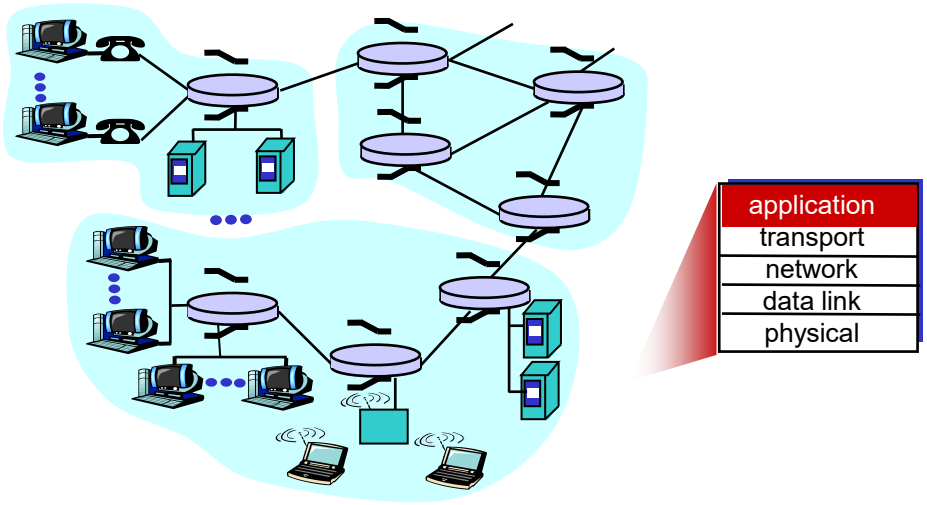
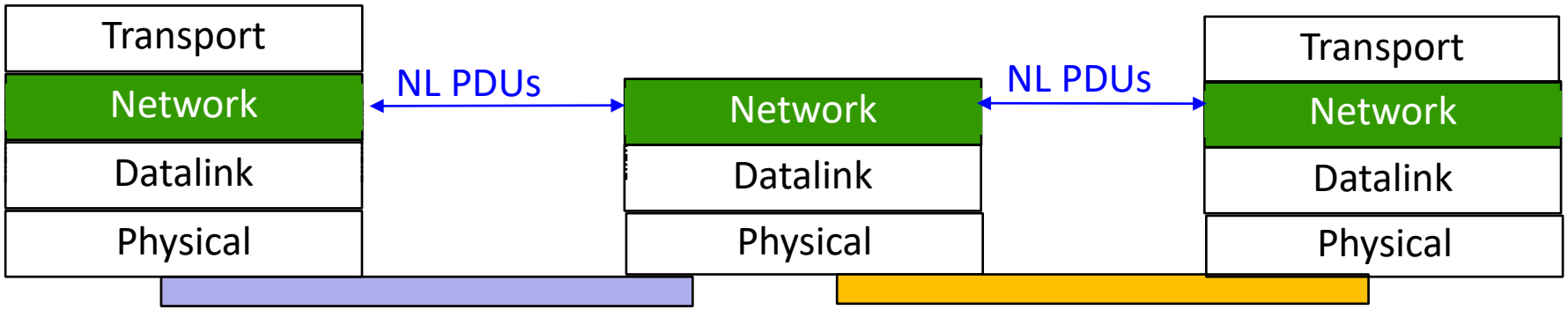
# CS 4390
# Computer Networks




## *Network Layer – Introduction*

# Network Layer

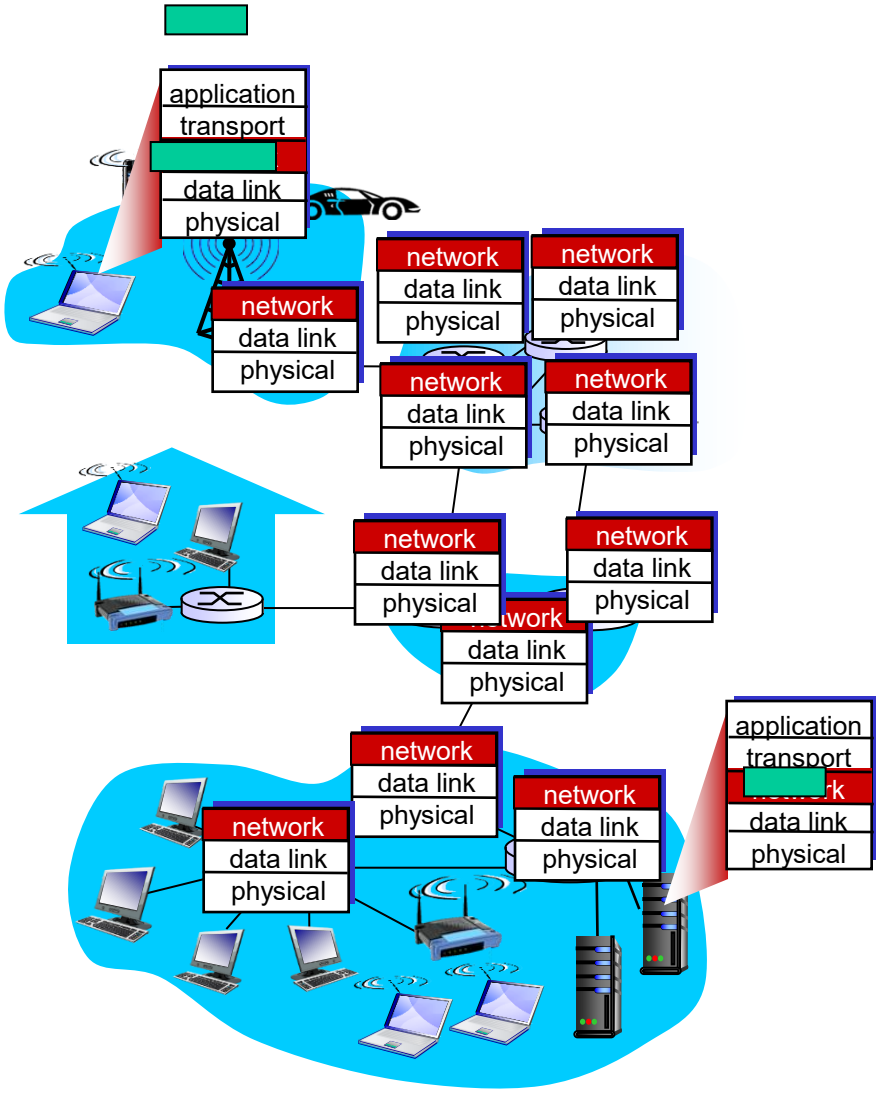



- Goal
  - Allow **network layer PDUs** to be forwarded from any *source host* to any *destination host* through *heterogeneous networks and intermediary nodes* (e.g. routers), using services provided by the datalink layer
- Services
  - Unreliable connectionless service
  - Reliable connection-oriented service

*Credits: adapted from slide by Prof. O. Bonadventure*

# Network Layer – cont'd

- transport segments from a sending node to a receiving node (host)
  - source to destination
- on source host encapsulates segments into network layer PDUs (e.g. datagrams)
- on destination host, delivers segments to transport layer
- network layer protocols run in *every* host, router
- router examines header fields in all datagrams passing through it



# Network Layer – Basic Requirements

- Each host or intermediate node (e.g. router) must be identified by a *network layer address* which is independent from its datalink layer address
- Network layer forwards segments from source host to destination host through multiple networks
- Network layer services must be completely independent from the services provided by the datalink layer
- Network layer user should not need to know anything about the internal structure of the network layer to be able to send or receive segments

*Credits: adapted from slide by Prof. O. Bonadventure*

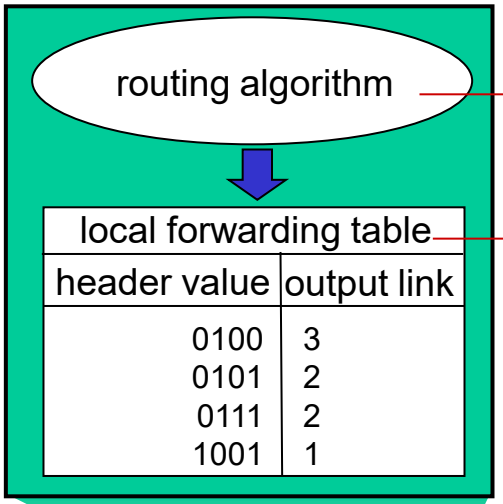
# Key Network-layer Functions

- ***routing:*** determine route taken by packets from source to dest.
  - *routing algorithms*
- ***forwarding:*** move packets from router's input to appropriate router output

*analogy:*

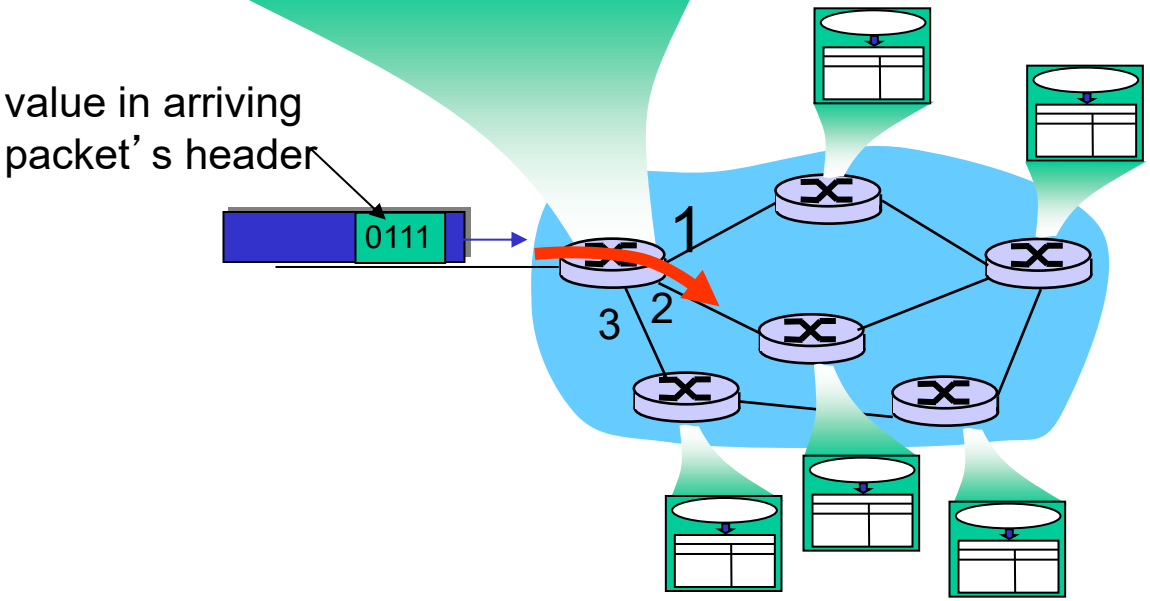
- *routing:* process of planning trip from source to dest
- *forwarding:* process of getting through single interchange

# Routing v.s. Forwarding




***routing algorithm*** determines *end-end-path* through network

***forwarding table*** determines *local forwarding* at this router



# Network Connection Setup

- 3rd important function in *some* network architectures:
  - ATM, frame relay, X.25
- before data can be exchanged, two end hosts *and* intervening routers establish *virtual connection*
  - routers get involved
- network vs transport layer connection service:
  - *network:* **between two hosts** (may also involve intervening routers in case of VCs)
  - *transport:* **between two processes**

# Network Service Model

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

*example services for individual datagrams:*

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

*example services for a flow of datagrams:*

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

# Network Layer Service Models:

| Network Architecture | Service Model | Guarantees ? | | | | Congestion feedback |
|---|---|---|---|---|---|---|
| | | Bandwidth | Loss | Order | Timing | |
| Internet | best effort | none | no | no | no | no (inferred via loss) |
| ATM | CBR | constant rate | yes | yes | yes | no congestion |
| ATM | VBR | guaranteed rate | yes | yes | yes | no congestion |
| ATM | ABR | guaranteed minimum | no | yes | no | yes |
| ATM | UBR | none | no | yes | no | no |

# Connection, Connection-less Service

- *datagram* network provides network-layer *connectionless* service
- *virtual-circuit* network provides network-layer *connection* service
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host
  - *no choice:* network provides one only
  - *implementation:* in network core

# Virtual Circuits

"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

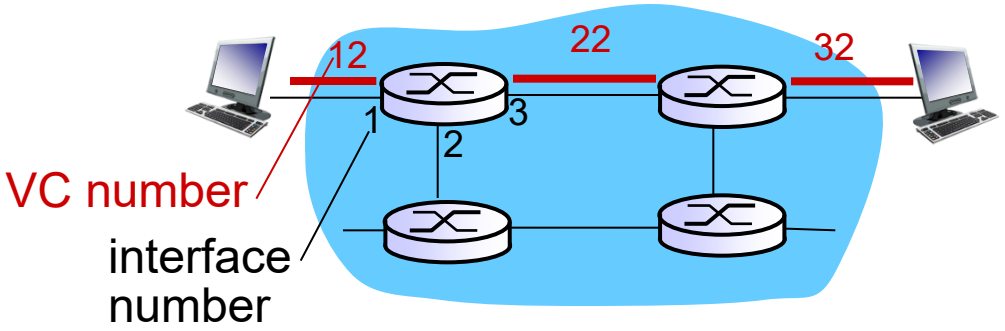
# VC Implementation

*a VC consists of:*

1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *entries in forwarding tables* in routers along path

- packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - new VC number comes from forwarding table

# VC Forwarding Table



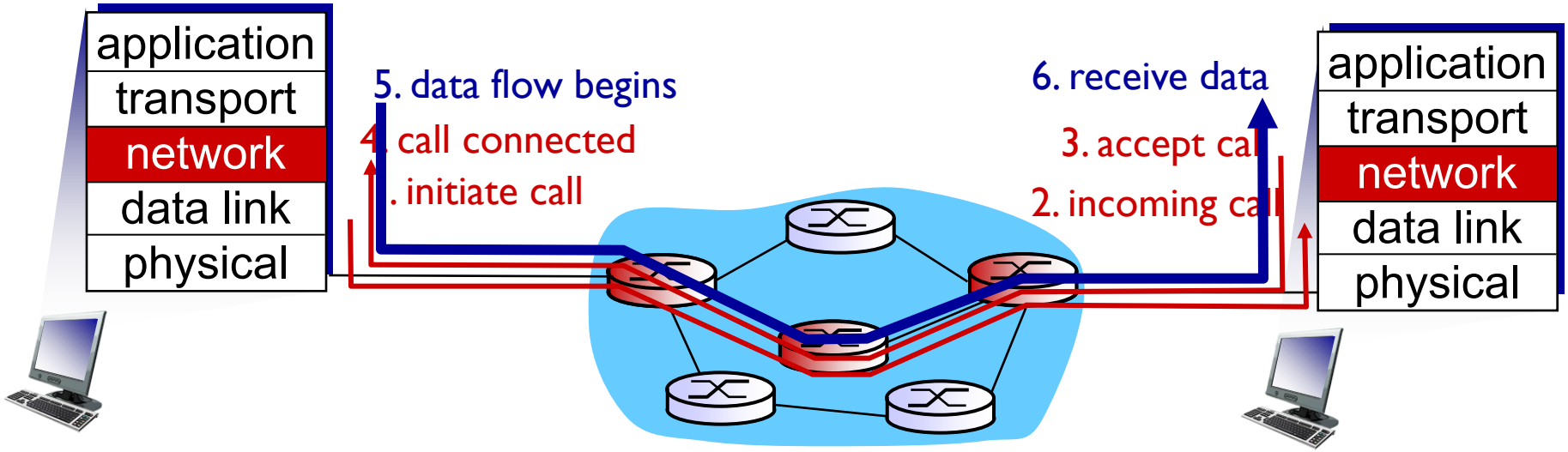

*forwarding table in northwest router:*

| Incoming interface | Incoming VC # | Outgoing interface | Outgoing VC # |
|---|---|---|---|
| 1 | 12 | 3 | 22 |
| 2 | 63 | 1 | 18 |
| 3 | 7 | 2 | 17 |
| 1 | 97 | 3 | 87 |
| … | … | … | … |

*VC routers maintain connection state information!*

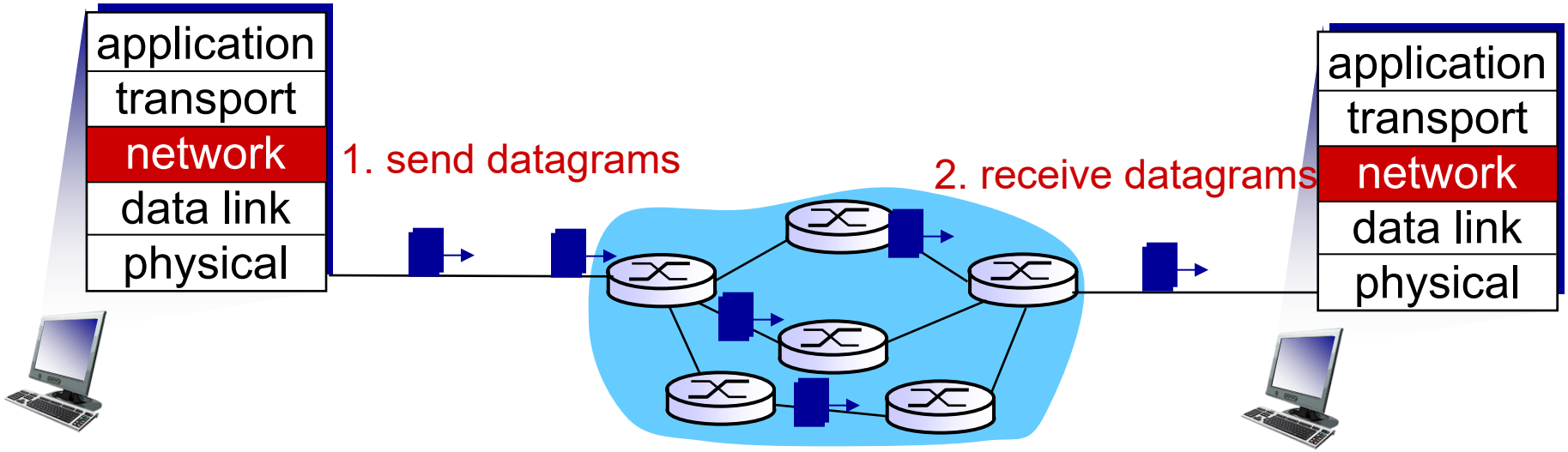
# Virtual Circuits: Signaling Protocols

- used to setup, maintain  teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet

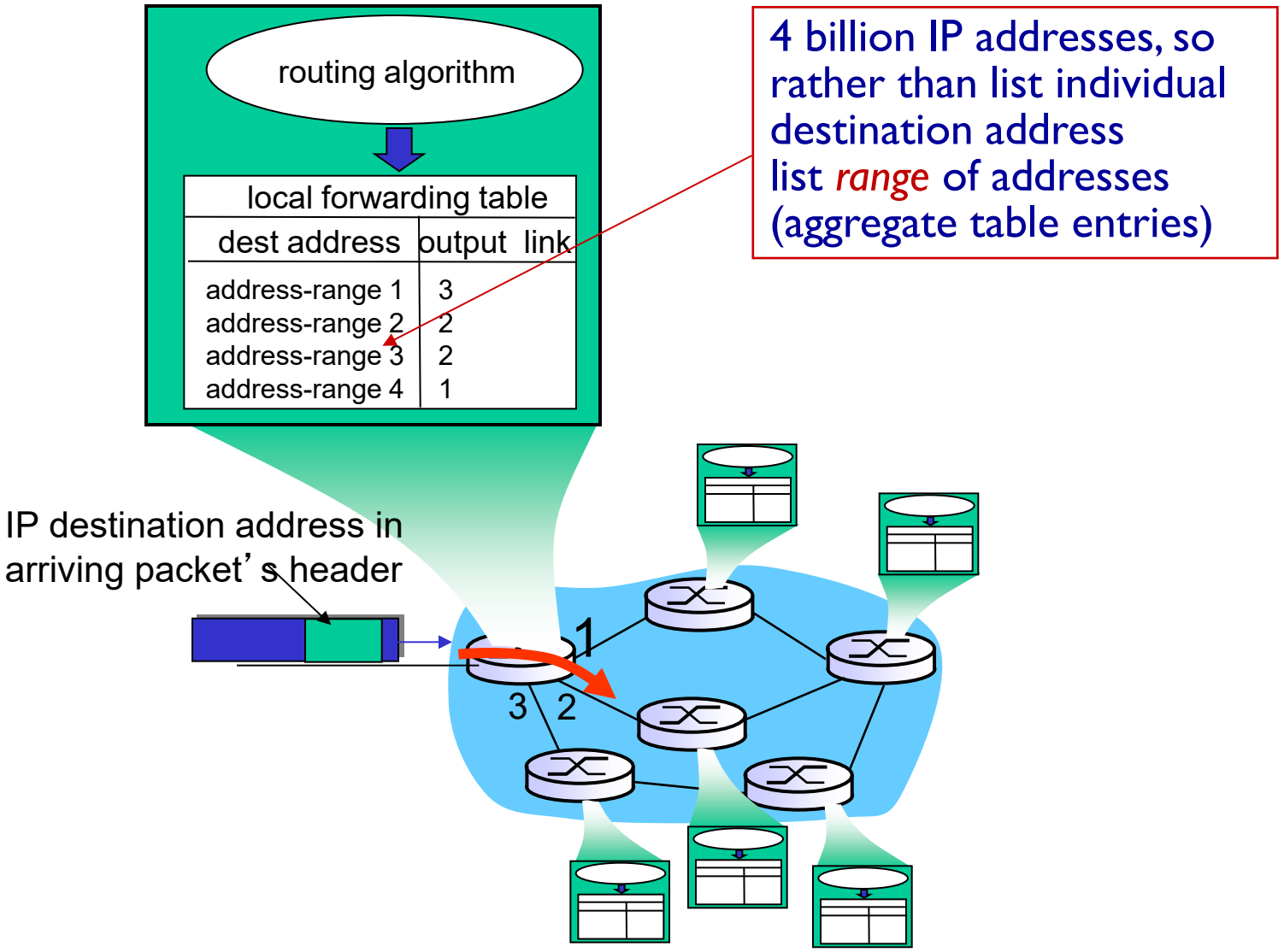


# Datagram Networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address



# Datagram Forwarding Table



# Datagram Forwarding Table – Example

| Destination Address Range | Link Interface |
|---|---|
| 11001000 00010111 00010000 00000000<br>*through*<br>11001000 00010111 00010111 11111111 | 0 |
| 11001000 00010111 00011000 00000000<br>*through*<br>11001000 00010111 00011000 11111111 | 1 |
| 11001000 00010111 00011001 00000000<br>*through*<br>11001000 00010111 00011111 11111111 | 2 |
| *otherwise* | 3 |

*Q: but what happens if ranges don't divide up so nicely?*

# Longest Prefix Matching

*longest prefix matching*

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

| Destination Address Range | Link interface |
|---|---|
| 11001000 00010111 00010*** ********* | 0 |
| 11001000 00010111 00011000 ********* | 1 |
| 11001000 00010111 00011*** ********* | 2 |
| otherwise | 3 |

Examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

# Datagram or VC Network: Why?

## *Internet (datagram)*

- data exchange among computers
  - "elastic" service, no strict timing req.
- many link types
  - different characteristics
  - uniform service difficult
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - ***simple inside network, complexity at "edge"***

## *ATM (VC)*

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - ***complexity inside network***

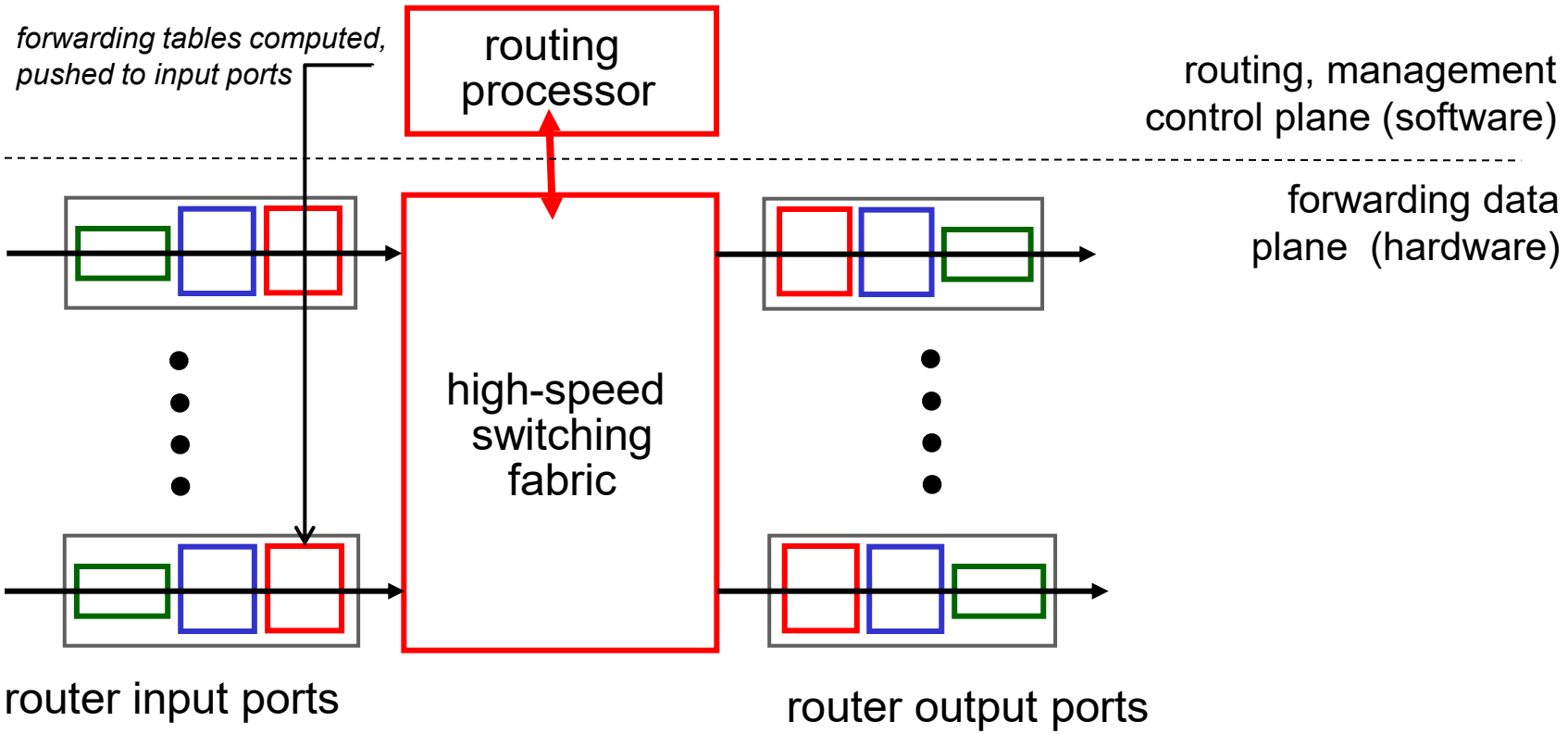
*The end-to-end arguments*

http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

# Router Architecture Overview

two key router functions:

- run *routing* algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link



# Internet Standards Philosophy

"We reject: kings, presidents and voting.

We believe in: rough consensus and running code."

*- David Clark, IETF24, July 1992*