CAN 1011: Data Communication

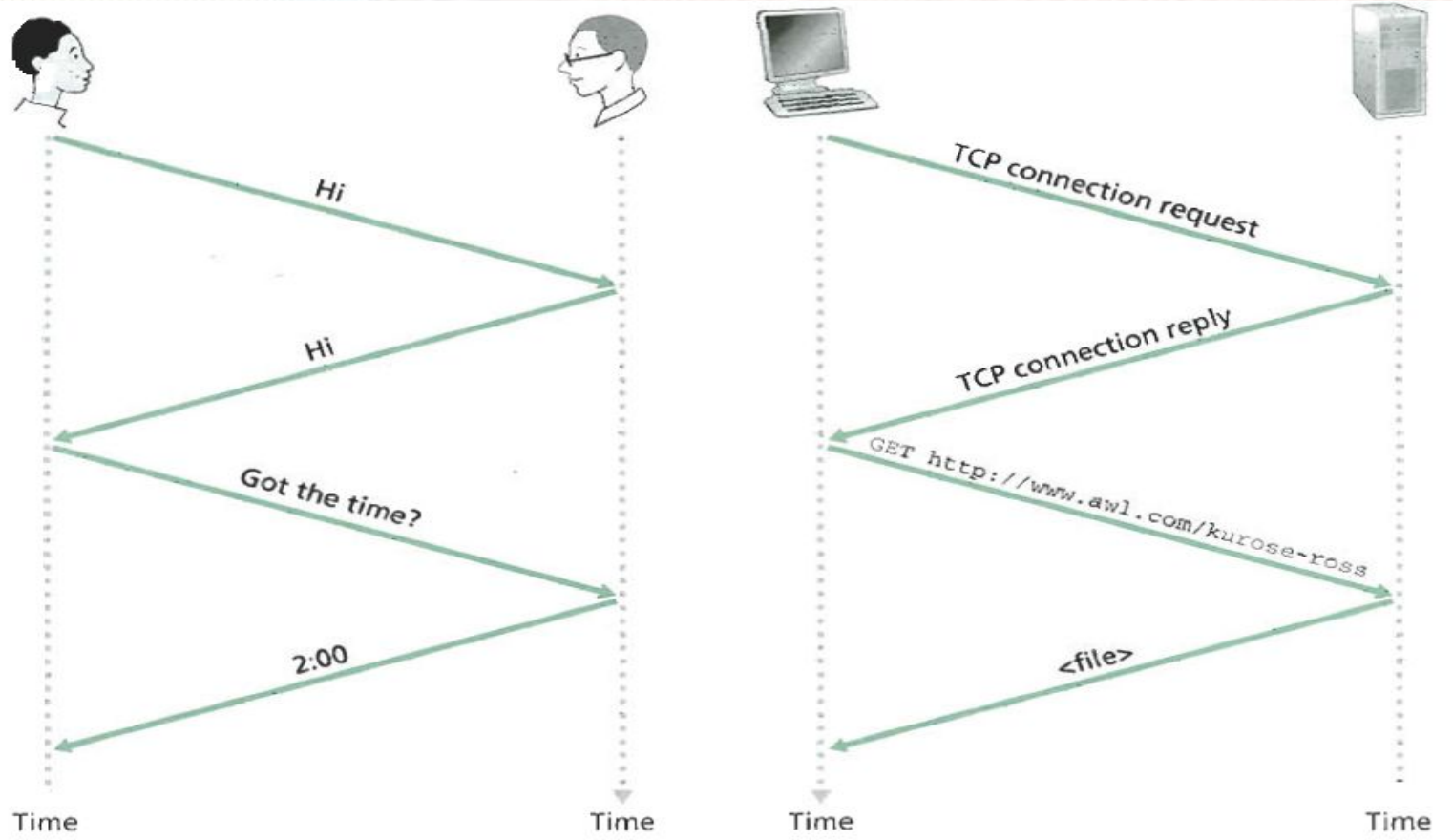
- Protocol characteristics and functions
- Architecture

Contents

- Key Elements of a Protocol
- Layered protocol architecture
- Framework for Standardization
- Simplified File Transfer Architecture
  - A Three-Layer Model
  - Protocol Architectures and Networks: Addressing
  - Primitives and Parameters
  - Protocol operation: Protocol Data Unit (PDU)
- Standard Protocol Architectures
  - The ISO/OSI Protocol Architecture
  - The TCP/IP Protocol Architecture

What is a Protocol?

- A set of rules agreed upon and observed by two communicating entities for successful exchange of data



A human protocol and a computer network protocol

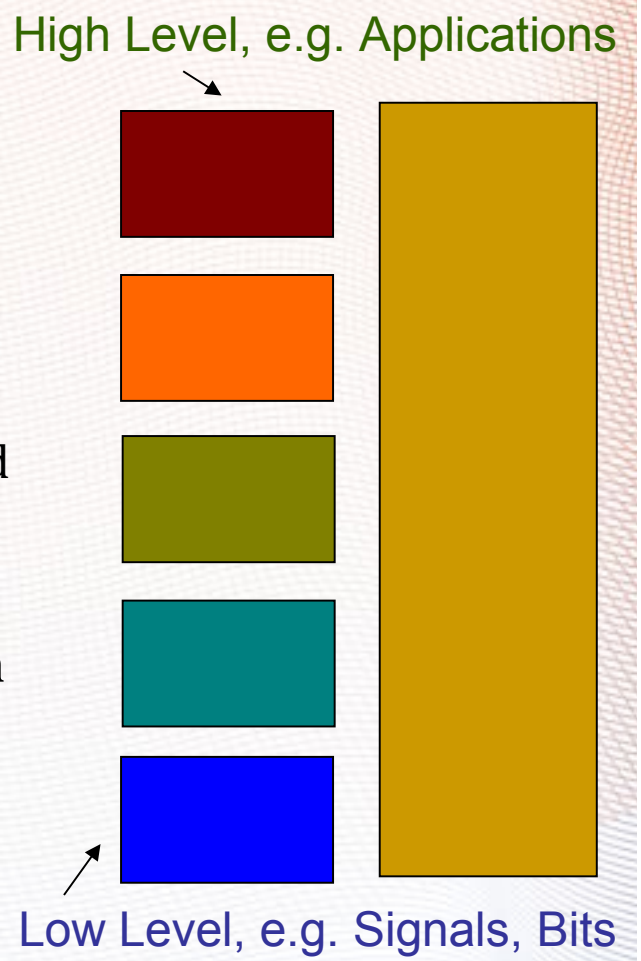
Why we Need Protocols?

- Equipment come from different vendors and run different operating systems… Need common standard procedures and language to communicate properly

- Example: File transfer
  - Source must:
    - Activate communications link
    - Check if destination is prepared to receive data
  - Source file transfer application must:
    - Check if destination file management system will accept and can store file
    - May need file format translation, etc…

Need for layered Protocol Architecture?

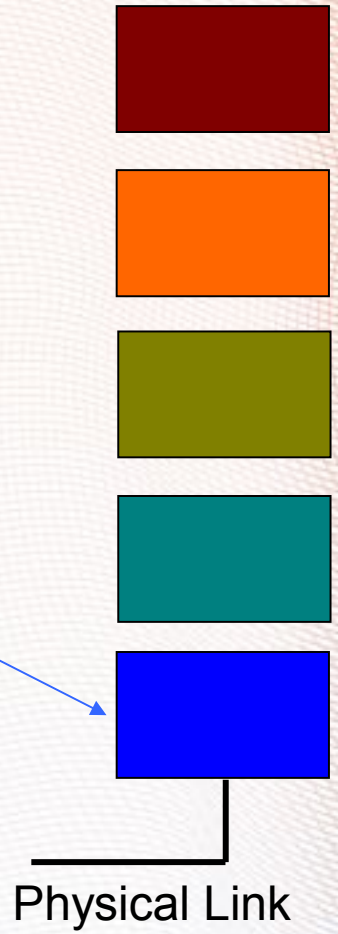
- A large task is better handled through splitting it into smaller subtasks (divide & conquer)
- Can be developed simultaneously by different teams using different platforms (hardware, software)
- Hierarchical layered architecture:
  - The subtasks are implemented separately in layers forming a stack
  - The stack is implemented in all communicating end systems
  - Higher layers handle higher-level tasks
  - A layer provides/requests services to/from an adjacent layer on a system
  - Peer layers in the two communicating systems interact with each other using a protocol
- Examples from everyday life



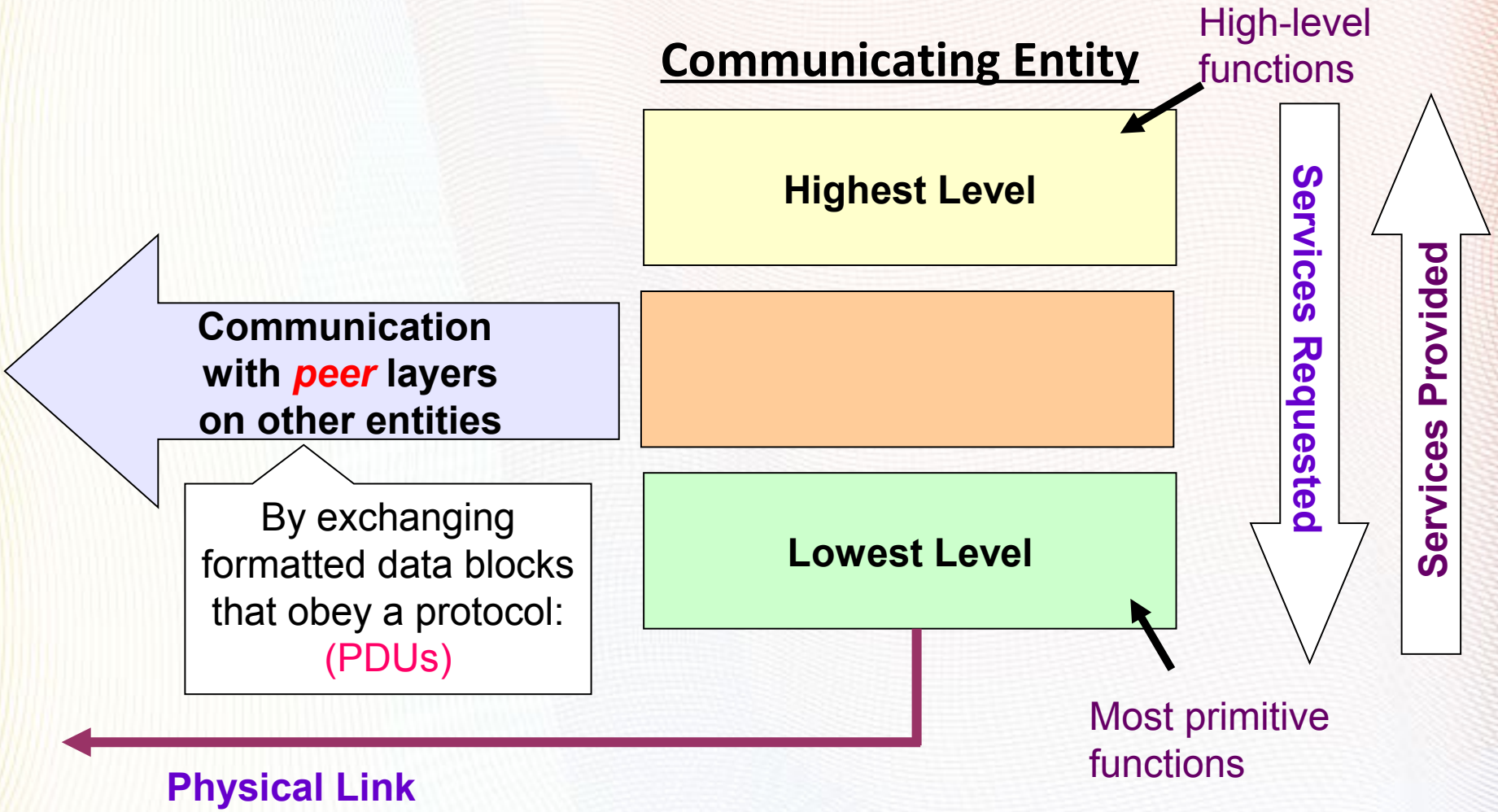
Guidelines for layering

- Each layer performs a subset of *related* subtasks
- Each layer provides services to the higher layer and requests services from the lower layer
- Ideally, changes in one layer should *not* require changes to other layers,

  e.g. changing the link from wire to optical fibre should require changing only the physical layer
- Information hiding: Each layer sees *only* what it needs to see
- Not too few layers:
  - Not enough splitting of functionality
- Not too many layers:
  - Difficult to manage
  - Large communication overhead between them



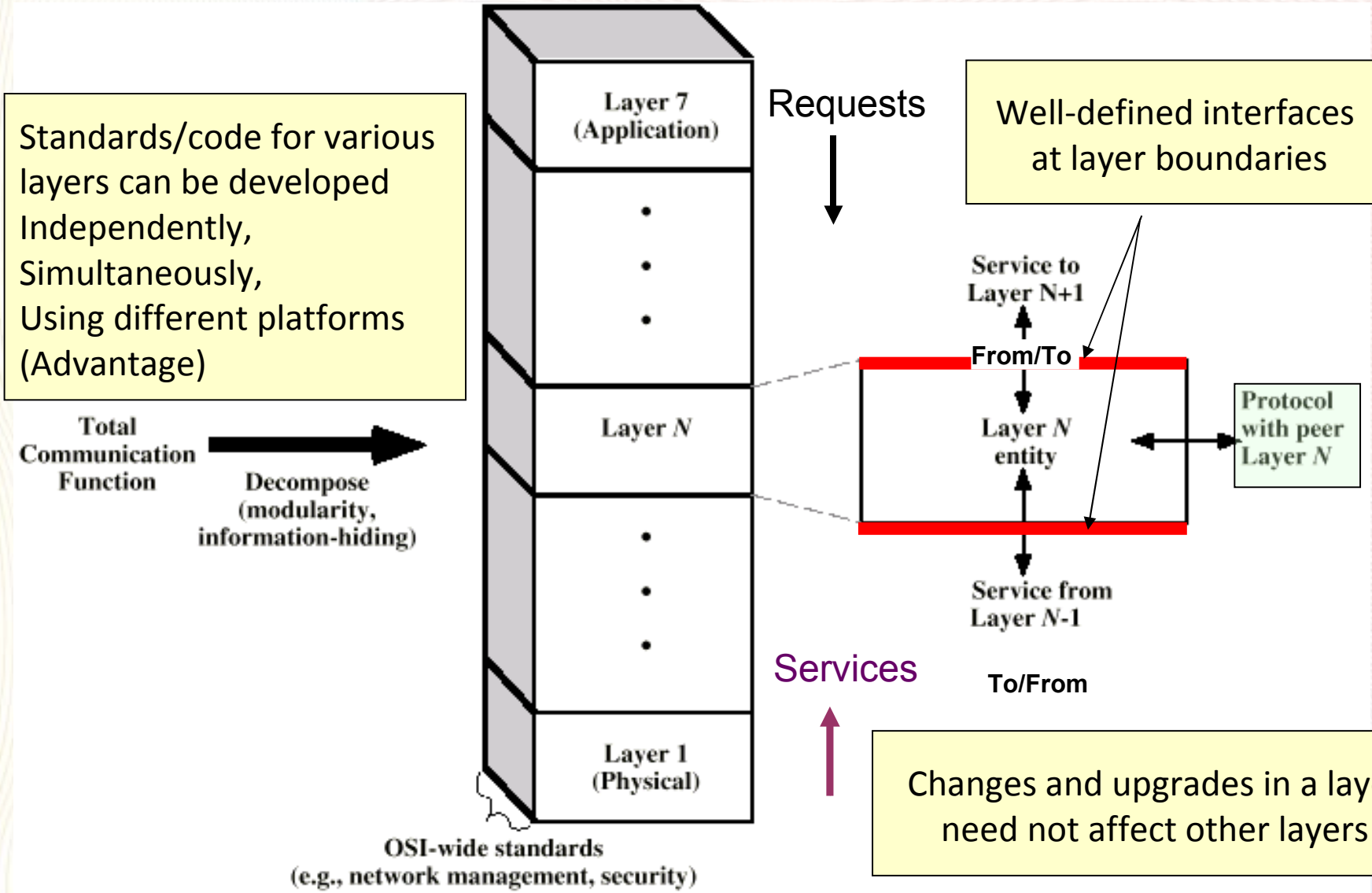
Architecture: Layered Structure



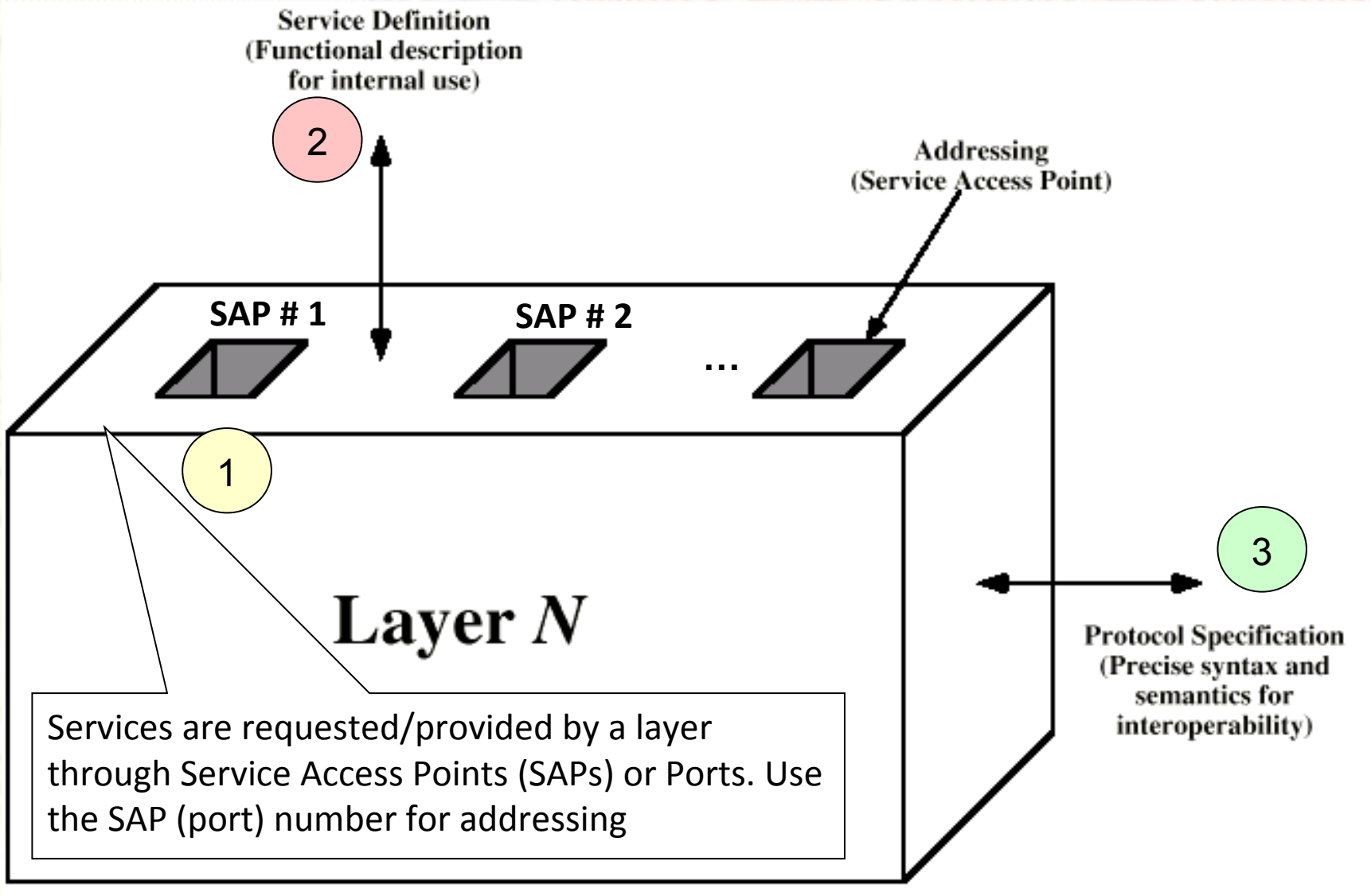
Protocol stack on an end system

Framework for Standardization

Standards/code for various layers can be developed Independently, Simultaneously, Using different platforms (Advantage)



Scope for Layers Standardization



Three standardization elements

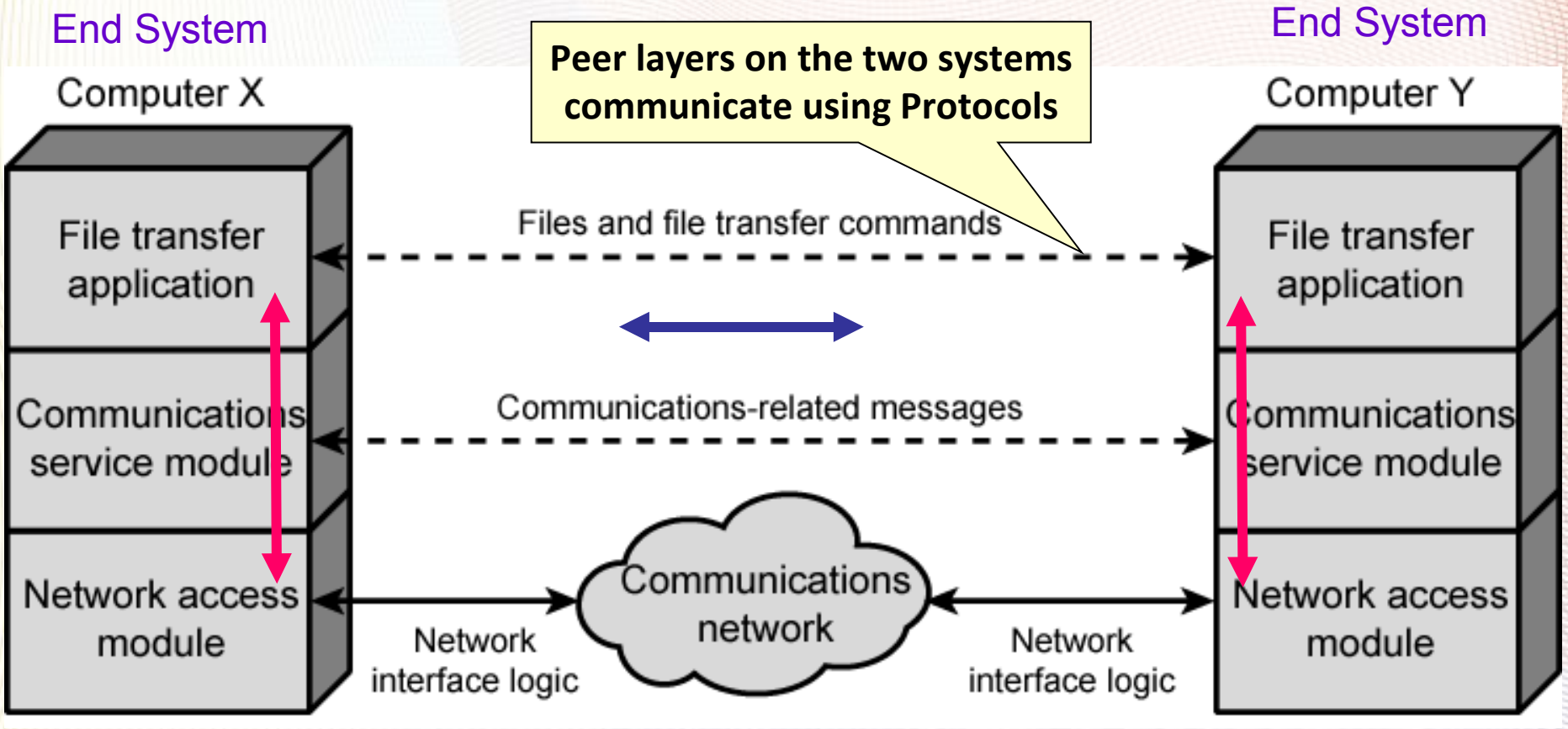
- Addressing within the same system
  - Referencing by SAP numbers: We name the entity in a layer requesting a service from a lower layer by the SAP used on that lower layer
- Service definition (**vertically** within a system)
  - ***Only Functional*** description of what is required between layers within the same system
- Protocol specification (**Horizontally**, more strict)
  - Operates between the same layer on the two communicating entities (peer layers)
  - May involve *different* operating systems
  - So, protocol specifications must be defined precisely
    - Format of data units
    - Semantics (meanings) of all fields

Example: Simplified Protocol Architecture

- The task of file transfer is broken down into three modules (layers) that handle:
  - File transfer application → Application layer (top)
  - Communication services → Transport layer
  - Network access → Network access layer (bottom)

Simplified Architecture for File Transfer

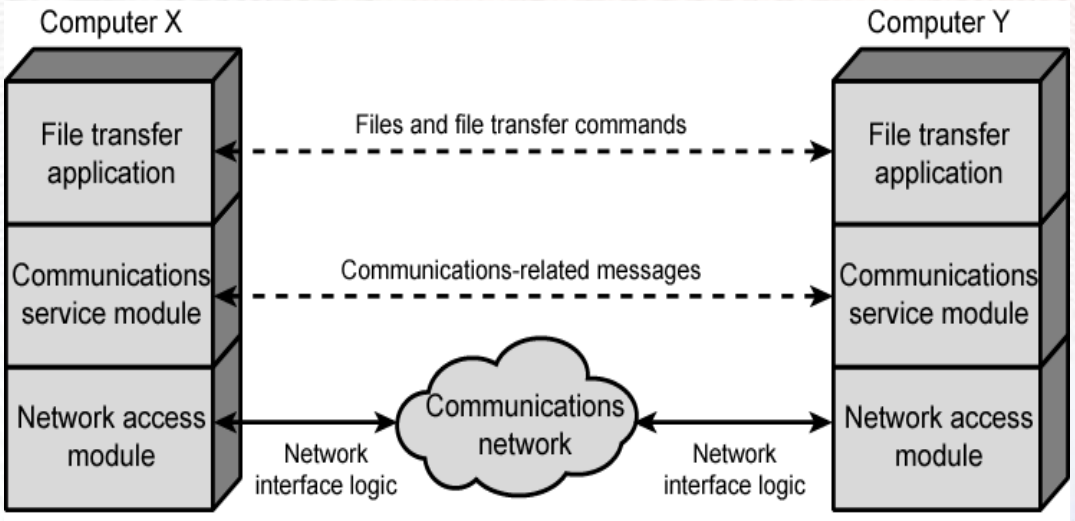
Example: A Three-Layer Model



Assuming Services Requested/ Provided on the same system

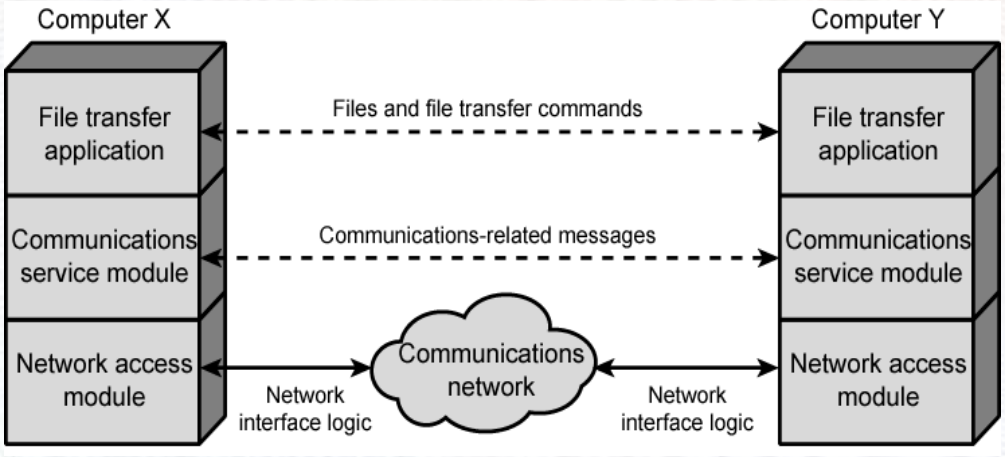
Network Access Layer

- Function: Exchange of data between computer and network
- Depends on type of network used (Ethernet, ATM, etc...) should be the only layer that worries about such details.
- Should specify:
  - Address of destination computer on the network
  - Possibly the level of service required from the network, example priority, delay, etc...



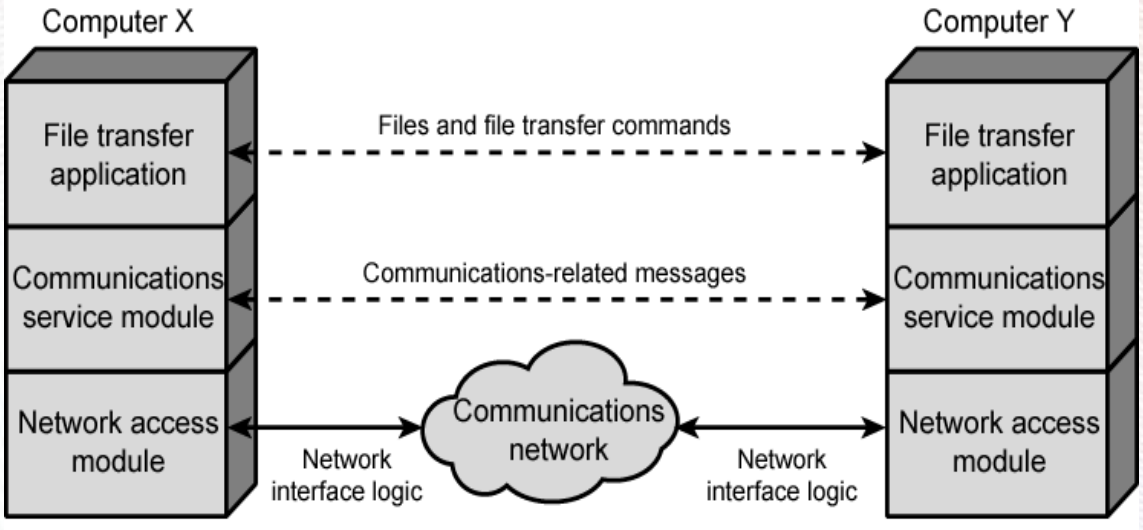
Transport Layer (communication services)

- Function: Reliable exchange of data (error and flow control, etc)
- Independent of network used (significance)
- Independent of application (serves all the higher-level applications- sharing of resources)

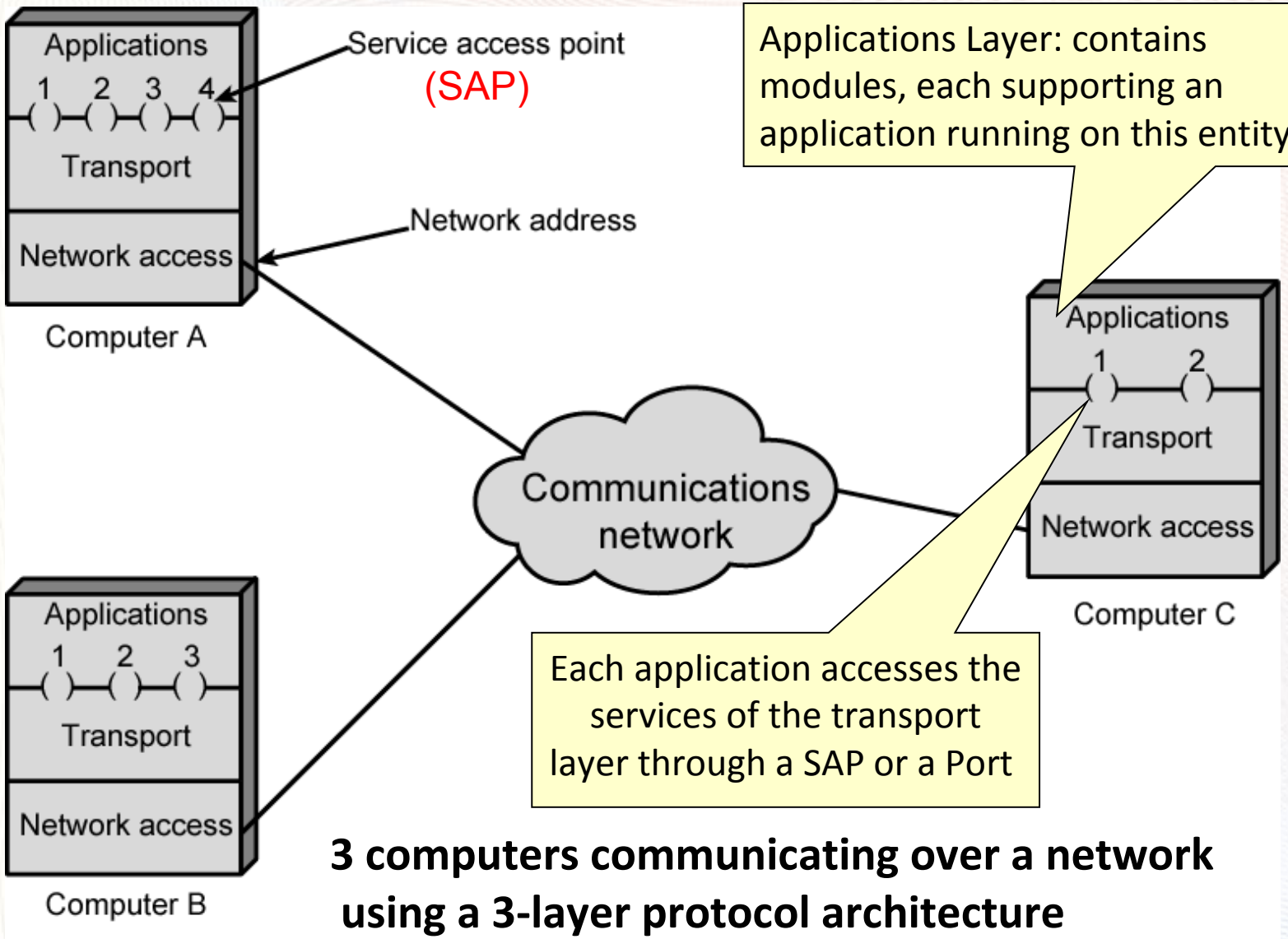


Application Layer

- Function: Supports different user applications running on the communicating entity.
- Example: e-mail, file transfer, web surfing, etc...



Protocol Architecture and Networks



3 computers communicating over a network using a 3-layer protocol architecture

Addressing Requirements

- **Two levels** of addressing:
  - **Each computer on the network needs a unique network computer address**
  - **Each application on a (multi-tasking) computer needs a unique application address within that computer**
- We identify the application on a computer by the SAP number it uses:
  - "Service Access Point or SAP", example SAP #3
  - "Port", example Port #80
- Each of the two addressing levels is handled only by the appropriate layer (information hiding):
  - **Computer address** is handled by the **Network Access layer**
  - **Application address** (SAP#) is handled by the **Transport layer**

Protocol Data Units (PDU)

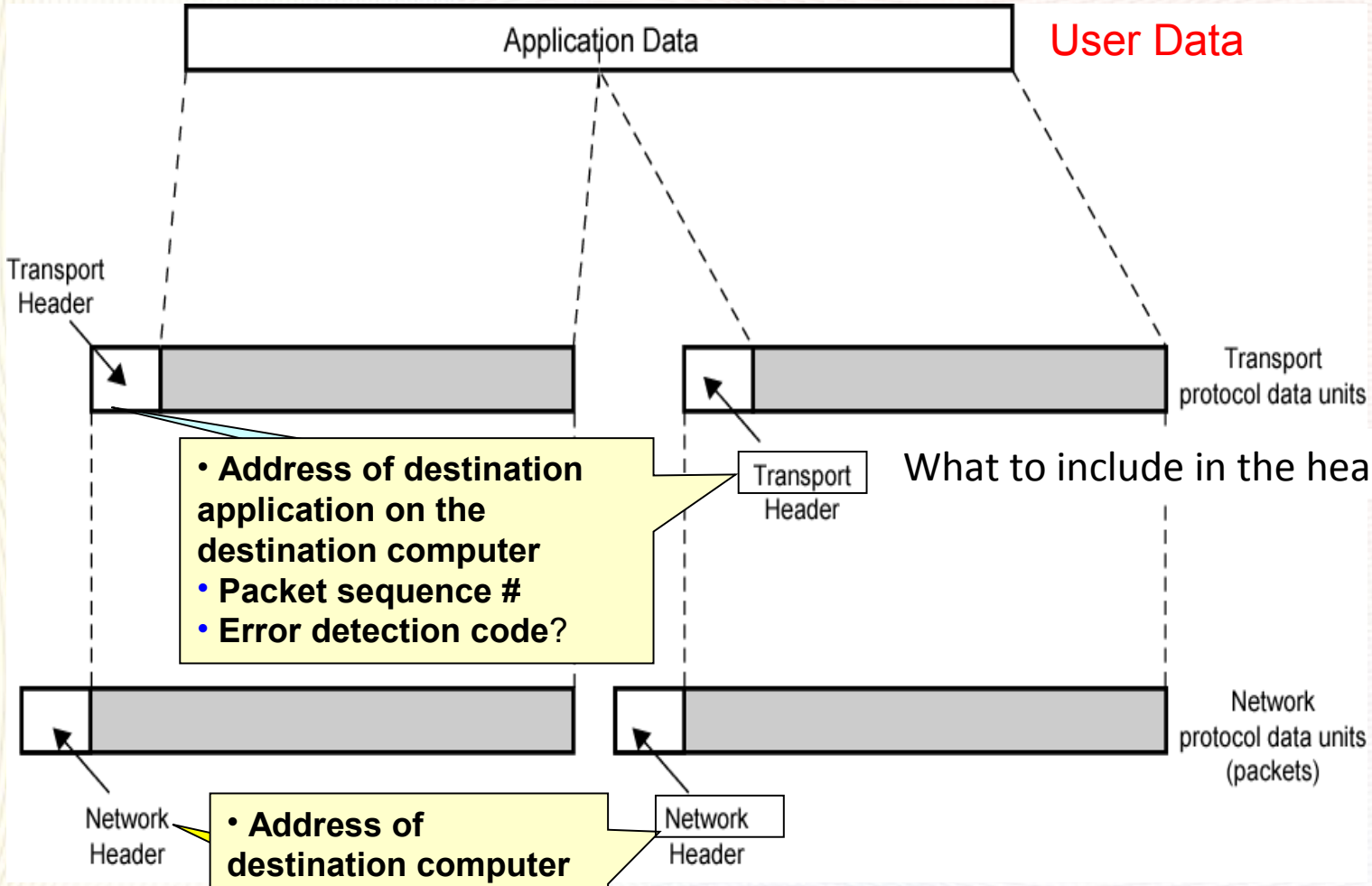
Peer layers communicate by exchanging **PDU messages**.

At source: Each layer adds its relevant control information (header) to the data it receives from a higher layer, thus forming its PDU, and pushes it down to the lower layer: **Encapsulation**

At destination: that PDU is handled by the corresponding peer layer: The relevant control header is removed, used, and the remaining part of the PDU is pushed up to the higher layer: O**pposite of encapsulation**

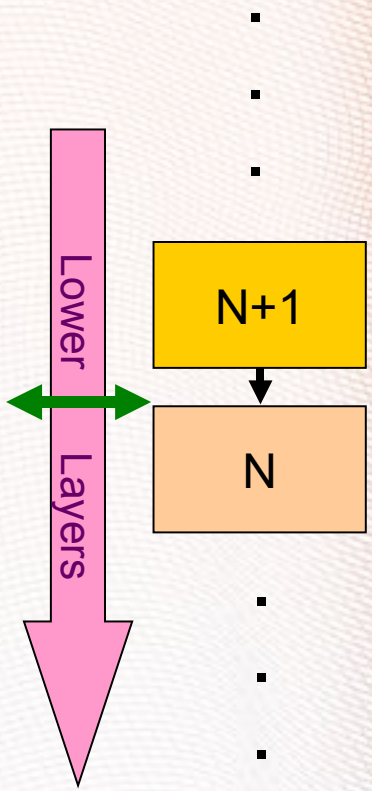
- Transport layer may fragment user data into smaller packets
- A transport header is appended to each packet, which would include:
  - Destination SAP
  - Sequence number of the data fragment
  - Possibly, error detection/correction code
- This makes the transport layer protocol data unit (PDU) or **segment**

Protocol Data Units (PDUs)



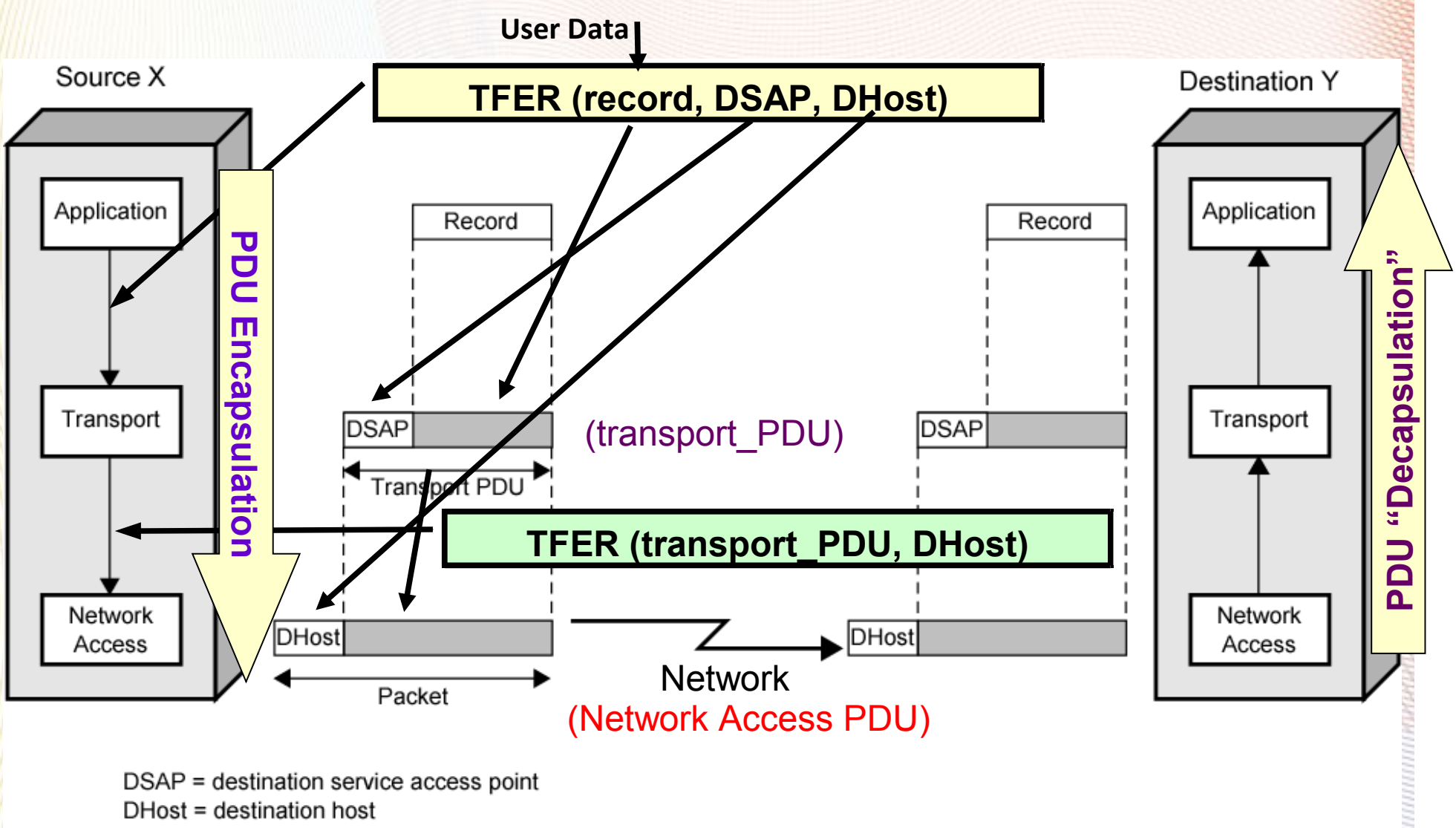
Generating PDUs at Source layers

- Layer N generates its PDU by appending a control part to data received from the next higher layer (N+1)
- Appended control part takes the form of a header
- That header will be used by the peer layer N on the destination entity
- Both the new PDU and header are labeled (identified) by the layer generating/using them, i.e.
- **Header (N) + PDU (N+1) = PDU (N)**
- **As PDU (N+1) = SDU (N), therefore**
- **Header (N) + SDU (N) = PDU (N)**



Source End System

Operation of a Protocol Architecture



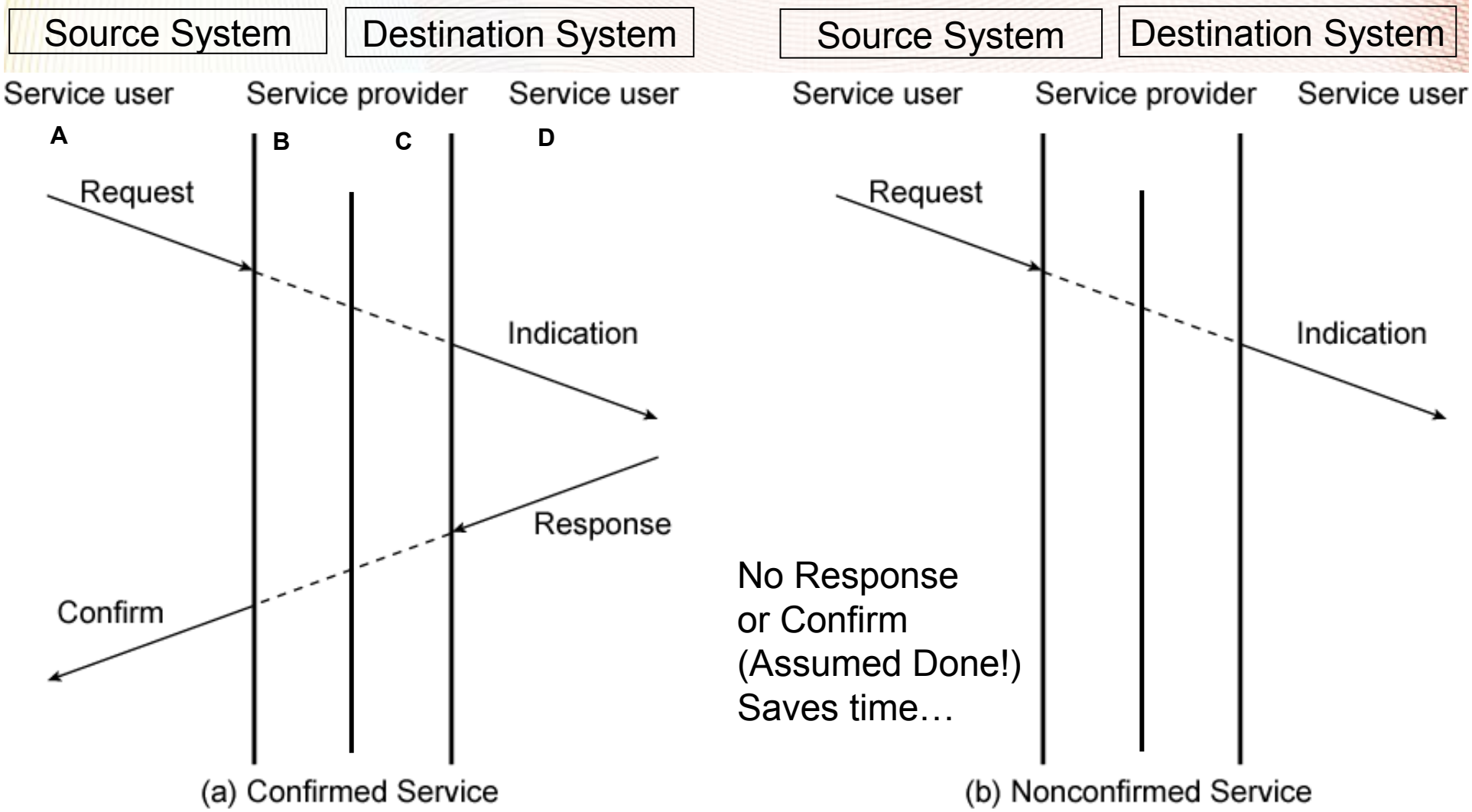
Standardizing Vertical Communication: Service Primitives and Parameters

- Services between vertically adjacent layers within the same system are expressed in terms of *primitives* and *parameters*
- *Primitives* specify the action to be performed
- *Parameters* pass data and control information required to perform the action

TFER (record, DSAP, DHost)

- Services are requested by a service *user* layer
- … and performed by a service *provider* layer

Timing Sequence for Service Primitives



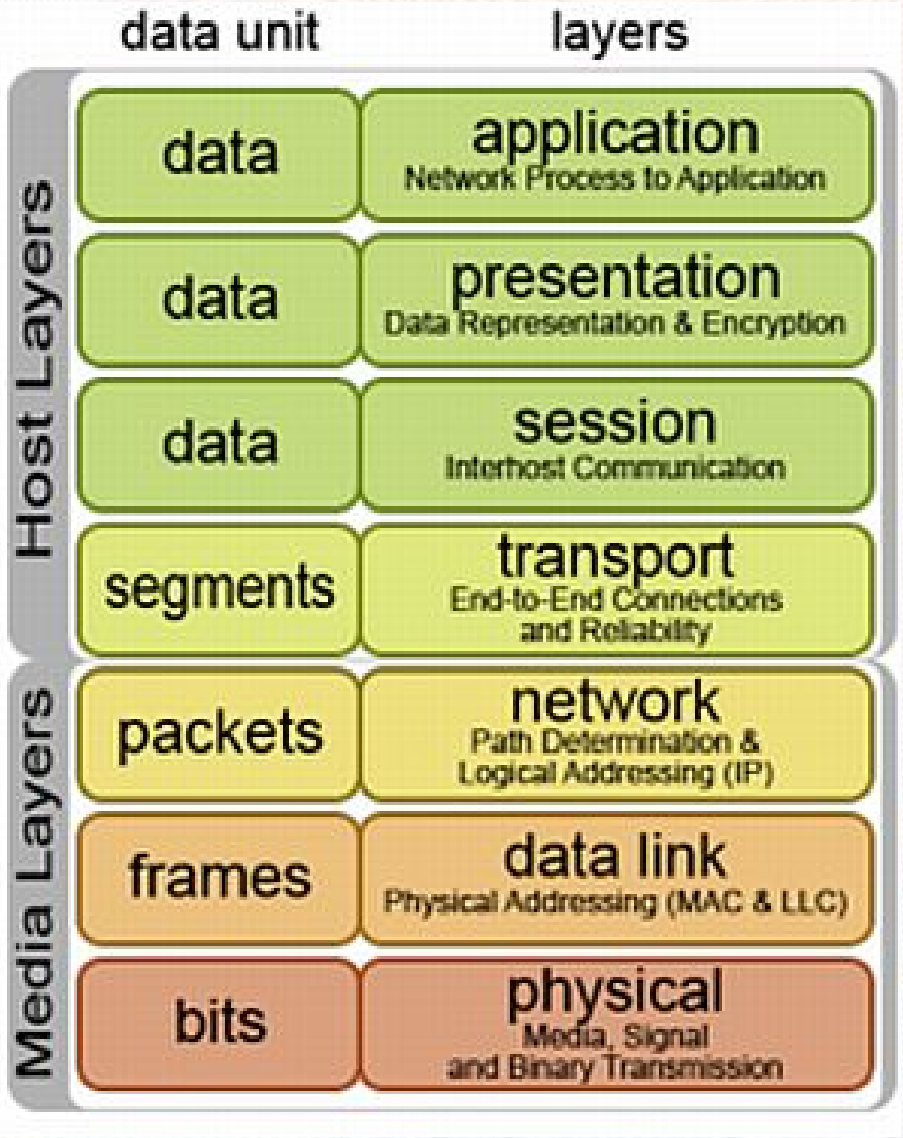
Standardized Protocol Architectures

- Required to allow devices from different manufacturers to communicate (interoperability)
- A win-win environment:
  - Helps different vendors market their products better
  - Helps customers shop around for best deals from different manufacturers (assured that equipment will work together)
- Main standard networking architectures:
  - OSI Reference model (X.200, 1977)
    - A theoretical system delivered too late! Never lived up to early promises
  - TCP/IP protocol suite (Developed for early forms of the internet)
    - (The Internet protocol) Most widely used: *de facto* Standard

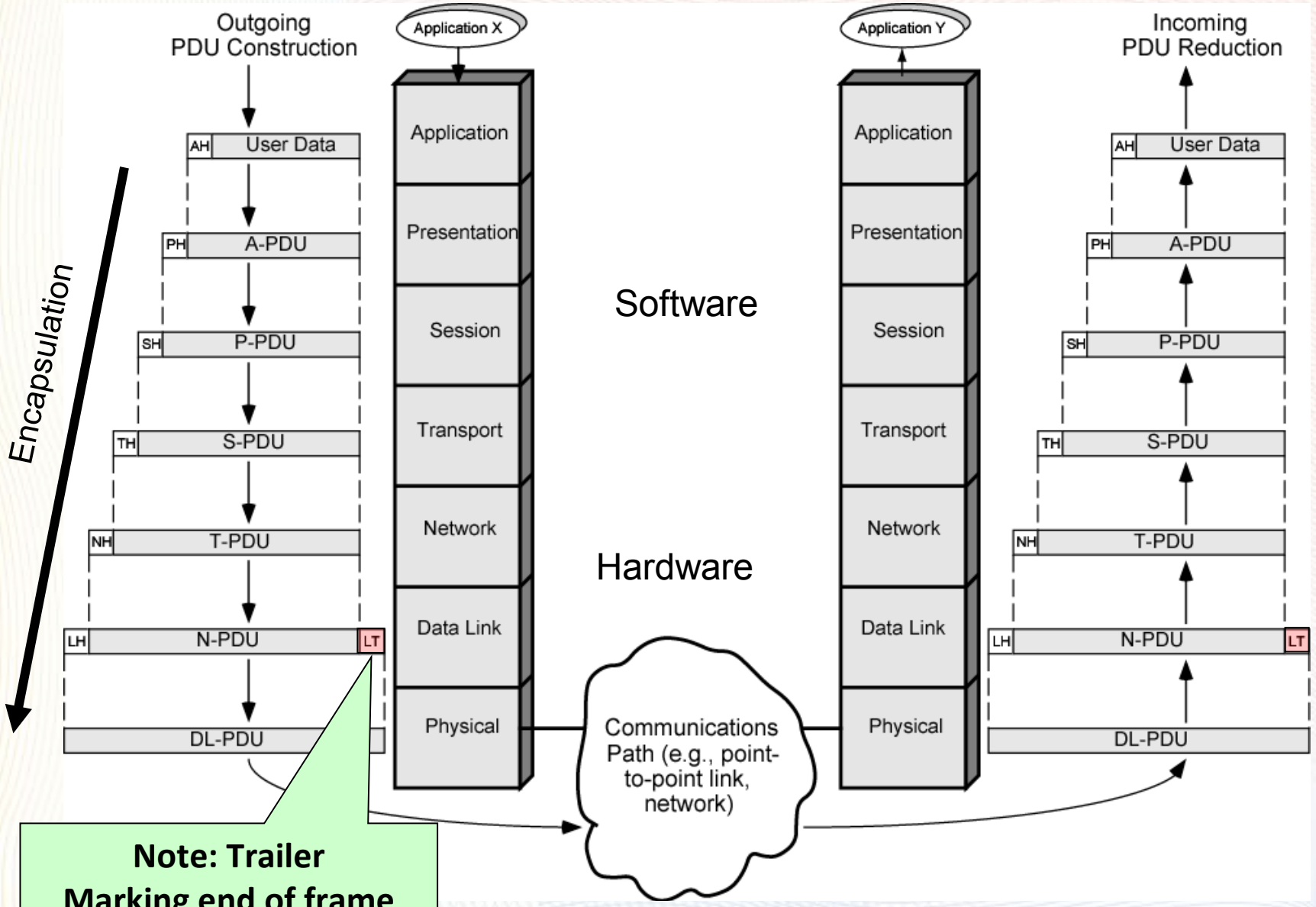
ISO:OSI Reference Model

- Developed by the International Standardization Organization (ISO) in 1977
- A 7-layer model
- Each layer:
  - performs a subset of the required communication functions
  - relies on the next lower layer to perform *more primitive* functions
  - provides services to the next higher layer
- Changes in one layer should not require changes in other layers

The ISO:OSI Model



Encapsulation within ISO:OSI



OSI Layers with examples

- **Physical** Example: EIA-RS-232, V.92, etc...

  Physical interface between devices
  - Mechanical: Connectors, etc.
  - Electrical: Bit representation, Voltage levels, Data rates,...
  - Functional: Functions of individual interface circuits
  - Procedural: Sequence of events for exchanging bit streams
- **Data Link** Example: HDLC (High level data link control)
  - Activating, maintaining and deactivating the data link (link management):
  - Reliable data transfer across the link: synchronization, error detection and control, flow control, so that higher layers may assume error free transmission over the link

OSI Layers with examples

- **Network** Example: Ipv4, Ipv6, IPX, etc...
  - Data transportation over a network
  - Relieves higher layers from worrying about underlying network details (routing, switching technology, etc)
  - Only needs info on destination host address and required facilities
  - Not needed (bypassed) on direct (point-to-point) connections
- **Transport** Example: TCP (Transmission Control Protocol)
  - Reliable exchange of data between end systems:

    (possibly across multiple networks!)
    - Without errors
    - Without loss
    - Without duplication
  - Allows different levels of Quality of Service (QoS): Regarding acceptable error rates, maximum delay, priority, and security

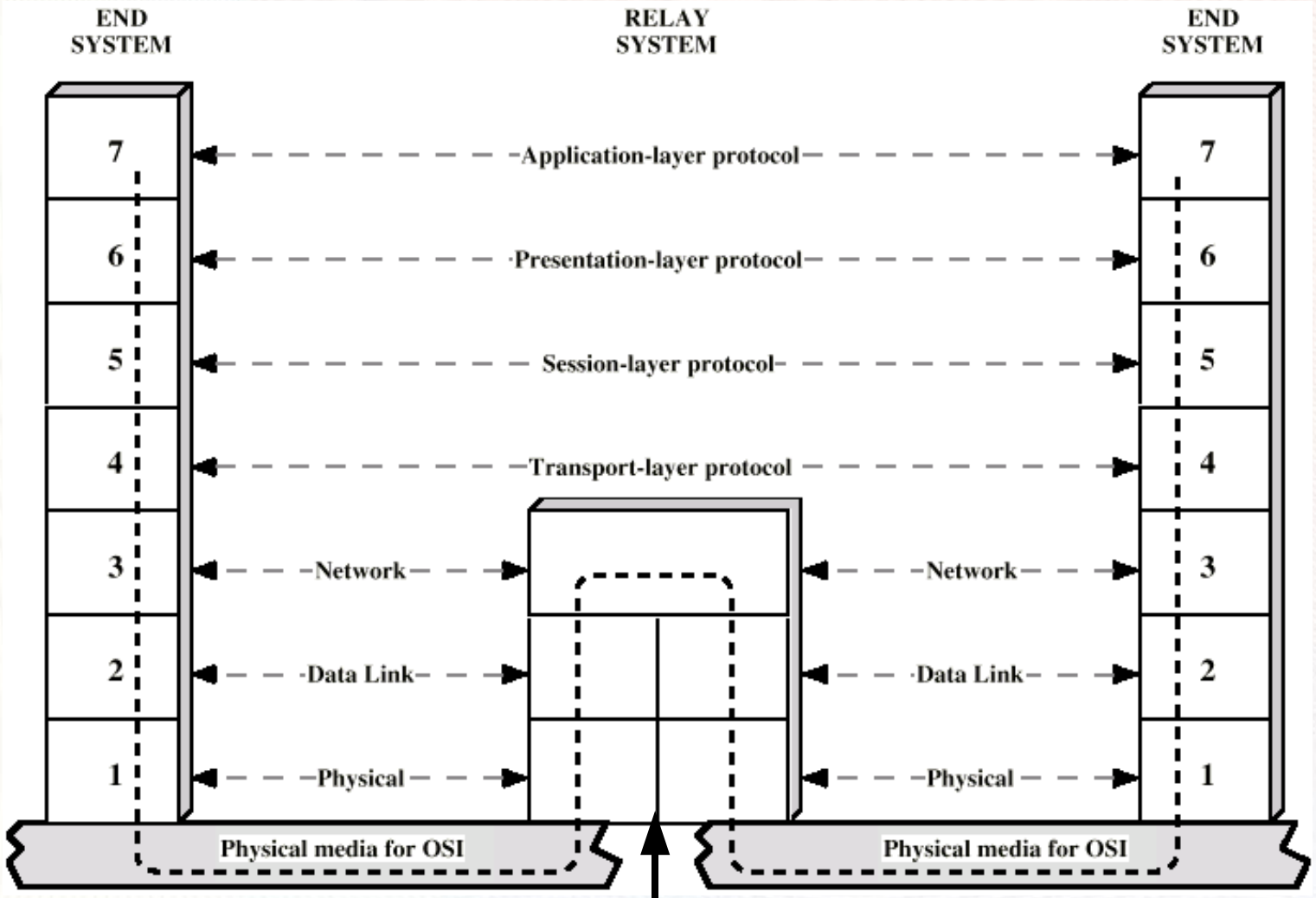
OSI Layers with examples

- **Session**
  - Control dialogues between applications running on end systems
  - Dialogue disciplines, e.g. full duplex or half duplex
  - Grouping: Marking of data to indicate different groups
  - Recovery: Use of data check points
- **Presentation**
  - Data formats and coding
  - Data compression
  - Data Encryption
- **Application**
  - Allows user applications running on a host to access the ISO:OSI environment
  - Support for e-mail, file transfer, terminal access to remote computers (telnet), web surfing, etc...

Do we always need all the 7 protocol layers?

- Only hosts at end points need to have all 7 layers
  - Point-to-Point on a single link:
    - Network layer is bypassed
  - Intermediate nodes in network are not interested in user data!:
  - Within the same network: An intermediate node needs only the bottom 2 layers (layer 2 switch)
  - Between multiple networks: A node linking two networks (router) needs only the bottom 3 layers

Layers for a Relay node in a network



(Intermediate network node (router) is only interested in the 3 lower layers)

TCP/IP Protocol Architecture

- Developed by the US Defense Advanced Research Project Agency (DARPA) for its packet switched network ARPANET (ancestor of the present Internet, 1966)
- Used today by the Internet
- Did not start as a formal model.. but as a working one
- Four (sometimes five) Layers:
  - Application layer
  - Host to host (end to end) Transport layer
  - Internet layer (multiple interconnected networks)
  - Network Interface layer (Data link + Physical layer)

ISO/OSI vs. TCP/IP