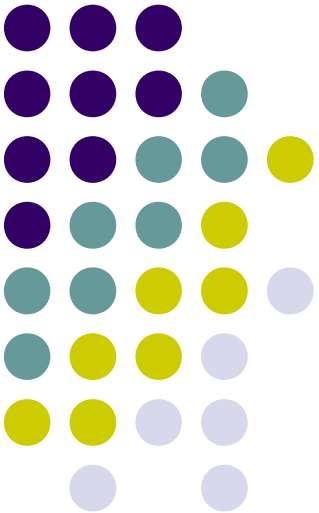
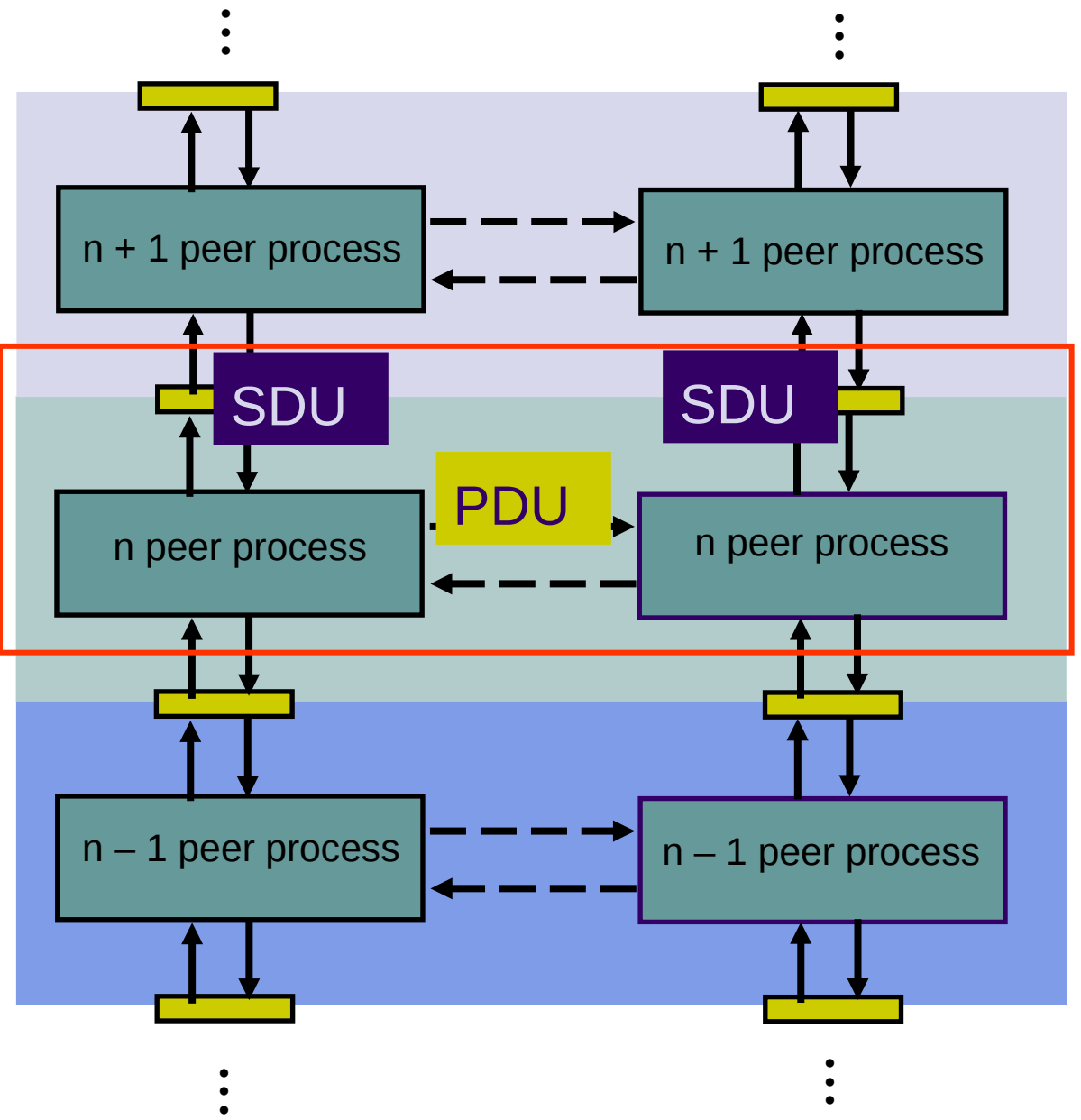
Peer-to-Peer Protocols and Service Models



Peer-to-Peer Protocols



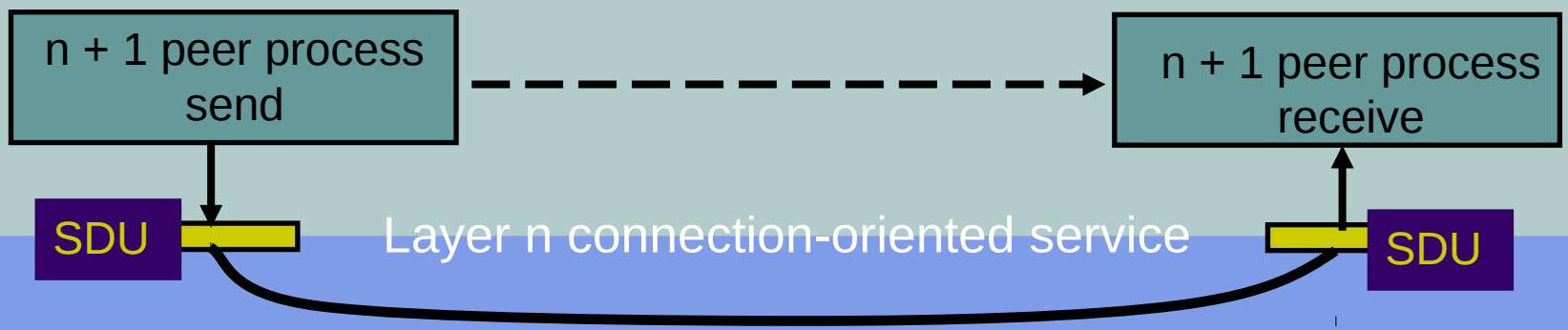
- *Peer-to-Peer processes* execute layer-n protocol to provide service to layer-(n+1)
- Layer-(n+1) peer calls layer-n and passes Service Data Units (SDUs) for transfer
- Layer-n peers exchange Protocol Data Units (PDUs) to effect transfer
- Layer-n delivers SDUs to destination layer-(n+1) peer

Service Models

- The ***service model*** specifies the information transfer service layer-n provides to layer-(n+1)
- The most important distinction is whether the service is:
  - **Connection-oriented**
  - **Connectionless**
- Service model possible features:
  - Arbitrary message size or structure
  - Sequencing and Reliability
  - Timing and Flow control
  - Multiplexing
  - Privacy, integrity, and authentication

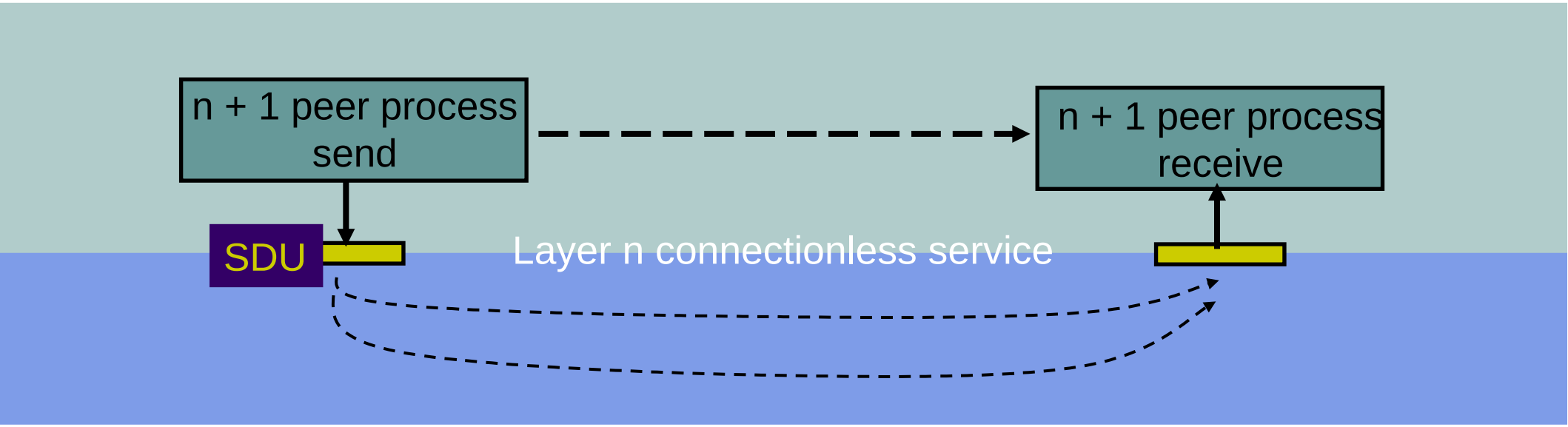
Connection-Oriented Transfer Service

- Connection Establishment
  - Connection must be established between layer-(n+1) peers
  - Layer-n protocol must: Set initial parameters, e.g., sequence numbers; and allocate resources, e.g., buffers
- Message transfer phase
  - Exchange of SDUs
- Disconnect phase
- Example: TCP, PPP



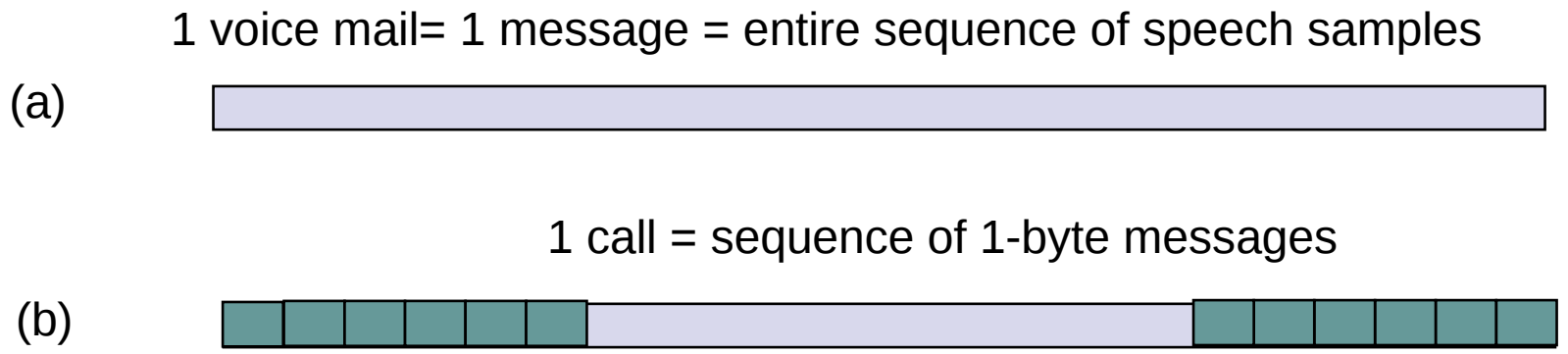
Connectionless Transfer Service

- No Connection setup, simply send SDU
- Each message is sent independently
- Must provide all address information per message
- Simple & quick
- Example: UDP, IP



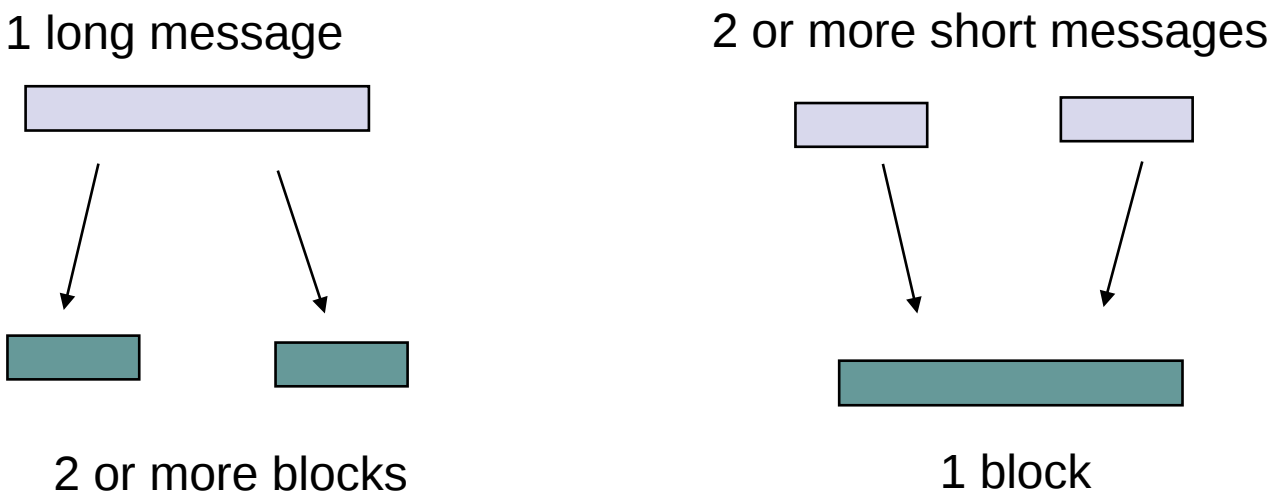
Message Size and Structure

- What message size and structure will a service model accept?
  - Different services impose restrictions on size & structure of data it will transfer
  - Single bit? Block of bytes? Byte stream?
  - Ex: Transfer of voice mail = 1 long message
  - Ex: Transfer of voice call = byte stream



Segmentation & Blocking

- To accommodate arbitrary message size, a layer may have to deal with messages that are too long or too short for its protocol
- ***Segmentation & Reassembly***: a layer breaks long messages into smaller blocks and reassembles these at the destination
- ***Blocking & Unblocking***: a layer combines small messages into bigger blocks prior to transfer



Reliability & Sequencing

- ***Reliability***: Are messages or information stream delivered error-free and without loss or duplication?
- ***Sequencing***: Are messages or information stream delivered in order?
- ***ARQ protocols*** combine error detection, retransmission, and sequence numbering to provide reliability & sequencing
- Examples: TCP and HDLC

Flow Control

- Messages can be lost if receiving system does not have sufficient buffering to store arriving messages
- If destination layer-(n+1) does not retrieve its information fast enough, destination layer-n buffers may overflow
- ***Flow Control*** provide backpressure mechanisms that control transfer according to availability of buffers at the destination
- Examples: TCP and HDLC

Timing

- Applications involving voice and video generate units of information that are related temporally
- Destination application must reconstruct temporal relation in voice/video units
- Network transfer introduces delay & jitter
- ***Timing Recovery*** protocols use *timestamps* & *sequence numbering* to control the delay & jitter in delivered information
- Examples: RTP & associated protocols in Voice over IP

Multiplexing

- ***Multiplexing*** enables multiple layer-(n+1) users to share a layer-n service
- A multiplexing tag is required to identify specific users at the destination
- Examples: UDP, IP

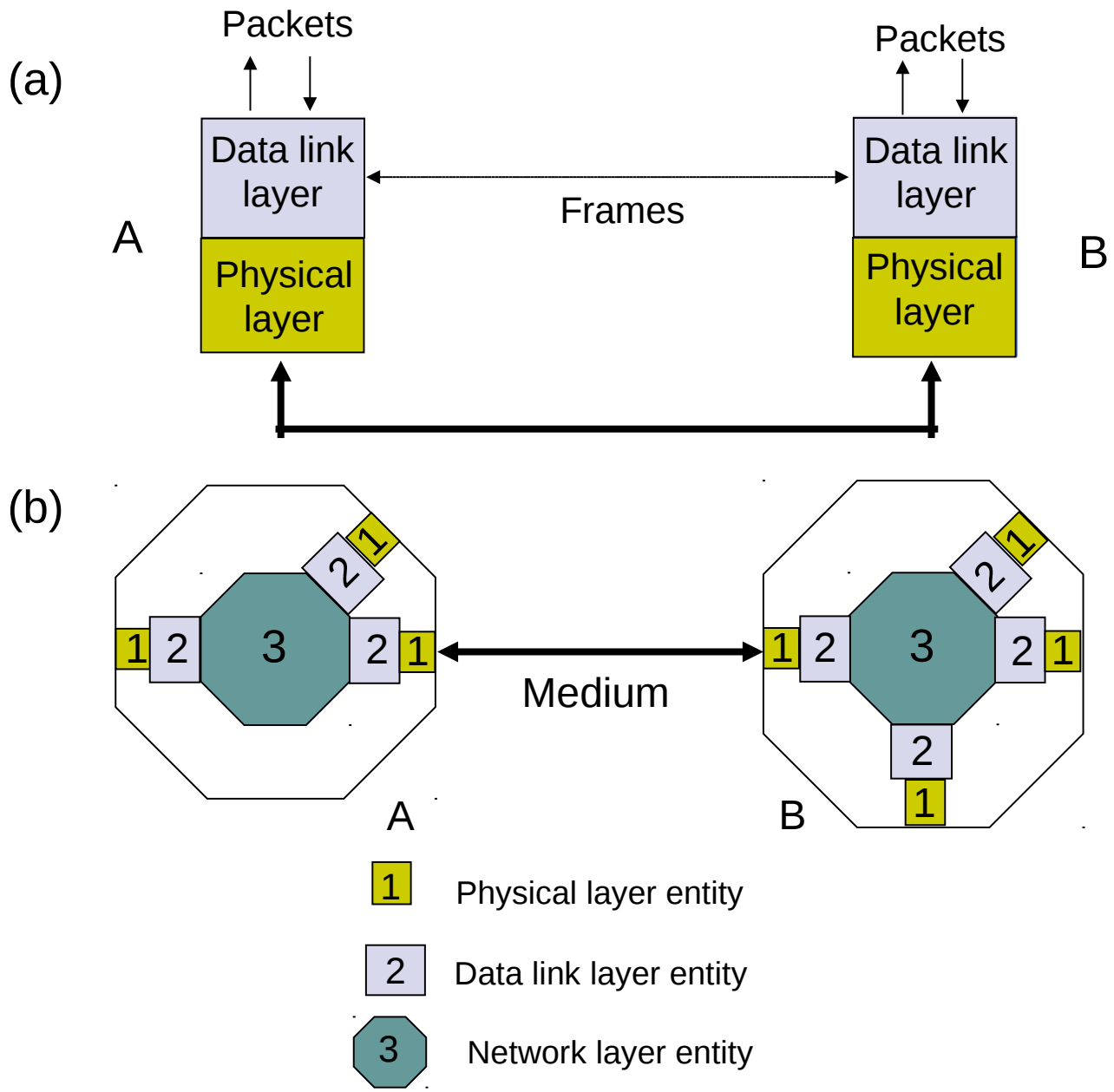
Privacy, Integrity & Authentication

- ***Privacy***: ensuring that information transferred cannot be read by others
- ***Integrity***: ensuring that information is not altered during transfer
- ***Authentication***: verifying that sender and/or receiver are who they claim to be
- ***Security protocols*** provide these services and are discussed in Chapter 11
- Examples: IPSec, SSL

End-to-End vs. Hop-by-Hop

- A service feature can be provided by implementing a protocol
  - end-to-end across the network
  - across every hop in the network
- Example:
  - Perform error control at every hop in the network or only between the source and destination?
  - Perform flow control between every hop in the network or only between source & destination?
- We next consider the tradeoffs between the two approaches

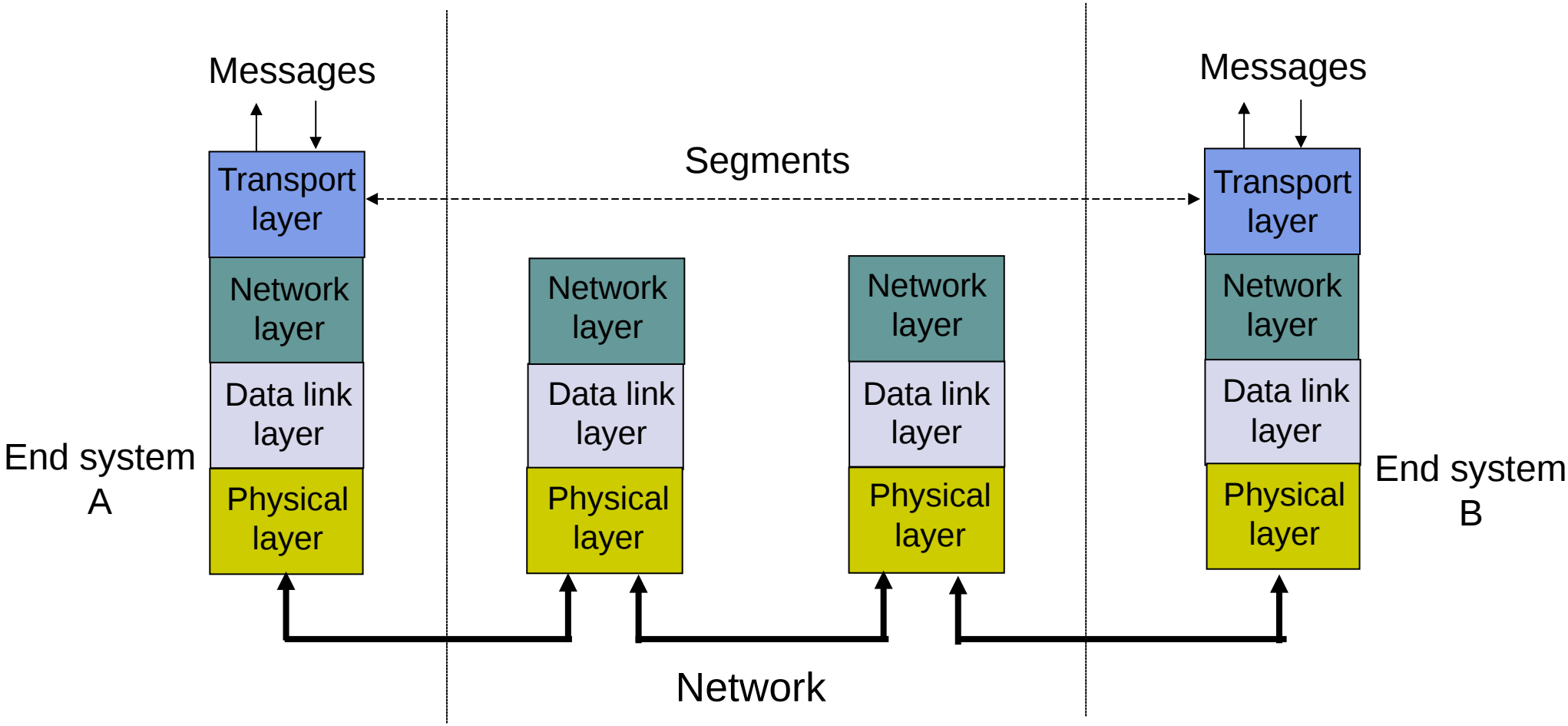
Error control in Data Link Layer



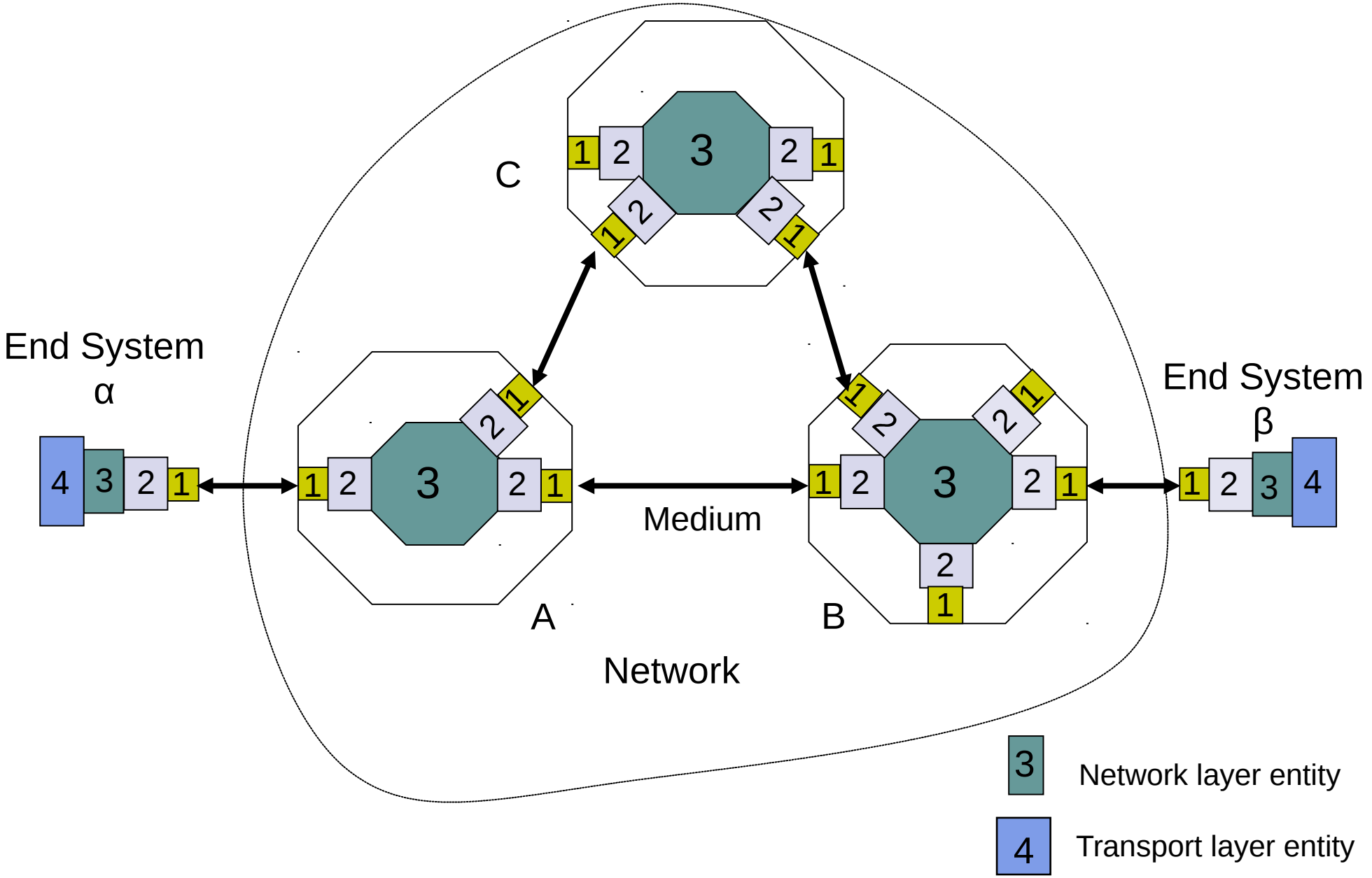
- Data Link operates over wire-like, directly-connected systems
- Frames can be corrupted or lost, but arrive in order
- Data link performs error-checking & retransmission
- Ensures error-free packet transfer between two systems

Error Control in Transport Layer

- Transport layer protocol (e.g., TCP) sends segments across network and performs end-to-end error checking & retransmission
- Underlying network is assumed to be unreliable

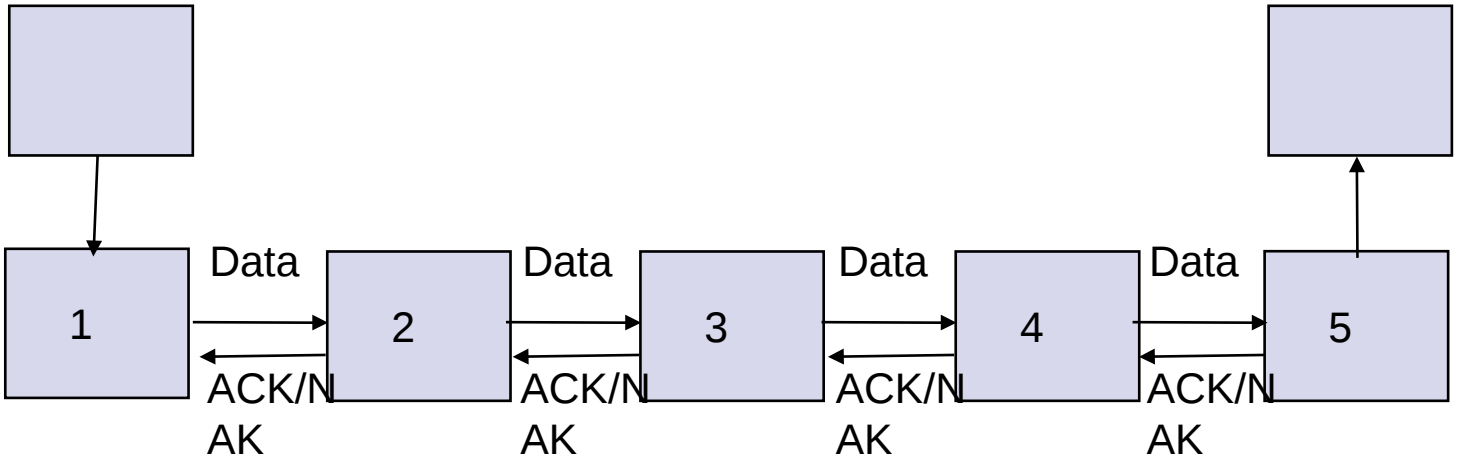


- Segments can experience long delays, can be lost, or arrive out-of-order because packets can follow different paths across network
- End-to-end error control protocol is more difficult



End-to-End Approach Preferred

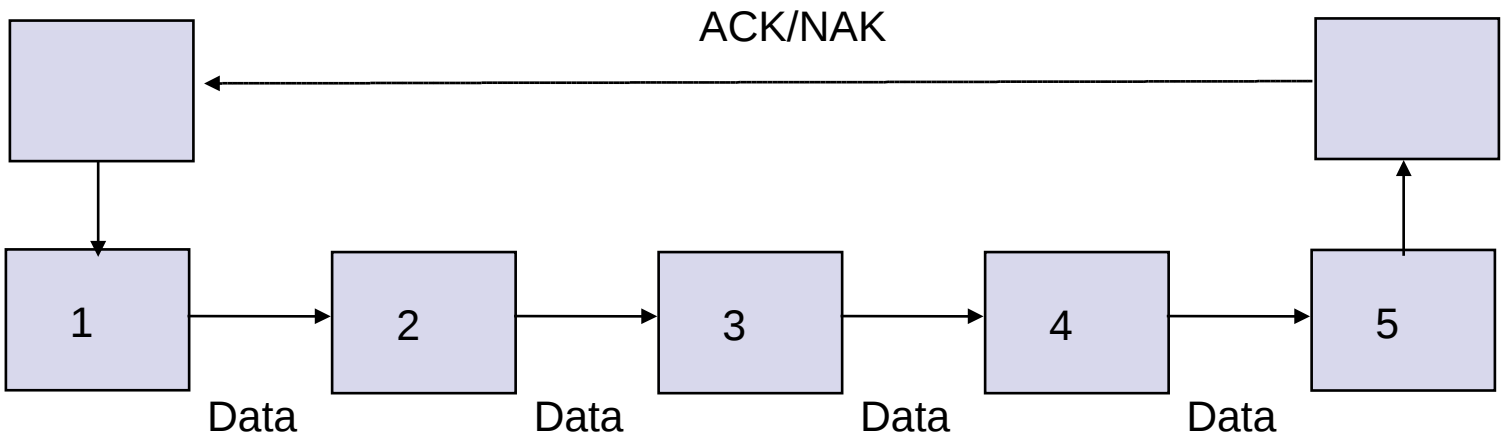
Hop-by-hop



Hop-by-hop cannot ensure E2E correctness

Faster recovery

End-to-end



Simple inside the network

More scalable if complexity at the edge