Communication Networks

MAP-TELE

2011/12

José Ruela

Network basic mechanisms

Network Architectures

Protocol Layering

Network architecture – concept

- A network architecture is an abstract model used to describe the organization and behaviour of a network and the systems that compose it
  - The model must be based on a set of general principles and define behaviour rules
  - The abstract nature of a network model allows describing concepts and relations between network components in a clear and concise way
- A network architecture must
  - Identify the functions required for the whole communication process
  - Organize the functions into components (modules) by grouping or decomposing functions based on their similarity (e.g., use of common mechanisms) or differences
  - Establish relations between the functional components (modules)
  - Define behaviour rules and relations between systems and their components for communication and cooperation purposes

Layered architectures

- A network architecture should not be based on a monolithic model
  - It is difficult to design and develop
  - It is difficult to maintain and change (e.g., to profit from technological evolution or to satisfy new requirements)
  - It is not flexible (difficult to apply to different cases or to adapt to new communication scenarios)
- Instead, a modular approach, consisting in decomposing a global and complex problem into simple and more tractable problems, allows handling the whole communication process in a systematic, flexible and adaptable way
- Architectural models so far adopted in networks are based on organizing functions in modules and structuring them hierarchically, which results in so-called *Layered Architectures*

Principles of layered architectures

- The basic principle of *Layered Architectures* is layer independence
  - A layer hides implementation details from other layers – the functions it realizes are encapsulated and only the service it provides is visible (but not the details of how it is realized)
- Adjacent layers communicate (interact) through an interface (*service interface*)
  - A layer provides a service to the upper layer through a (service) interface
- A layer uses the service provided by the layer below to perform its own functions, thus adding value to the service it provides to the layer above

Advantages of layered architectures

- Reduction of complexity in design, development and maintenance
- Possibility of independent developments of each layer (provided that interfaces are well defined)
- Flexibility in implementation (choice of technologies and mechanisms best suited to each function)
- Possibility of making changes on a layer (to change or improve an algorithm or exploit a new technology) without impacting the other layers
- Possibility of supporting a variety of applications based on a reduced number of common interfaces / services
- Design and analysis of systems with different levels of abstraction
- Development and adoption of standards is easier, thus favouring mass production and ample support from manufacturers (more choices and more flexible solutions to users, at lower cost)

but

- Information hiding may cause function mismatches (insufficient information may lead to duplication of functions or wrong assumptions) and ultimately to performance degradation – *cross-layer* design is an attempt to overcome such limitations

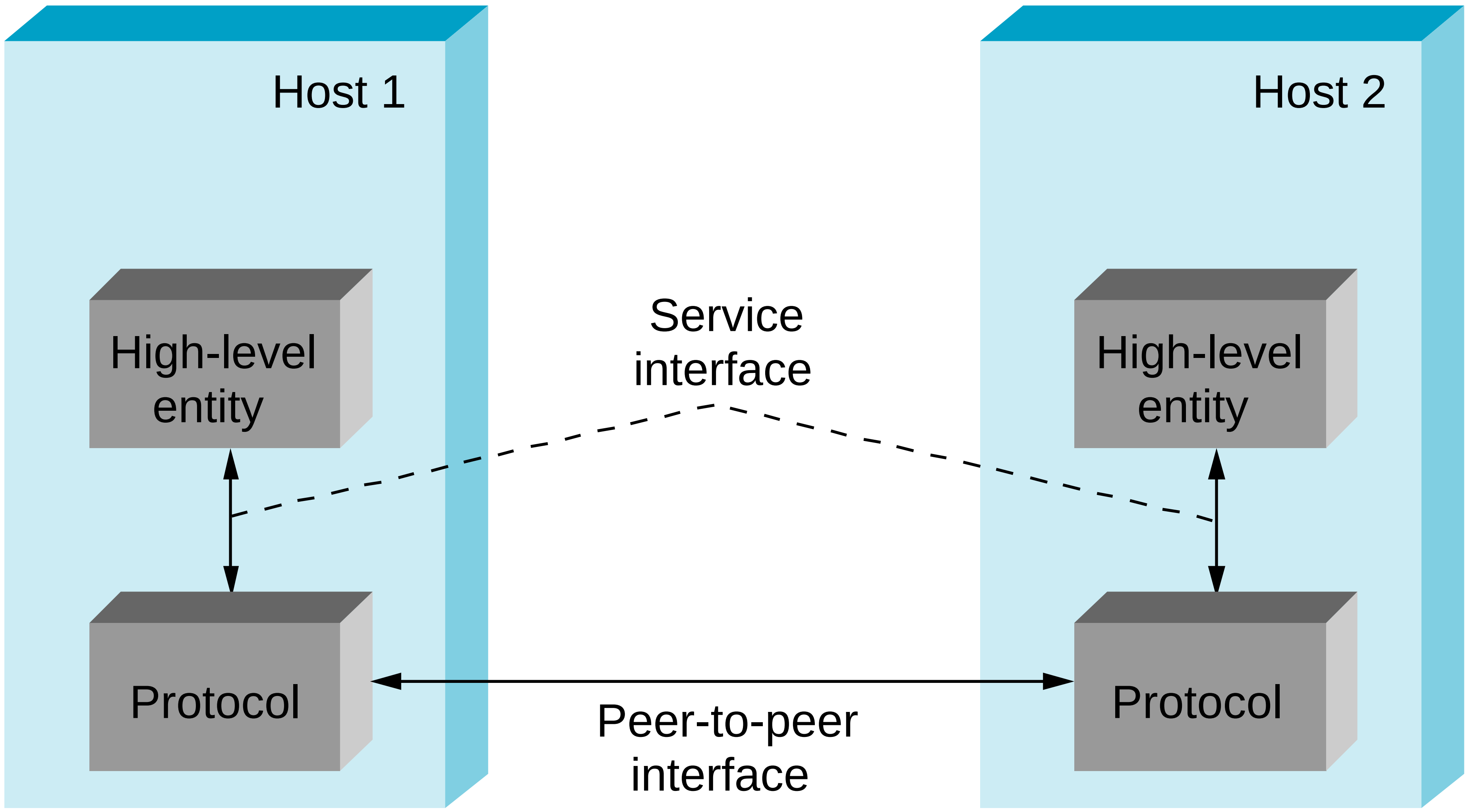
Protocols

- In a layered architecture, the systems that constitute the network are organized in layers – a layer crosses (horizontally) multiple systems
- Each layer is composed of entities (processes, resources) that implement the corresponding layer functions
- *Peer entities* (residing on the same layer on different systems) must cooperate to build the service provided by the layer
  - This requires the exchange of control and synchronization messages (besides data messages that are the ultimate goal of communication) based on well defined rules – a *protocol*
- A protocol is the set of rules that govern the communication between peer entities (usually on different systems)
  - A protocol defines a *peer interface* between such entities
- A protocol must define the syntax (formats) and semantics (meaning) of messages, provide synchronization mechanisms (to enforce a required temporal behaviour) and specify the actions (procedures) to carry out, on the occurrence of events, depending on the state

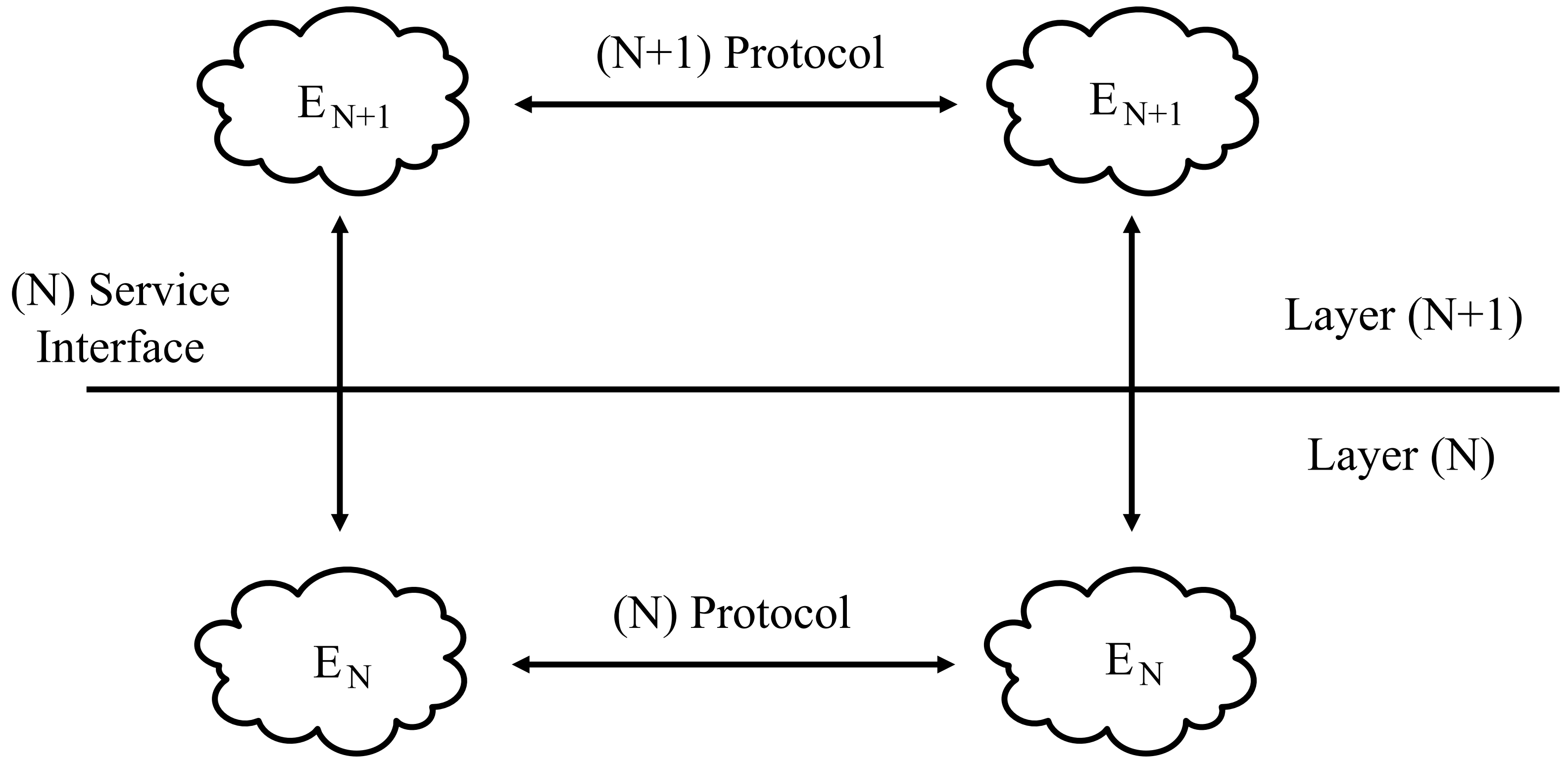
Protocol stack

- In a layered architecture, protocols are layered as well, that is, a protocol stack exists on each system
  - On each system, layers interact through service interfaces along the protocol stack (vertical view) – this corresponds to the real flow of information (up and down the stack) on each system
  - The communication between systems is the result of the communication that occurs on each layer (horizontal view) – this communication is logical (virtual) since it uses services provided by lower layers
- End-systems, where the users' applications reside, must implement a complete protocol stack
- Intermediate systems (network nodes) only have to implement a subset of the protocol stack, corresponding to those functions related with the transport of packets across a network, which include moving frames between adjacent systems and transmitting sequences of raw bits over transmission links
- Higher layer protocols (application oriented) are therefore executed end-to-end, while lower layer protocols (communication oriented) are executed on a node-to-node (hop-by-hop) basis

Protocol and service interfaces

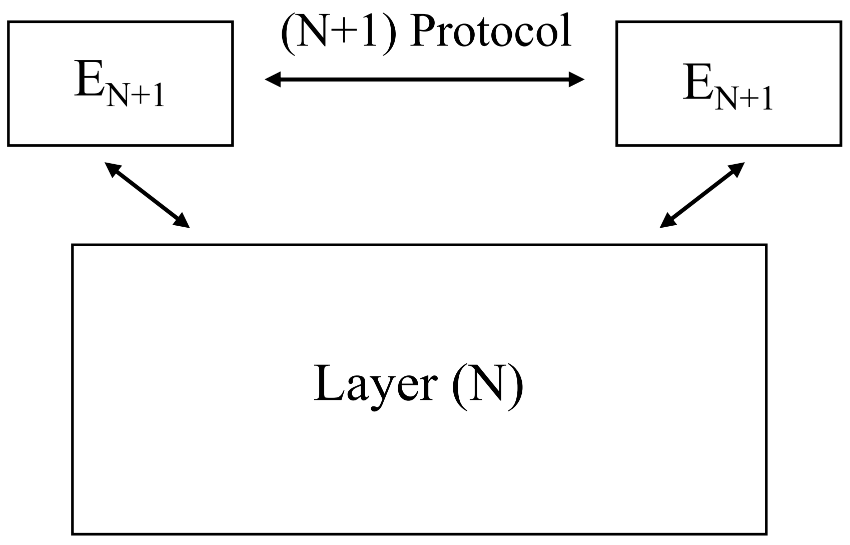


Communication model

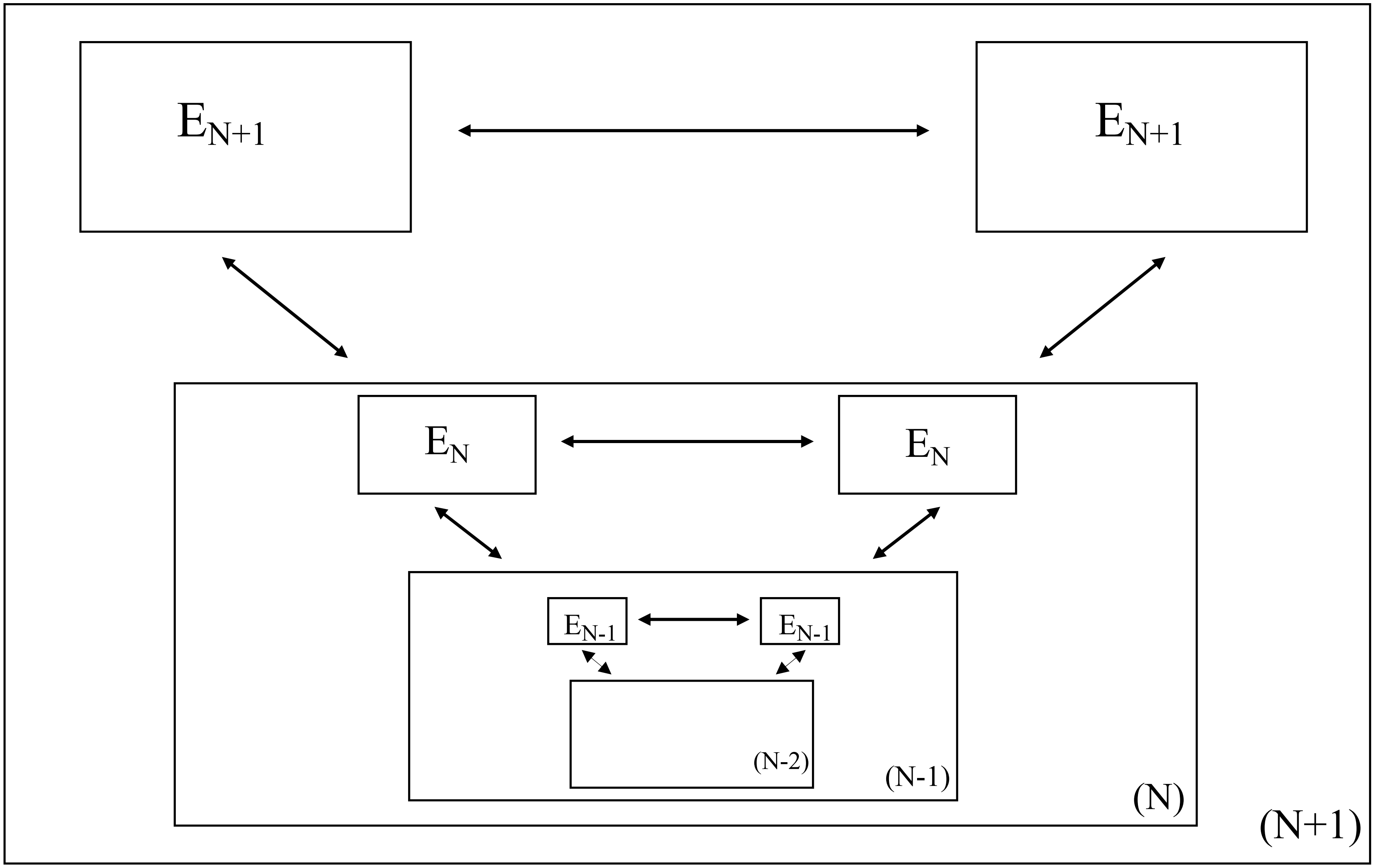


Service

- (N+1) entities communicate by means of an (N+1) Protocol, using the service provided by Layer (N) through the service interface between layers (N) and (N+1)
- The way the Layer (N) service is realized is hidden from Layer (N+1)
- Services are recursively used by the layers



Recursive use of services



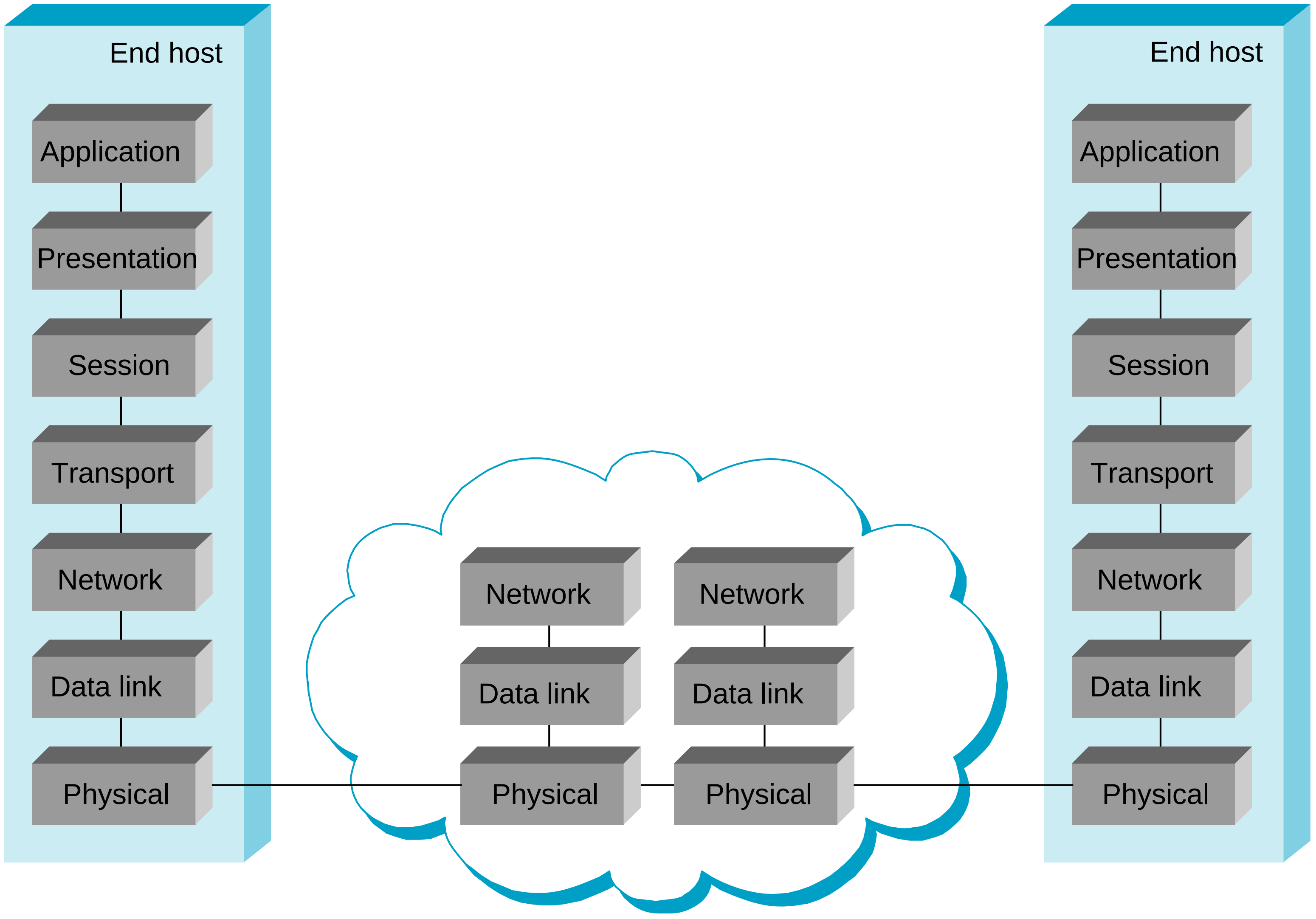
Need for a reference model

- In the early days of computer communications, the main computer manufacturers developed proprietary network architectures that soon proved to be totally incompatible (even if based on similar concepts and principles)
- This was in the first place an obstacle to the use of computers of different manufacturers on corporate networks and to the possibility of wide and open communications among computers of different organizations
- The offer of the first public packet switched data services (such as X.25) by many telecom operators (mainly in Europe) and the initial steps that ultimately led to the development of the Internet raised the necessity of a framework that could help the creation of an open communication environment
- This was the goal of the Open Systems Interconnection (OSI) Reference Model, developed by ISO (International Organization for Standards)

OSI reference model

- The OSI reference model defines general rules of interaction among *open systems*, that is, systems that obey universal rules for communication (as opposed to closed or proprietary systems) and whose external behaviour is in conformity with the rules prescribed by the model
- The OSI model creates the conditions for the specification and development of standards, which may be approved by recognized organizations (ITU, ISO, IEEE) or accepted as *de facto* standards by industry and users (but standards are not part of the model)
- The model defines principles, concepts and relations between components – it is essentially an abstract model for describing the interaction between systems (and not an implementation guide)
- The model is general and flexible, although it was developed in the framework of the networking environment of the 1970's, and thus did not take into account some scenarios that emerged at later stages (e.g., Local Area Networks, internetworking, service integration, etc.)

OSI seven layers



One or more nodes
within the network

OSI seven layers

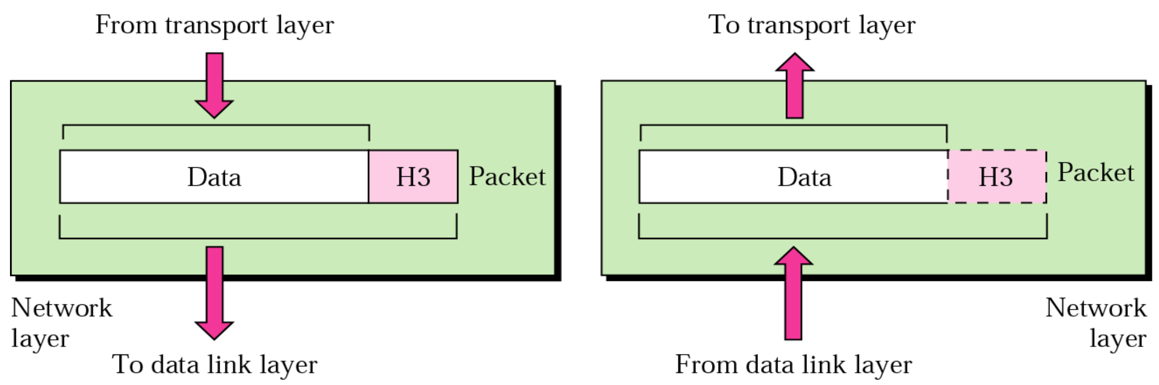
- Application
  - Provides services and creates the environment for the execution of applications (deals with the semantic aspects of communication)
- Presentation
  - Deals with the representation of information independently of content (syntactic aspects aimed at solving syntactic differences, negotiation of transfer syntax, data encryption, etc.)
- Session
  - Controls the dialog between processes (e.g., provision of a full-duplex service) and provides synchronization mechanisms at process level
- Transport
  - Provides an end-to-end information transfer service, independent of the network service, with support for the multiplexing of traffic flows (hides the details and differences of real networks from the applications)
  - The service may be reliable or unreliable, and may include flow and congestion control functions

OSI seven layers

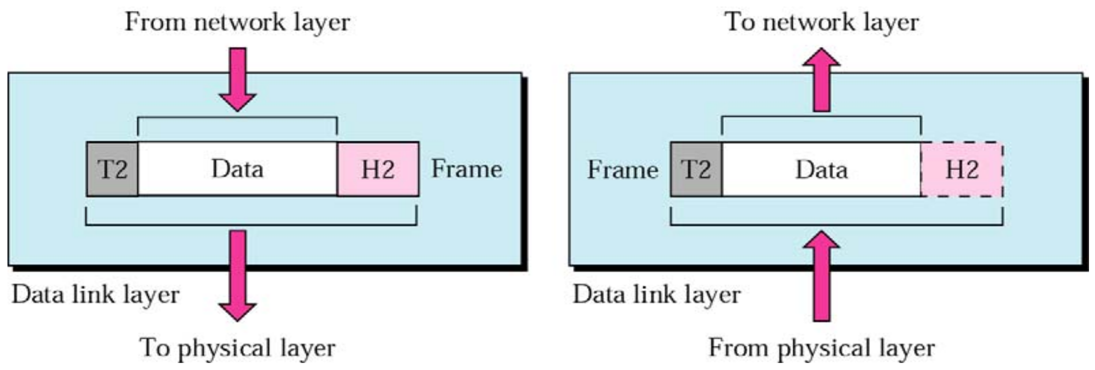
- Network
  - Provides the transfer (switching and routing) of data units (packets) within each network and across different networks (internetworking), thus providing a communications service to end-systems
- Data Link
  - Organizes the basic units for data communications (frames) out of a sequence of raw bits and manages the transfer of frames over a transmission link, which may include establishment of logical connections, error and flow control
- Physical
  - Provides a bit level transport service, dealing with the physical interface to the transmission medium (connectors, signal levels, transmission codes, bit-level synchronization, etc.)

The lower communications layers

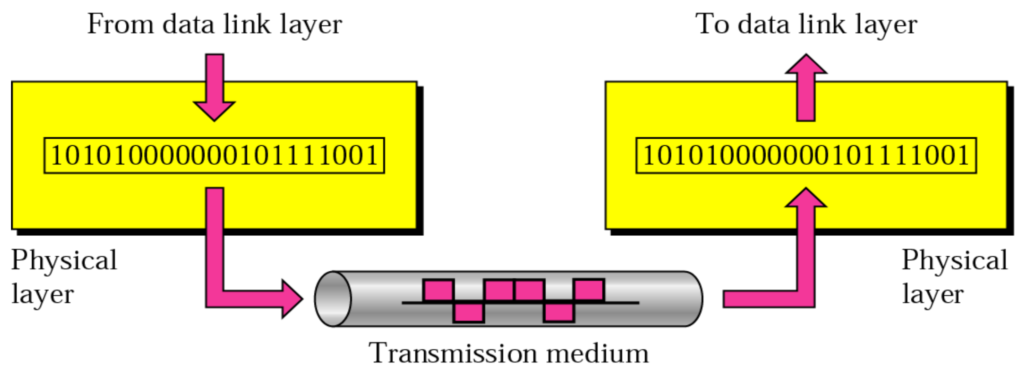
- Delivery of packets from the original source to the final destination, possibly through a number of intermediate nodes



- Transmission of frames from one node to the next

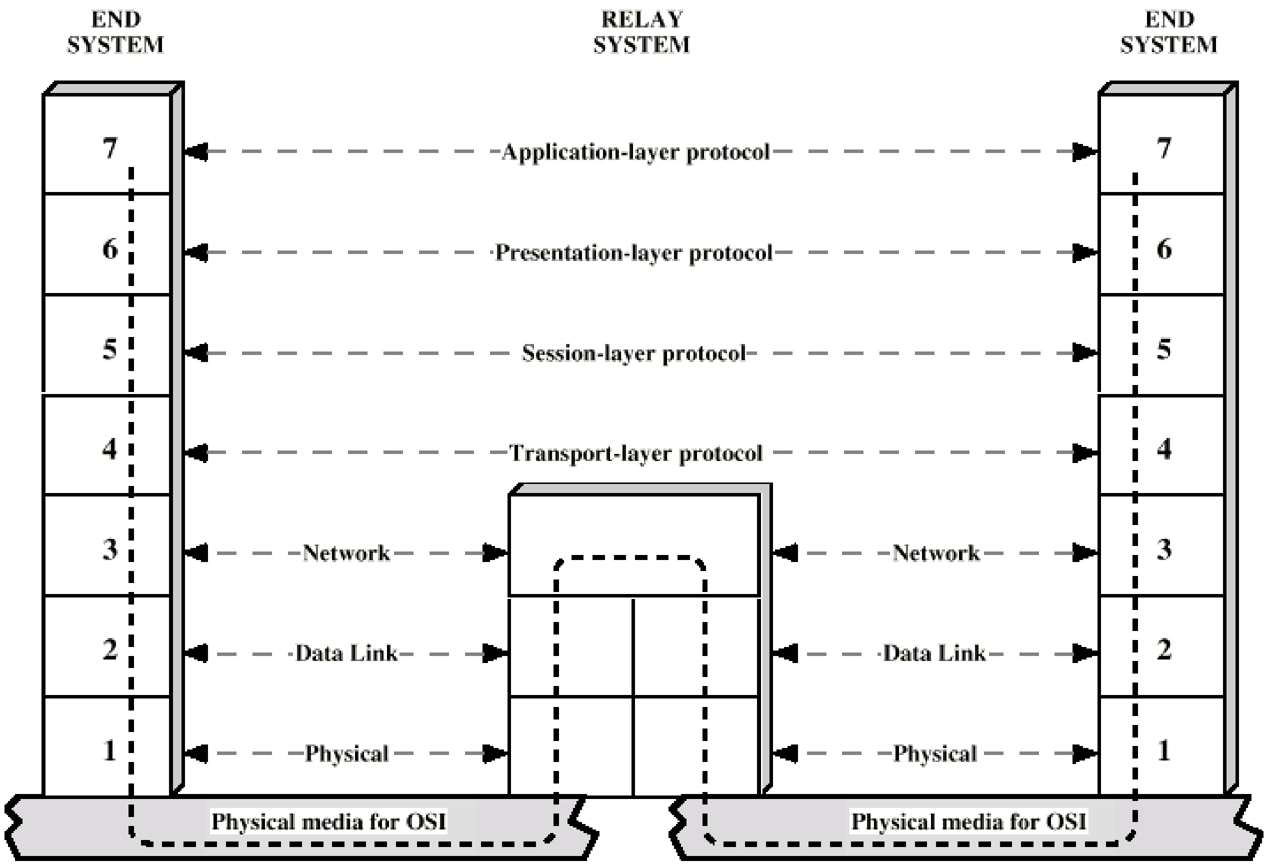


- Transmission of individual bits from one node to the next



Data flow and protocol communication

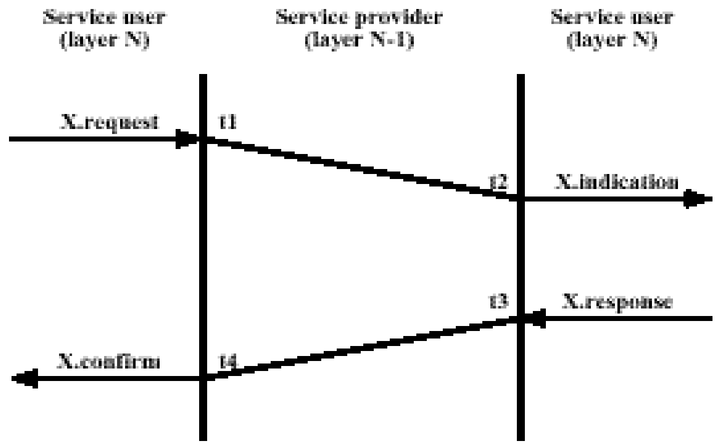
- The figure illustrates the real flow of data through service interfaces across the OSI protocol stack (vertical) and the logical communication that takes place within each protocol layer (horizontal)
- It also illustrates hop-by-hop and end-to-end protocols



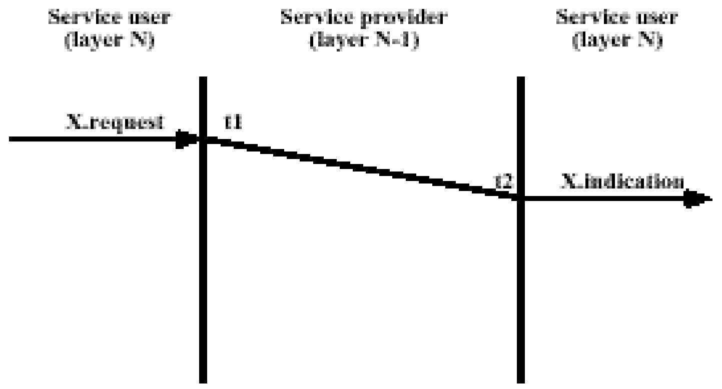
Description of services – service primitives

- A service is described in an abstract way as a set of capabilities offered by a layer (*service provider*) to the layer above (*service user*)
- A service (i.e., its attributes) is described by means of *service primitives*
  - *Request*
    - Invoked by a service user
    - Invokes a service (action, procedure) specified by means of parameters
  - *Indication*
    - Invoked by a service provider
    - Indicates that a service has been invoked by a peer user or notifies the service user of an event or an action initiated by the service provider
  - *Response*
    - Invoked by a service user
    - Response to an *Indication*
  - *Confirmation*
    - Invoked by a service provider
    - Indicates that a service invoked by a user (by means of a *Request*) was triggered or has completed

Examples



(a) Confirmed Service



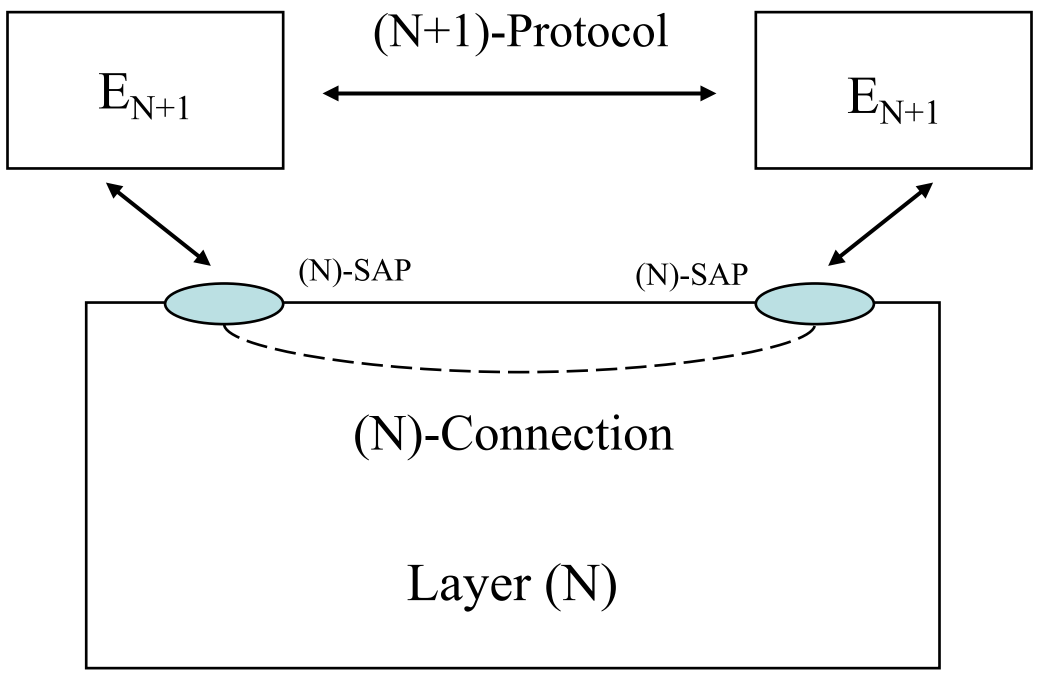
(b) Nonconfirmed Service

Concepts

- Service Access Points
  - A *Service Access Point* (SAP) is the logical interface between two layers
  - A layer provides a service to the layer above on a SAP
- Connections
  - A connection is an association established by a layer protocol for the transfer of data units between entities of the layer above
  - A connection is a logical association between SAPs
- Data Units
  - Data units transferred across the interface between layers (service interface) are called *Service Data Units* (SDU)
  - Data units transferred between peer entities (protocol interface) are called *Protocol Data Units* (PDU) – frames and packets are examples of PDUs
- Encapsulation
  - A layer encapsulates SDUs received across the interface with the layer above, thus creating PDUs that contain *Protocol Control Information* (PCI) required for the operation of the layer protocol
  - It is possible to establish different relations between SDUs and PDUs

SAPs and connections

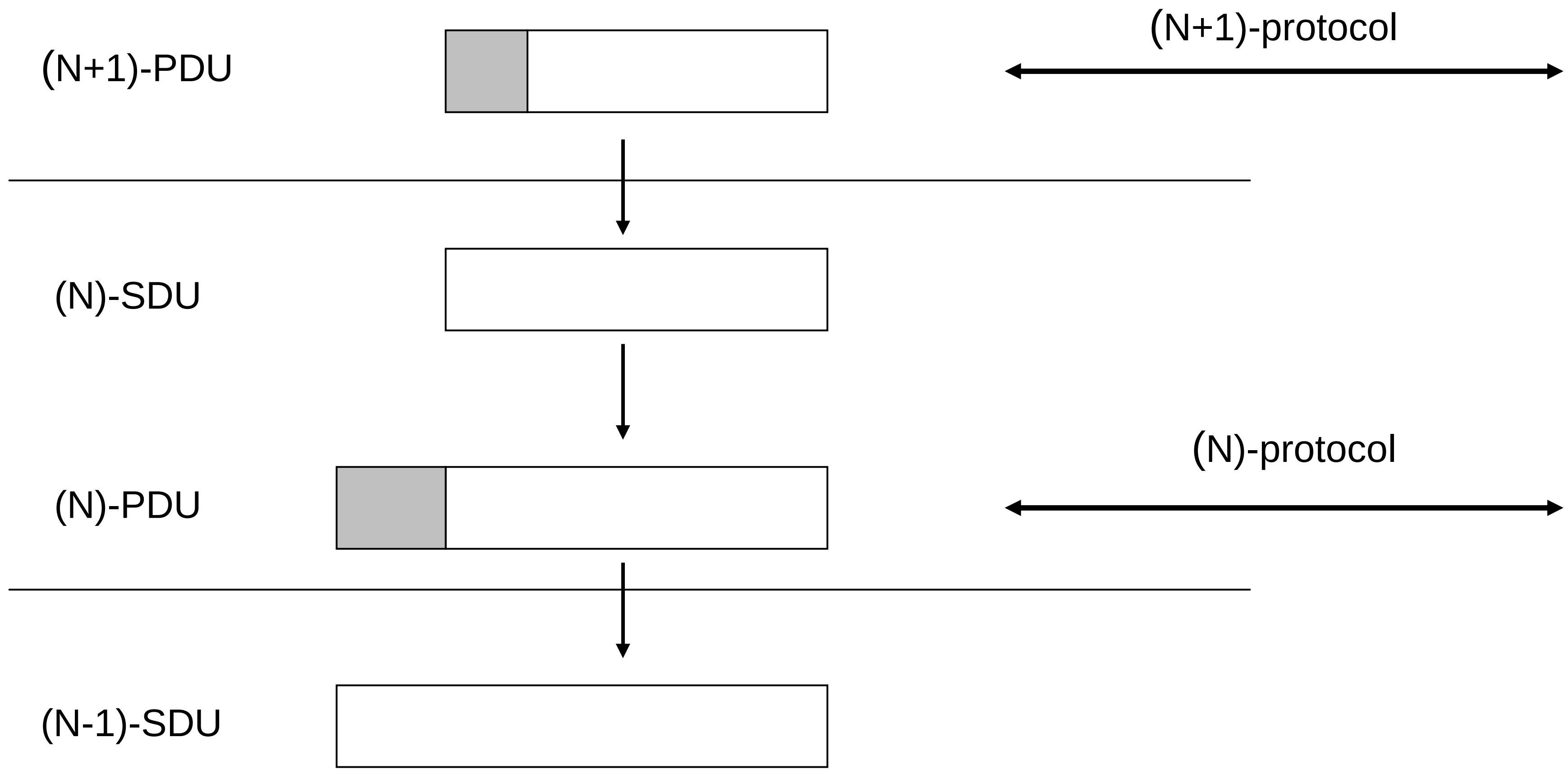
- An (N)-Connection is a service provided by Layer (N) to entities of Layer (N+1)



Protocol Data Units and Service Data Units

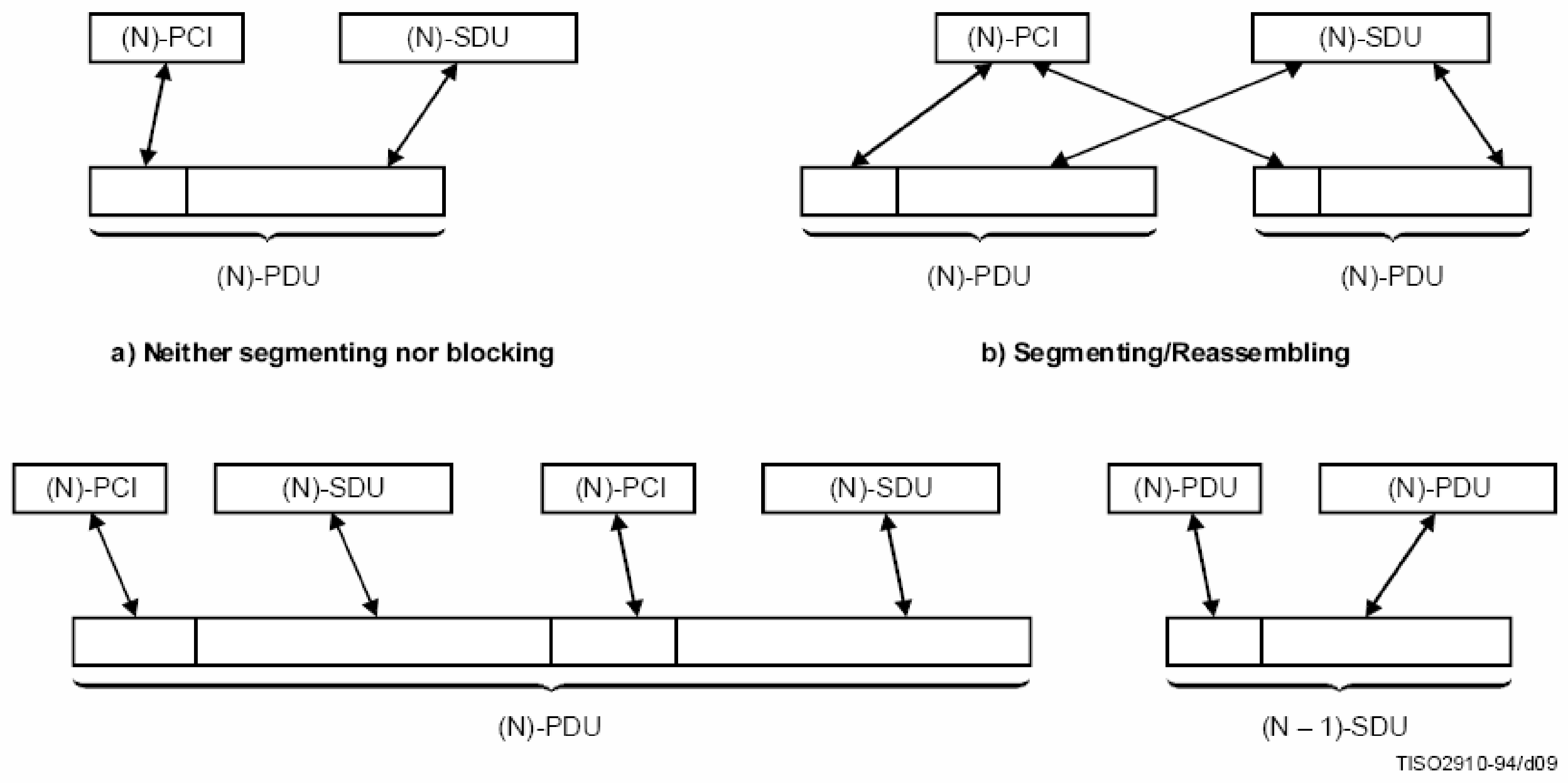
- In the basic case, one (N)-SDU is encapsulated with (N)-PCI to form a single (N)-PDU
- A single (N)-SDU may be segmented and encapsulated in multiple (N)-PDUs
  - It is necessary to provide information on the (N)-PCI of each (N)-PDU to reassemble the original (N)-SDU at the destination peer
  - This is called *Segmentation and Reassembly*
- Multiple (N)-SDUs may be combined and carried on a single (N)-PDU
  - It is necessary to provide information on the (N)-PCI of the (N)-PDU to extract the original (N)-SDUs at the destination peer
  - This is called *Blocking and Deblocking*
- In both cases, the operations take place within a single layer, unlike a third case that consists in combining multiple (N)-PDUs to form a single (N-1)-SDU
  - This is called *Concatenation and Separation*

PDUs and SDUs – basic encapsulation



▨ PCI – Protocol Control Information

Relations between PDUs and SDUs – examples



a) Neither segmenting nor blocking

b) Segmenting/Reassembling

c) Blocking/Deblocking

d) Concatenation/Separation

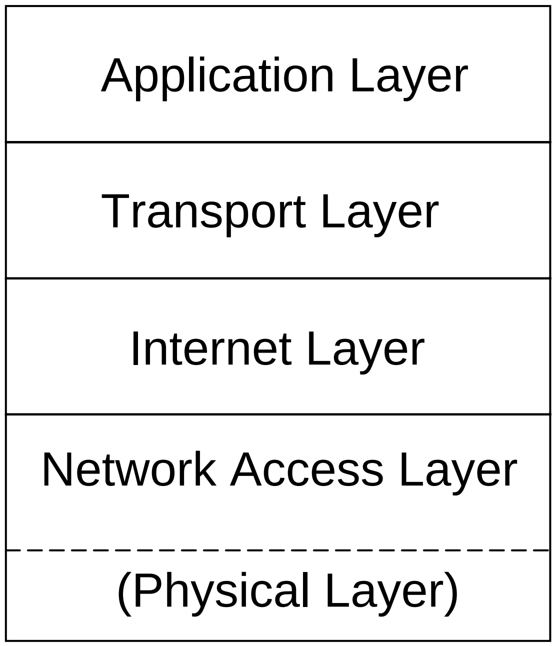
SDU	Service-data-unit
PCI	Protocol-control-information
PDU	Protocol-data-unit

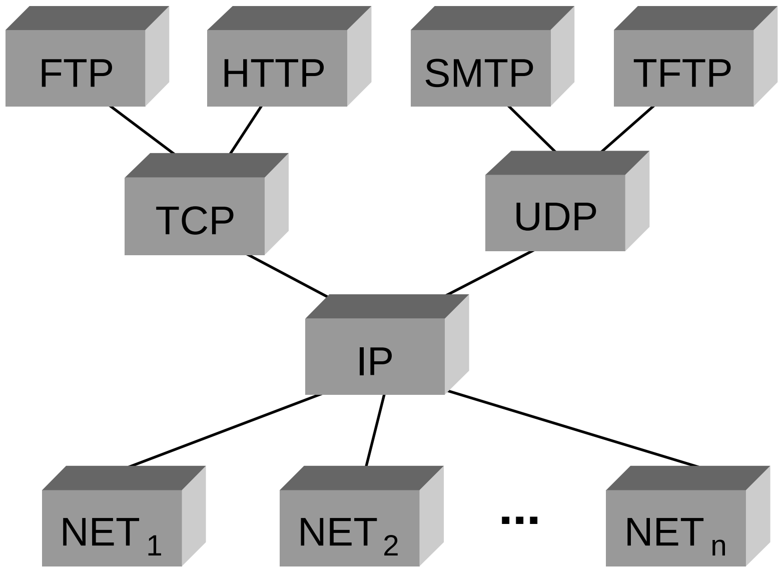
Connection oriented and connectionless services

- A protocol layer may provide connection oriented (CO) services, connectionless services (CL) or both
- A CO service provided by a given layer may be built on top of a CO or a CL service
  - CO over CO – ATM VC over a physical circuit
  - CO over CL – TCP over IP
- A CL service provided by a given layer may be built on top of a CO or a CL service
  - CL over CO – IP over ATM, IP over PPP
  - CL over CL – UDP over IP, IP over MAC (LAN)
- It is possible to map connections on adjacent layers in different ways
  - One to one
  - Multiplexing – multiple connections on a layer share a lower layer connection
  - Splitting or Inverse Multiplexing – one connection on a layer uses multiple lower layer connections

Internet architecture

- The Internet architecture, also known as TCP/IP architecture (from its two nuclear protocols), evolved from research and experimental work in early Packet Switched networks, like the ARPANET
- A number of protocols have been developed and improved along the years and form a protocol stack that may be described by means of layers (although there is not a unanimous view on its representation)

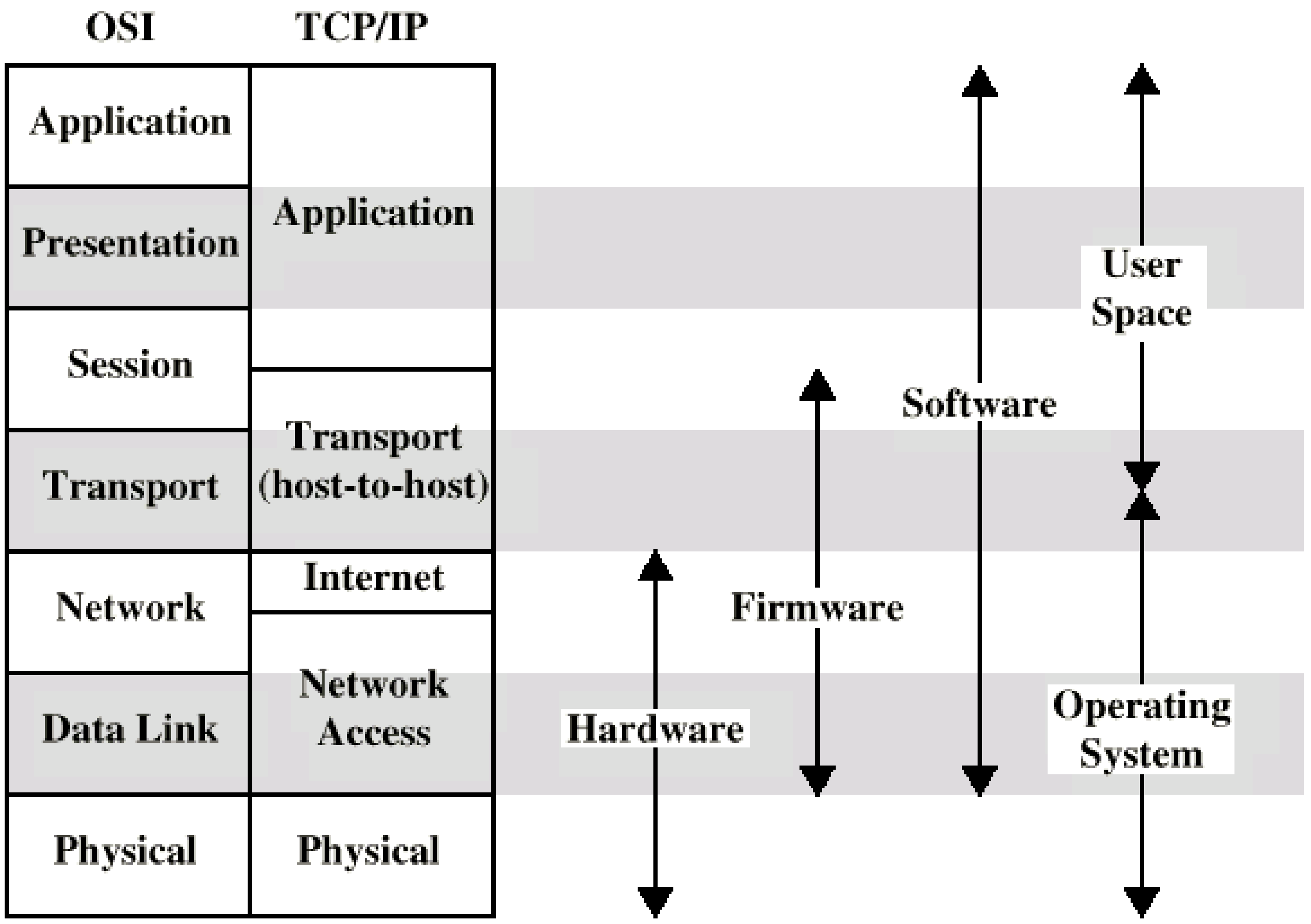




Internet layers

- Application Layer – provides services to users (e.g., client-server model)
  - Examples: Telnet, FTP, HTTP
  - It covers the functions of the three OSI higher layers
- Transport Layer – provides an end-to-end transport service, independent of the network service, with support for multiplexing of application flows
  - Examples: TCP (reliable), UDP (unreliable)
  - It corresponds to the OSI Transport layer
- Internet Layer – provides the transfer of packets between end-systems on the same or different networks (*internetworking*)
  - IP (Internet Protocol)
  - It includes some functions of the OSI Network Layer
- Network Access Layer – provides access to "physical" networks
  - IP runs virtually over any network technology
  - This "layer" provides functions of the Network, Data Link and Physical layers
  - In practical terms, IP uses a Data Link service – possible examples are PPP (on point to point links), a LAN service or even a VC service, such as ATM (from the IP point of view, a VC service is equivalent to a Data Link service and, for this reason, it is sometimes called a pseudo-link)

OSI versus TCP/IP


OSI
TCP/IP
Application
Presentation
Session
Transport
Network
Data Link
Physical
Application
Transport (host-to-host)
Internet
Network Access
Physical
Hardware
Firmware
Software
User Space
Operating System