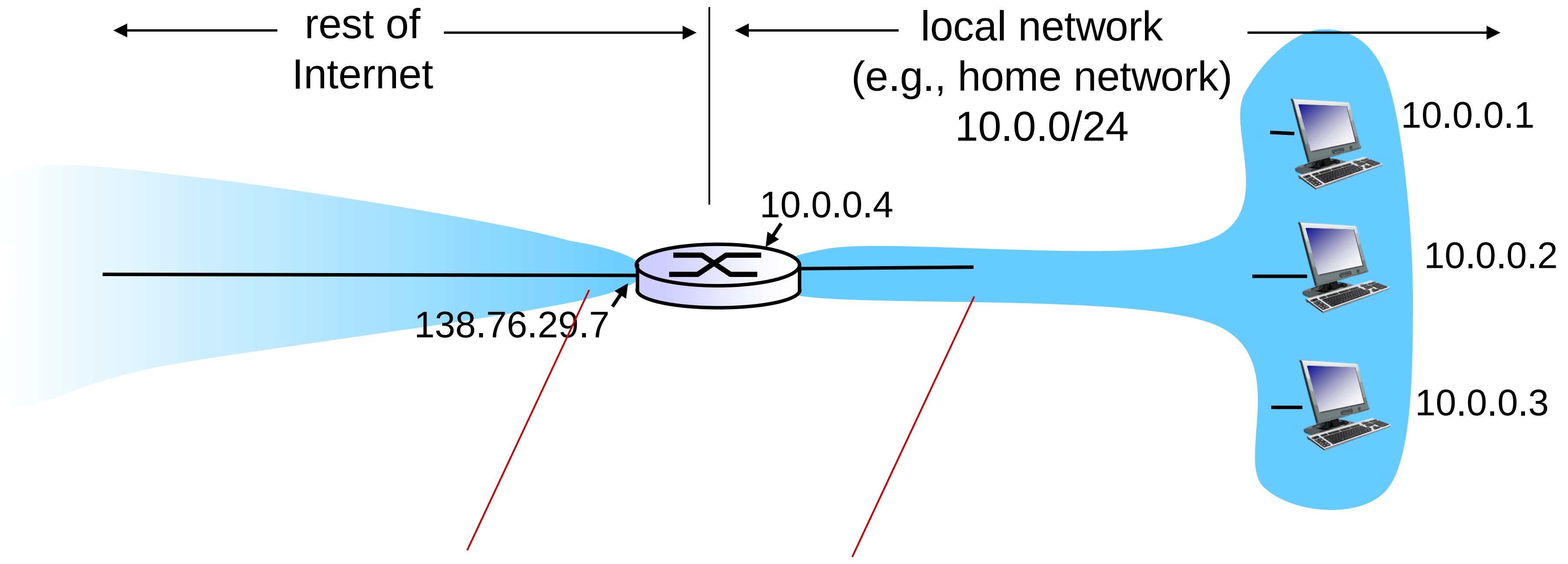
IPv4 addresses are unfairly allocated



Country or entity	IP addresses[3]	%	Population (mostly 2012)[4]	IP addresses per 1000
World	4,294,967,296	100.0	7,021,836,029	611.66
United States	1,541,605,760	35.9	313,847,465	4,911.96
Bogons	875,310,464	20.4		
China	330,321,408	7.7	1,343,239,923	245.91
United Kingdom	123,500,144	2.9	63,047,162	1,958.85
France	95,078,032	2.2	65,630,692	1,448.68
Canada	79,989,760	1.9	34,300,083	2,332.06
Italy	50,999,712	1.2	61,261,254	832.50
India	34,685,952	0.8	1,205,073,612	28.78

https://en.wikipedia.org/wiki/List_of_countries_by_IPv4_address_allocation

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

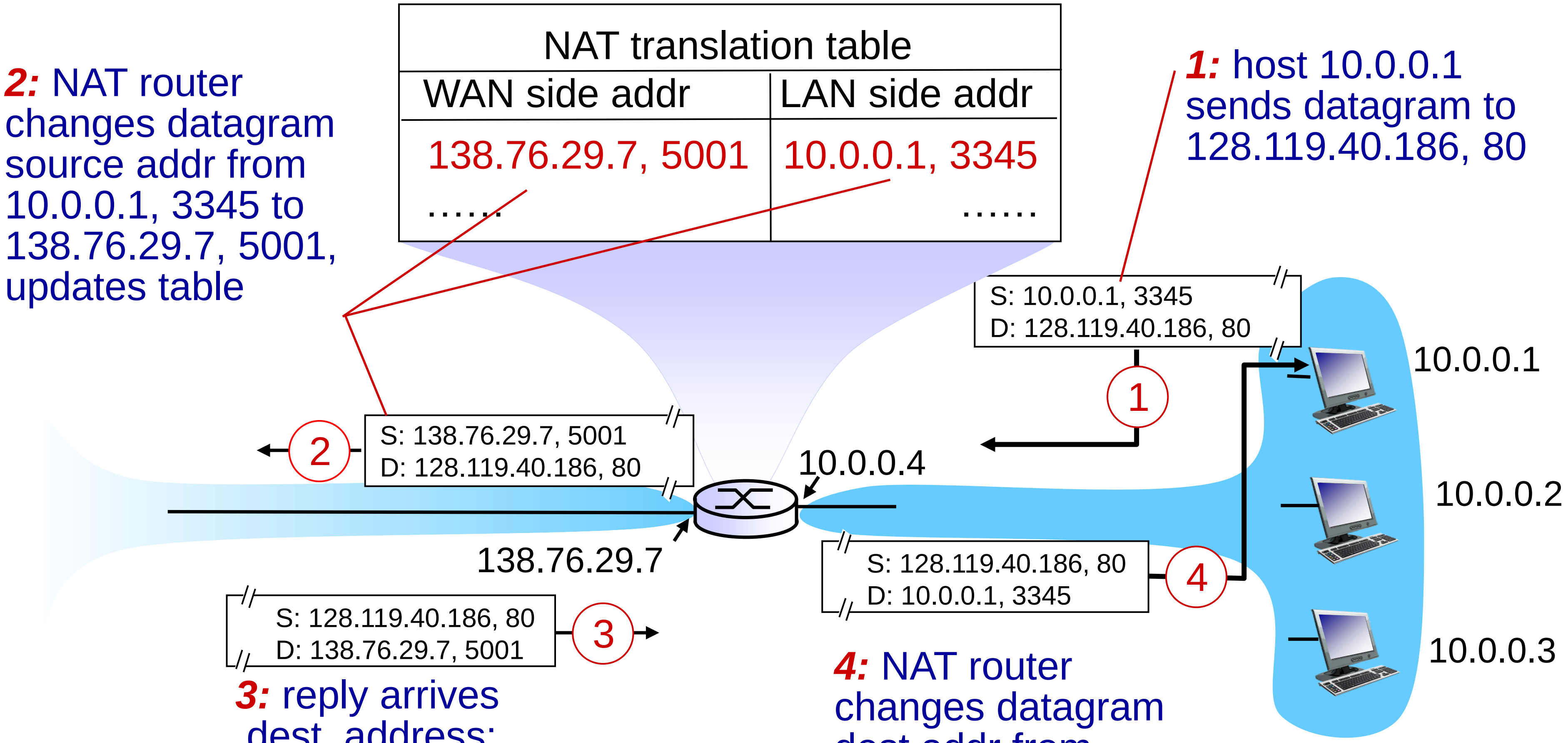
NAT: network address translation

NAT translation table

WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
……	……

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



3: reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument (complexity should be at network "ends" not middle)
    - NAT possibility must be taken into account by app designers, e.g., P2P applications

Network Data Plane: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What' s inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IPv6: motivation

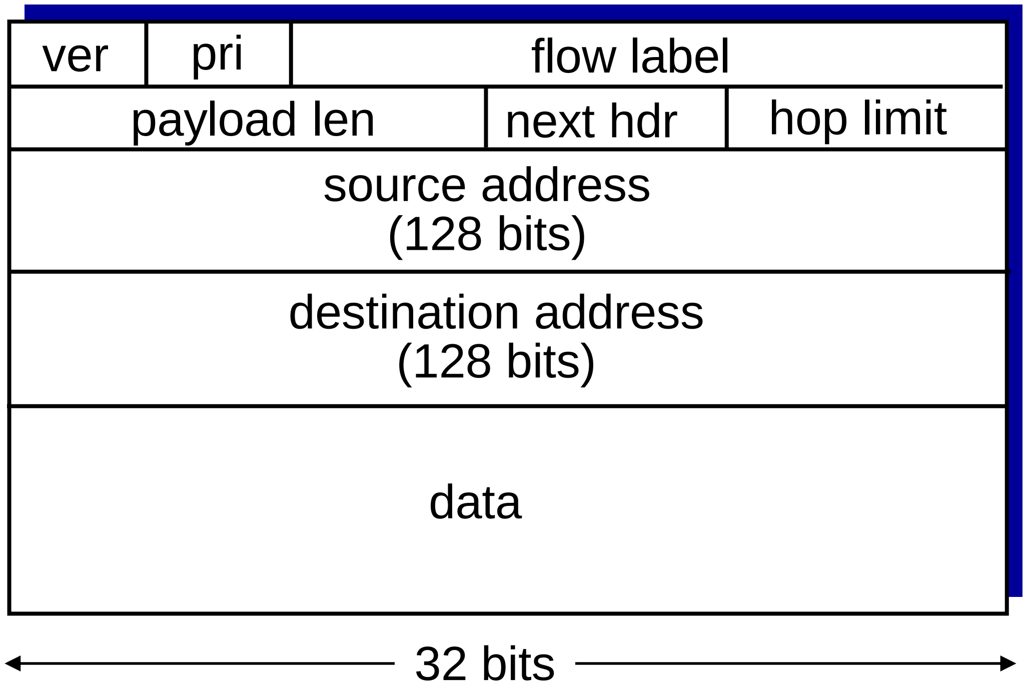
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

IPv6 datagram format

priority: identify priority among datagrams in flow
flow Label: identify datagrams in same "flow."
next header: identify upper layer protocol for data



Other changes from IPv4

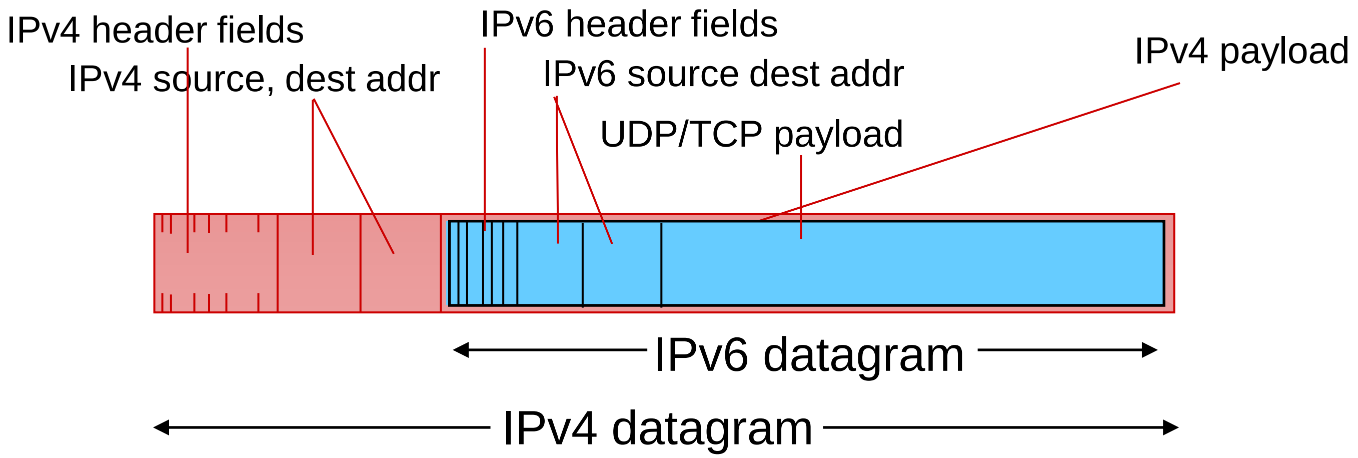
- *checksum:* removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *ICMPv6:* new version of ICMP (see later lecture)
  - additional message types, e.g. "Packet Too Big"
  - multicast group management functions

IPV4 and IPV6:

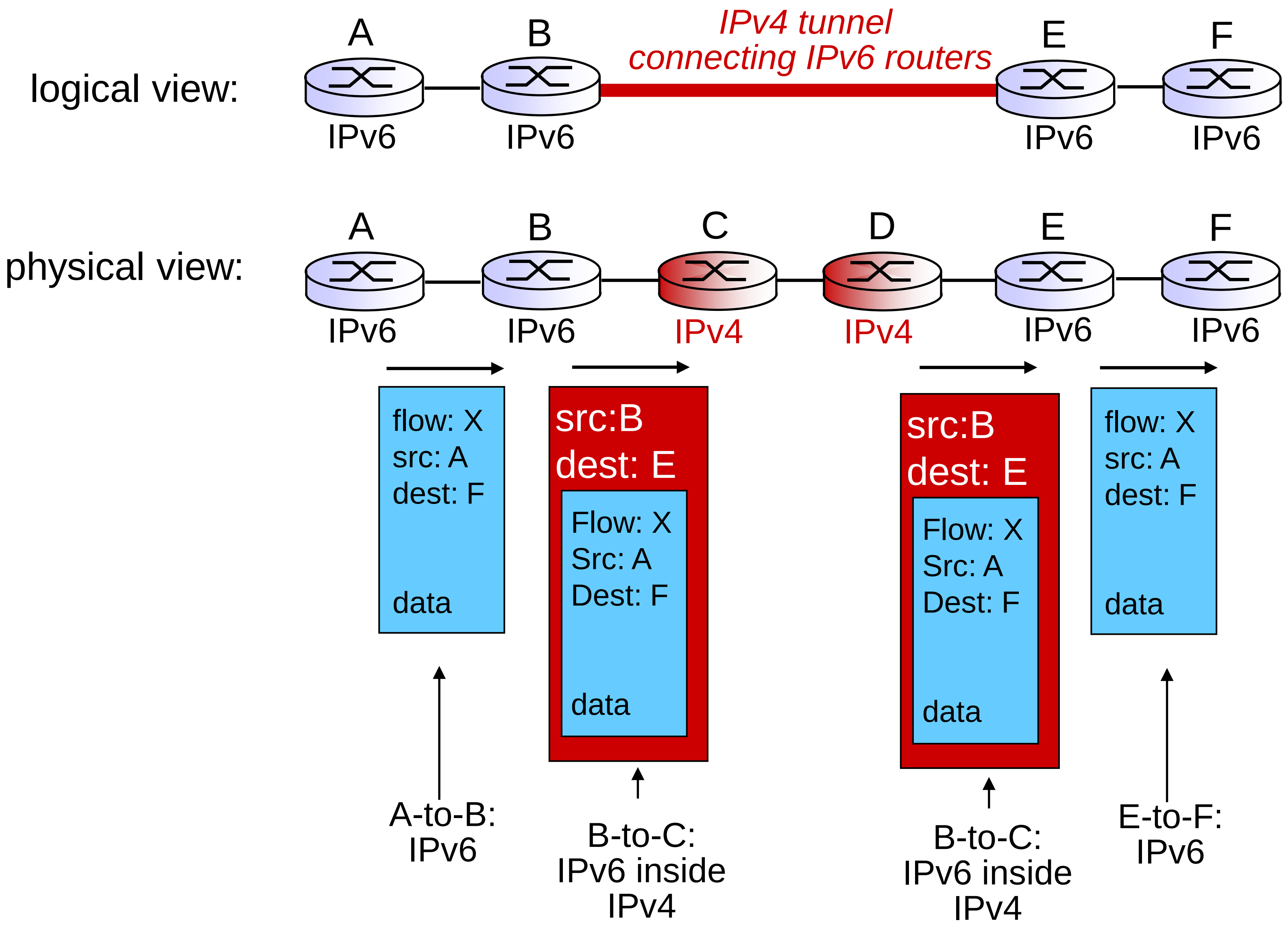
Address bits: 32bits and 128bits

Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling


logical view:
A
B
IPv4 tunnel connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B: IPv6
B-to-C: IPv6 inside IPv4
B-to-C: IPv6 inside IPv4
E-to-F: IPv6

Test your understanding

- How many bits in an IPv6 address?
  a) 32 bits
  b) 64 bits
  c) 128 bits
- Which of the following not in the IPv6 header?
  a) Checksum
  b) Next Header
  c) Hop Limit

Test your understanding

- How many bits in an IPv6 address?
  a) 32 bits
  b) 64 bits
  c) 128 bits
- Which of the following not in the IPv6 header?
  a) Checksum
  b) Next Header
  c) Hop Limit

Network Data Plane: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

What is Software Defined Networking?

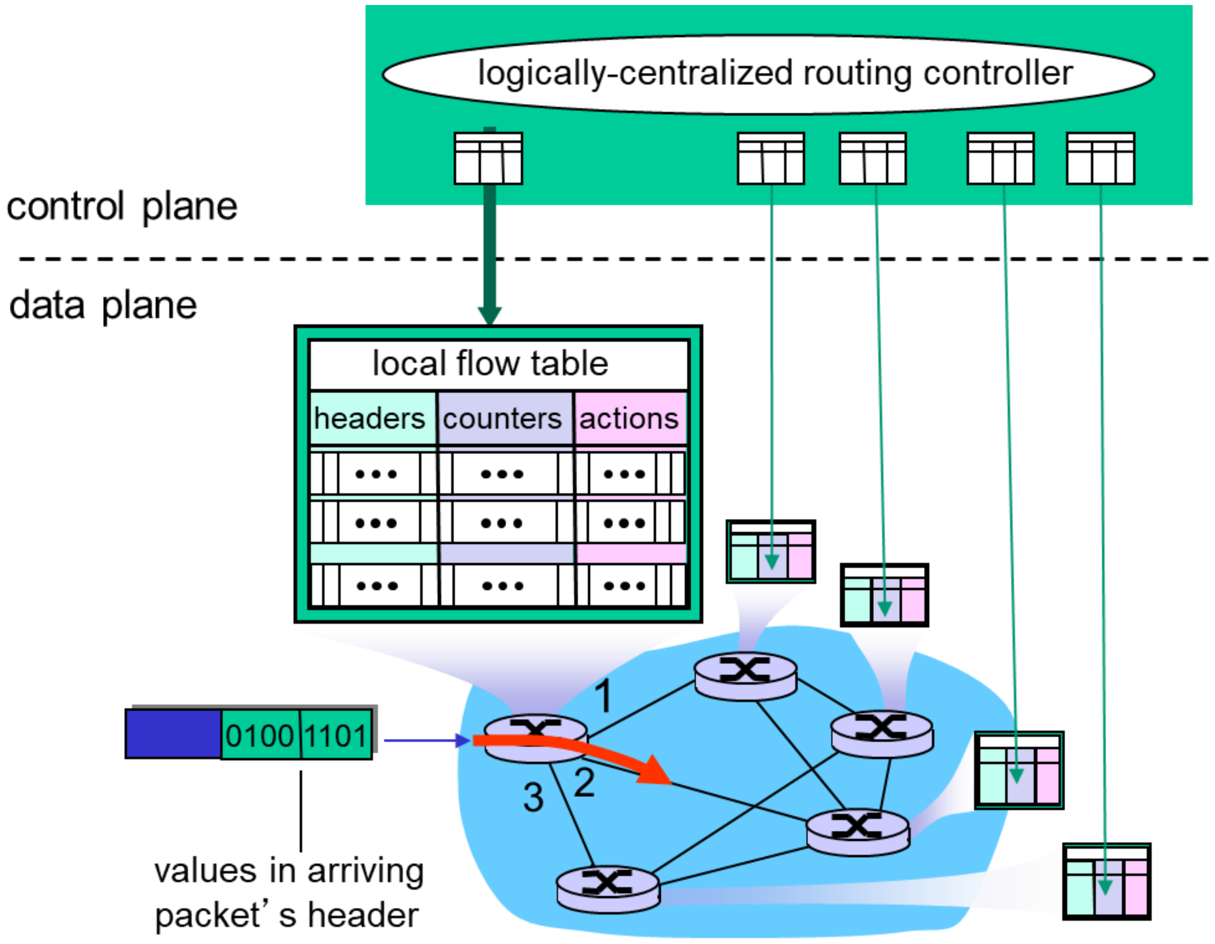
- Traditional routing methods:
  - Data Plane is longest-prefix match forwarding using a forwarding table.
  - The control plane calculates the forwarding table for each router (we will see how later).
  - The forwarding table can forward packets by their IP address and nothing else.
- SDN allows more flexibility in the control and data plane.
  - Program your own control algorithms in language you know (Java, python etc).
  - Forwarding can use any part of the packet header.

How does SDN work?

- Data plane:
  - A set of "match-action" rules send by a controller can do many different things to packets. (Forward them, change data etc).
  - Much more flexible (route video packets differently? route private data separately? Drop suspicious data!)
- Control plane (later lectures):
  - Not a distributed system, but a single centralised control point.
  - Programmable, not fixed – you can program the controller in a high-level language that you know.
  - Create your own algorithms and test them on the network without spending a million dollars to create a new hardware router.

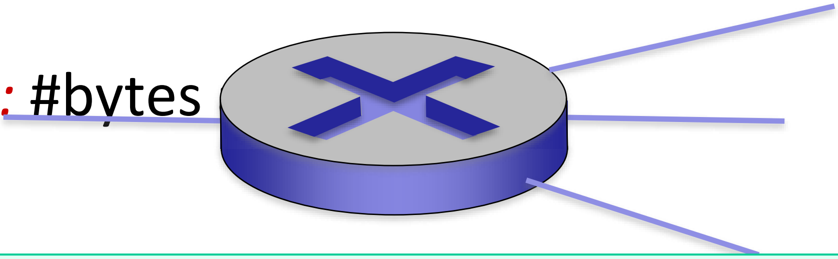
Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow data plane abstraction

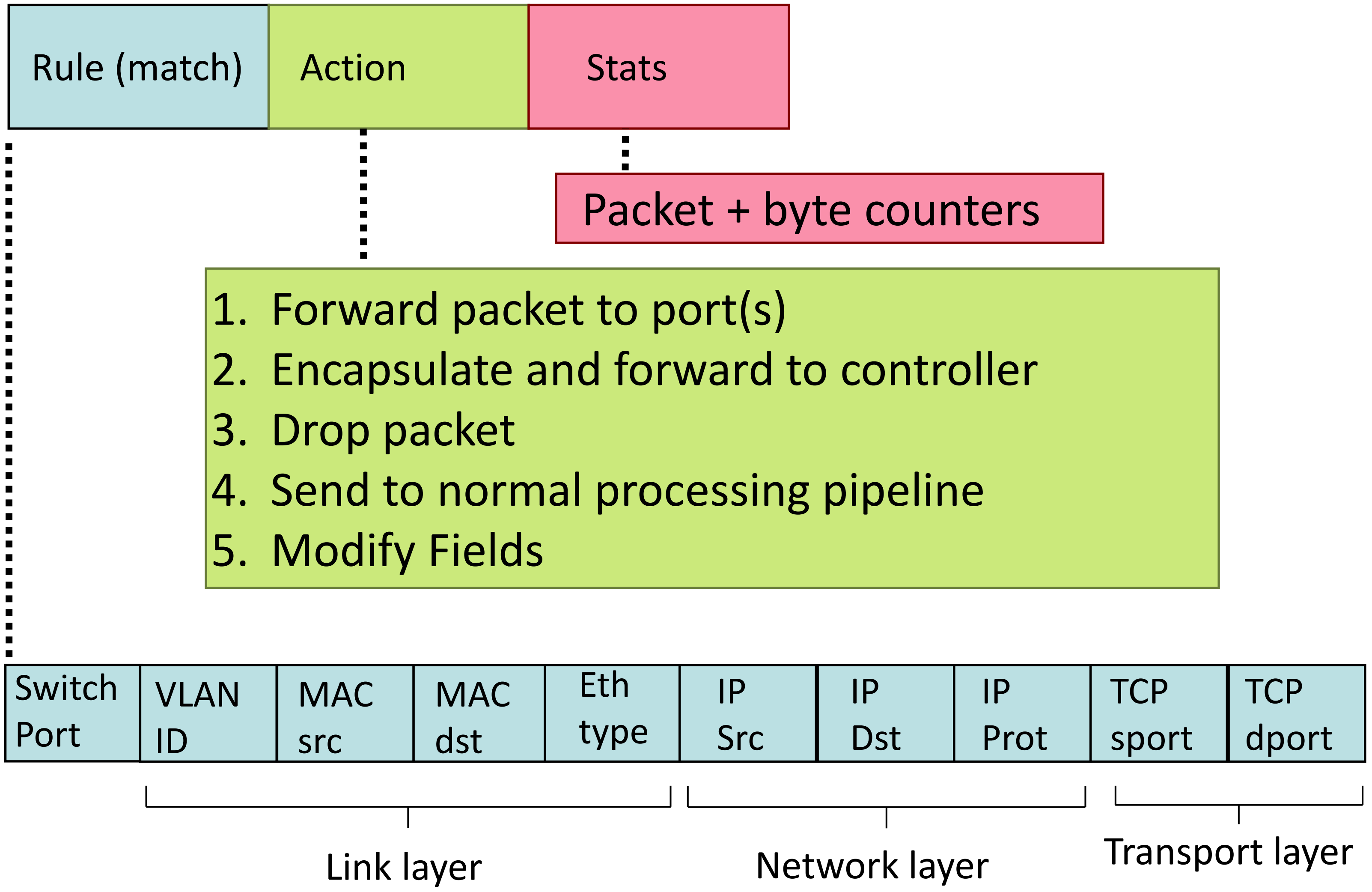
- OpenFlow is a specific SDN protocol
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate (tell difference between) overlapping patterns
  - *Counters:* #bytes



* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

OpenFlow: Flow Table Entries



1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

OpenFlow abstraction

- *match+action:* different kinds of devices become one

- Router
  - *match:* longest destination IP prefix
  - *action:* forward out a link
- Switch
  - *match:* destination MAC address
  - *action:* forward or flood
- Firewall
  - *match*: IP addresses and TCP/UDP port numbers
  - *action:* permit or deny
- NAT
  - *match:* IP address and port
  - *action:* rewrite address and port

Network Layer Data Plane: *done!*

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

4.4 Generalized Forward and SDN
- match plus action
- OpenFlow example

Test your understanding

Which of the following information can be used for “match-action” in OpenFlow?

a) IP addresses
b) MAC addresses
c) Port addresses
d) Any of the above

Network Control Plane: outline

Network-layer functions

Recall: two network-layer functions:

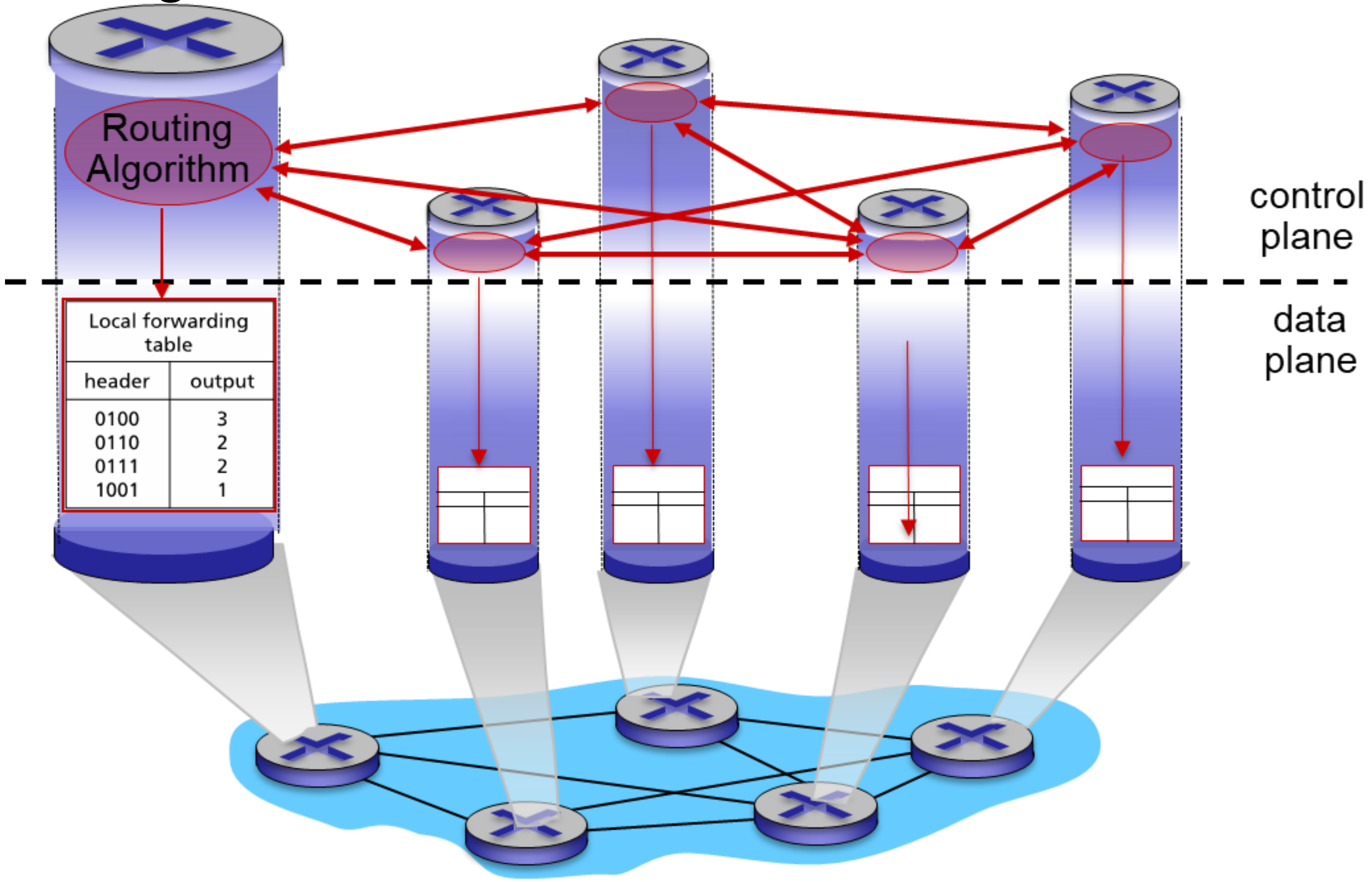
- *forwarding:* move packets from router's input to appropriate router output — *data plane*
- *routing:* determine route taken by packets from source to destination — *control plane*

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

Per-router control plane (revision)

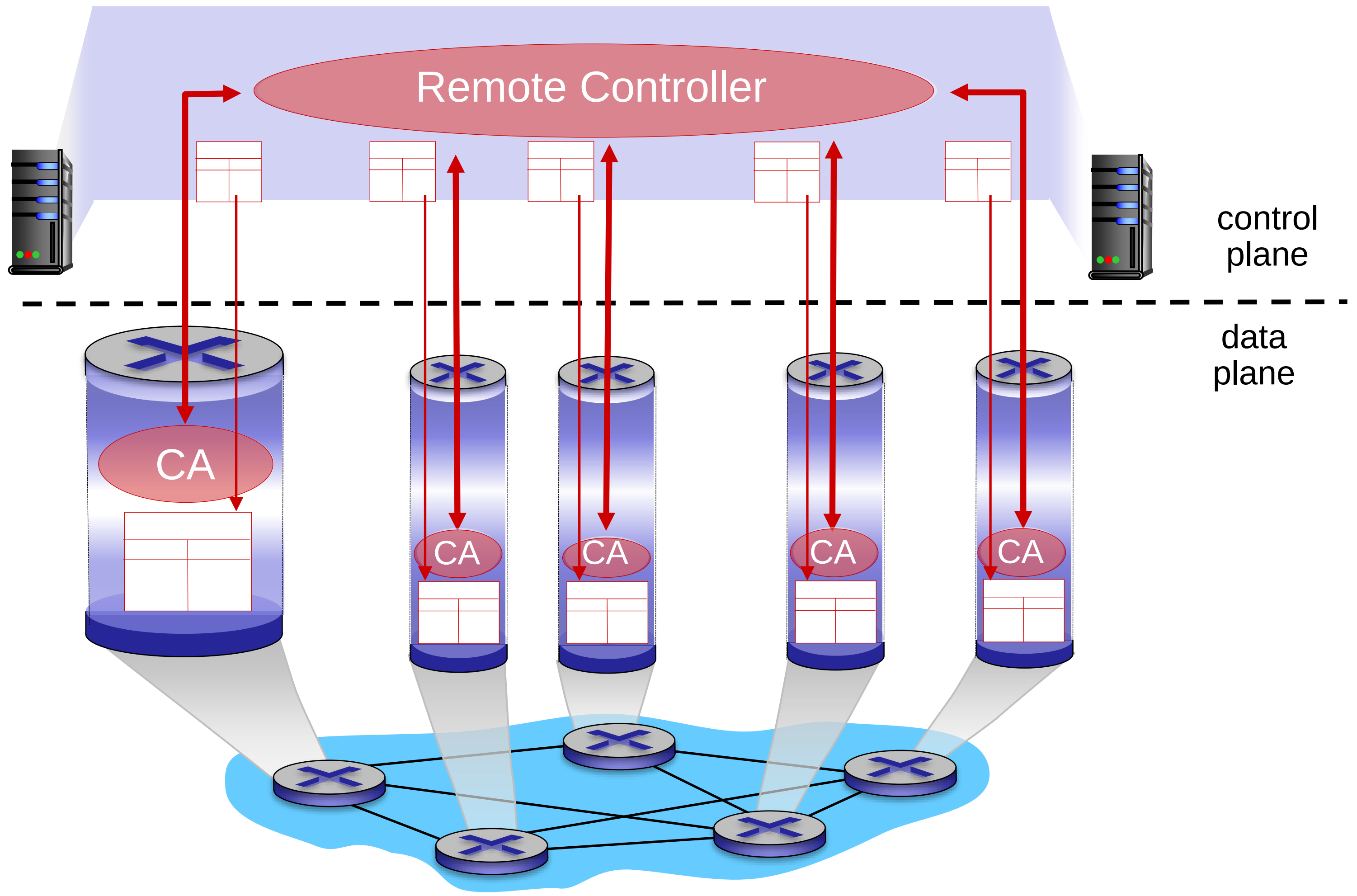
Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Local forwarding table	
header	output
0100	3
0110	2
0111	2
1001	1

Logically centralized control plane (revision)

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



What have we learned?

- NAT (Network Address Translation) is a way of making one IPv4 address usable by many hosts
  - NAT is really common (especially in China)
  - NAT has its problems (getting data back into computer)
- IPv6 improves upon IPv4 in very simple ways.
  - IPv6 deployment slowly growing but not "there" yet.
- Software Defined Networks (for example OpenFlow) is a new technology for forwarding
  - Becoming very popular, extremely powerful
  - We will see later this week how SDN and OpenFlow are used in the control plane.