Network Software

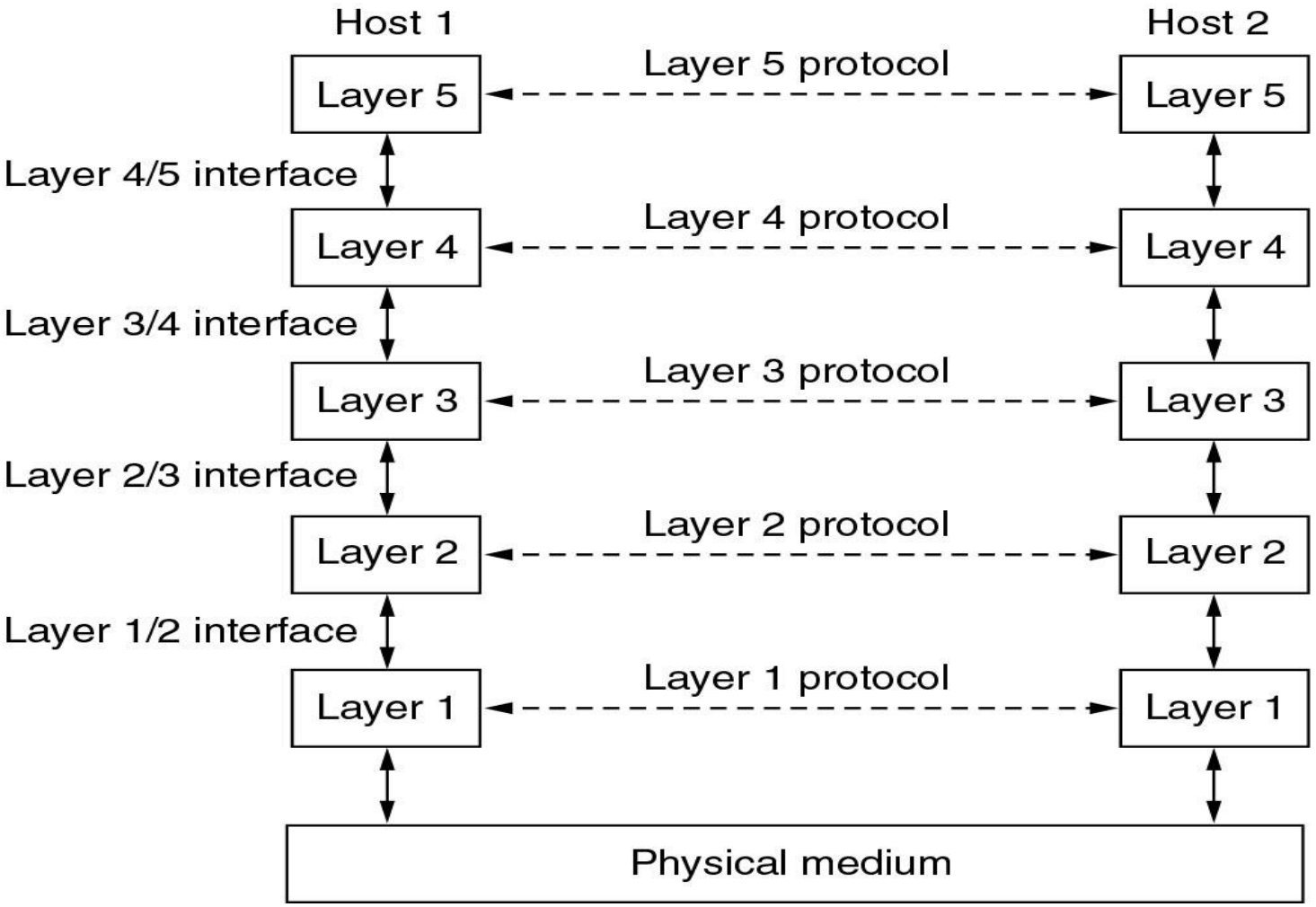
- Network software is now days are highly structured.
- Here, we examine the software structuring techniques

1. Protocol Hierarchy
2. Design Issues for the Layers
3. Connection-Oriented and Connectionless Services
4. Service Primitives
5. The Relationship of Services to Protocols

1. Protocol Hierarchy

- To reduce design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it.
- The name of each layer, its content and functions, the number of layers, differ from network to network.
- The purpose of each layer is to offer certain services to the higher layers.
- Layer ***n*** in one machine carries on a conversation with layer ***n*** on another machine, the rules and conventions used in this conversation are collectively known as layer ***n*** "protocol".

Example: A five layer network



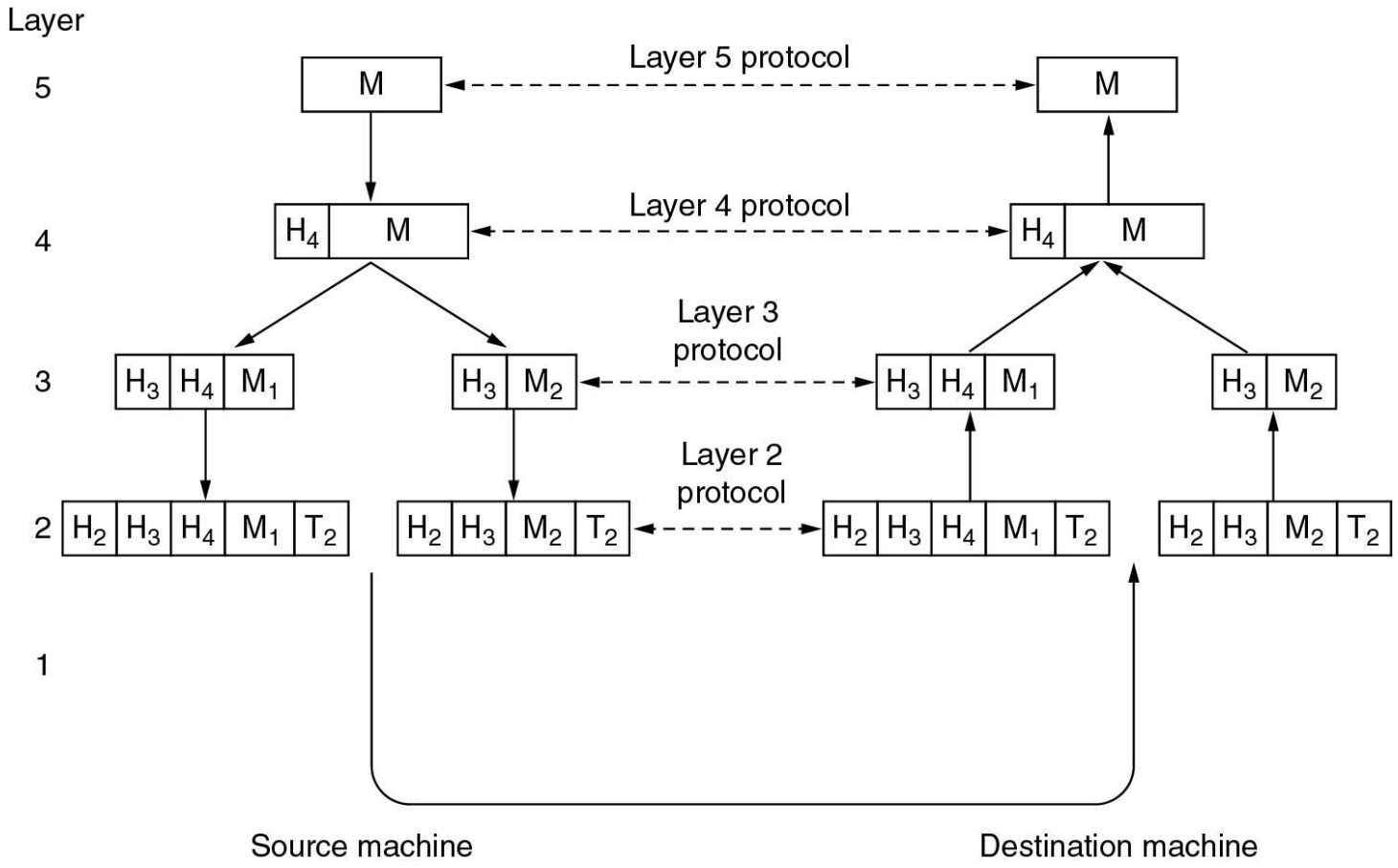
Protocol Hierarchy: Some Terminology

- **Layered structure**:
  - **Protocol** - an agreement between the communicating parties on how communication is to proceed.
  - **Peers -** entities that locally implement the functionality of a given layer ; peers communicate by using the protocol
  - **Interface -** the set of primitive operations and services that lower layer provides to the upper.
  - **Physical media -** through it the actual communication occurs, there is the signal carrier.
  - **Network architecture -** the set of layers and protocols; ignores the interfaces as the interfaces of the hosts in a network may differ.
  - **Protocol stack -** the list of protocol hierarchy in the network; matches the layered structure.

How to provide communication to the top layer of a five-layer network?

- A message, M, is produced by an application process running in layer 5 and given to layer 4 for transmission.
- Layer 4 puts a header in front of the message to identify the message and passes the result to layer 3.
- The header includes control information, such as sequence numbers, to allow layer 4 on the destination machine to deliver messages in the right order if the lower layers do not maintain sequence.
- In most networks, there is nearly always a limit imposed by the layer 3 protocol, so layer 3 must break up the incoming messages into smaller units, packets, adding a layer 3 header to each packet - here M is split into two parts, M_1 and M_2.
- Layer 2 adds not only a header to each piece, but also a trailer, and gives the resulting unit to layer 1 for physical transmission.
- At the receiving machine the message moves upward, from layer to layer, with headers being stripped off as it progresses. None of the headers for layers below n are passed up to layer n.

Fig: Communication to the top layer of a five-layer network



2. Design Issues for the layers

- Design issues can be discussed under the following topics:

1. Addressing
2. Error Control
3. Flow Control
4. Multiplexing
5. Routing

2.1 Addressing

- A network normally has many computers, some of which have multiple processes.
- A mechanism is needed for a process on one machine to specify with whom it wants to talk.
- As a consequence of having multiple destinations, some form of addressing is needed in order to specify a specific destination.
- There should be two addresses:
  - Destination Address
  - Source Address

2.2 Error Control

- Physical communication circuits are not perfect; errors occur.
- Many error-detecting and error-correcting codes are known, but both ends of the connection must agree on which one is being used.
- The receiver must have some way of telling the sender which messages have been correctly received and which have not.

2.3 Flow Control

- Not all communication channels preserve the order of messages sent on them.
- The protocol must make explicit provision for the receiver to allow the pieces to be reassembled properly.
- How to keep a fast sender from swamping a slow receiver with data; either have some kind of feedback from the receiver to the sender, or limit the sender to an agreed-on transmission rate

2.4 Multiplexing

- Sometimes it is expensive to set up a separate connection for each pair of communicating processes.
- So the underlying layer may decide to use the same connection for multiple, unrelated conversations – multiplexing
- Must be done transparently and can be used by any layer.
- Needed in the physical layer, for example, where all the traffic for all connections has to be sent over at most a few physical circuits.

2.5 Routing

- When there are multiple paths between source and destination, a route must be chosen.
- Sometimes this decision must be split over two or more layers.
- A low-level decision might have to made to select one of the available circuits based on the current traffic load.

3. Connection-Oriented and Connectionless Services

- There are the two services given by the layers to layers above them.
- These services are :

1. Connection Oriented Service
2. Connectionless Services

3.1 Connection-Oriented Services

- There is a sequence of operation to be followed by the users of connection oriented service. These are :

    1. Connection is established
    2. Information is sent
    3. Connection is released

- In connection oriented service we have to establish a connection before starting the communication.
- When connection is established we send the message or the information and then we release the connection.

3.2 Connectionless Services

- Each message (letter) carries the full destination address, and each one is routed through the intermediate nodes inside the system independent of all the subsequent messages.
- When the intermediate nodes receive a message in full before sending it on to the next node, this is called **store-and-forward switching.**
- The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called **cut-through switching**.

4. Service Primitives

- A service is formally specified by a set of primitives (operations) available to a user process to access the service.
- Primitives:
  - request some elementary service action;
  - inform the service process for some event in the peer entity
- When the protocol stack is located in the operating system, the primitives are normally **system calls**
- The primitives for connection-oriented service are different from those of connectionless service

Service primitives for implementing a simple connection-oriented service

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
ACCEPT	Accept an incoming connection from a peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

5. Relationship Services - Protocols

- Services and protocols are distinct concepts
- A **service** is a set of operations that a layer provides to the layer above it
- The service defines **what operations** the layer is prepared to **perform** on behalf of its users
- A service says **nothing** at all about **how** these operations are **implemented**
- A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user

Relationship Services - Protocols

- A **protocol** is a **set of rules** governing the **format and meaning** of the **packets/messages** that are exchanged by the peer entities within a layer
- Entities use protocols to implement their service definitions
- Entities are free to change their protocols at will, provided they do not change the service visible to their users
- the service and the protocol are completely decoupled
- **services** relate to the **interfaces between layers**
- **protocols** relate to the packets sent **between peer entities** on different machines

Relationship Services - Protocols