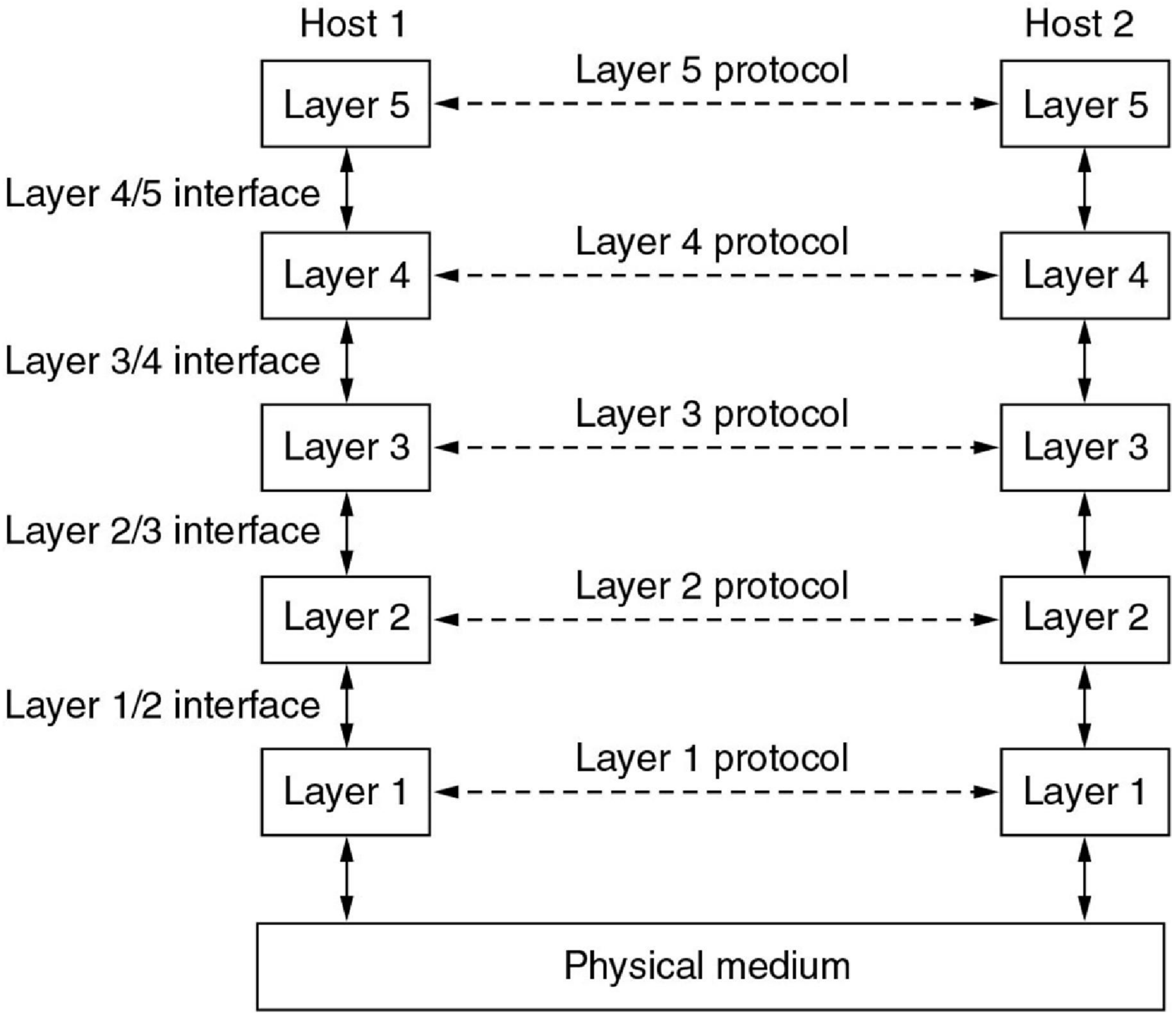
Protocol Hierarchies/Network Software

- Computer networks are generally comprised of numerous pieces of hardware and software
- To simplify network design most networks are organized as a stack of layers of hardware **and/or** software
- The purpose of each layer is to offer *services* to higher layers
- This concept is common in programming viz. objects or libraries which perform specific operations
  - Importantly the object/library function keeps details of its internal state and algorithms hidden from the main program
- The example five-layer network illustrates the layers

Protocol Hierarchies/Network Software



Protocol Hierarchies/Network Software

- *Peer* entities exist at corresponding layers on the source and destination stations
  - These may be processes, hardware devices, or even human beings
- These *peer entities* communicate with each other across a layer
  - The communications rules and conventions used between *peer entities* are collectively known as the *Layer N protocol*
  - A protocol is an *agreement* between communicating parties on how communication should take place

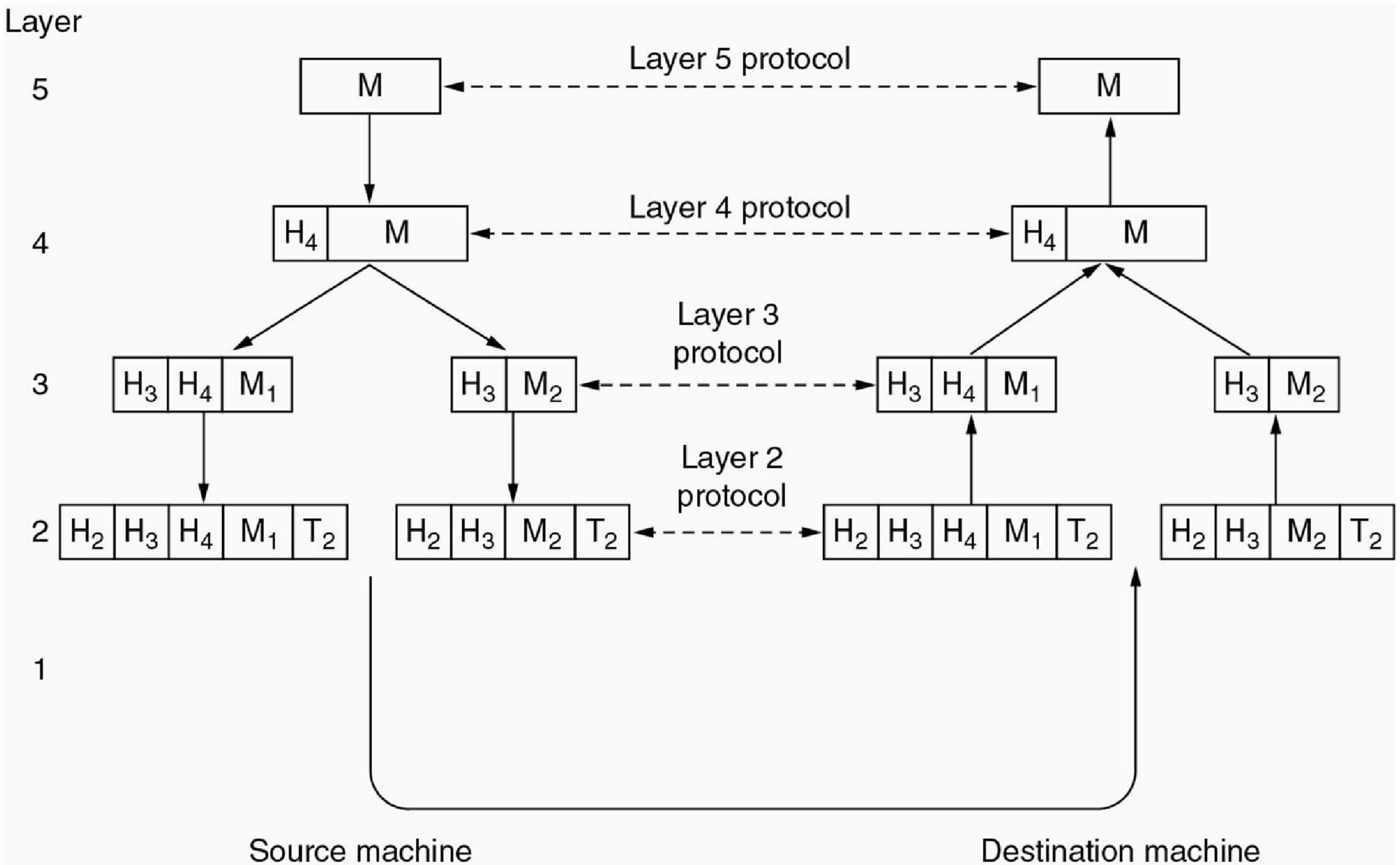
Protocol Hierarchies/Network Software

- In reality, no data is passed directly between peer entities
  - Instead data passes upwards towards the *network applications* or downwards towards the *physical medium*
- *Virtual* communications between peer entities are shown as dotted lines
- *Physical* communications are shown with solid lines

Protocol Hierarchies/Network Software

- Between each pair of adjacent layers is an *interface*
  - A layer's interface reveals the specific functions performed within the layer
- Well defined *interfaces* are an essential design feature of protocol software because:
  - They minimize the amount of information passed between layers
  - They make it easier to replace *layer entities* without affecting the ability of the hosts to communicate with each other

Information flow supporting virtual communication in layer 5



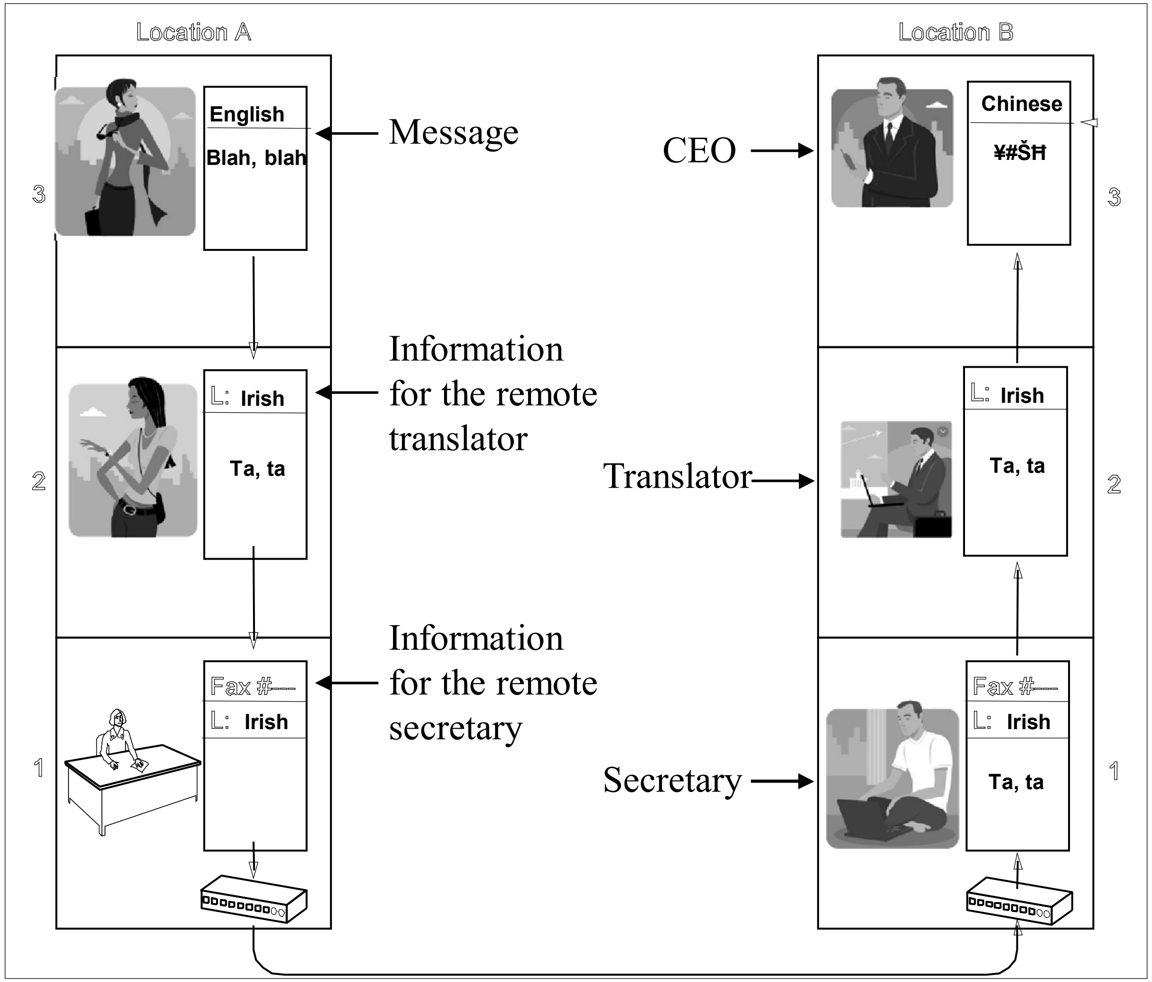
The *Layering Principle*

- When designing layered protocol software it is important to adhere to the *layering principle*:
  - *"Layer N software on the Destination machine must receive the exact message sent by Layer N software on the Source machine"*
- Any transformations or additions made by layers below Layer N on the sending side must be reversed/removed before a message is passed to Layer N software on the receiving side.

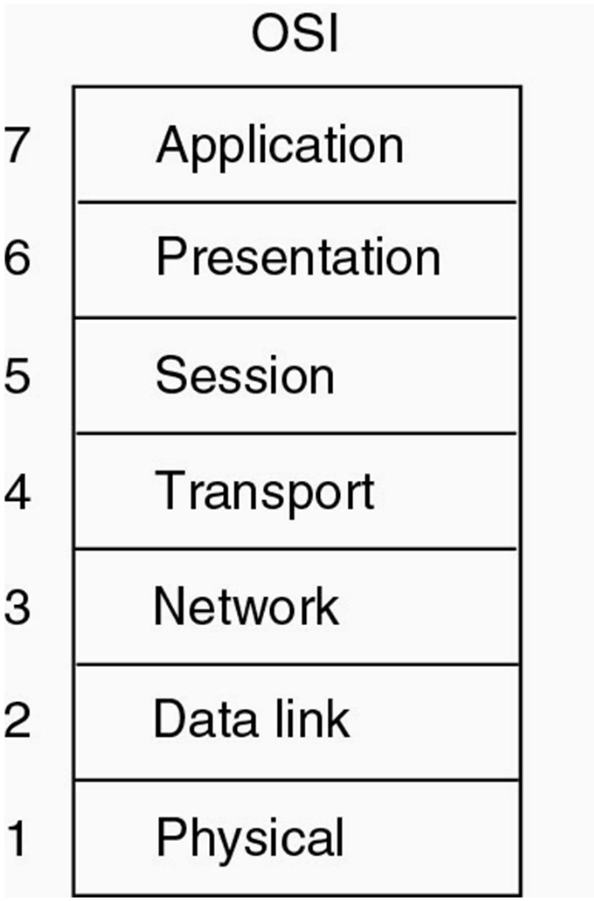
Network Architecture

- Some terminology:
  - A set or stack of layers and protocols is called a *protocol architecture*
    - The architecture only specifies the functions associated with each layer. It does not specify the *implementation* or the *interfaces* of the layer
  - The set of protocols (one per layer) used within an actual network system is called a *protocol stack*
- An analogy may help explain the idea of multilayer communication

The CEO-Translator-Secretary Scenario



Reference Models - The *ISO OSI Model*



Reference Models - The OSI ISO Model

- The "Internal Standards Organisation Open Systems Interconnect" model:
  - AKA: "*The OSI model"*, " *The 7-layer model",*
  - It deals with connecting open systems, Systems that are open for communication with other systems
- The OSI Reference model is a Protocol Architecture Reference Model :
- It is not a Protocol Stack:
  - It does not specify how functionality is to be provided only what the functionality of each layer should be,
  - It is not something one can purchase or install on a networked machine.

Reference Models - The OSI ISO Model

- The principles that were applied to arrive at the seven layers are as follows:
  - Each layer was created when a different level of abstraction was required
  - Each layer performs a well-defined function with each function chosen carefully to facilitate internationally standardized protocols
  - The layer interfaces (boundaries) were carefully defined to *minimize* information flow across the interfaces
  - The number of layers chosen was sufficient enough to ensure that distinct functions were not lumped together without becoming unwieldy

The *ISO Reference Model* - Layer-by-layer

- The *Application* Layer:
  - Contains a variety of protocols commonly used on the Internet
    - E.g. HTTP (HyperText Transfer Protocol) is the underlying protocol for the World Wide Web
    - Other protocols include FTP, E-mail etc.

The *ISO Reference Model* - Layer-by-layer

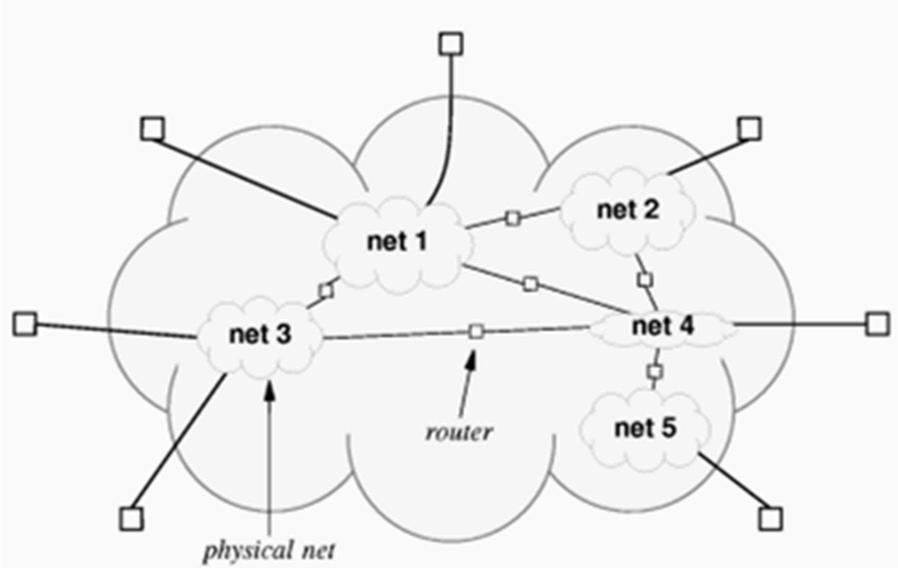
- The *Presentation* Layer: Concerned with the *syntax* and *semantics* of the information transmitted
  - Facilitates communication between *big-endian* computers e.g. Sun Sparcs and *little-endian* computers e.g. Windows machines
- The *Session* Layer: Facilitates the use of *sessions* between end stations. During a session the user and the computer system engage in a *dialogue*
  - The session layer establishes and maintains dialogues
  - It also determines:
    - The type of control to be used i.e. two-way simultaneous communication, two-way alternate comm. or one-way comm.
    - *Re-synchronization* of the dialogue after a crash

The *ISO Reference Model* - Layer-by-layer

- The *Transport* Layer: THIS IS KEY LAYER from our perspective.
- It is a true end-to-end layer in that it operates between end-hosts.
- Its function is to isolate the applications from the underlying network hardware technology:
  - It uses the network as a reliable ‘deliverer’ of data
- It splits the **source** data into manageable chunks and passes them to the network layer for onward delivery.

The *ISO Reference Model* - Layer-by-layer

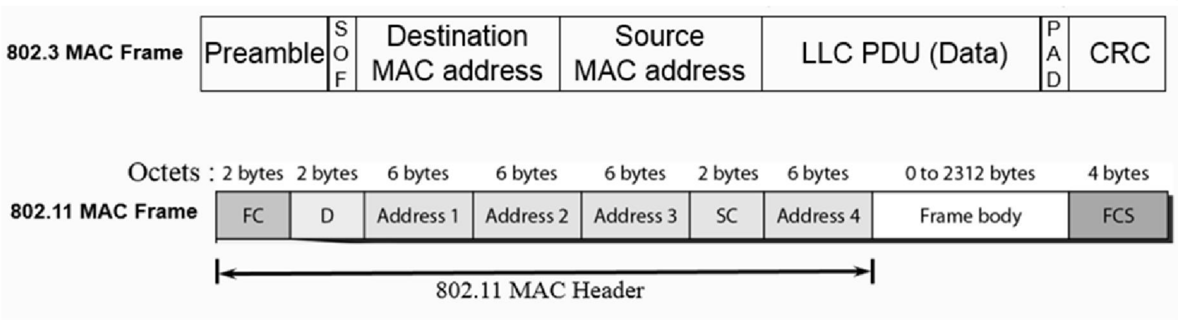
- The *Network* Layer:
- Concerned with controlling the operation of the *sub-network* (subnet)
  - Deals with the routing of *packets* from the *source* station towards the *destination* station across ***sub-networks***
  - It handles the different sub-net addressing formats
  - Essentially this layer is responsible for interconnecting *heterogeneous* networks

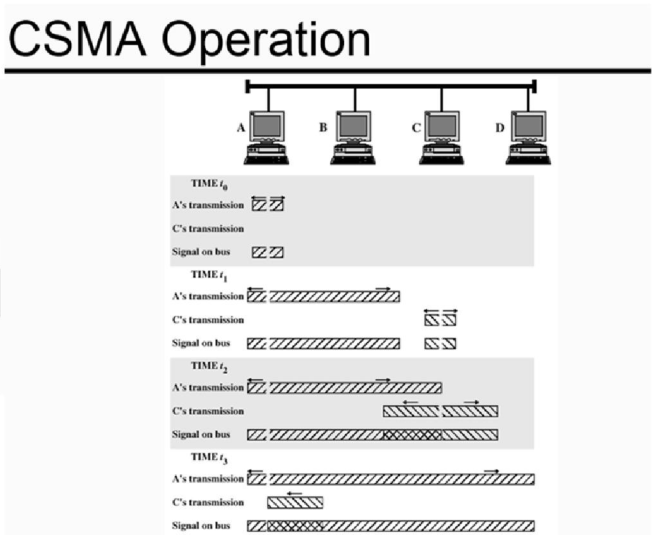


(b)

The *ISO Reference Model* - Layer-by-layer

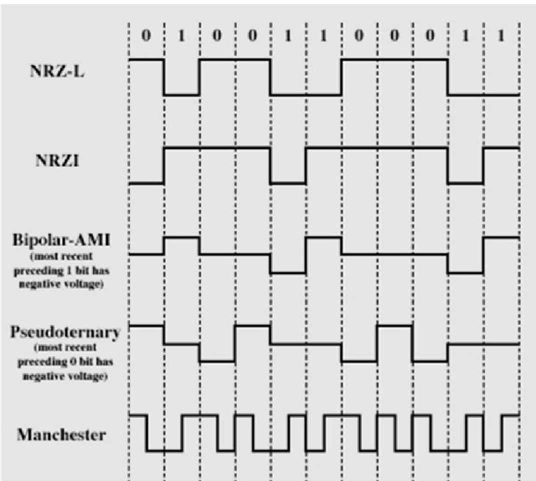
- The *Data Link* Layer: Concerned with getting data across an individual link
  - Essentially it transforms a raw transmission facility into a *data communications channel* that *appears* free of transmission errors
  - Breaks up the data into *data frames.* Also deals with flow control, controlling access to a shared channel etc.

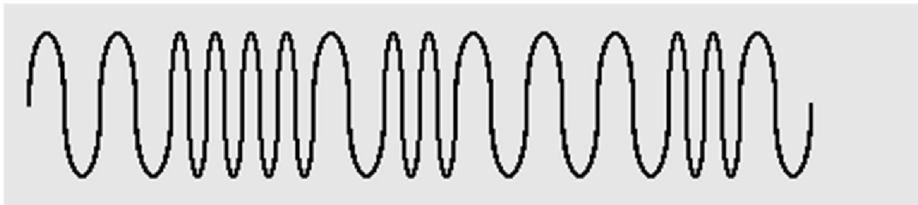




The *ISO Reference Model* - Layer-by-layer

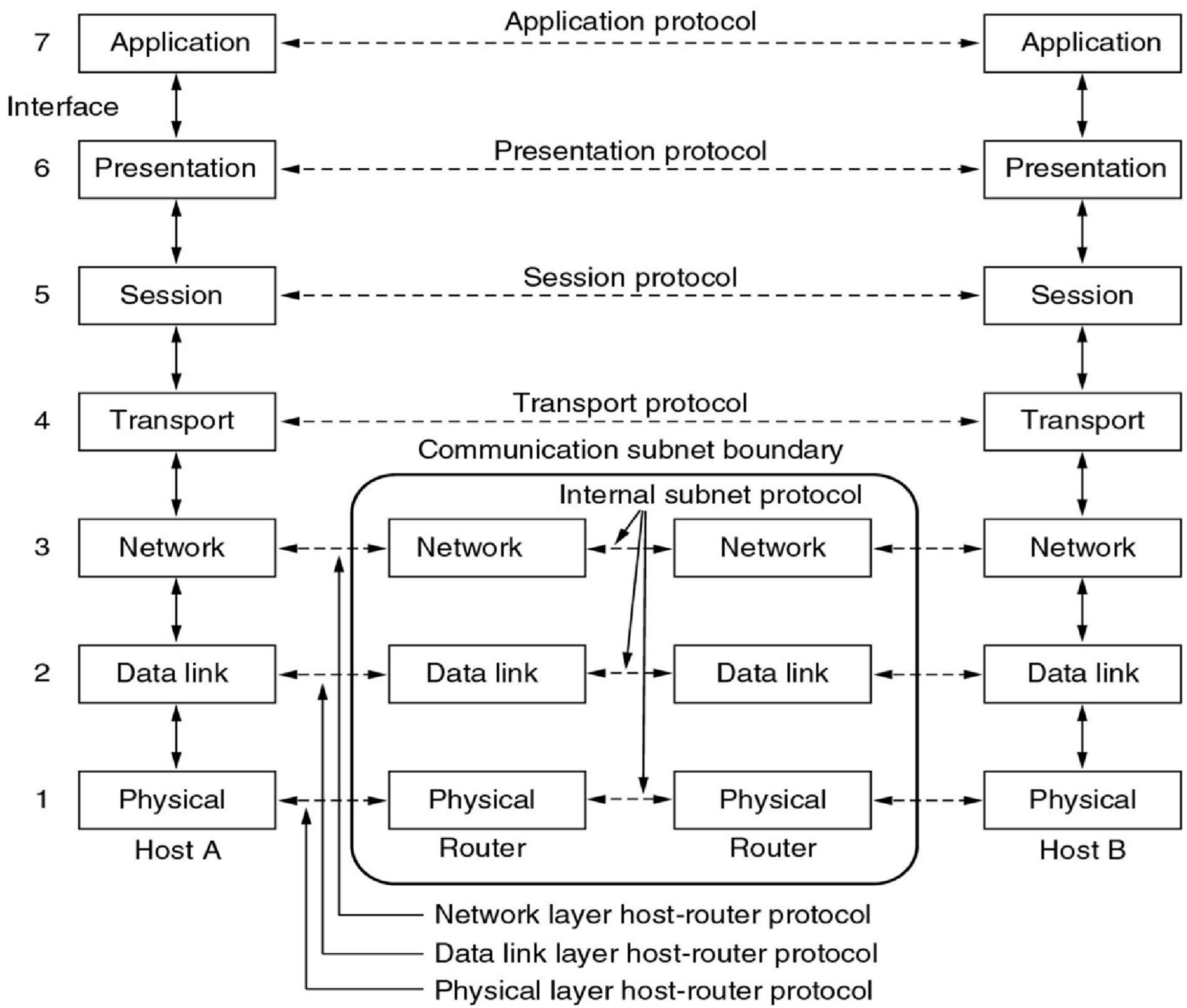
- The *Physical* Layer: Concerned with transmitting raw bits over a *communication channel*. Must ensure that when a binary 1 is sent it is received as such by the *receiver*
  - Deals with voltage levels used, bit duration etc.
  - Design issues deal with *mechanical*, *electrical*, and *timing interfaces*, and the physical *transmission medium*



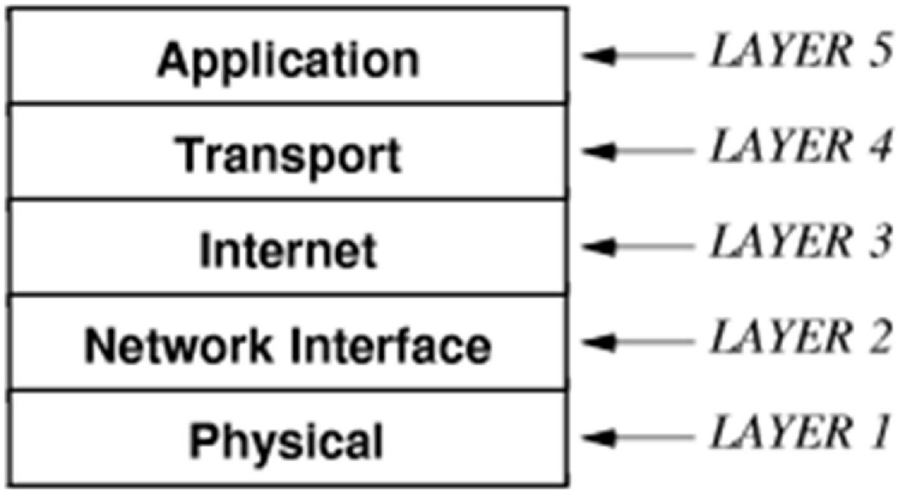


(b) Frequency-shift keying

The *ISO OSI Model* – Layers 1-3 Versus Layers 4-7



Reference Models – The *TCP/IP Reference Model*



The *TCP/IP Reference Model*

- Transport Control Protocol / Internetwork Protocol Reference Model:
  - AKA: “The TCP/IP Model”
- The protocols upon which this model is based (*TCP and IP*) fuelled the early growth of the *Internet*:
  - TCP and IP were adapted because there were available,
  - The ISO protocols on the other hand were still being developed
- The *TCP/IP Reference Model* was developed <u>after</u> the protocols
  - This similar to retrospectively drawing plans for a house <u>after</u> it is built.

The *TCP/IP Reference Model*

- Specific design goals of this *Reference Model* included:
  - Ability to survive the loss of subnet hardware. Specifically in a "Theatre of War" (a Battlefield),
  - Ability to handle multiple types of data including files and real-time speech.
- These requirement led to the adoption of a <u>connectionless</u> packet-switching network within the *internet* layer

The *TCP/IP Reference Model* - Layer-by-layer

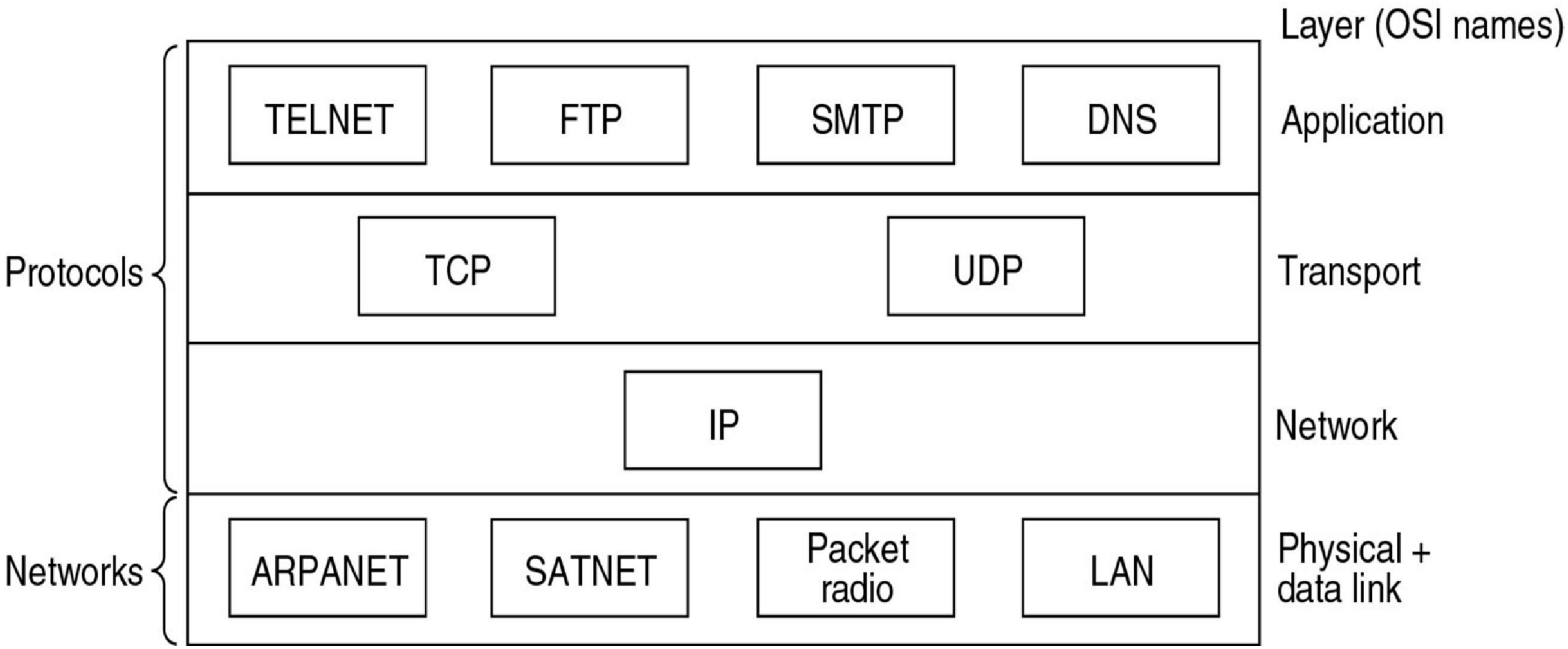
- The *Application* Layer: This layer contains all of the higher-level protocols including *FTP, E-mail*, the *Domain Name System (DNS)* and HTTP.

- The *Transport* Layer: Facilitates *end-to-end* communication between the *Source* and *Destination* hosts
- Two end-to-end transport protocols have been defined:
  - *TCP (Transmission Control Protocol)*: This is a reliable, connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error to any other machine in the internet.
  - *UDP (User Datagram Protocol)*: This is an unreliable, connectionless protocol for applications that provide their own *sequencing* and *flow control* functionality

The *TCP/IP Reference Model* - Layer-by-layer

- The *Internet* Layer: This layer is key to the whole architecture
  - It facilitates hosts injecting *packets* into <u>any</u> network,
  - It ensures correct *routing* of packets to the *Destination* station
- The *Host-to-Network* Layer: This layer is meant to deal with hosts connecting to the network in order to transmit packets
  - It is not well defined within the TCP/IP reference model

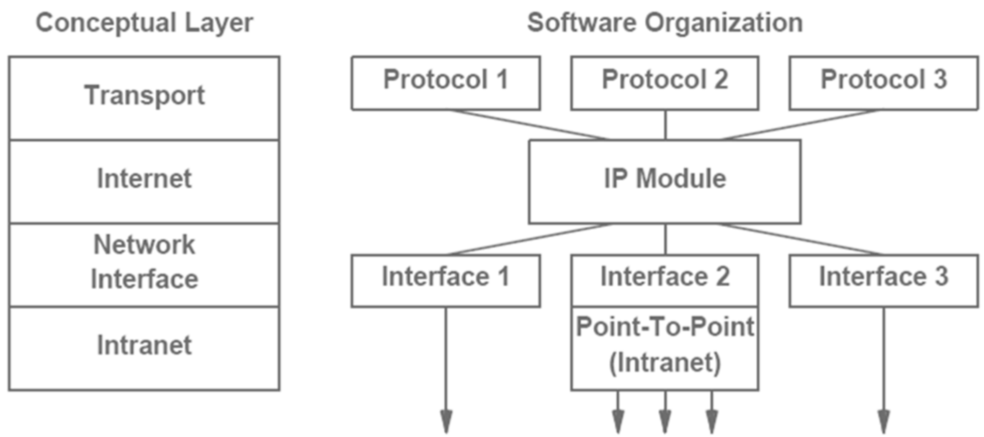
The *TCP/IP Reference Model* – Relationship between *TCP, UDP* and *IP*



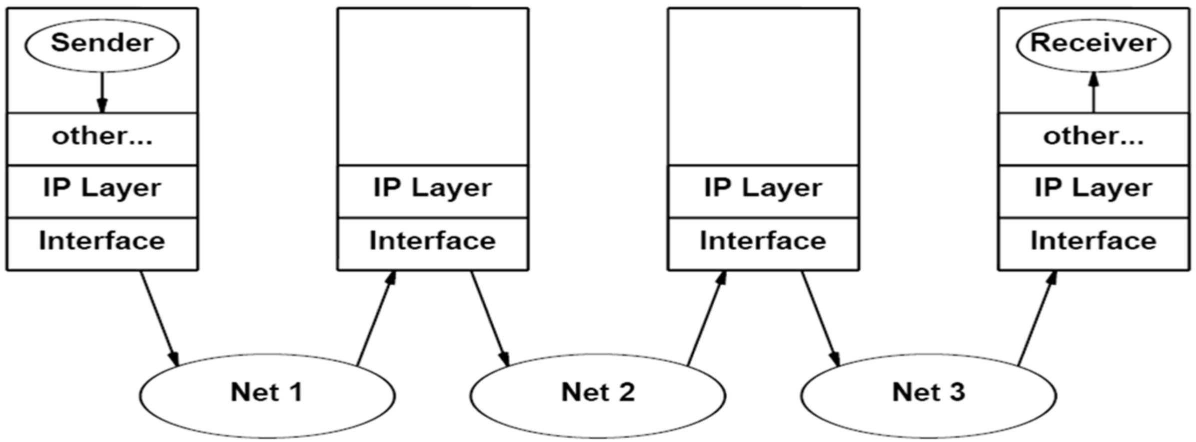
A Comparison of OSI and TCP/IP Reference Models

- Both models are similar in many ways viz.:
  - Both use the concept of a stack of independent protocols
  - Both transport layers provide an end-to-end, network-independent transport service to applications
- However, there are some notable differences as follows:
  - *Number of layers*: OSI model has **7** layers, TCP/IP has **5** layers
  - *Services versus Interfaces/Protocols*:
    - OSI clearly defines what each layer does using *service definitions*
    - TCP/IP did not originally clearly distinguish between *service*, *interface* and *protocol*. This hindered switching-out protocols to facilitate technological change
  - *Timing*:
    - OSI model was developed before the protocols were invented
    - With TCP/IP the protocols came first and then the model

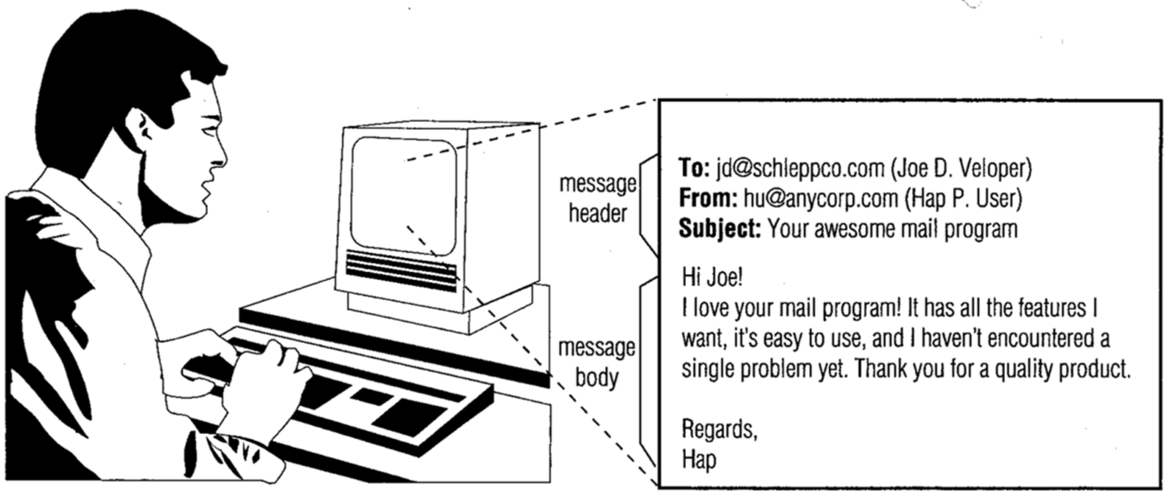
Conceptual versus *Realistic* view of protocol layering

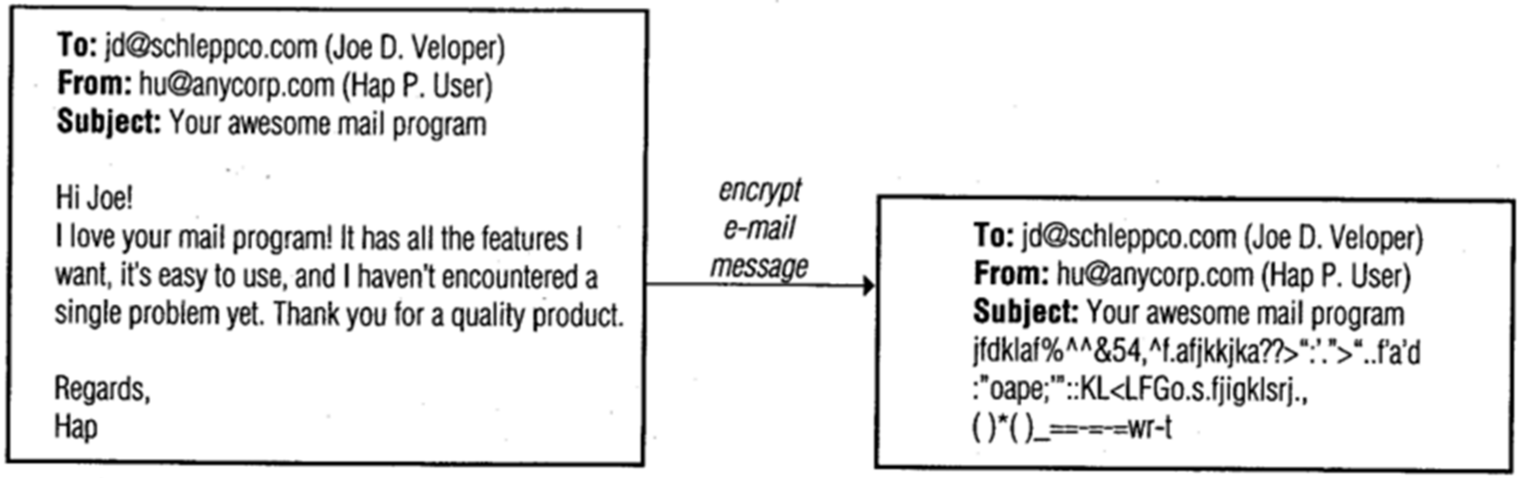


(a) (b)



The *Layers* in operation – Application and Presentation Layers





The *Layers* in operation – The Session Layer

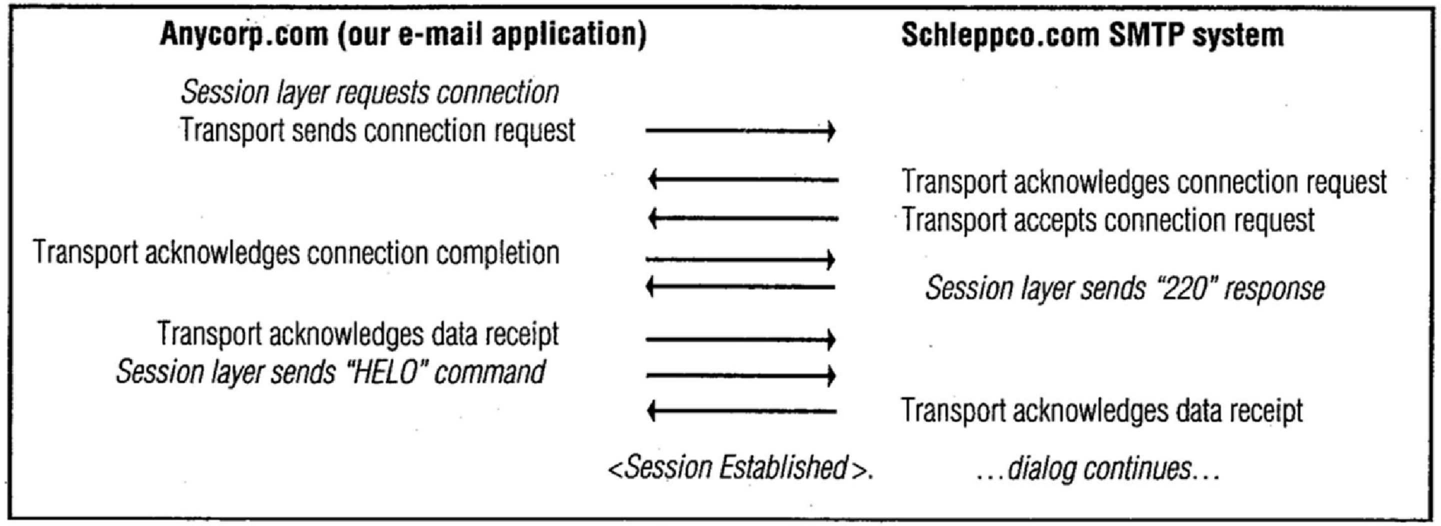
Server: 220 Schleppco.COM Simple Mail Transfer Protocol (SMTP) Server ready to serve you!
App: HELO AnyCorp.COM
Server: 250 Ok to proceed!
<Session Established>
App: MAIL FROM:<hu@anycorp.com>
Server: 250 Ok to proceed!

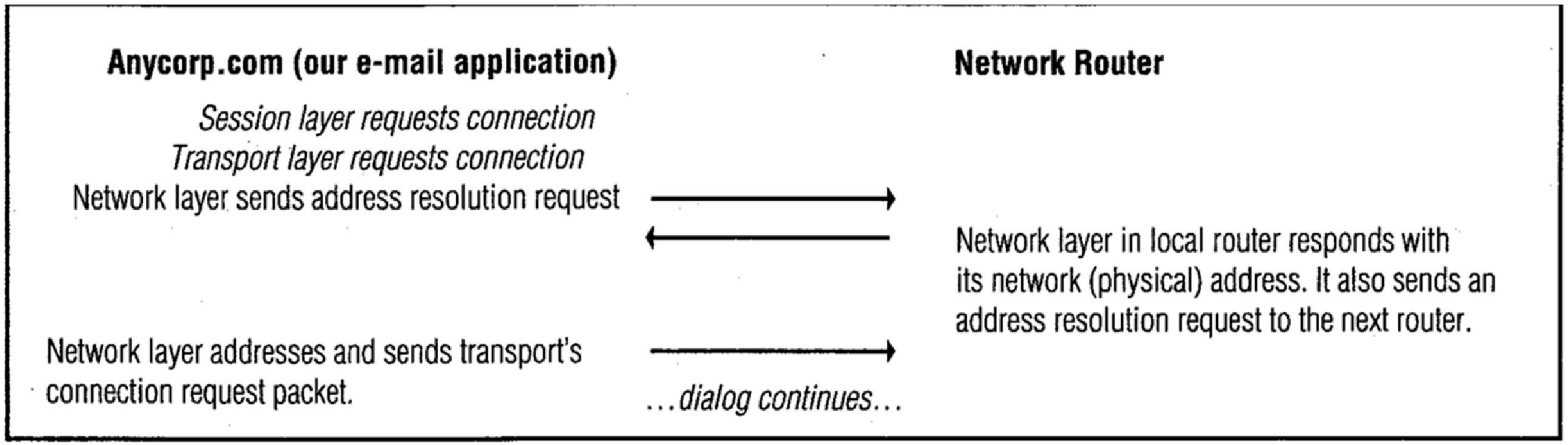
App: RCPT TO:<jdv@schleppco.com>
Server: 250 Ok to proceed!

App: DATA
Server: 354 Send mail message and end it with '.' alone on a line
App: Date: 24 June 94 14:20:00 EST
App: jfdklaf%^^&54,^f.afjkkjka??>":'."..f 'a' d
(our encrypted & compressed mail message)
App: :"oape;'"::KL<LFGo.s.fjigklsrj.,
App: ( )*( )_=.=-=-=wr-t
App: .
Server: 250 Ok to proceed!

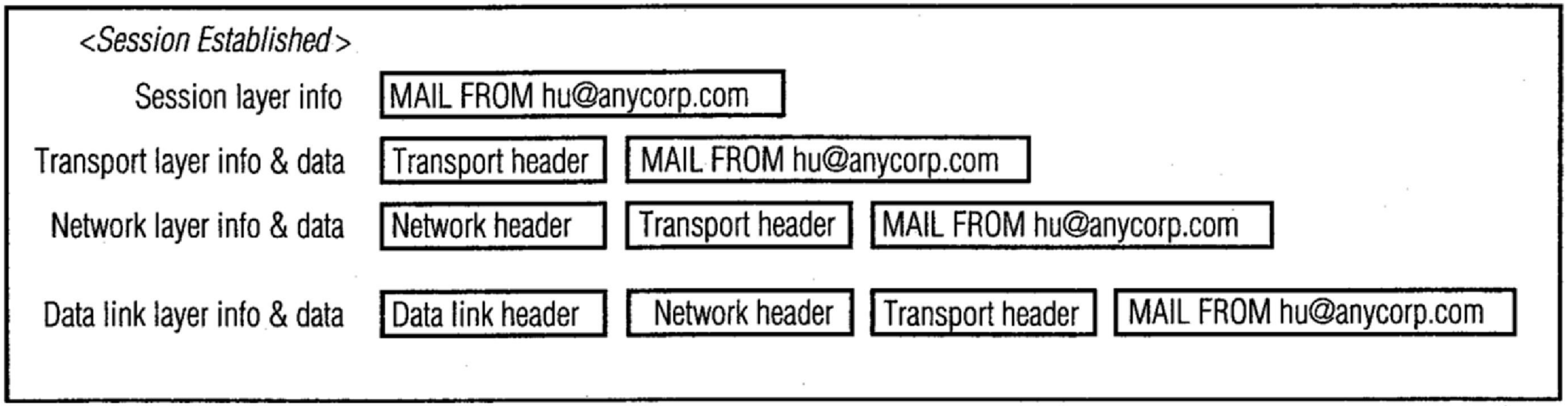
App: QUIT
Server: 221 Schleppco.COM over and out Thanks for the visit!
<Session Ended>

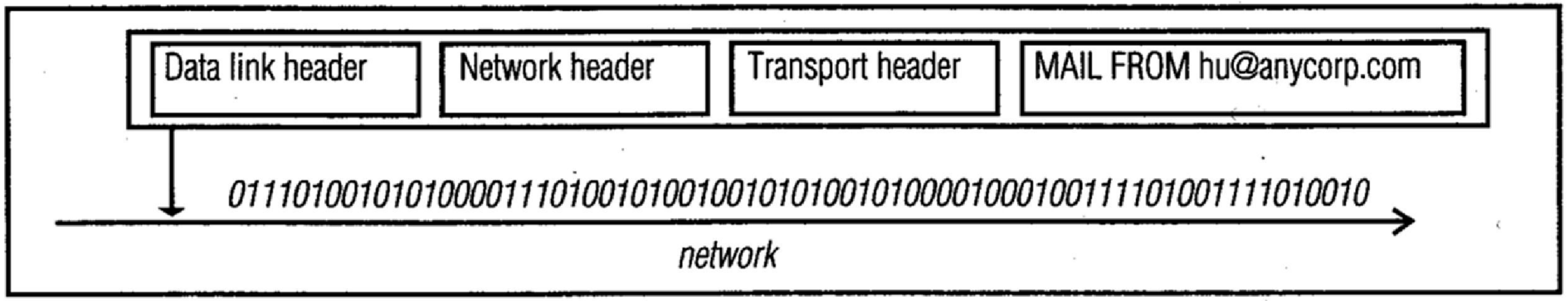
The *Layers* in operation – Transport and Network Layers





The *Layers* in operation – Data Link and Physical Layers

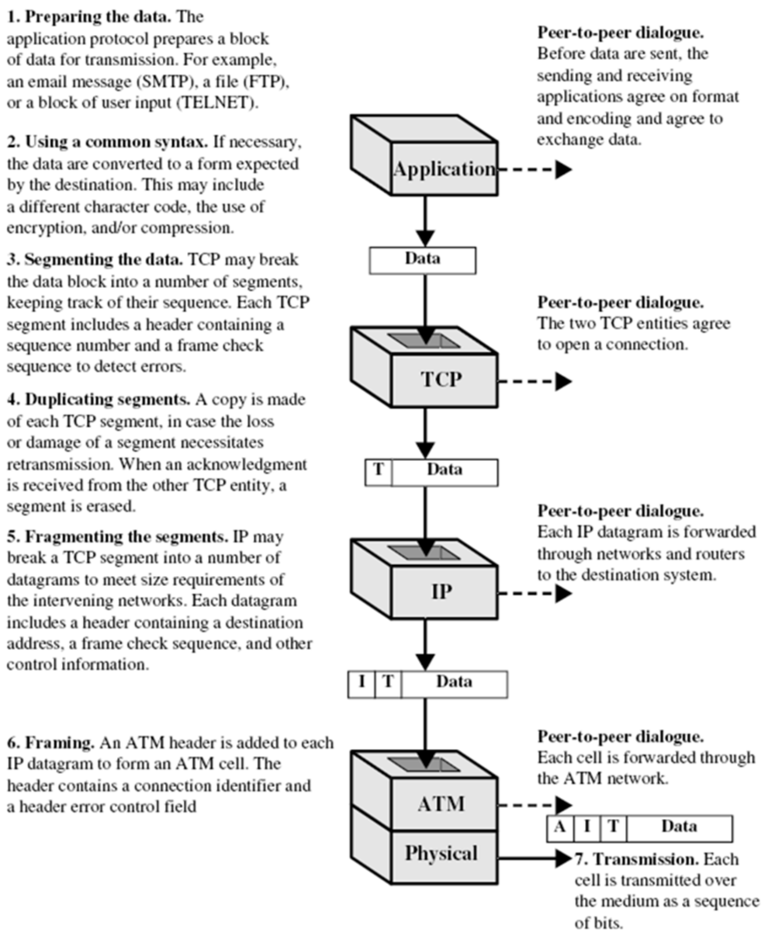




Design Issues for Layered Software

- There are a number of key design issues common to a several layers:
  - *Addressing* – Each layer needs to be able to identify *senders* and *receivers*. Some form of *addressing* is required
  - *Error control* - The *receiver* must be able to tell the *sender* which messages have been correctly received and which have not
  - *Sequencing* - The protocol software on the the receiver must be able to resequence incoming messages
  - *Flow Control* – The receiver must be able to control the flow of information from the sender
- The following are two examples of network *models* namely the **OSI** and **TCP/IP** reference models
  - These form the basis for many of today's *network architectures*

Operation of TCP/IP - Sender



Operation of TCP/IP – Router and Receiver