Technische Universität Berlin

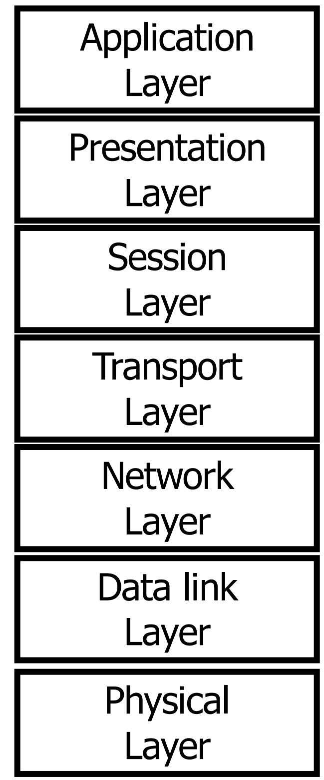
Computer Networks

Protocols

Chapter

1. Introduction
2. **Protocols**
   - **Protocol stacks**
   - **Internet protocol stack**
3. Application layer
4. Web services
5. Distributed hash tables
6. Time synchronization
7. Transport layer
8. UDP and TCP
9. TCP performance
10. Network layer
11. Internet protocol
12. Data link layer

Top-Down-Approach



Protocols

Protocols

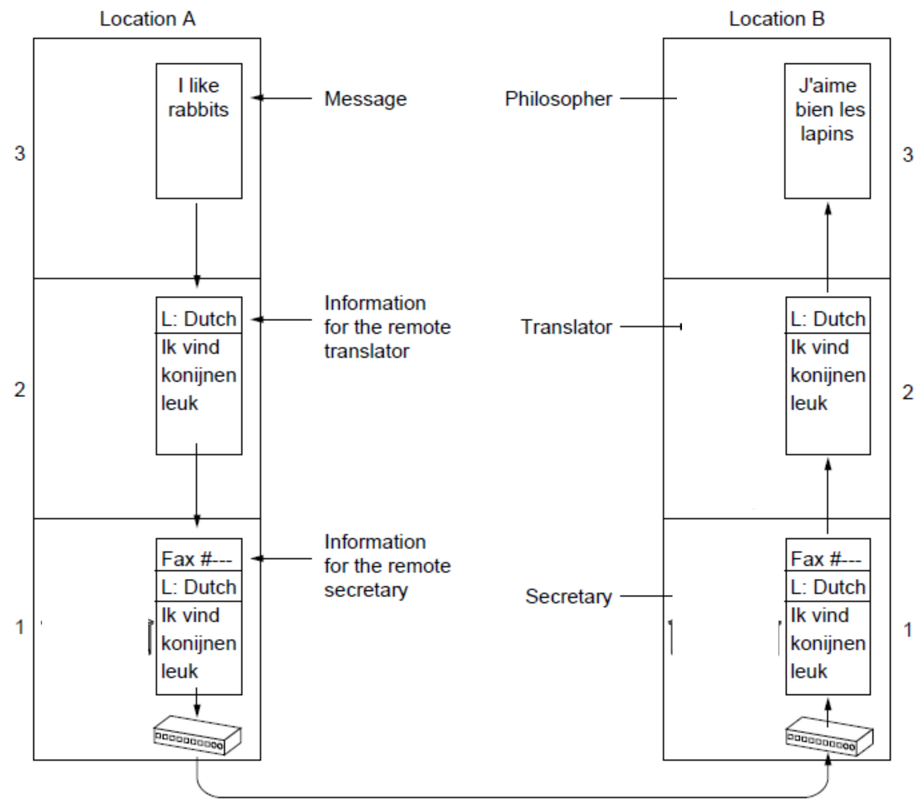
- Computer networks are quite complex
  - End systems, switches, router, network interface cards (NIC), cables, channels, connections
  - Messages
  - Mechanisms for error control, flow control, congestion control, addressing, routing, forwarding, medium access, …
- Protocols
  - define **message format** and **behavior** of communicating nodes
  - Example: Hypertext Transfer Protocol (HTTP)
    - HTTP client requests content from HTTP server
    - Two types of messages: request and response
    - Well-defined message formats
    - Well-defined behavior of HTTP client and HTTP server

Analogy: The philosopher-assistant-clerk architecture

Philosophers

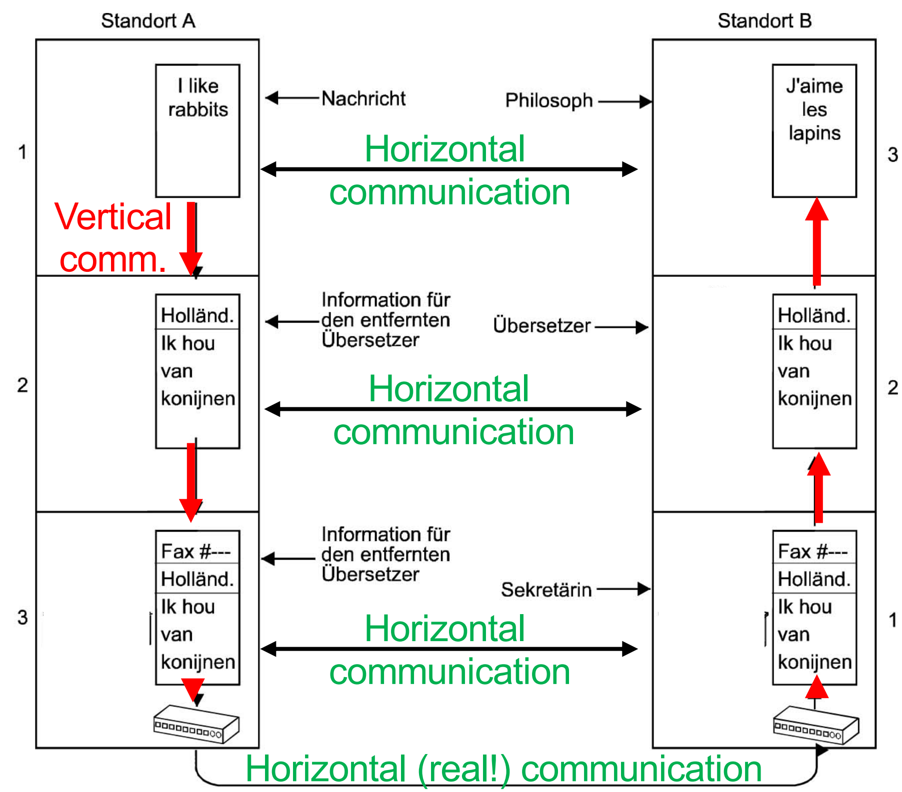
Assistants

Office clerks



Analogy: Nested Layers as nested Translations

- Vertical vs. horizontal communication
  - Vertical: always real
  - Horizontal: may be real or virtual
- Note: protocols interchangeable as long as the interface remains unchanged, e.g.:
  - Layer 2: Dutch => French
  - Layer 3: Fax => E-Mail

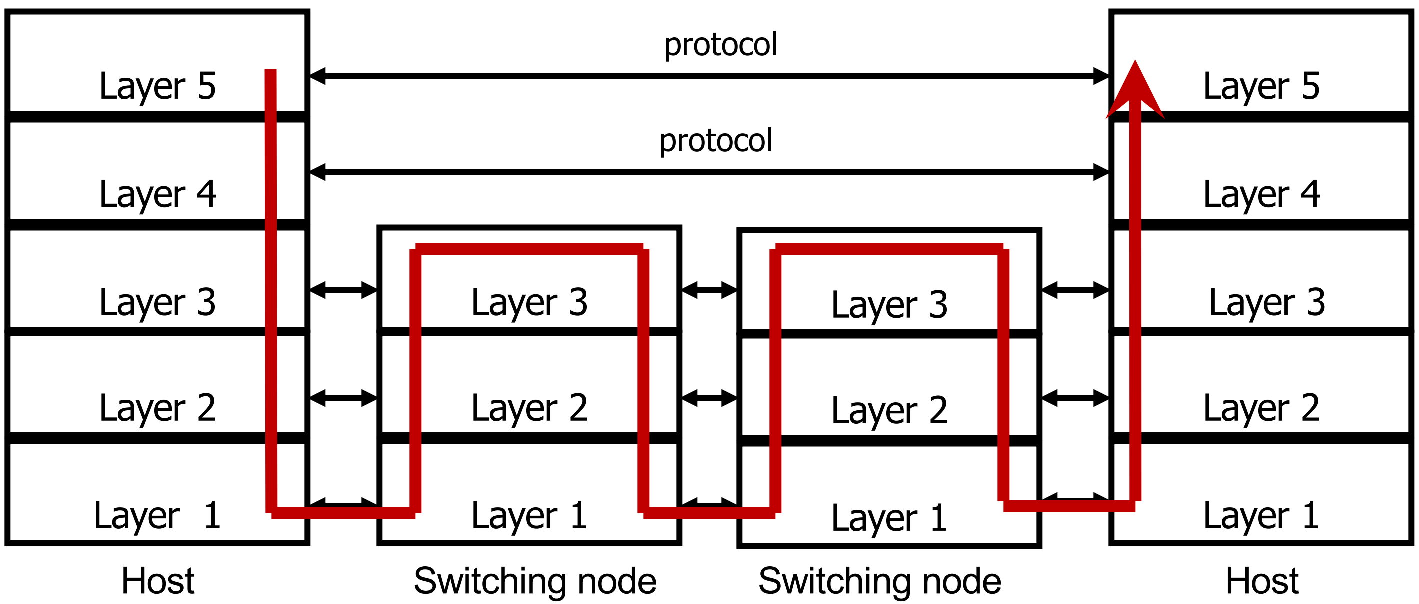


The Reference Model

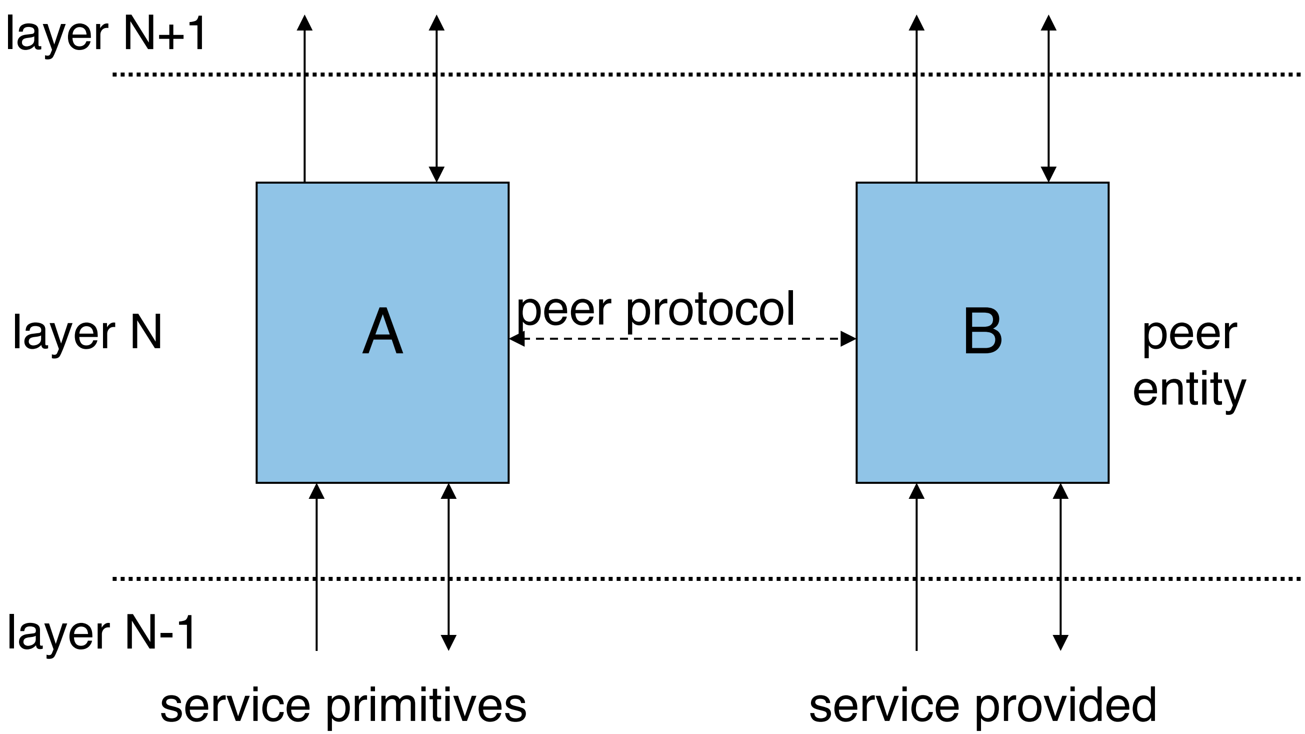
- To keep **complexity** of communication systems tractable:
  - Division in subsystems with clearly assigned responsibilities – layering
- Each layer offers a particular **service**
  - More abstract and more powerful the higher up in the layering hierarchy
- To provide a service, a layer has to be distributed over remote devices
- Remote parts of a layer use a **protocol** to cooperate
  - Make use of service of the underlying layer to exchange data
  - Protocol is a horizontal relationship, service a vertical relationship
- Layers/protocols are arranged as a (protocol) stack
  - One atop the other, only using services from directly beneath
  - **Strict layering** (alternative: cross-layering)

Multi-layer Architecture

- Number of Layers,
  {services, naming, and addressing conventions} / Layer
- Functions to be executed in each layer
- Protocols: (host-to-host, node-to-node, host-to-node)

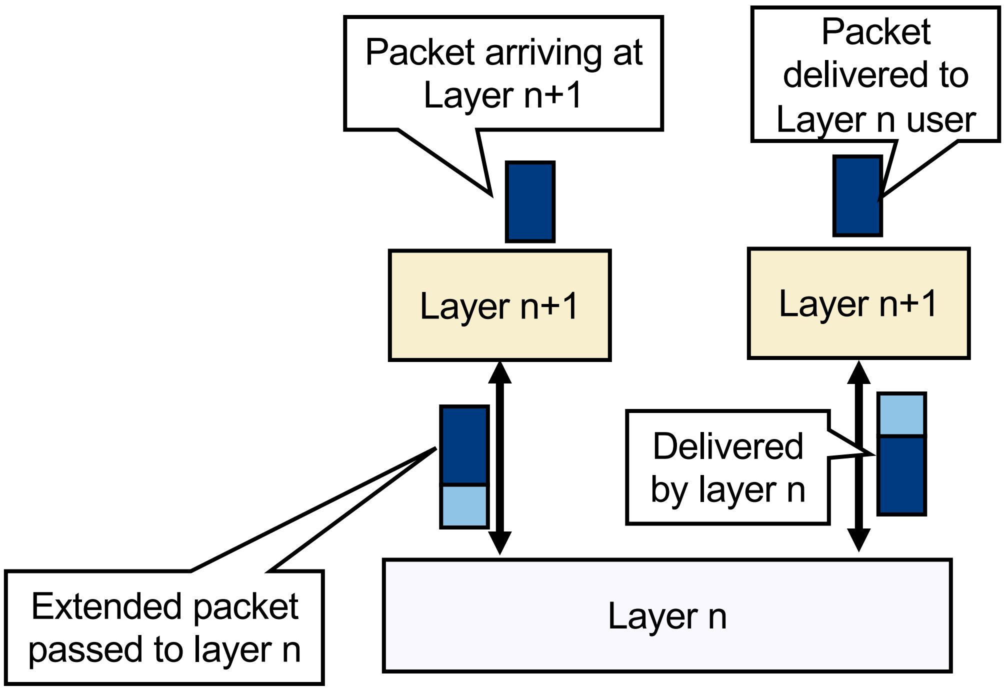


Multi-layer Architecture (II)



Protocols and Messages

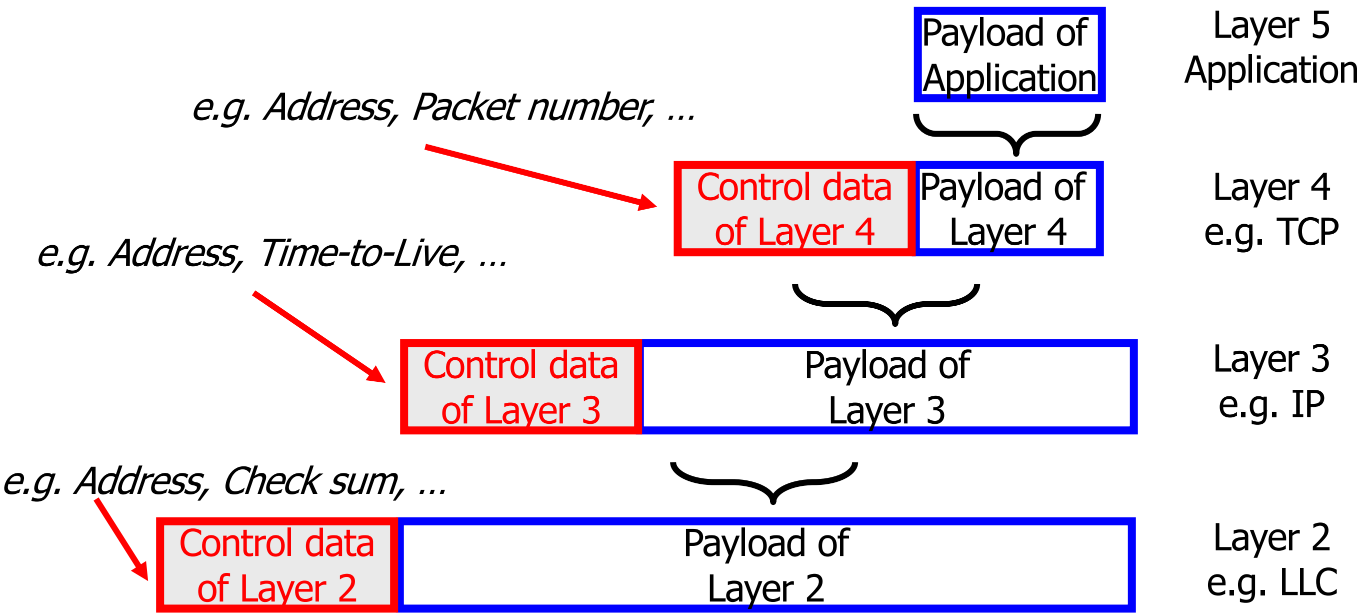
- When using lower-layer services to communicate with the remote peer, administrative data is usually included in those messages
- Typical example
  1. Protocol receives data from higher layer,
  2. Adds own administrative data,
  3. Passes the extended message down to the lower layer,
  4. Receiver will receive original message plus administrative data.
- Encapsulating
  - Header or trailer



Embedding Messages

- Messages from upper layers are used as payload for messages in lower layers

src: wiki



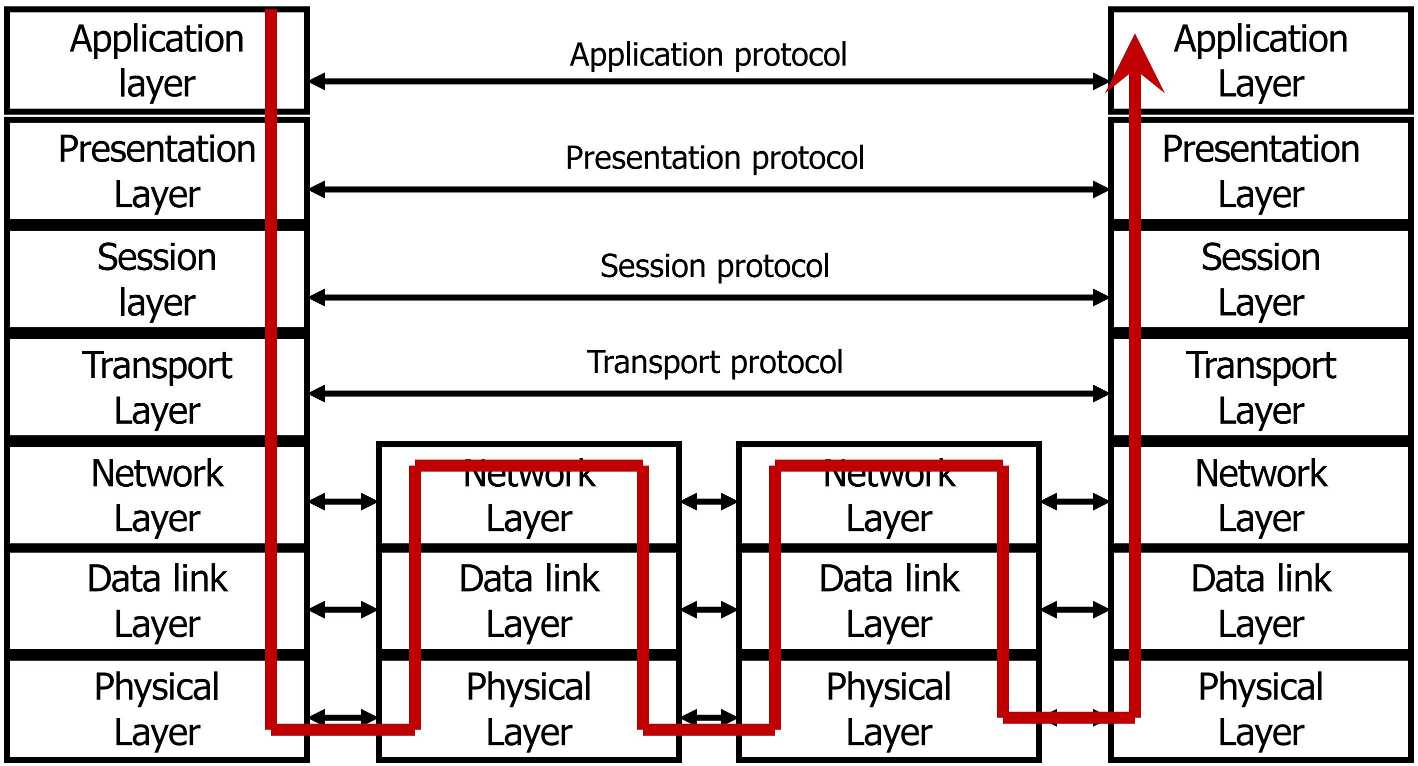
How to structure Functions/Layers?

- Many functions have to be realized
- Not each function is necessary in each layer
- How to actually assign them into layers to obtain a real, working communication system?
  - This is the role of a specific **reference model**
- Two main reference models exist
  - ISO/OSI reference model (International Standards Organization Open Systems Interconnection)
  - TCP/IP reference model (by IETF – Internet Engineering Taskforce)

ISO/OSI Reference Model

- Basic **design principles**
  - One layer per abstraction of the “set of duties”,
  - Choose layer boundaries such that information flow across the boundary is minimized (minimize inter-layer interaction),
  - Enough layers to keep separate things separate, few enough to keep architecture manageable.
- Result: **7-layer model**
  - Not strictly speaking an architecture, because
  - Precise interfaces are not specified (nor protocol details!)
  - Only general duties of each layer are defined

ISO/OSI Model



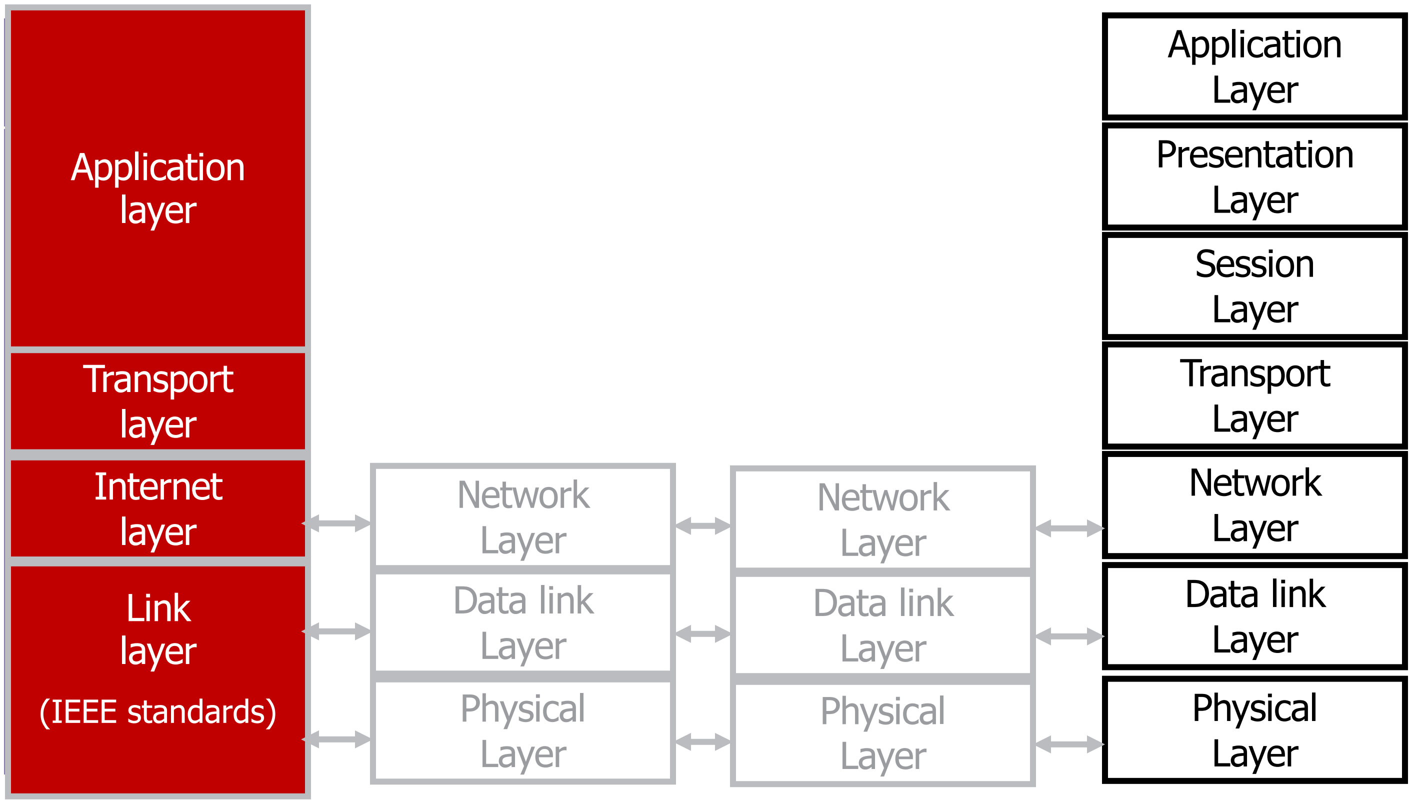
Brief Overview of the 7 Layers

- **Physical layer**: Transmit raw bits over a physical medium
- **Data Link layer**: Provide a (more or less) error-free transmission service for data frames - also over a shared medium!
- **Network layer**: Solve the forwarding and routing problem for a network - bring data to a desired host
- **Transport layer**: Provide (possibly reliable, in order) end-to-end communication, overload protection, fragmentation to processes

  "Bringing data from process A to B with sufficient quality"
- **Session layer**: Group communication into sessions which can be synchronized, checkpointed, …
- **Presentation layer**: Ensure that syntax and semantic of data is uniform between all types of terminals
- **Application layer**: Actual application, e.g., protocols to transport web pages

Internet Model (in red) vs. ISO/ OSI

- Presentation, session & physical layer not present in Internet model



Telecommunication Networks Group

Architecture, Protocols

- A communication architectures needs **standard protocols** in addition to a layering structure
- And some **generic rules & principles** which are not really a protocol but needed nonetheless
  - Example principle: **end-to-end**
  - Example rule: naming & addressing scheme
- Popular protocols of the Internet reference model
  - Data link layer: Ethernet & CSMA/CD (defined in IEEE standard)
  - Network layer: Internet Protocol (IP)
  - Transport layer: Transmission Control Protocol (TCP)

Internet Reference Model

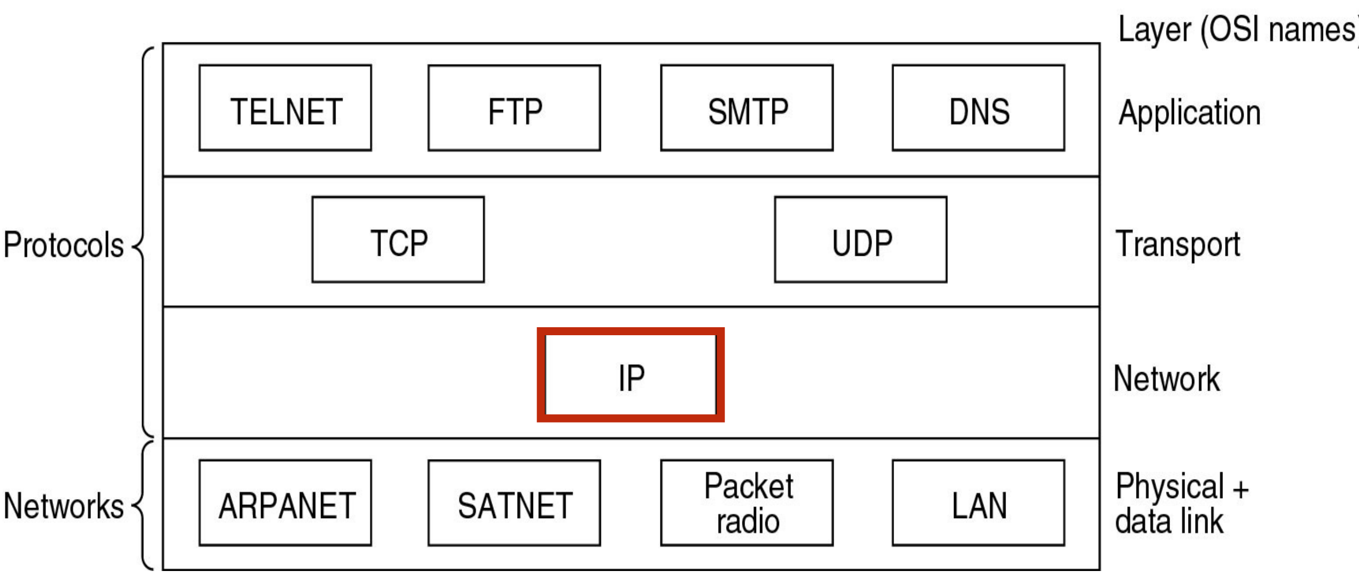
- Historically based on ARPANET, evolving to the Internet
  - Started out as little university networks, which had to be interconnected
- Some generic rules & principles
  - Internet connects **networks**
    - Minimum functionality assumed (just unreliable packet delivery)
    - Internet layer (IP): packet switching, addressing, routing & forwarding

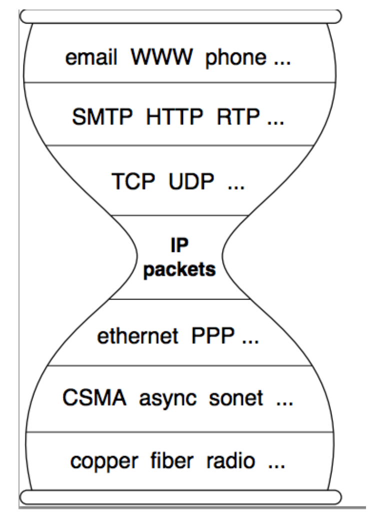
    → **Internet over everything**
  - End-to-end
    - Any functionality should be pushed to the instance needing it
  - Fate sharing
- In effect only two layers really defined: Internet and Transport Layer - lower and higher layers not really defined

  → **Anything over Internet**
- New applications do not need any changes in the network
  - Compare with the telephone network

The Internet Suite of Protocols

- Over time, suite of protocols evolved around core TCP/IP protocols
- → Internet Protocol Suite is also refereed to as TCP/IP Protocol Suite
- → "**hourglass model**": thin waist of the protocol stack at IP, above technological layers





Naming & Addressing in the Internet Stack

- **Names**: Data to **identify** an entity exist on different levels
  - Alphanumerical names for resources:
    e.g., ccs.tkn.tu-berlin.de, www.tkn.tu-berlin.de
- **Address**: Data how/where to **find** an entity
  1. Address of a network device in an IP network: an IP address
     - IPv4: 32 bits, structured into 4x8 bits
     - Example: 130.149.110.75 (dotted decimal notation)
  2. Address of a network: Some of the initial bits of an IP address
- Address of a networked device in the Local Area (IEEE 802 standardized) Network (LAN): a MAC address
  - 48 bits, hexadecimal notation, example: ac:1f:6b:0e:a2:d0

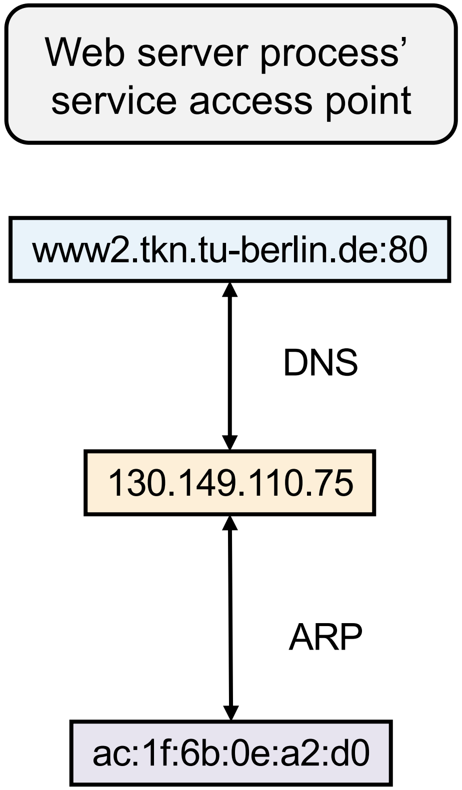
Mapping

- Needed: mapping from name to address

→ realized by separate protocols

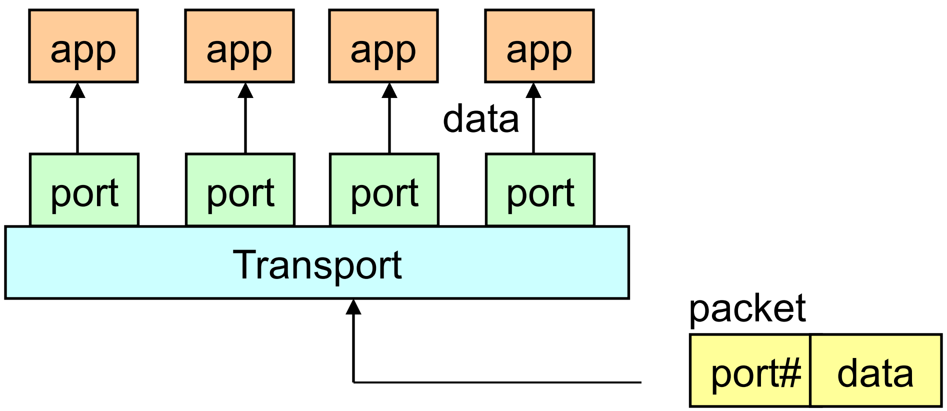
- From alphanumerical name to IP address: **Domain Name System** (DNS)

- Often also needed: mapping from IP address to MAC address: **Address Resolution Protocol** (ARP)



Understanding Ports

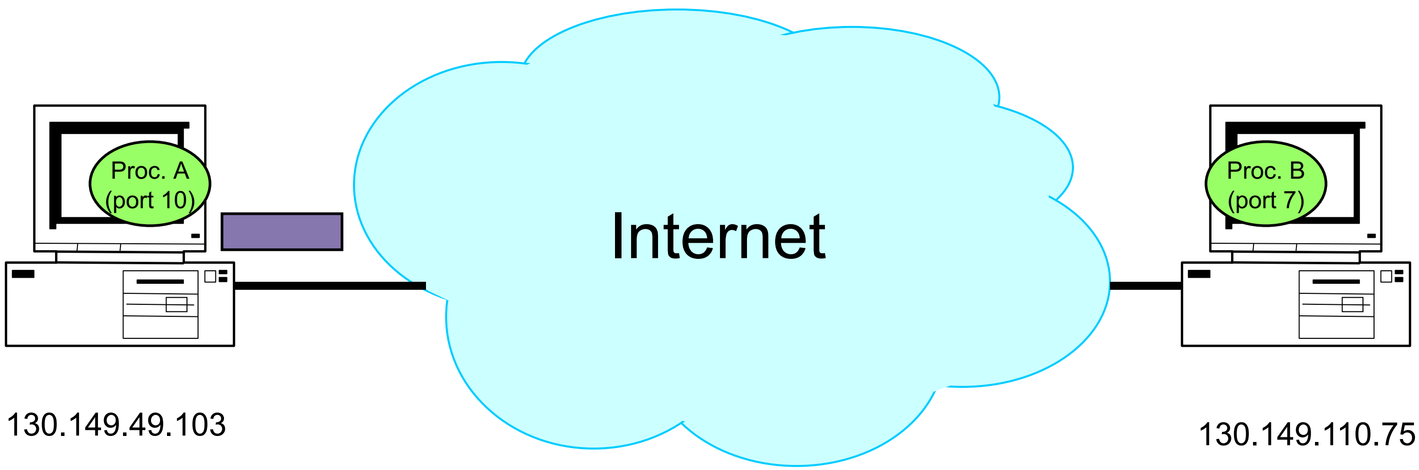
- … to distinguish between individual processes
- Port is represented by a positive (16-bit) integer value
- Some ports have been reserved to support common/well known services: http 80/tcp; ftp 21/tcp; telnet 23/tcp; smtp 25/tcp;
- User level process/services generally use port number value >= 1024



[Buya, op. cit.]

Internet End-to-End View

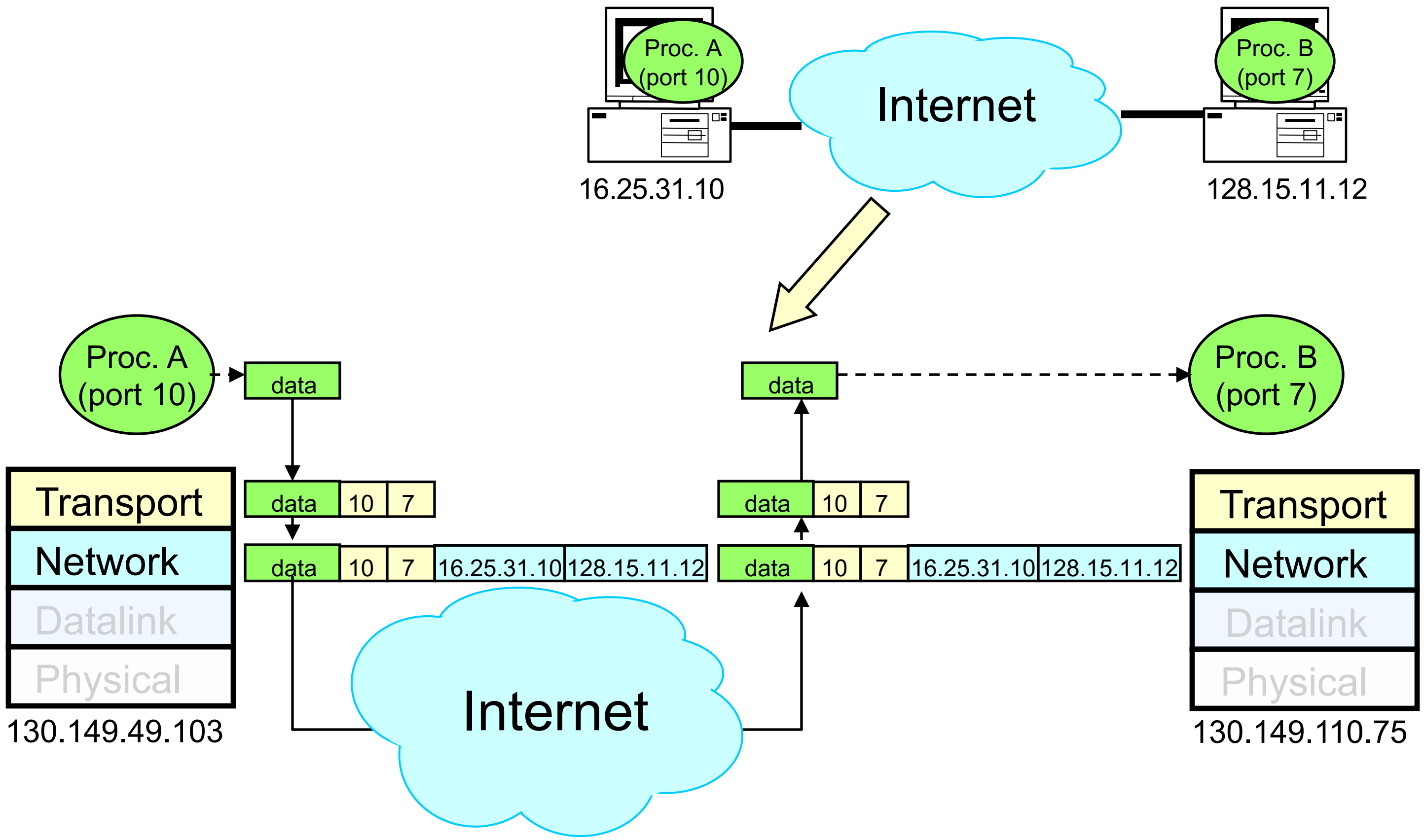
- Process A sends a packet to process B



IP address:
A four-part "number" used by Network Layer to route a packet from one computer to another

[Stoica, op. cit.]

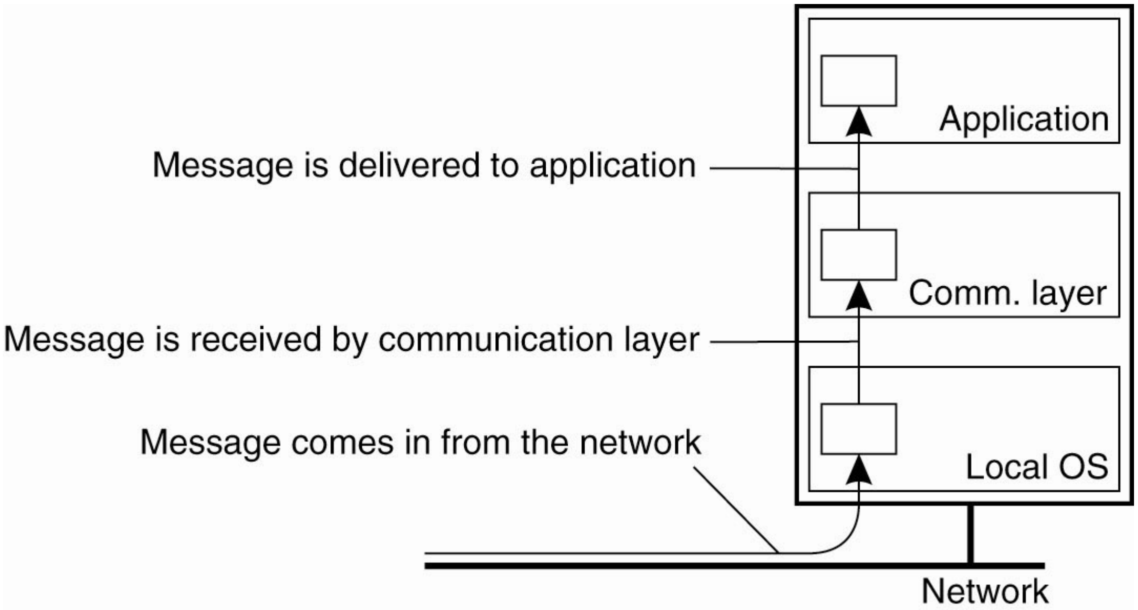
End-to-End Layering View



[Stoica, op. cit.]

Message Receipt vs. Message Delivery

- The logical organization of a distributed system to distinguish between message receipt and message delivery.

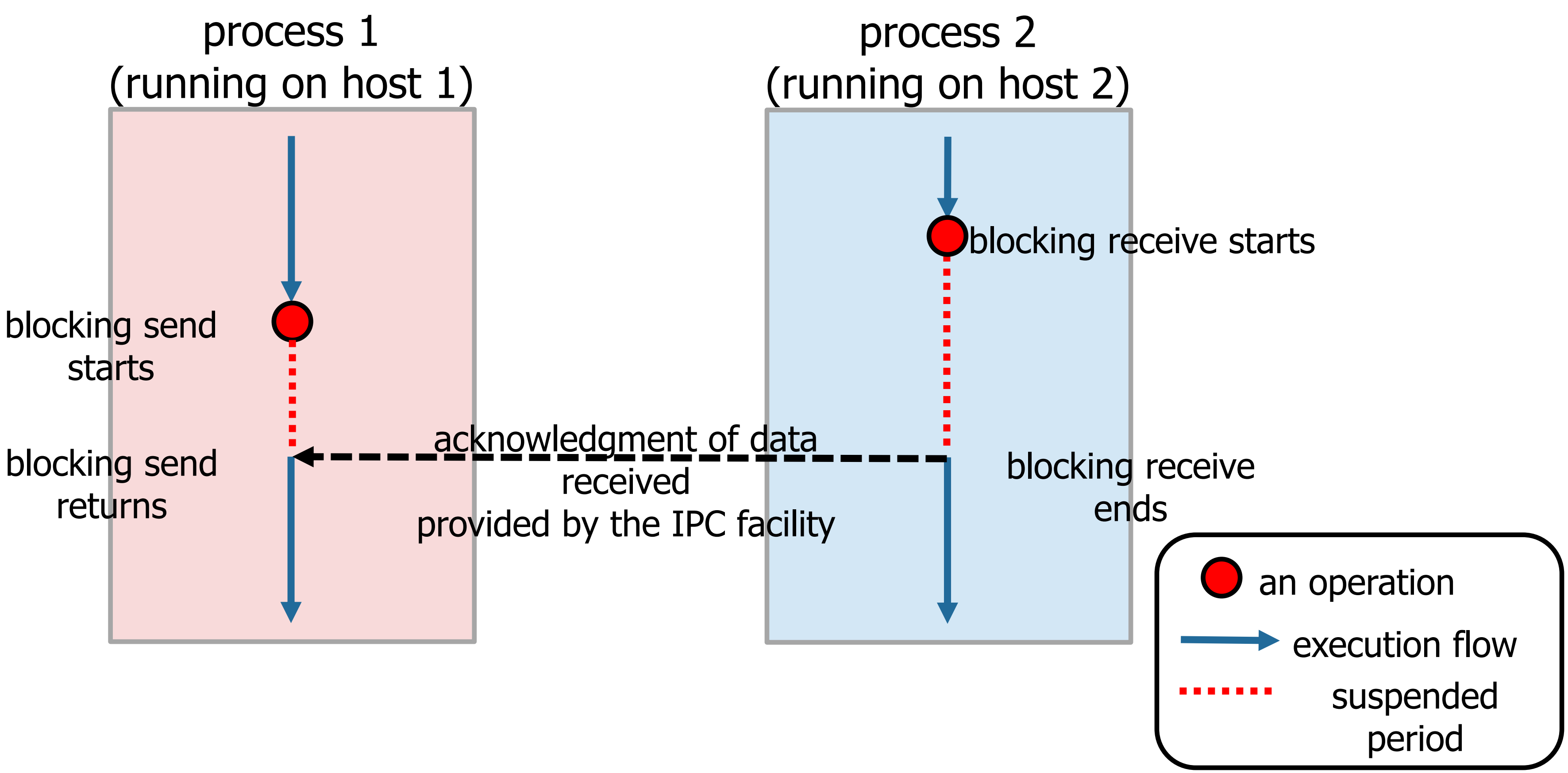


Interaction Principles: Synchronous Interaction

- Blocking send
  - Blocks until message is transmitted
  - Blocks until message acknowledged
- Blocking receive
  - Waits for message to be received

- **You should know**: upper/lower bounds on execution speeds, message transmission delays and clock drift rates

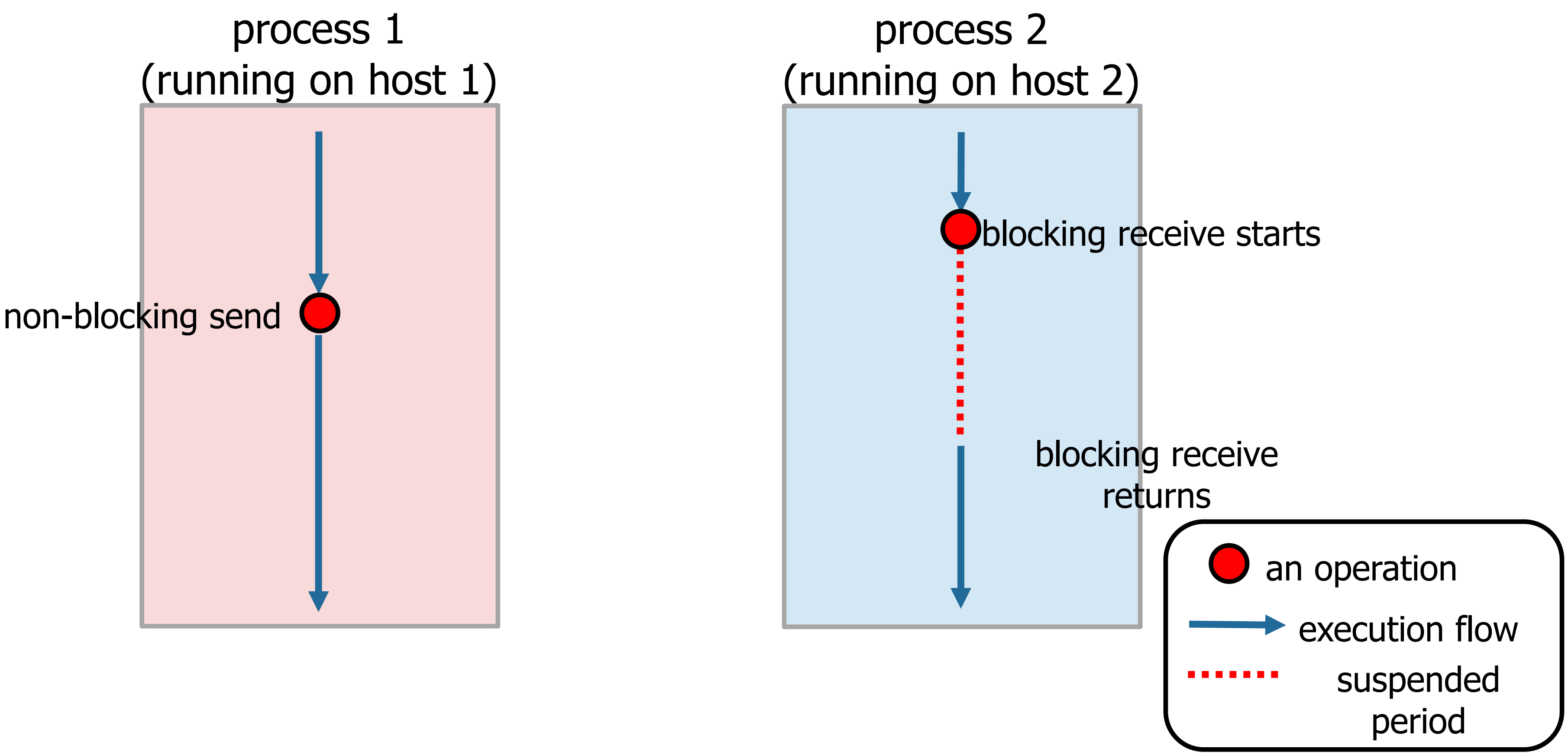
Synchronous Send & Receive



Interaction Principles: Asynchronous Interaction

- Non-blocking send: sending process continues as soon message is **queued**
- Blocking or non-blocking receive:
  - Blocking:
    - Timeout
    - Threads
  - Non-blocking: proceeds while waiting for message
    - Message is queued upon arrival
    - Process needs to poll or be interrupted
- **Advantage**: arbitrary process execution speeds, message transmission delays and clock drift rates
- Some problems impossible to solve (e.g., agreement)

Asynchronous Send & Synchronous Receive



Asynchronous Send & Asynchronous Receive