Layered Network Architecture and Protocol Stack

1 Introduction to Layering

A **layered architecture** organizes a complex system into distinct layers, each implementing a specific service through its internal actions while relying on the services provided by the layer below.

Key Idea

Each layer:

- Implements a service via internal-layer actions.
- Uses services provided by the layer below.
- Provides abstraction, hiding internal details from upper layers.

1.1 Why Layering?

Layering is a systematic approach to designing and discussing complex systems:

a) Explicit structure allows identification and understanding of system components.

b) Provides a **reference model** for discussion.

c) **Modularization** simplifies maintenance and updating.

d) A change in one layer's implementation is transparent to other layers.
Example: Changing gate procedures does not affect other system parts.

2 Layered Internet Protocol Stack

The Internet protocol stack consists of five layers, each responsible for specific tasks:

1. **Application Layer:** Supports network applications.
Examples: HTTP, IMAP, SMTP, DNS.

2. **Transport Layer:** Provides process-to-process data transfer.
Examples: TCP, UDP.

3. **Network Layer:** Responsible for routing datagrams from source to destination.
Examples: IP, routing protocols.

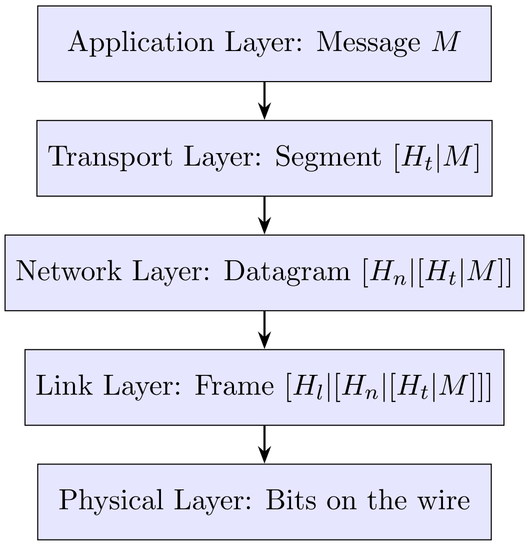
4. **Link Layer:** Transfers data between neighboring network elements.
Examples: Ethernet, 802.11 (WiFi), PPP.

5. **Physical Layer:** Transmits raw bits over a physical medium.

Layer Interaction Example

- Application exchanges messages using the transport layer services.
- Transport layer encapsulates application message M with transport header H_t to form a segment $[H_t|M]$.
- Network layer encapsulates transport segment with network header H_n to form datagram $[H_n|[H_t|M]]$.
- Link layer encapsulates network datagram with link header H_l to form frame $[H_l|[H_n|[H_t|M]]]$.

3 Data Encapsulation Across Layers


Application Layer: Message M
Transport Layer: Segment $[H_t|M]$
Network Layer: Datagram $[H_n|[H_t|M]]$
Link Layer: Frame $[H_l|[H_n|[H_t|M]]]$
Physical Layer: Bits on the wire

4 Summary

- Layering simplifies design, discussion, and maintenance.
- Each layer provides a specific service, encapsulating data with its own header.
- Encapsulation ensures modularity and transparency, making the system robust and flexible.