EEL-4736/5737 Principles of Computer System Design

Lecture Slides 15

Textbook Chapter 7

Networks

Introduction

- Communication link abstraction
  - SEND, RECEIVE
- A network provides a communication system interconnecting multiple entities
- Networks have interesting properties with significant design implications
- In this class, we will introduce basics of networks, as a computer system and system component
  - A major topic of its own – treatment will not be extensive

Interesting properties

- Speed of light is finite
  - Boston to L.A.
    - Direct cable route: 20 ms propagation delay
    - Through geo-stationary satellite: 244 ms
  - Within a data center
    - Nanoseconds
  - Mars and back: 6 minutes
- Implication:
  - Design needs to deal with orders of magnitude differences

Interesting properties

- Harsh environments
  - Cables under the floor, deep in ocean
  - Waves subject to weather interference
- Implication:
  - Design must deal with various failure modes

Interesting properties

- Limited bandwidth
  - Different communication channels can have very different bandwidth limitations
    - Modem, wireless, broadband, local-area, wide-area
  - Available data rate is another parameter that may vary by orders of magnitude

Other considerations

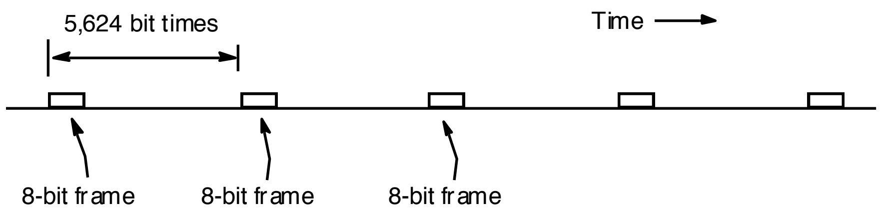
- Networks are often shared
  - Building separate links for each pair of entities does not scale
  - Even in small-scale networks, sharing is often the most economical and simpler to manage approach
    - E.g. Ethernet
  - Number of connected entities can also vary by orders of magnitude
    - Home LAN: <10; Internet: billions

Other considerations

- Several parameters vary by orders of magnitude
  - Latency
  - Data rates
  - Number of nodes
- Also, traffic demand
  - Bytes/second casual browsing
  - MB/s transfers of large files

Sharing and multiplexing

- Multiplexing (e.g. time, frequency) key to sharing
- Capacity is limited
  - How is it apportioned?

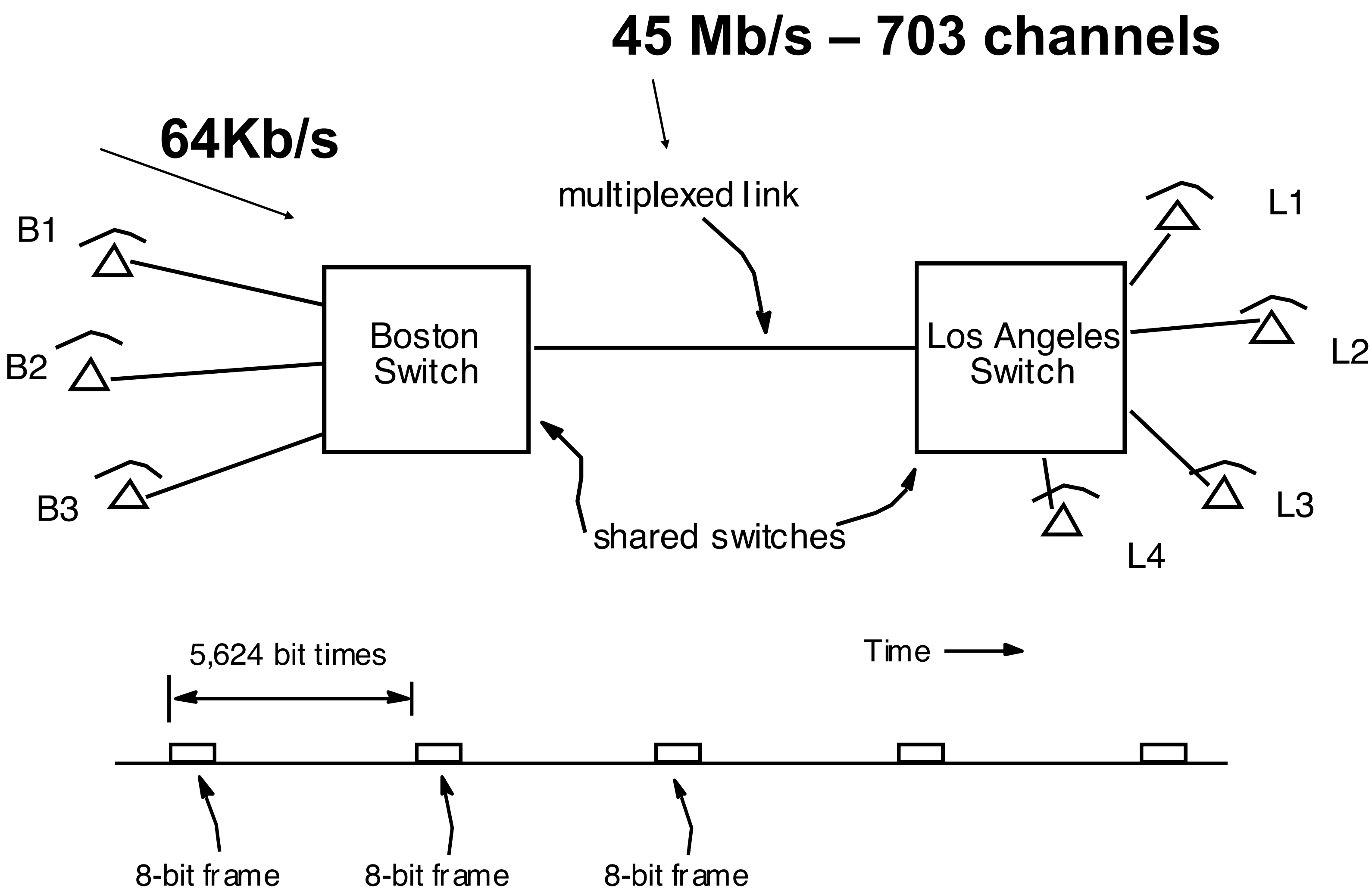


- Isochronous
  - "Equally-timed" communication
  - E.g.: telephone system
    - 64Kbit/s needed per communication channel
    - 45Mb/s can carry up to 703 conversations
    - 703 calls accepted with 64Kb/s rate and fixed delay; 704^{th} is denied – hard-edged limiting scheme

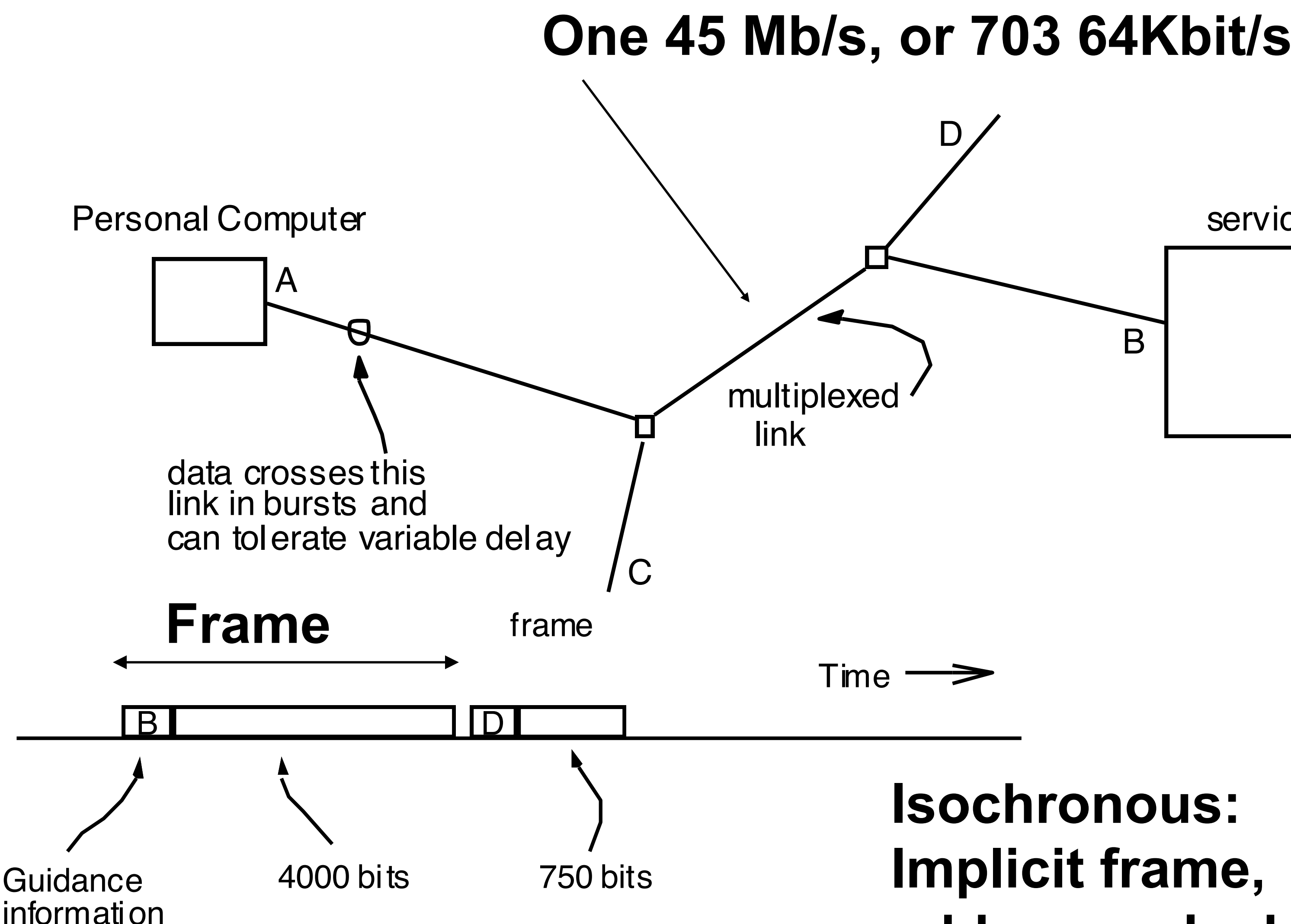
Asynchronous communication

- Data applications are typically bursty and require variable rates
- Isochronous approach:
  – Channels can be under-utilized most of the time, impose limitation during bursts
- Data communication – usually asynchronous
  – Messages vs. streams
  – Forgo guarantee of uniform data rate/latency if in return messages can move in more quickly

Example


45 Mb/s – 703 channels
64Kb/s
multiplexed link
B1
B2
B3
Boston Switch
Los Angeles Switch
L1
L2
L3
L4
shared switches
5,624 bit times
Time
8-bit frame
8-bit frame
8-bit frame

Example



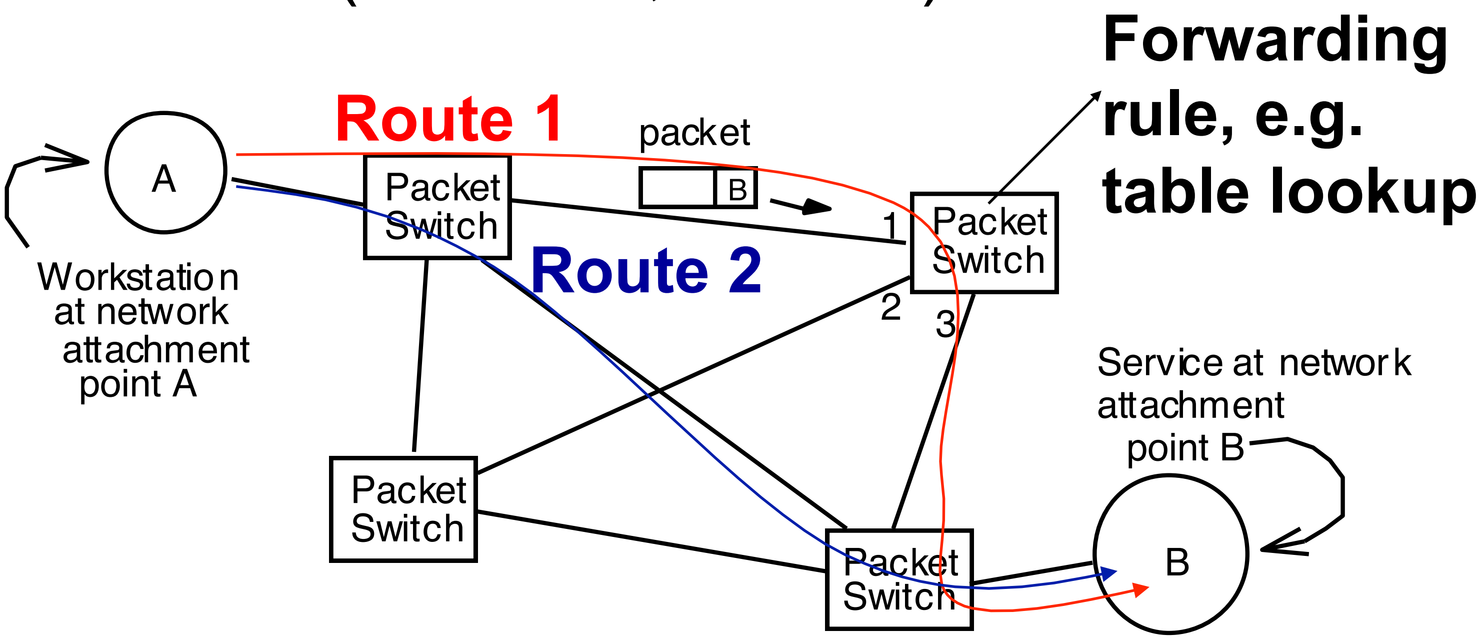
Isochronous: Implicit frame, address - clock

Packet switching

- Connection-less
  - Message carries destination address
- Most links place limit on frame sizes
  - Large messages broken down into smaller messages
- Asynchronous networks can also support stream communication
  - Split messages in smaller datagrams
  - Delay and rate may not be guaranteed as in the case of connection-oriented
  - E.g. voice over IP

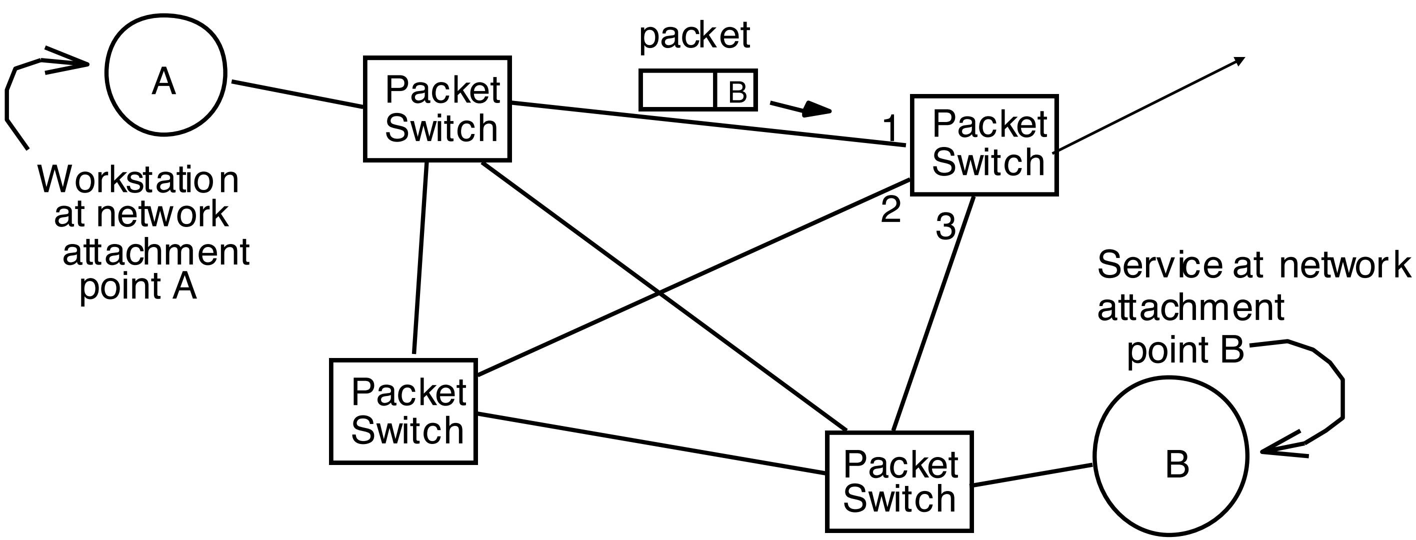
Packet forwarding

- Asynchronous communication links: organized as packet forwarding networks
- Messages originated at network attachment points, switched at intermediary packet forwarders (switches, routers)



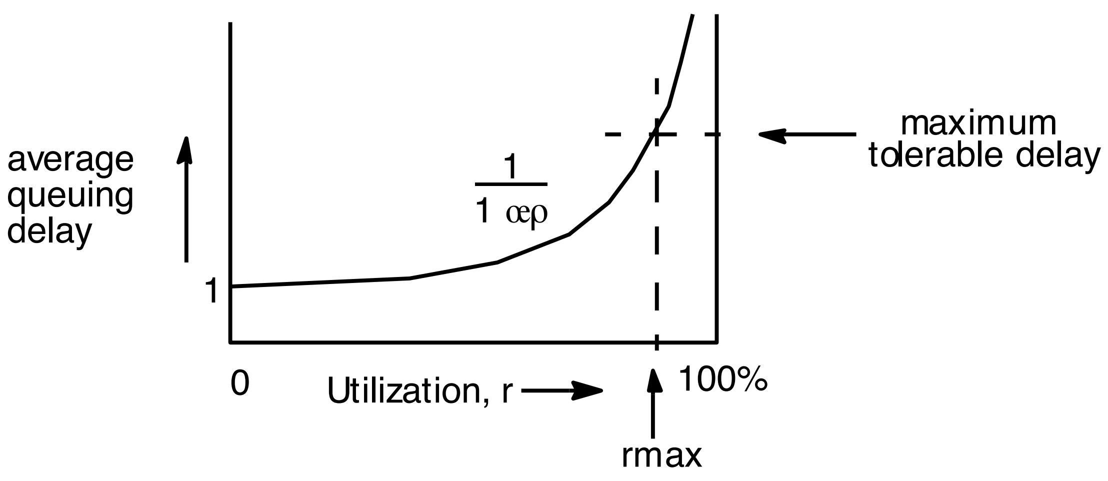
End-to-end latency

- Propagation delay: speed of light, link length
- Transmission delay: time to receive message a function of its size and link bandwidth
- Processing delay: each forwarding lookup
- Queuing delay: buffer in memory if busy



Performance

- Asynchronous vs isochronous:
  - Smooth degradation of service (longer latency, smaller rate) vs. abrupt admission control
  - However, as wehave seen in Chapter 6, queuing delays can grow unbounded at high utilization



Buffer overflows

- In practice, queues in systems attached to network are bounded
- How large should they be?
  - Plan for the worst case?
    - Estimate maximum traffic, allocate enough buffers for worst case
  - Plan for typical case and fight back?
    - Estimate typical traffic, send messages back when buffers fill up
  - Plan for typical case and discard overflow?

Strategy 1

- Plan for the worst case
  - In a large network, it may be impossible to predict
    - What is the worst-case traffic of an Internet router? Today vs next year?
  - Worst-case may be orders of magnitude than typical case
    - Cost considerations
  - Delays in the worst case will still be very large

Strategy 2

- Plan for typical case and fight back
  - When congestion happens or is about to
  - Low on buffers – send message back asking sender to stop or throttle down
    - "quench"
    - Back to forwarder at other side of link, perhaps all the way back to sender
  - Problems
    - Quench packets can add to the congestion – possible that entire path is congested
    - May cause deadlock with cyclic paths of forwarders asking others to stop sending
    - Sending quench all the way back to source – too long a latency may not swiftly slow down sender

Strategy 3

- Plan for typical case and discard
  - What most packet forwarding networks do in face of congestion
  - Implication: packets may be lost and need to be re-sent
  - *Rate adaptation* approaches can be built to achieve goal of slowing down sender
    - Use of acknowledgements – lack of them leads to re-transmission at slowed rate

Guaranteed vs. best effort

- Guaranteed delivery networks
  - Track messages and take every possible measure to deliver a packet or notify sender that it could not be delivered
    - E.g. first-class mail
- Best-effort networks
  - Designed for typical case, drop messages when congested
    - E.g. “junk” 3rd class mail – service contract allows discarding message

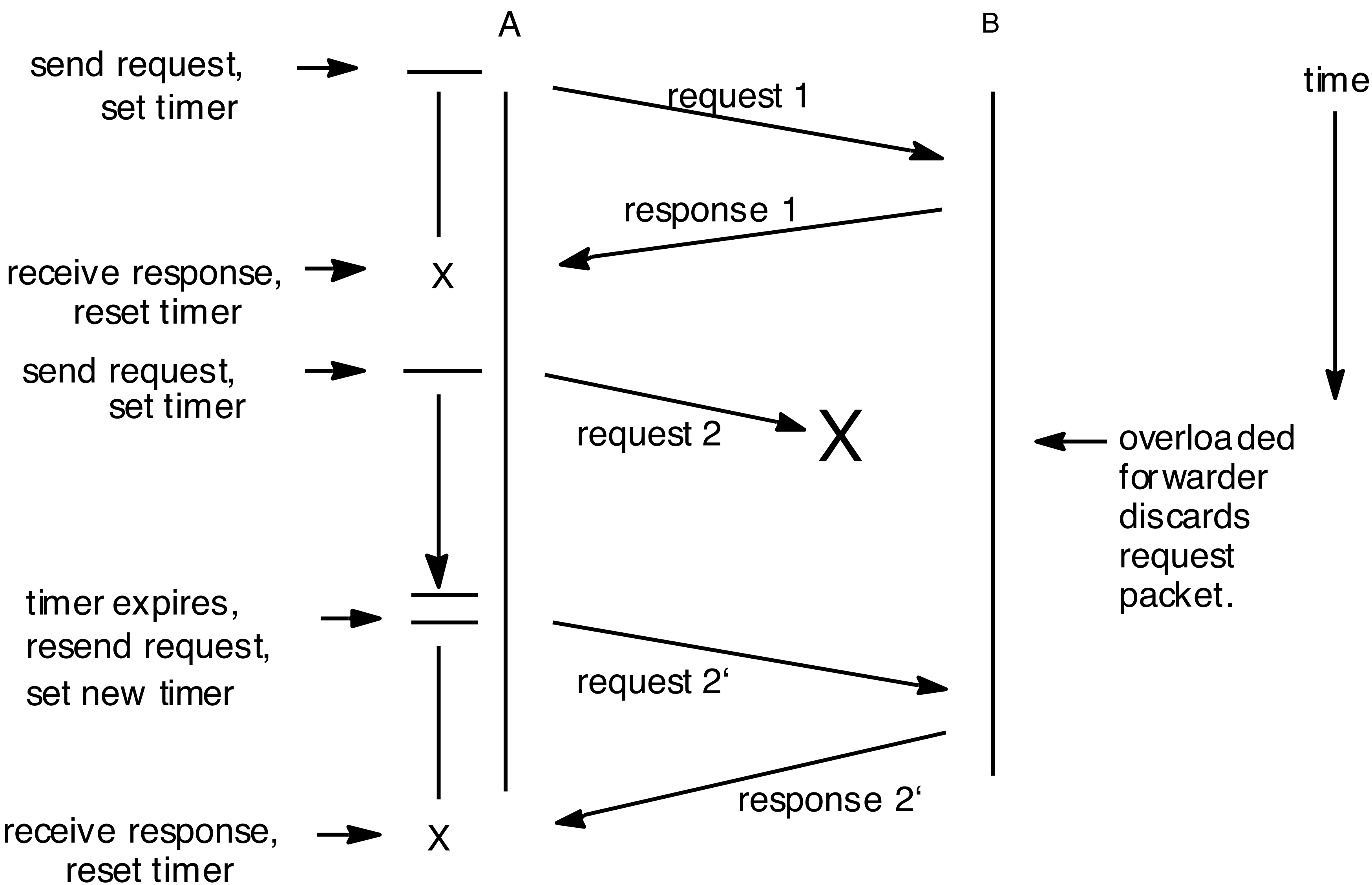
Guaranteed vs. best effort

- The Internet is designed as a best-effort network
- Guaranteed delivery services can be built on top of it
  - Organization in layers is important
    - To be discussed later

Dealing with lost messages

- Protocols need to deal with message loss because it can happen regardless of dropped packets
  - E.g. server unavailable, crashed
  - Timeout/resend mechanisms, as we've seen in RPC

Timeout/resend

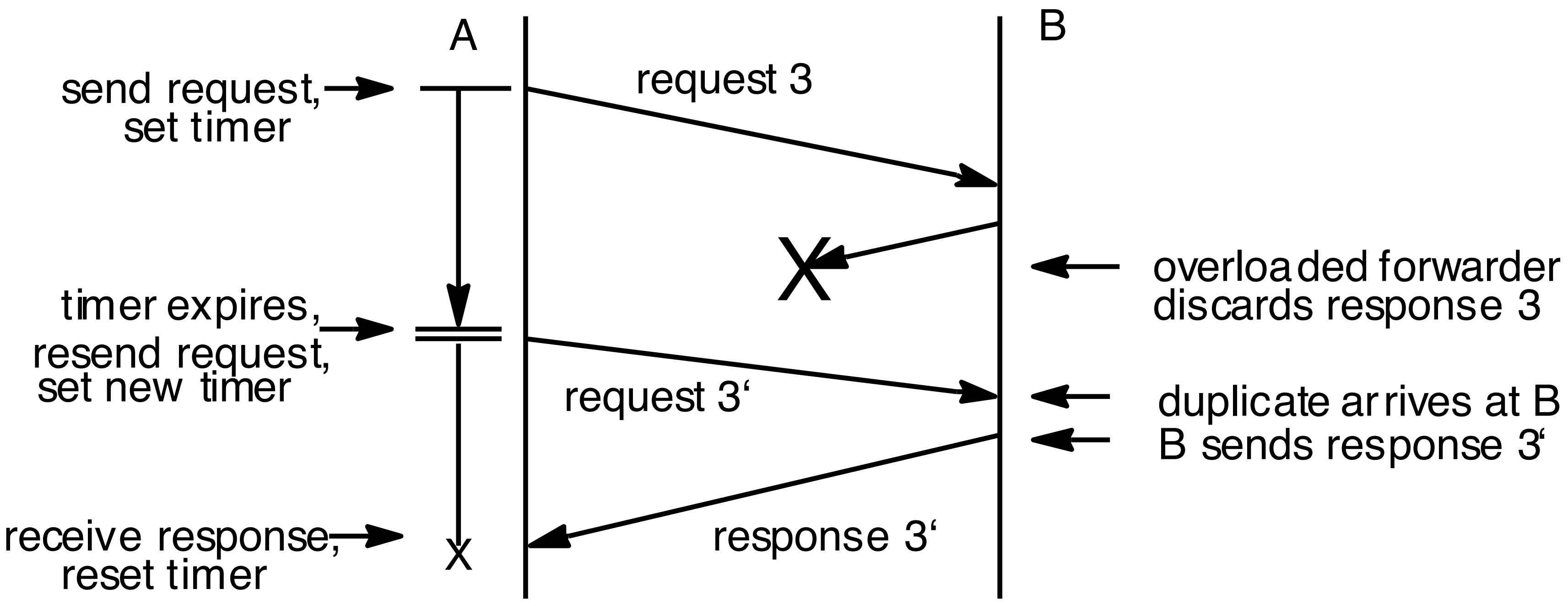


Dealing with duplicates

- From sender's perspective, when timer expires, cannot differentiate whether packet was lost or very slow
  – No matter how long the timeout is
    - Very long timeouts - bad idea from performance standpoint
  – Need to deal with the case where a duplicate packet is re-sent, and both arrive

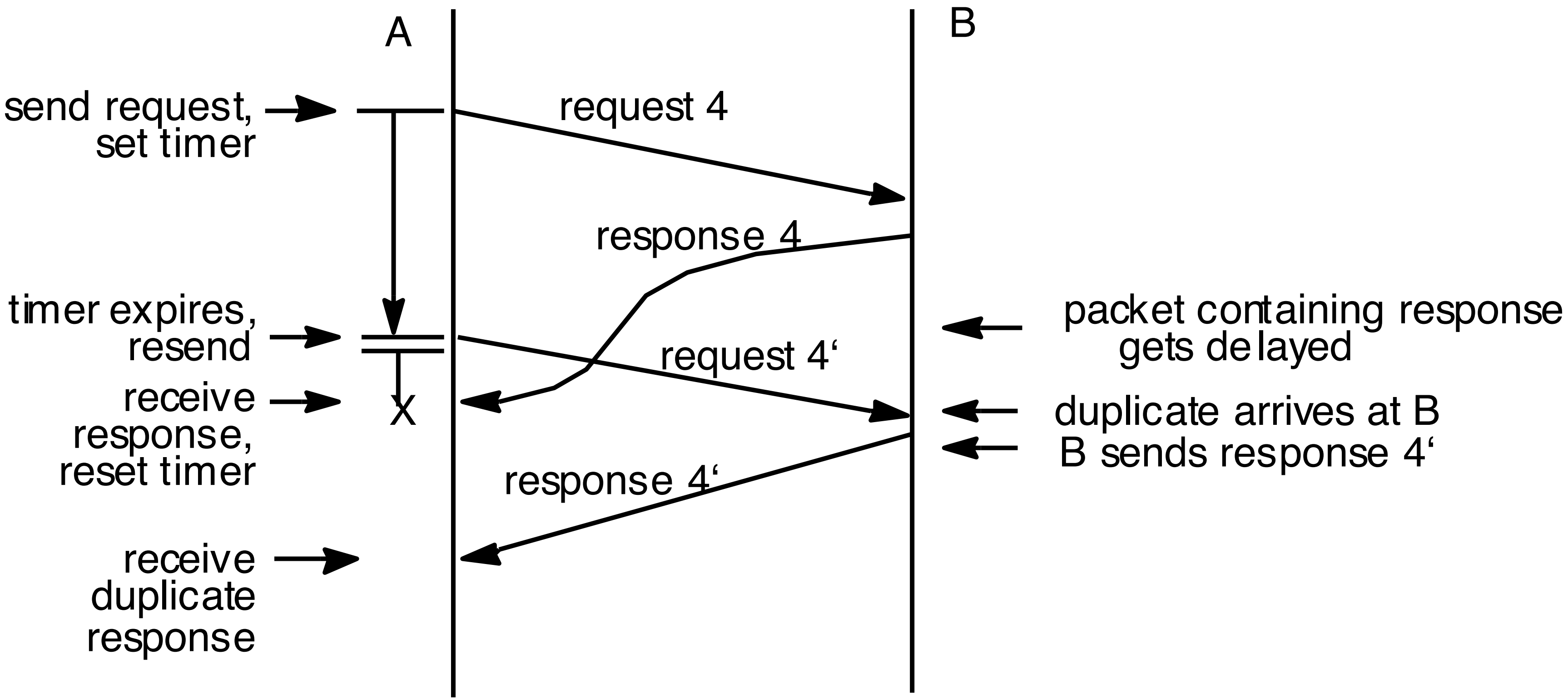
Dealing with duplicates

- Case 1 – response was lost
  - Receiver needs to be ready to re-send response



Dealing with duplicates

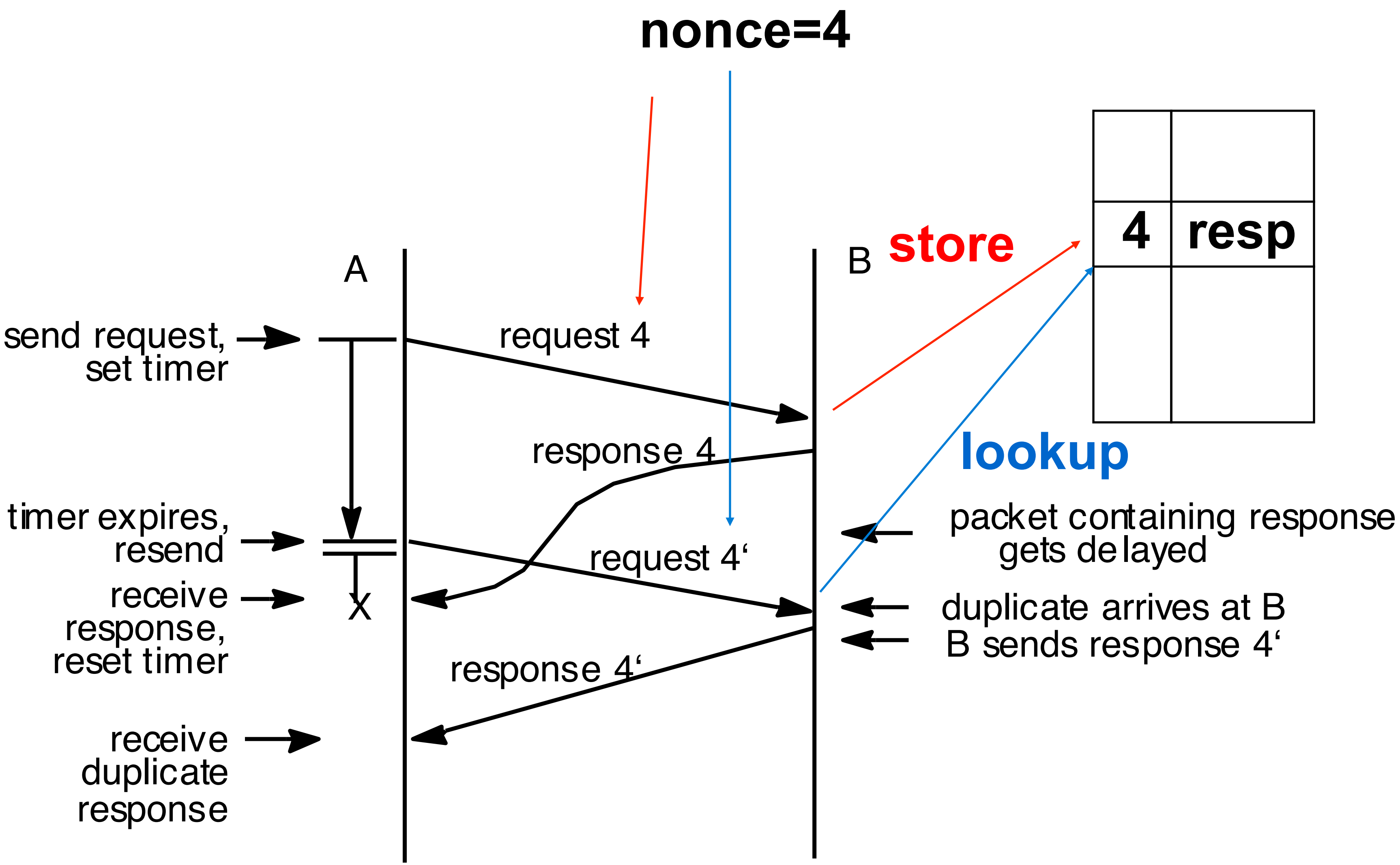
- Case 2 – long delay
  - Receiver ready to resend response, sender to deal with duplicate response



Dealing with duplicates

- General procedure
  - Tag messages with unique numbers
    - “nonce” – unique identifier for each message, never to be reused
      - E.g. monotonically increasing serial numbers, large identifier space
  - Record nonces that have been handled or are being handled
    - If it is a duplicate, does not perform action
    - Still, need to reconstruct/resend response
      - Keep actual response in nonce list

Dealing with duplicates



Dealing with duplicates

- Possible that sender sees duplicates for a single request it originated
  - If forwarders themselves have timer/retry
- Nonce can be echoed back on response, and sender also keeps list of nonces associated with outstanding requests
- Challenges to keep in mind:
  - This builds state
    - How many entries should be kept?
    - Crash, state stored in volatile memory?
  - If services can be made idempotent, duplicate suppression not needed

Dealing with message errors

- Messages can have errors in their data
  - Noise in link
  - Memory bit flip in forwarder
  - Bug in an interpreter
- First and foremost, must be able to *detect* errors
  - Information theory, coding techniques can provide certain guarantees
    - Will overview some later
- How should errors be dealt with?

Dealing with message errors

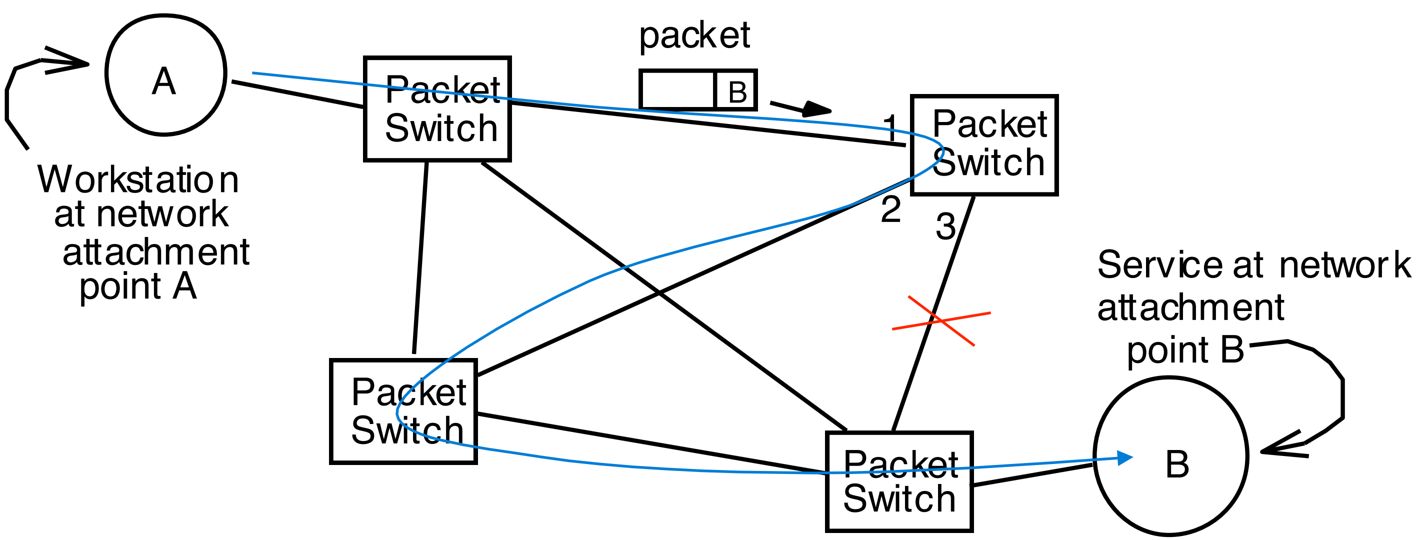
- One approach: once detected, attempt to correct an error
  - Example:
    - Error Correction Codes (ECC) used in DRAM memories – 64 + 8 bits can detect two and correct one bit error
  - Advantages?
  - Disadvantages?

Dealing with message errors

- Another approach: once detected, *discard*
  - Advantages?
  - Disadvantages?

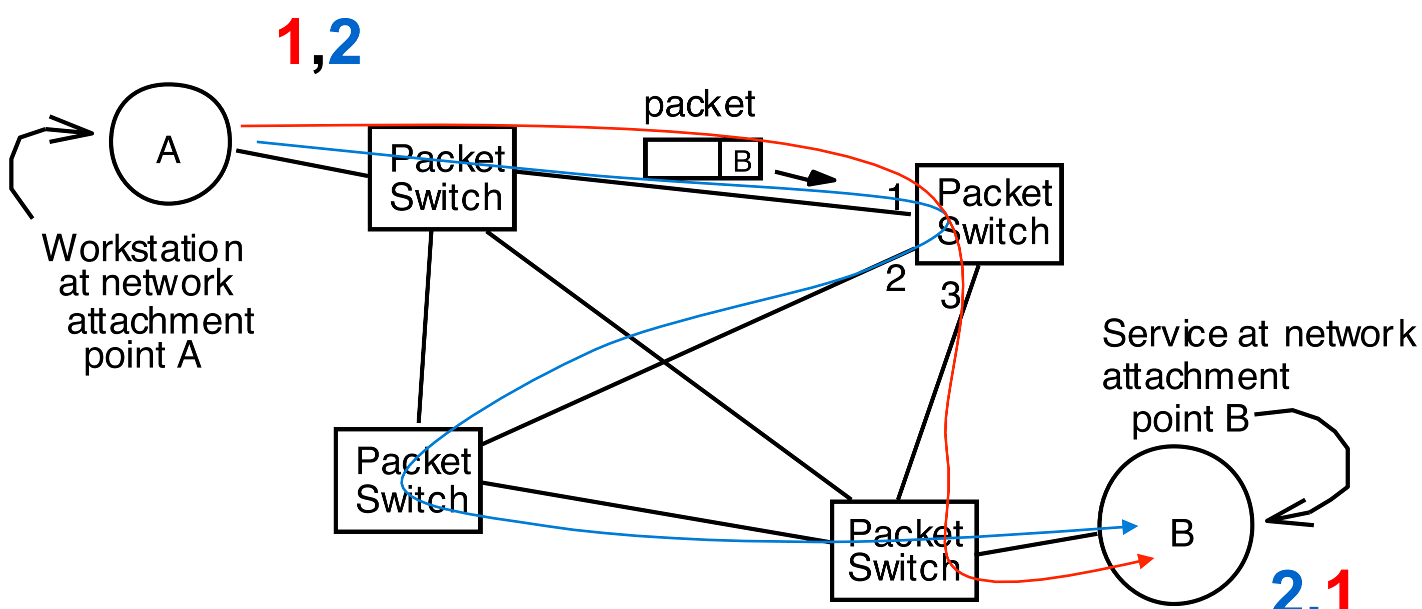
Dealing with broken links

- A communication link may be noisy and add error to a message, and it may also become unavailable (temporarily or permanently)
- Design of routing must deal with this case



Reordered delivery

- Guaranteeing in-order delivery is challenging
  - Multiple paths; temporary link failures; request re-sending
- In large networks, approaches again rely on identifiers for messages
  - In this case, they carry ordering information to reconstruct



Summary

- Best effort contract
  - Accept a message with the expectation that it is likely to deliver it, but no guarantee
  - Endpoints must deal with packet loss, duplication, varying delay, out-of-order

Network Type		Application characteristics: Continuous stream (e.g., interactive voice)	Bursts of data (most computer-to-computer data)	Response to load variations
	isochronous (e.g., telephone network)	good match	wastes capacity	(hard-edged) either accepts or blocks call
	asynchronous (e.g., Internet)	variable latency upsets application	good match	(gradual) 1 variable delay 2 discards data 3 rate adaptation

Reading

- Section 7.1-7.3