High Speed Communication Protocols

High Speed Transport Protocols

Why?

- Distributed processing

  - Generally characterized by client-server interactions

  - operating Systems provide Transparent and high-performance services such as:

    * remote procedure call (RPC)
    * remote memory access
    * remote database queries

  - The performance of transport protocols plays a critical role in distributed computing environments.

    *time-consuming setup procedures (not desirable)
    * multicast or broadcast capability (needed)
    *datagram service (might be preferable)

High Speed Transport Protocols

- Full Motion Video and Video-on-demand
  - these application require high-bandwidth
  - data delayed over a certain period of time might be useless.
- Computer Imaging
  - Medical applications
  - Weather related systems

Transport Protocol Responsibility

- Manage end to end connection between hosts
- Provide reliable data delivery
- In sequence delivery of packets to higher layers
- Provide flow control mechanism

=>Three popular standards:

TCP/IP, UDP/IP, TP4 (OSI Protocol)

Transport Protocol Functions

- The **objective** of the transport layer is to provide end-to-end reliable transmission of data
- To fulfill this objective, the transport layer protocol performs the following:
  - *Connection Management*
  - *Error Control*
  - *Flow Control*
  - *Synchronization*
  - *Transmitting and Receiving*

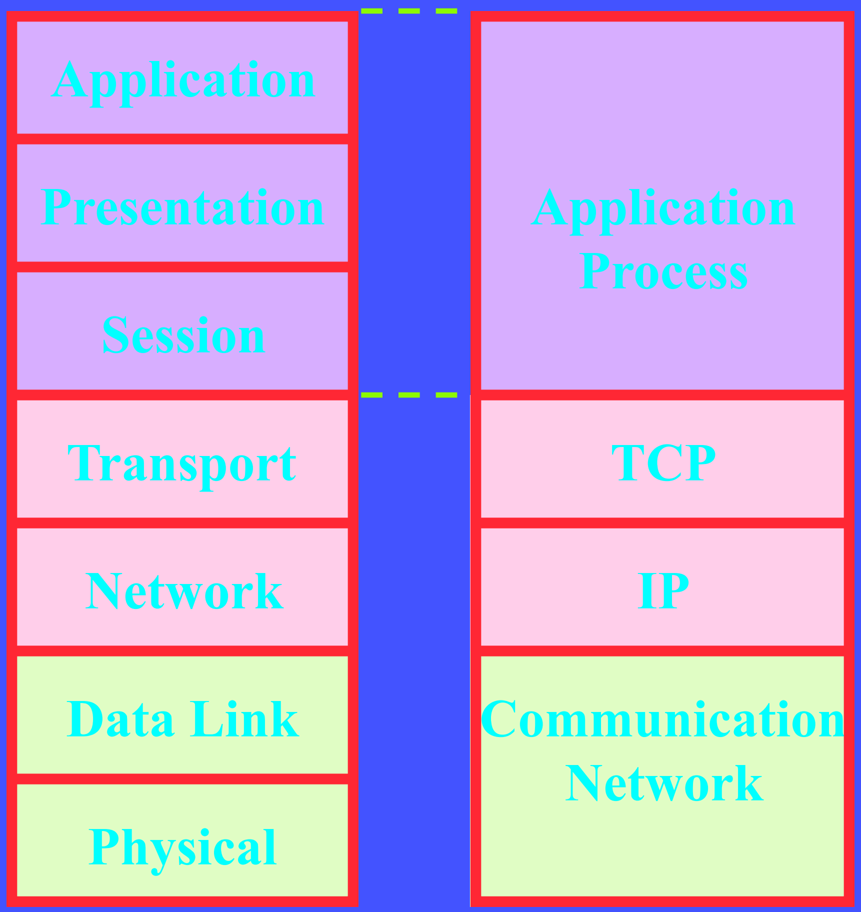
Transport Protocol Functions

- Connection Management
  - How to establish, maintain and release a connection
- Flow Control
  - To ensure that the receiver is not overwhelmed by a fast transmitter, and able to accept and process incoming packets

Transport Protocol Functions

- Error Control
  - Error control mechanisms required to recover from lost, corrupted or out of sequence packets
    - error detection
    - error correction
- Synchronization
  - Source - destination pair should be synchronized
  - it includes acknowledgments and other control data

TCP/IP Suite and OSI Model



Problems with Current Transport Protocols

1. Flow Control Algorithm

- Match data transmission rate with receivers data consumption rate

  Ex: Window controls the flow of data by limiting the number of units that can be transmitted without acknowledgement
- It can convey only how much data can be buffered, rather than how fast the transmission should be
- It ties flow control and error control together
- Go-back-N type of control degrades throughput severely and add unnecessary network congestion
- Flow control should be independent of error control

2. Protocol and Operating Systems Processing

- Most transactions are tied heavily to operating system
- Heavy usage of timers, interrupts, memory bus access degrades performance of CPU

3. Acknowledgement

- Current protocols use accumulative acknowledgement; when you ack N this implies that all packets up to N have been received successfully
- This restriction provides simple, but inefficient ack technique
- Selective ack is more efficient
- Block ack technique has been proposed to improve efficiency

4. Packet Format

- Traditionally packet formats were designed to minimize the number of transmitted bits
  - This has resulted in packets being bit-packed and require extensive decoding
  - Variable packet fields lead to slow processing
- Performance can be improved when packets can be processed in parallel, which requires fixed length packets like in ATM

5. Error Recovery

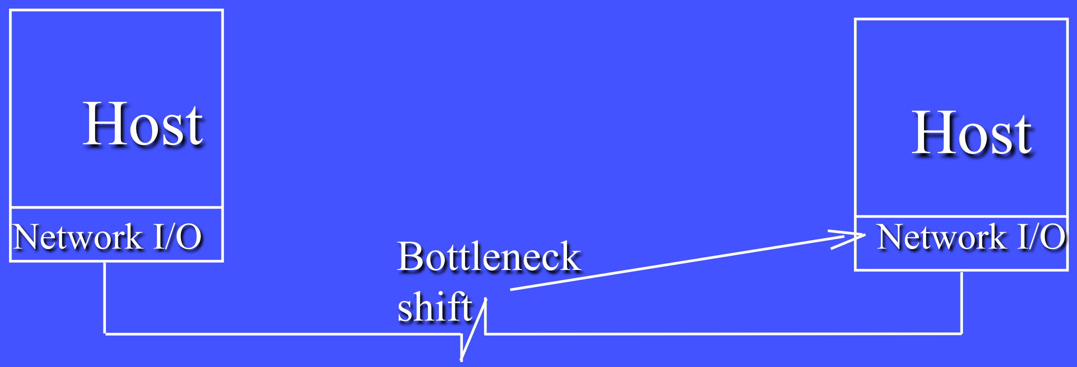
- Error recovery protocols are generally slow
  - When packets are lost, timers trigger the retransmission
  - Variation in Round-Trip Delay (RTD) enforces this to be too long
  - Performance loss is high when RTD is long

6. Flexibility of Protocols

- Existing protocols are not flexible enough for high speed applications and networks
- TCP does not supply mechanism for fast call setup or multicast transmission
  - These primitives are important for distributed computing
- We need to have several types of services over varying network topologies

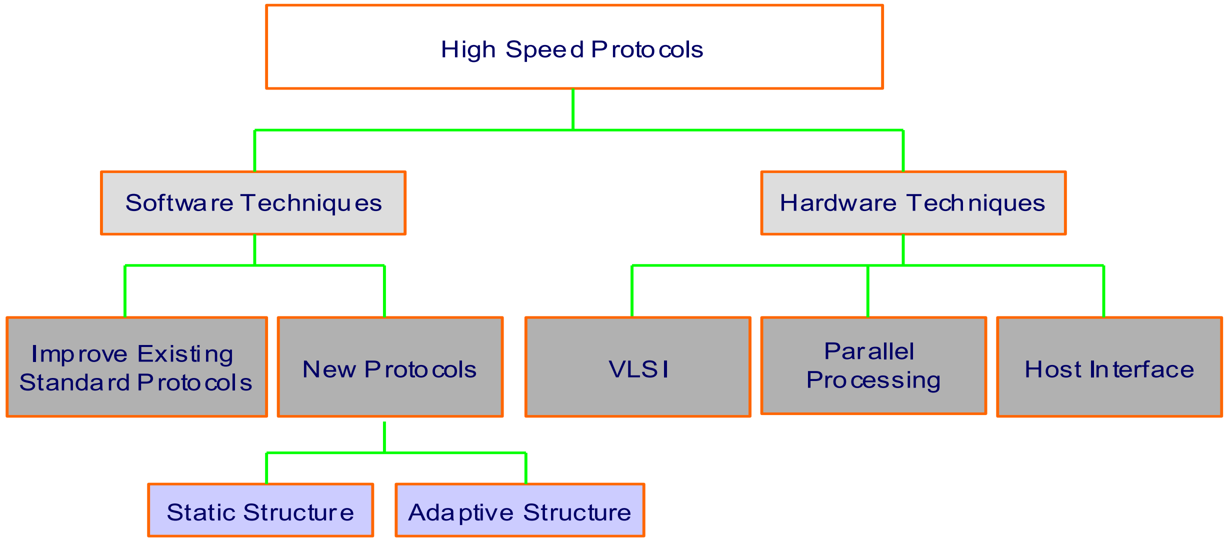
High - Speed Transport Protocols

- Existing standard protocols cannot utilize the performance of emerging high speed networks
  - Network design assumptions: Network is slow and unreliable



- Transfer Time:
  - at 1 kbps, 1M bits need s 1000 second
  - at 1 terrabit per second (10^{12}), 1 M bits needs 1 micro second
  - At one 1 Giga Instructions per second, we have one 1000 instructions to transmit 1 Mbits, while at 1 Kbps we have 10^{12} instructions to transmit the file

Classification of High-Speed Protocols



High Speed Protocol Methods Design Philosophy

- Make the protocol design a success oriented
- Emphasize streamlining data transmission
- Simplify transport protocol: the simpler the receiver, the faster incoming packets can be processed
- Should provide new capability needed for parallel/distributed computing

Architecture Approach

- Modify implementation approach of standard protocols
- Combine layers to facilitate implementing some of the layers in parallel

Hardware Implementation

- Protocol processing is responsible for 20% of all processing time
- The rest is spent on timers, memory access, and handling interrupts
- Separate protocol processing from operating system
- Run these tasks on special network adapter boards

Implementation Techniques in High Speed Transport Protocols

- Connection Management
  - Short lived connection needed
  - Desirable to have explicit connection set-up (e.g. VMTP, Xpress Transfer Protocol XTP); data is transmitted with connection request
  - Response to request can also have ACK data
  - Data can be sent with header (VMTP)

    for large data, difference between implicit and explicit call setup is negligible
  - Protocol should also support virtual connection, datagram and multicasting

Implementation Techniques

- Packet Organization and Packet Size
  - Packet Organization

  - all fields must be of fixed length
  - boundaries of fields must be on (multiples of) bytes or words
  - leads to simpler, faster implementation and simplifies hardware
  - header in proper place allows parallel processing of packets

    header addr, ID first

    two checksums: one for header, one for data

Implementation Techniques

- Packet Size
  - Efficient transmission requires least amount of overhead => burst transmission
  - For gigabit networks, packet size must be large (reduces overhead-to-data ratio)
  - VLSI implementation and use of parallel processing will play important role in achieving high speed transmission rates
- Flow Control
  - Must be *independent* of error recovery
  - Lock-step transmission must be avoided (setting incorrect window size could cause burst traffic, several pauses, while sender waits for permission to continue)
  - Credit scheme or block scheme

High Speed Transport Protocol Techniques

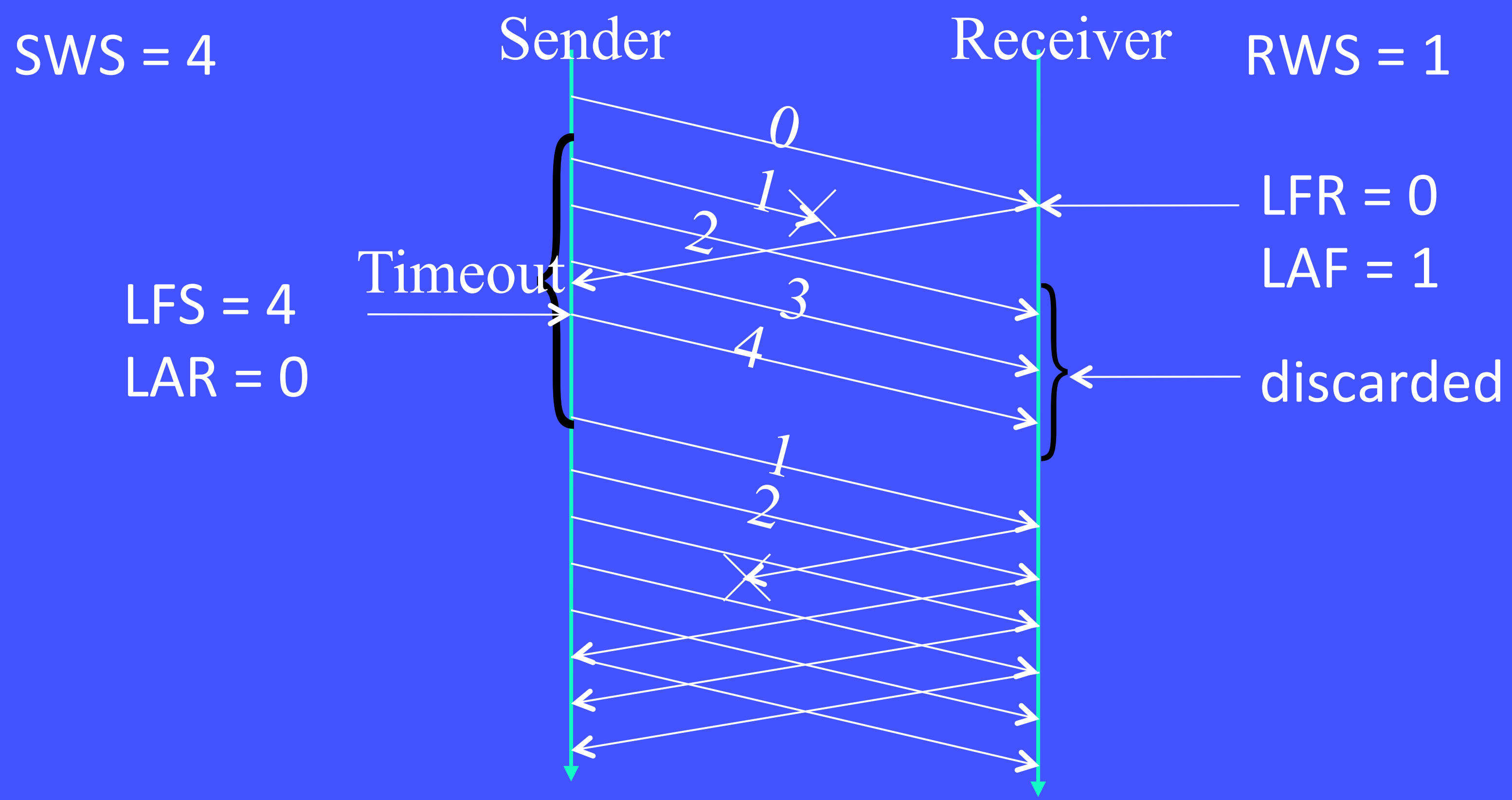
- Error Recovery
  - Retransmission can include an entire window (block) of data or only data lost to error

    Go-back-N vs. Selective Repeat Transmission
  - Go-back-N easy to implement
    - send all packets after receiving an erroneous packet
    - simplifies record keeping and buffer management
    - selective transmission continues storing data after out-of-order packets received

-9

- high-speed networks have low error rate (BER approx 10^{-10})
- packets will be mainly lost due to network overruns and receiver overruns
- selective repeat requires large tables and complex processing

Example of Go Back N

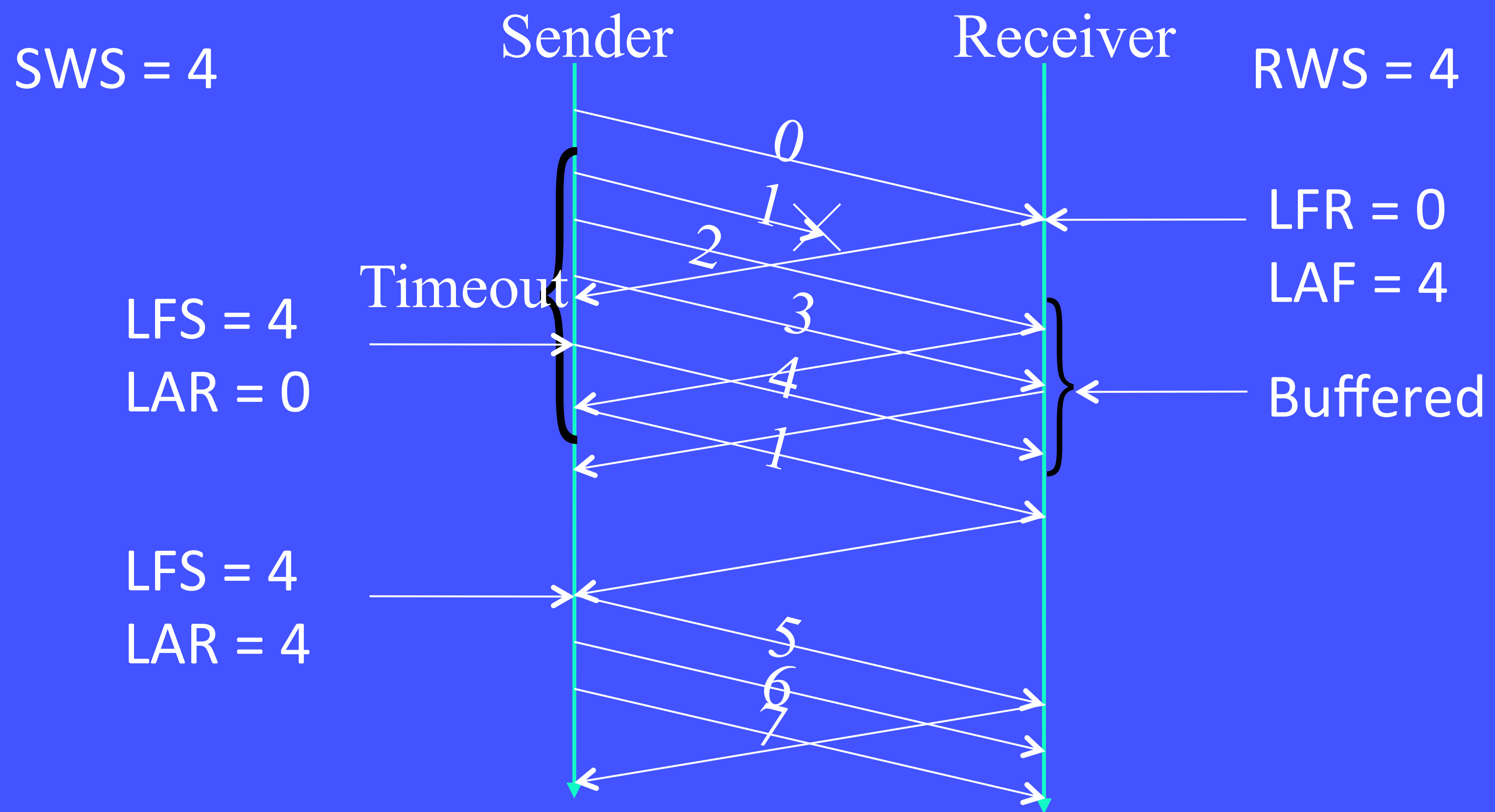


- Max Sequence Number ≥ SWS + 1
- Link does not re-arrange packets

Selective Repeat

- Window size can be very large for nets with large delay x bandwidth
  - $20 \times 10^{-3} \times 10^{12} = 20\ 10^{9}$
- Inefficient to retransmit all N frames if one is lost
- Selective repeat allows the re-transmission of only the lost packets
- Accepts out-of-order packets
- Simply increase the RWS up to SWS (does not make sense to allow for RWS > SWS)

Example of Selective Repeat



- Max Sequence Number ≥ 2 SWS

High Speed Transport Protocol Techniques

Buffer Management

- How much buffering is required?

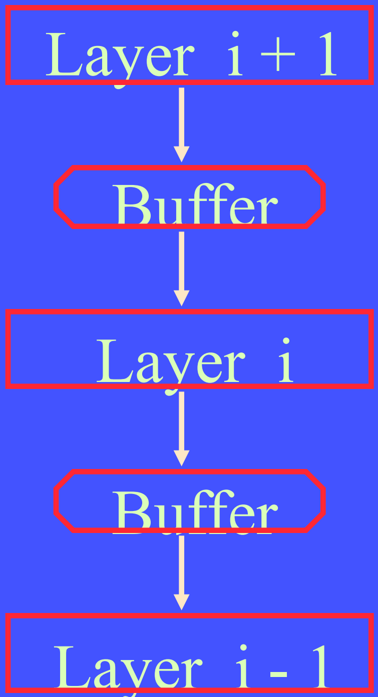
  delay x bandwidth

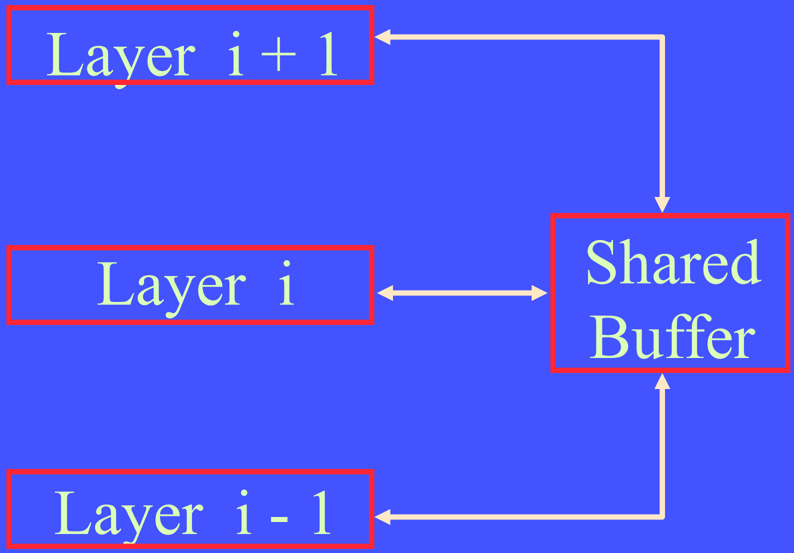
  for one round trip time over longest path

- Study has shown that 50% of TCP processing time is used for network memory copying
- Shared Buffer - Cut Through
  - buffer is shared among all layers
  - use streamlining and pipelining approach
  - map user buffer into network interface buffer

Software Approach: Buffer - Cut Through

- Instead of copying data between layers, data is stored in a shared buffer, only pointers are moved between layers.





Software Based Approach

- Integrated Layer Processing (ILP) (Clark90)
  - All data manipulations of different layers are combined together
- X-Kernel
  - A communication-oriented operating system that supports efficient implementation of standard protocols
  - Based on using upcalls and improved buffer management scheme

Adaptable Protocols

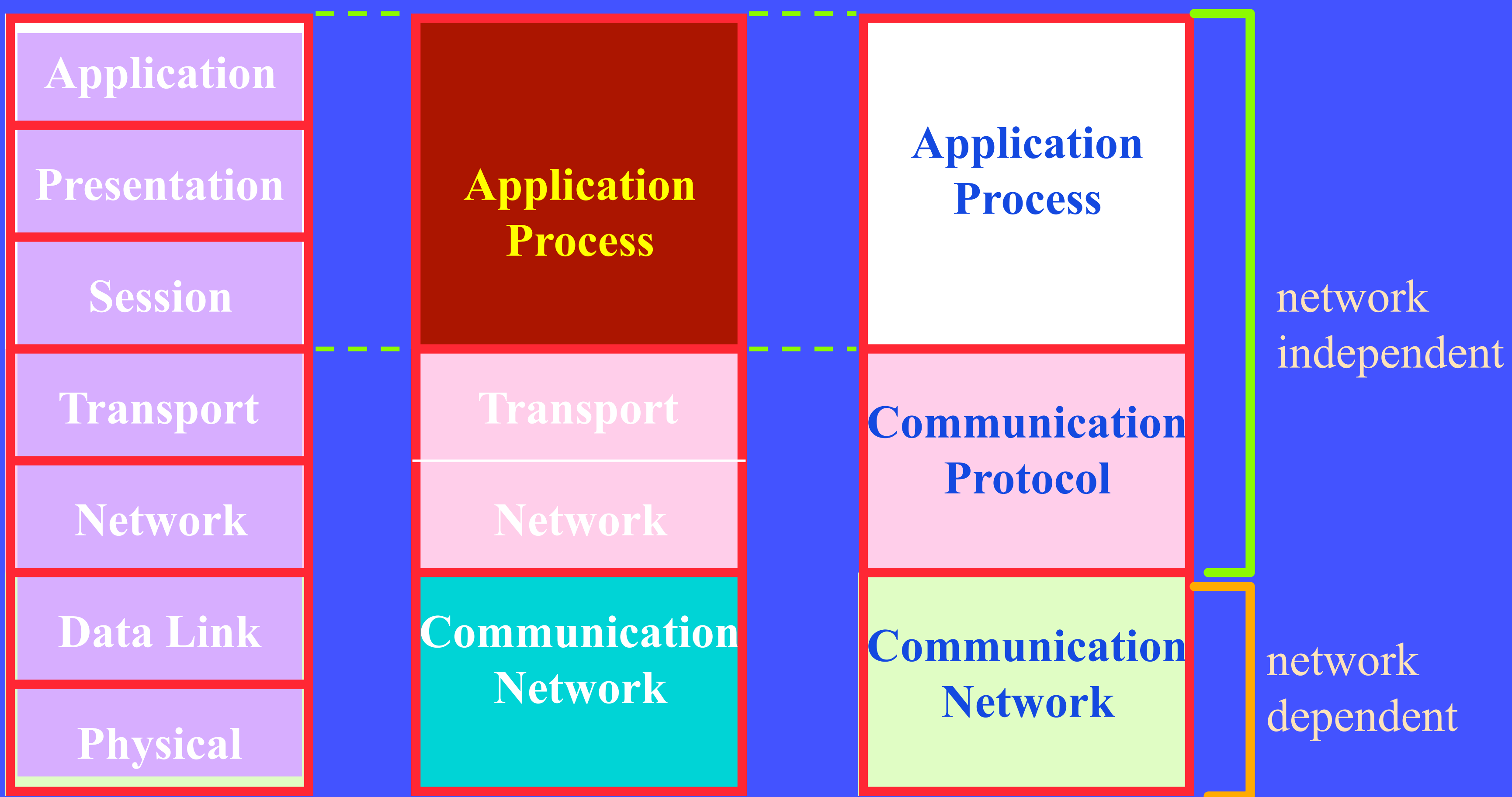
- Existing standard protocols are statically configured
- Emerging applications have diverse requirements
  - delay sensitive (Real time App)
  - bounded delay (RPC)
  - loss tolerance (voice traffic)
- A single "monolithic" protocol suite that integrates all functionalities is not realistic
- Solution : decompose protocols in term of functions instead of layers, functions represent "building blocks" of a protocol

Application Oriented Communication Protocol

- Framework Features
  - Communication protocols are adaptively configured by users to properly match applications requirements and networks characteristics.
  - Use parallel processing to enhance protocols performance
  - Inter - operability with standard protocols
  - Protocols are independent of the hardware platform used

- E. Al-Hajery and S. Hariri, "Application-Oriented Communication Protocols for High-Speed Networks," International Journal of Computers and Applications, Vol. 9, No. 2, 2002, pp. 90-101.
- E. Al-Hajery and S. Hariri, ``Quality of Service Guarantees at End-to-end Transport Protocols, `` *Proceedings of the IPCCC 98.*

Framework Description



Framework Description

- A communication protocol is represented as an abstraction that encapsulates a set of atomic modules, where each module represents a protocol function

```
Communication Protocol : protocol name
        {
        Function : flow control, error control, connection manager,
                   transmitter, receiver, sync, router

        FIC :    /* code that defines the interface between
                    the different functions                    */
        }
```

Communication protocol object

Framework Description

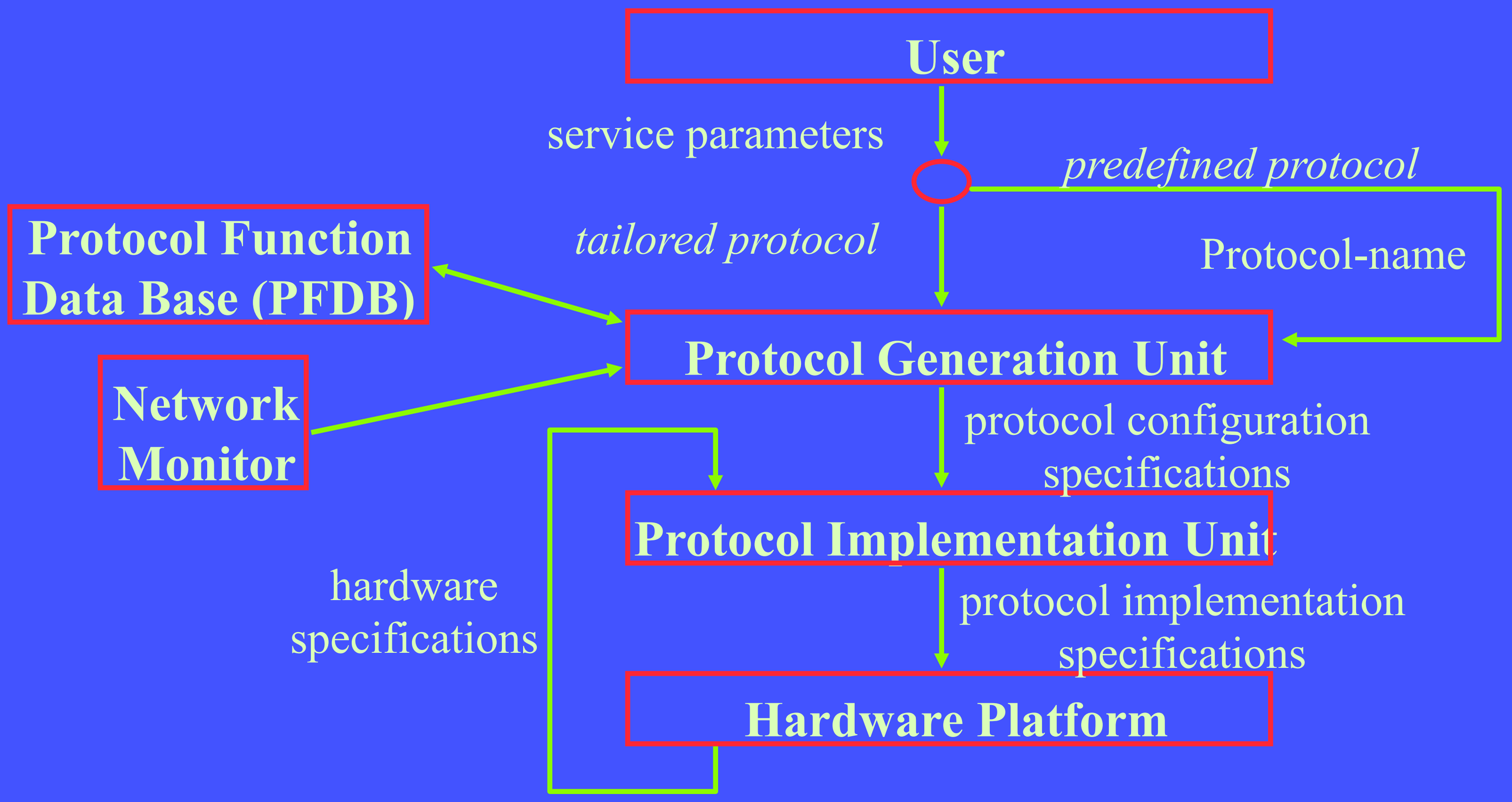
Protocol function class

```
class   function A   {
        . . . .      }
        mechanism 1 : function A          {
        input parameters : in1, in2, . . .
        /* code for mechanism 1 */
                                          }
        mechanism 2 : function A          {
        input parameters : in1, in2, . . .
        /* code for mechanism 2 */
                                          }
        mechanism 3 : function A          {
        input parameters : in1, in2, . . .
        /* code for mechanism 3 */
                                          }
```

- A protocol function class contains the different mechanisms supported by the protocol function

Communication Protocol Construction Process

- Protocol generation
- Protocol implementation

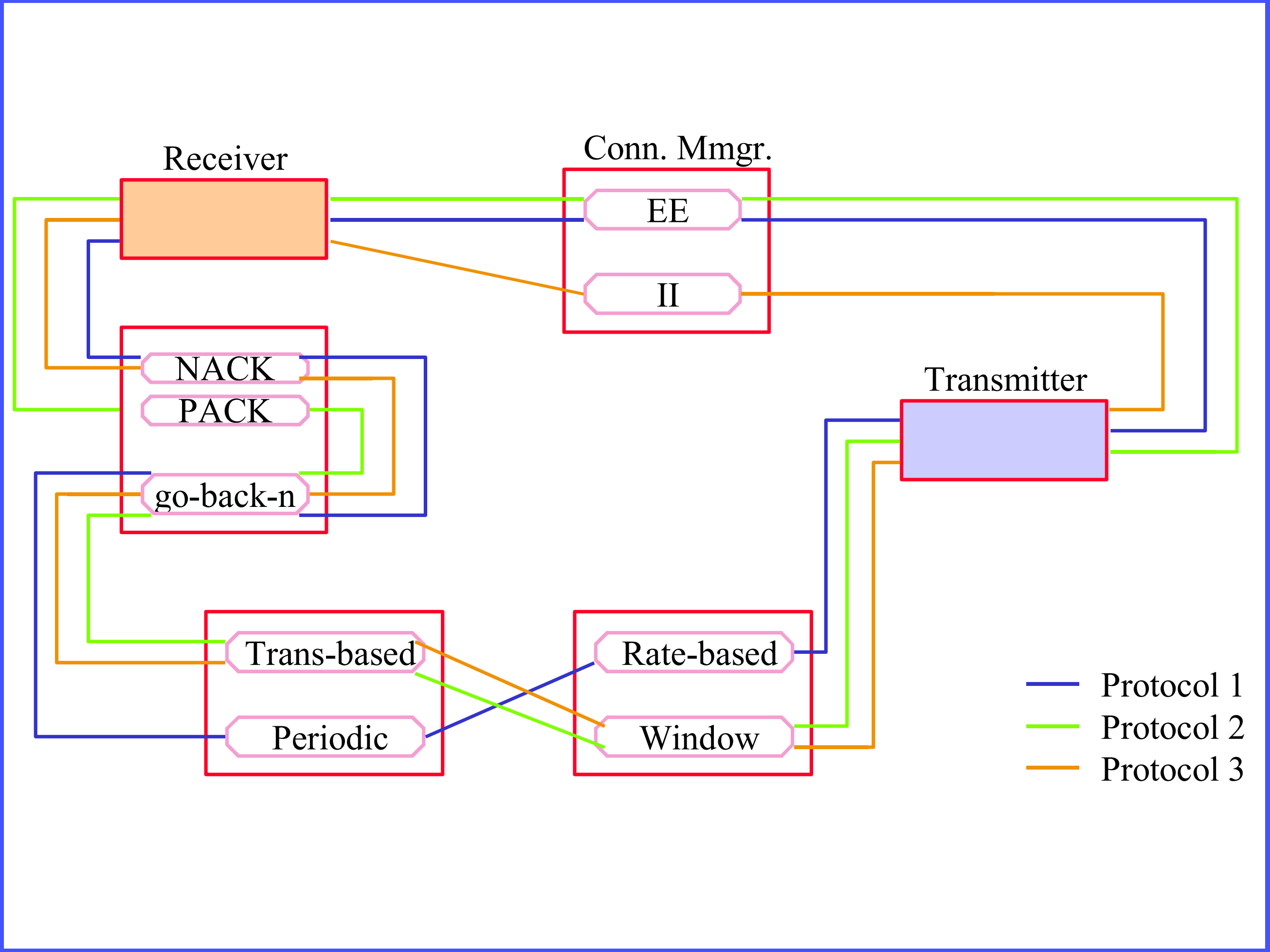


Protocol Configuration

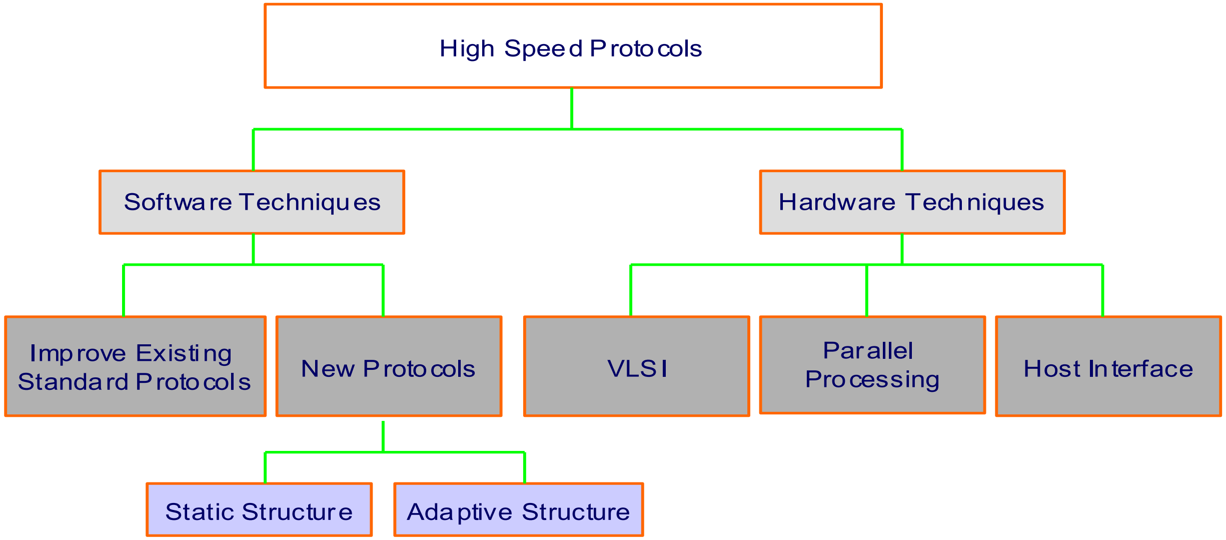
A general structure of configuration code

```
/* . . . . . mechanisms declarations . . . . . */
function : function 1
        mechanism : mechanism A
                input : in1, in2, . . .
function : function 2
        mechanism : mechanism C
                input : in1, in2, . . .
function : function 3
        mechanism : mechanism B
                input : in1, in2, . . .
function : function 4
        mechanism : mechanism D
                input : in1, in2, . . .
/* . . . . .  mechanisms operations  . . . . . */
on transmit (packet) do
        {mechanism A, mechanism C, mechanism D }
on receive (packet) do
        {mechanism A, mechanism B, mechanism C }
```

Application Based Protocol



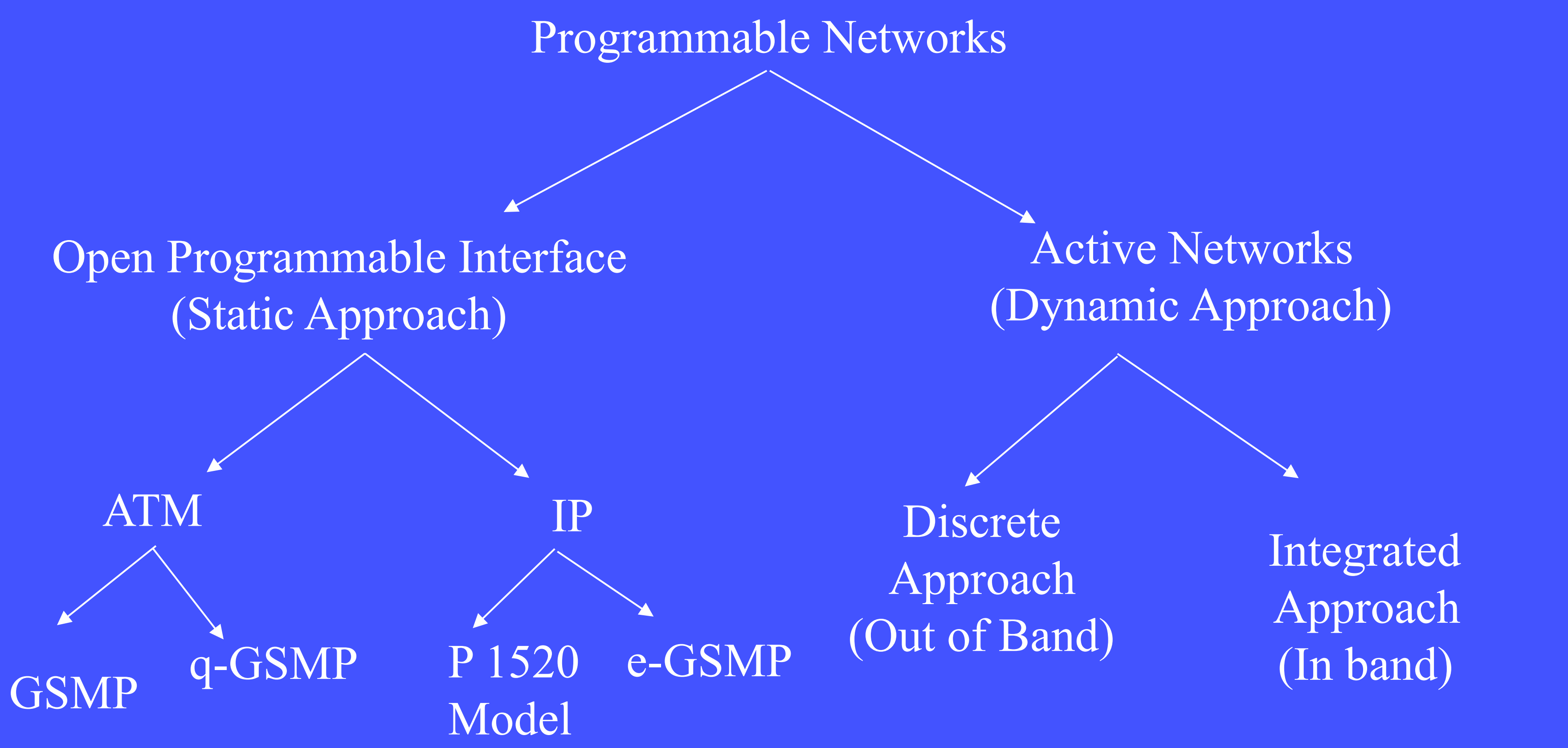
High-Speed Protocol Methods



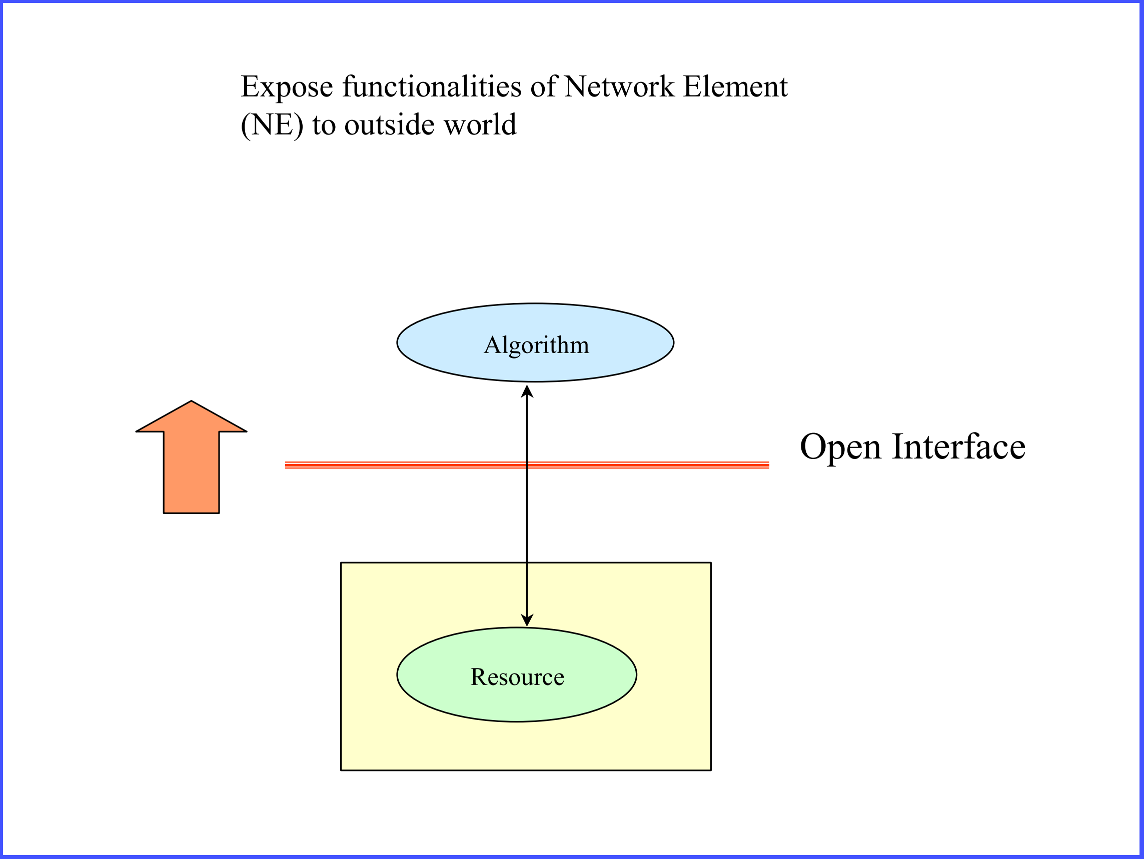
Hardware Based Techniques; Why Programmable Networks?

- Rapid creation, deployment and management of new services in response to user demands.
- Change in the nature of traffic due to the wide variety of applications and services.
- Application specific demands for resources.
- Need for the separation of communication hardware from control software.
- Better control over the network resources for its effective use.

Classification of Programmable Networks



Open Programmable Interface



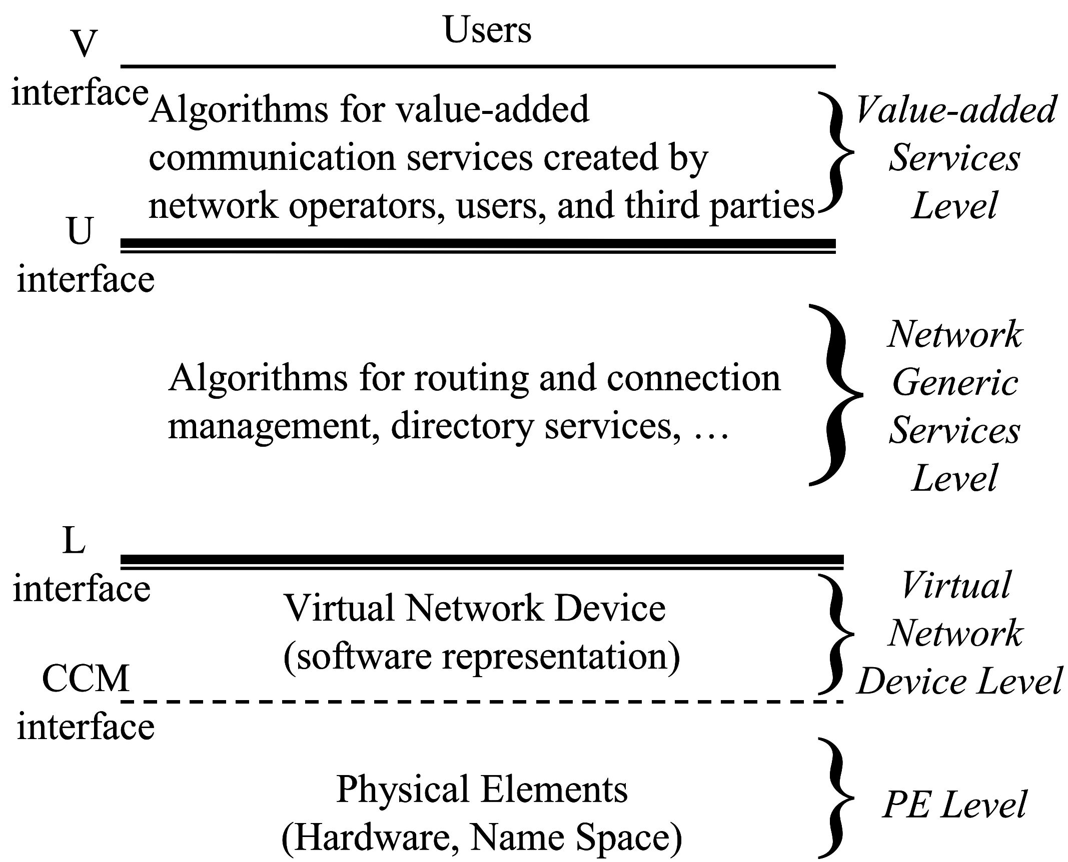
Open Interface Networks

- Provides abstractions in the layers of a node to define programmable interfaces.
- Allows applications and middle-ware to manipulate low-level network resources.
- Uses APIs to control the various layers.

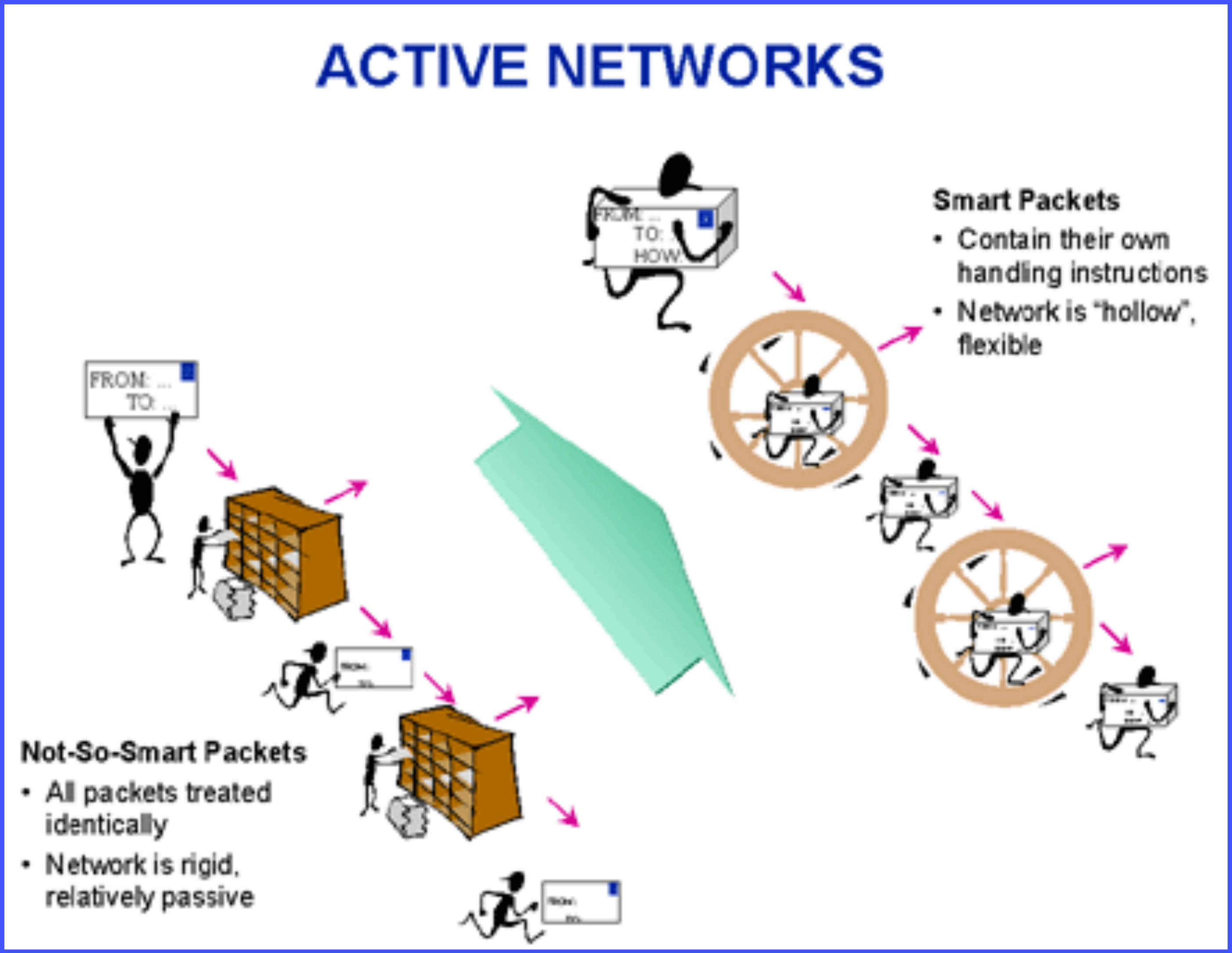
- **Advantages:**
  - Separation of service business.
  - Separation of vendor business.
  - Faster standardization.
  - Extensibility
  - Richer Semantics

IEEE P1520 Reference Model

P1520 Reference Model

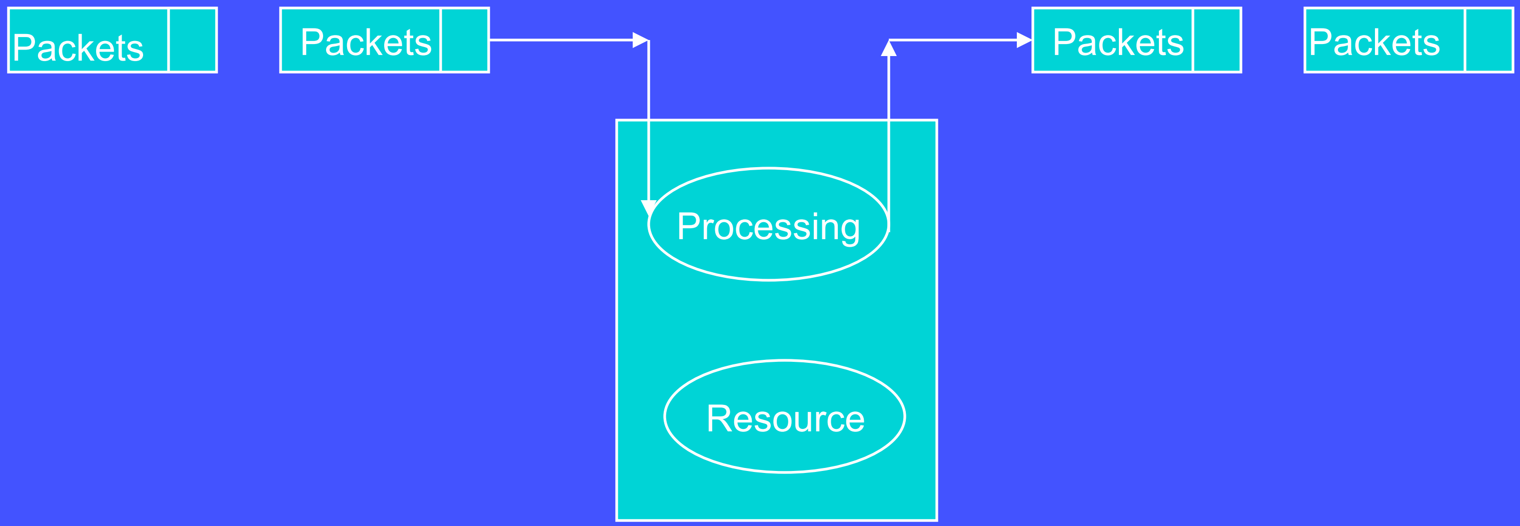


Active Networks

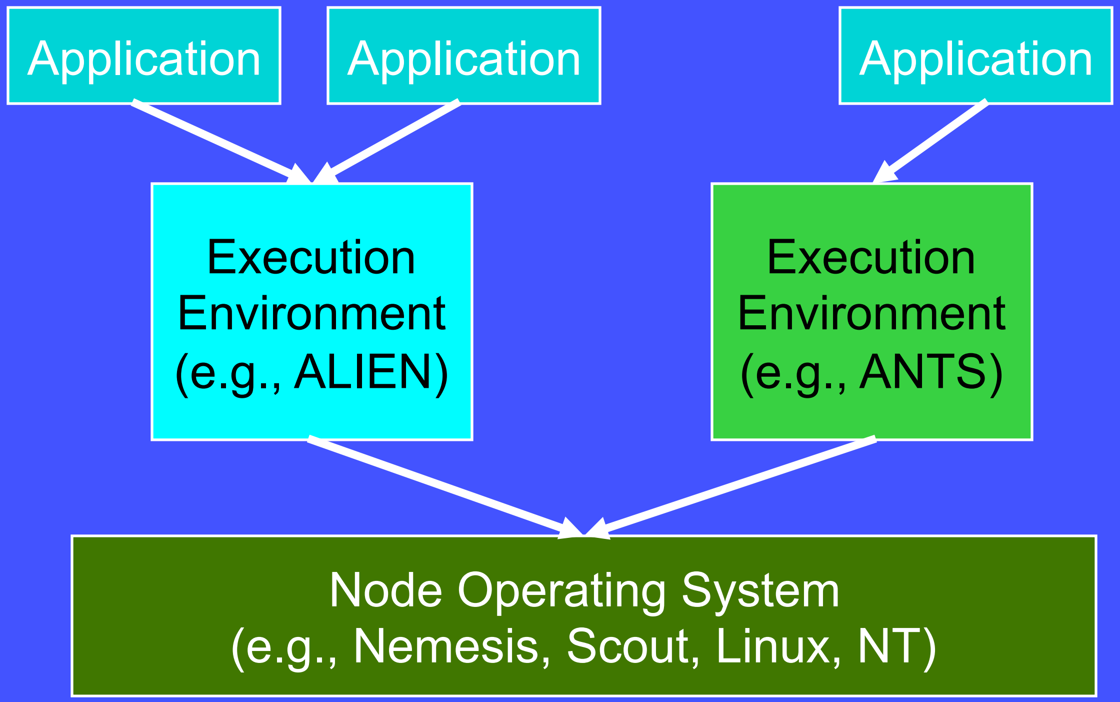


Active Networks

- Packets carry instructions regarding its processing.
- Provides for encapsulation of atomic programs in the packets.

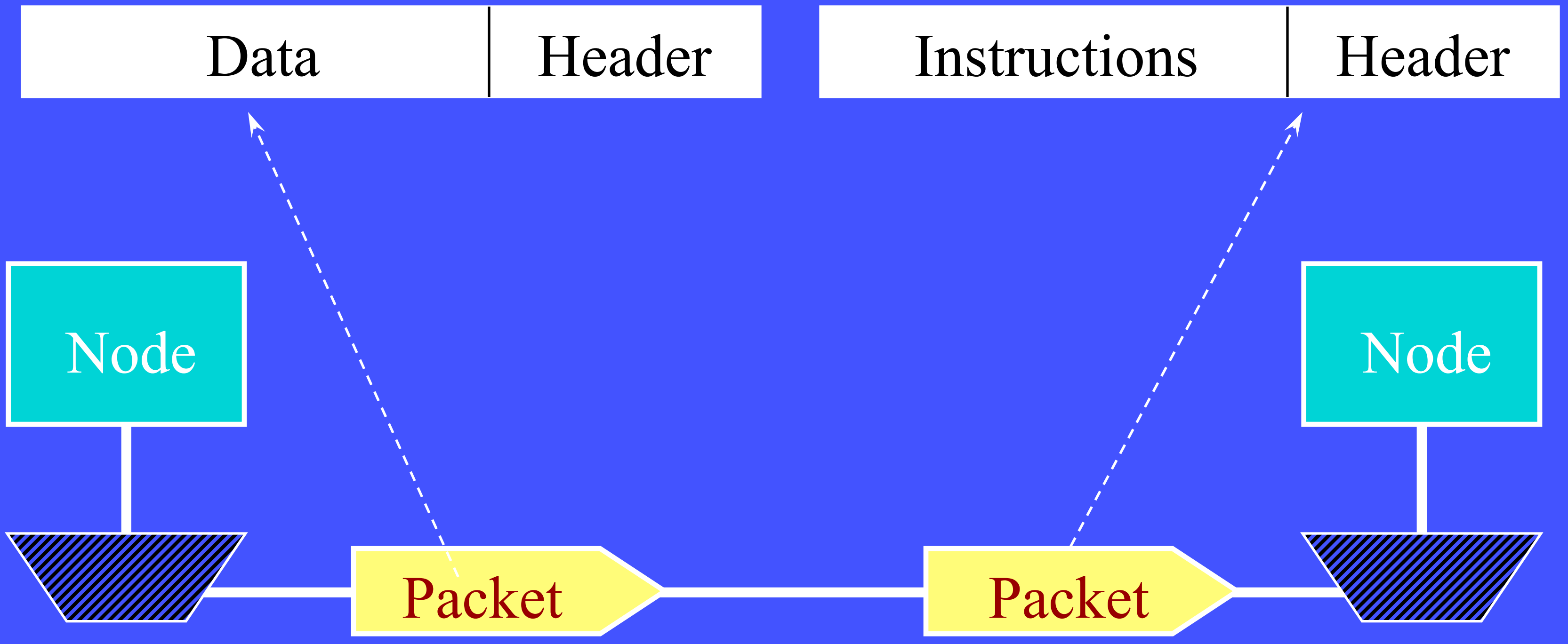


Elements of Active Networks



Discrete Approach

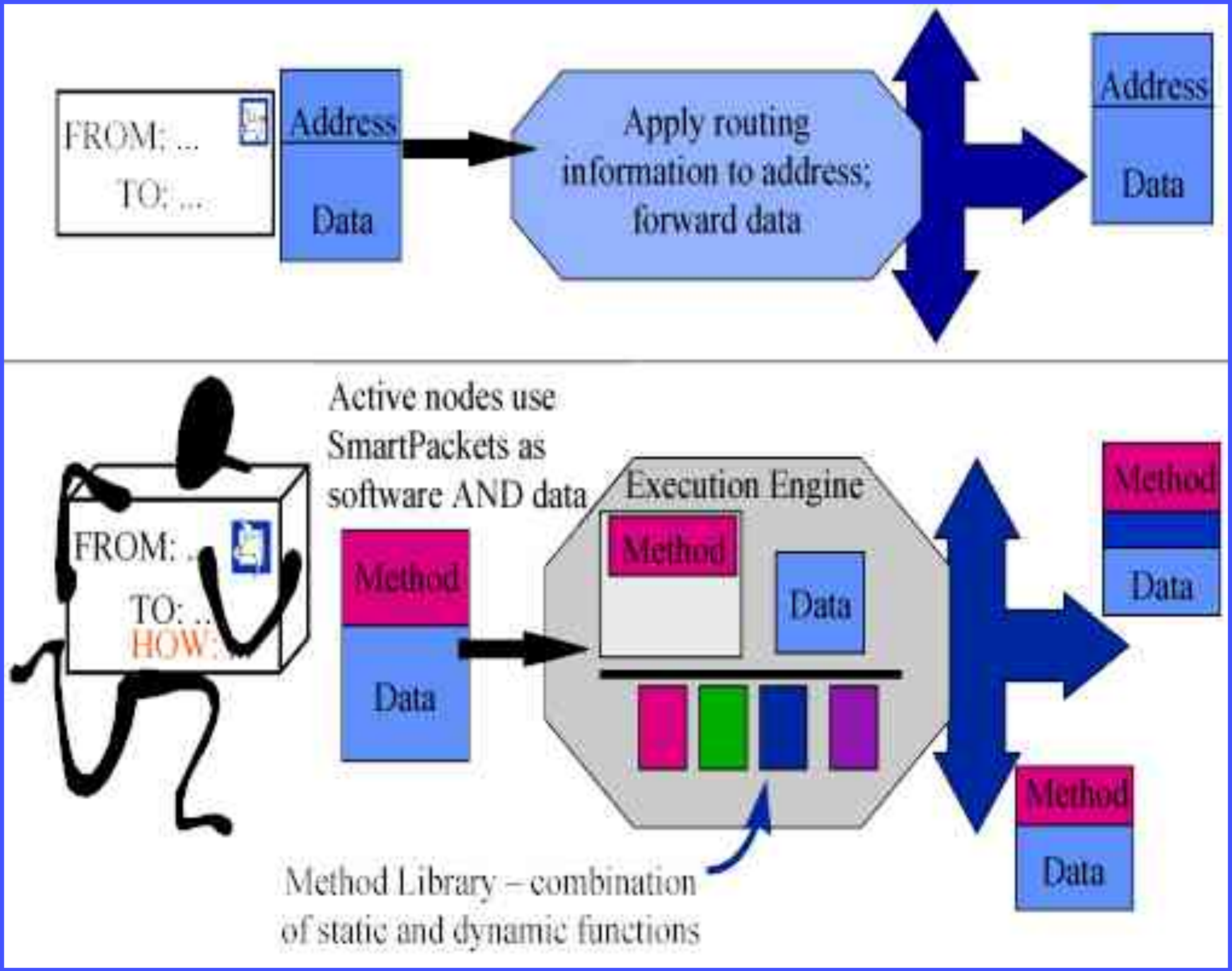
- The code injection is done out-of-band.
- The packets carrying the instructions configure the node.
- Subsequent data packets are processed by the node as per the configuration.



Integrated Approach

- The code injection happens in-band.
- Each packet carries information regarding the type of processing needed by it.
- Data and instructions are present in the same packet.
- Instructions are packet specific.

Integrated Approach



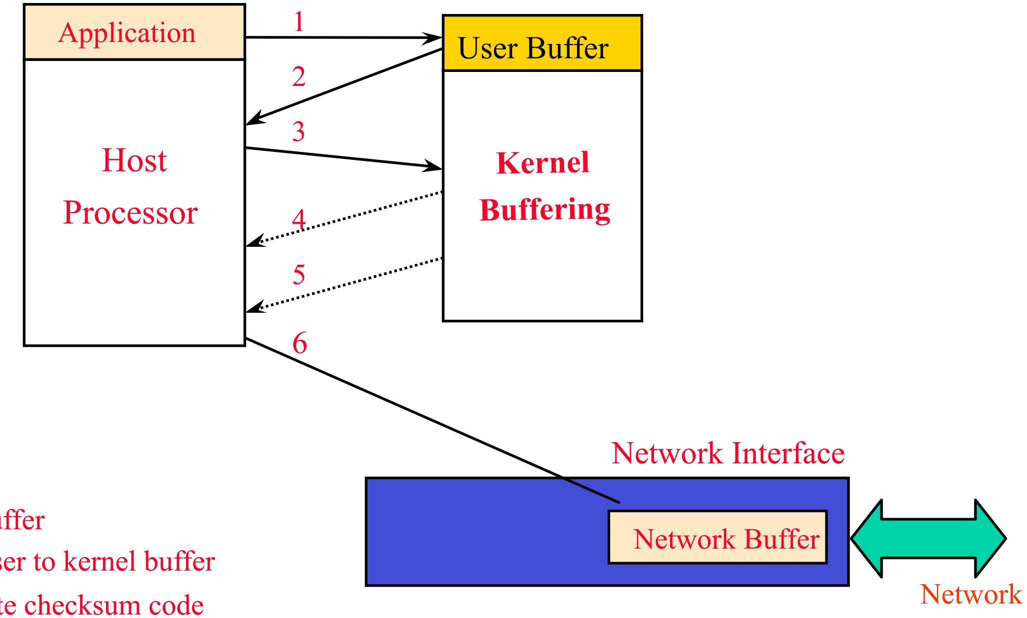
Active Networks

- **Advantages:**
  - Dynamic injection of code for realization of an application specific service logic.
  - Allows for rapid provision and update of protocol stacks by third parties.
  - Allows for services to be tailored to current network conditions.
- **Concerns:**
  - Security
  - Throughput

Hardware Based Approach

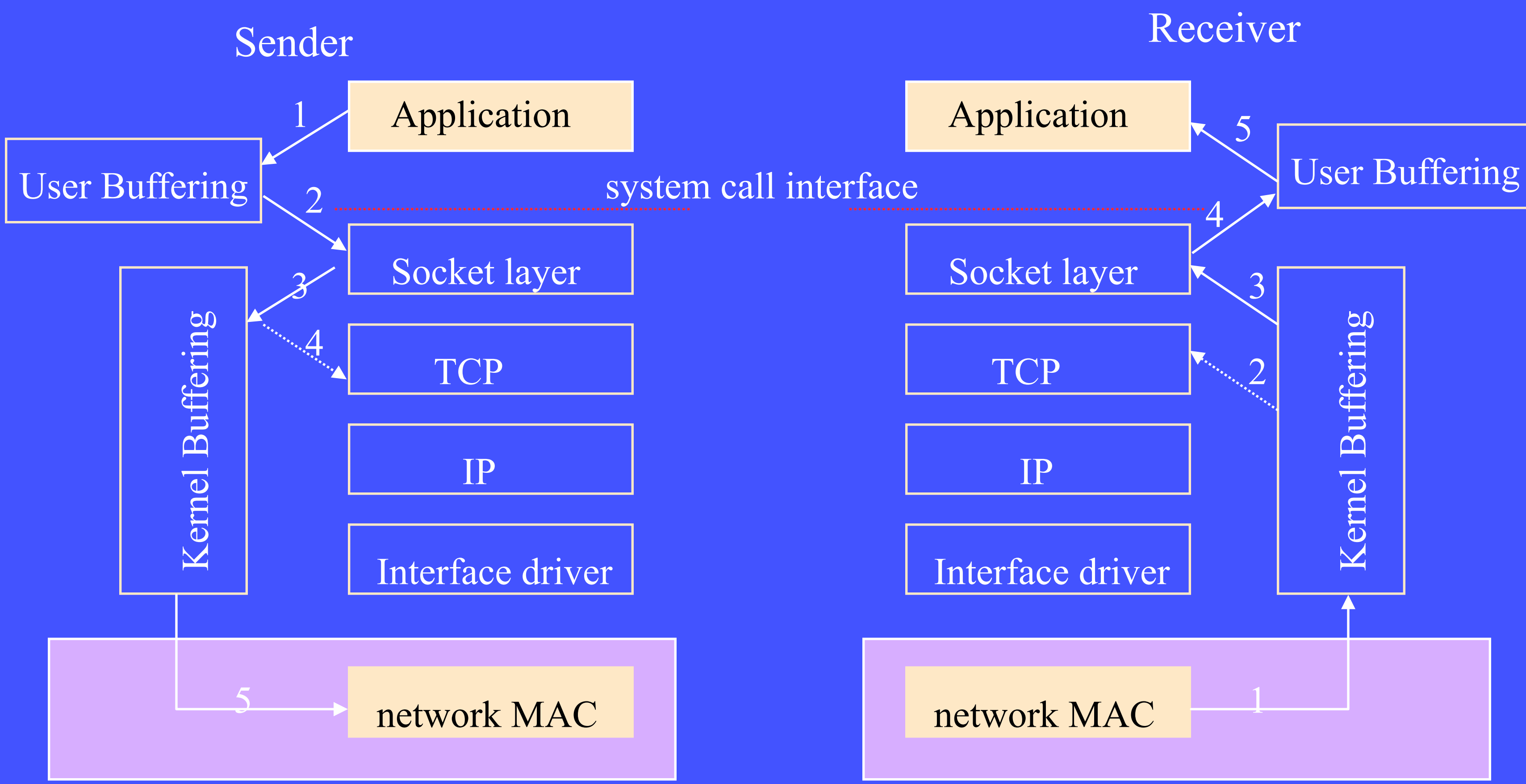
- VLSI Approach
  - XTP is designed and implemented using a VLSI chip set
  - XTP stream protocol functions, combine the transport and network layers and utilize VLSI and parallel processing capability

Host Network Interface



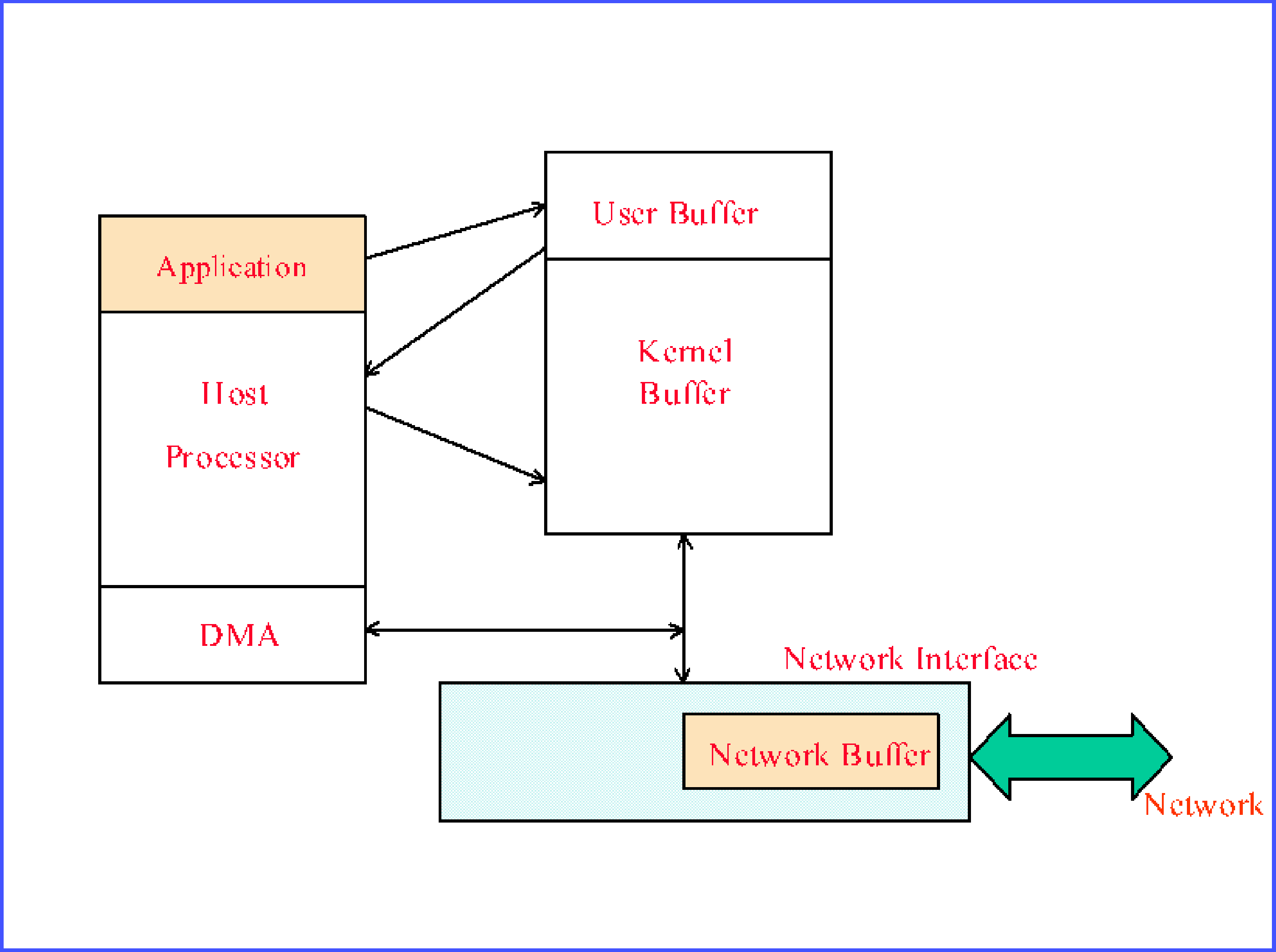
1 Write data to user buffer
2,3 Move data from user to kernel buffer
4 Read data to calculate checksum code
5,6 Move data to network buffer

Host Network Interface

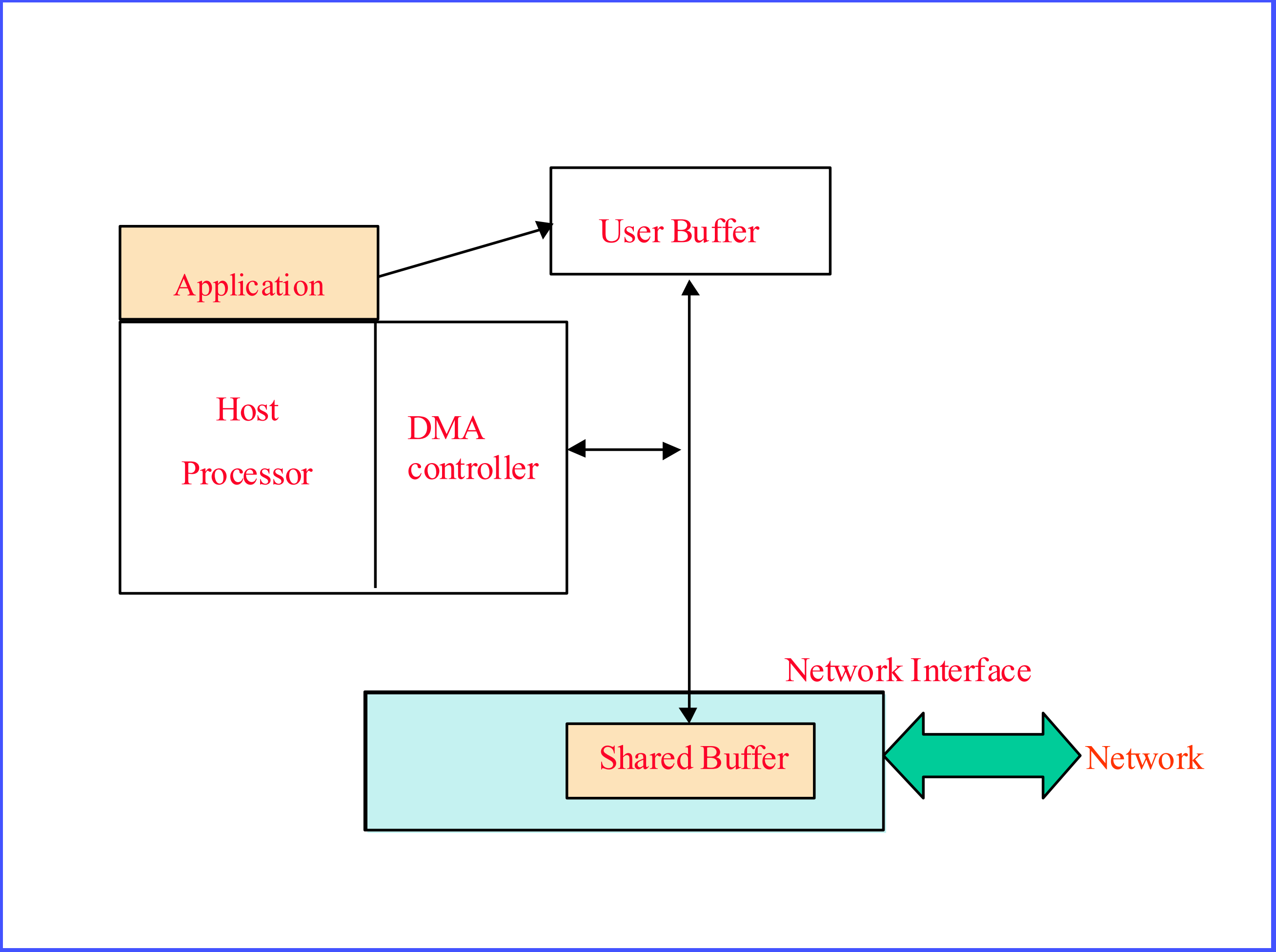


Data path in a conventional protocol stack

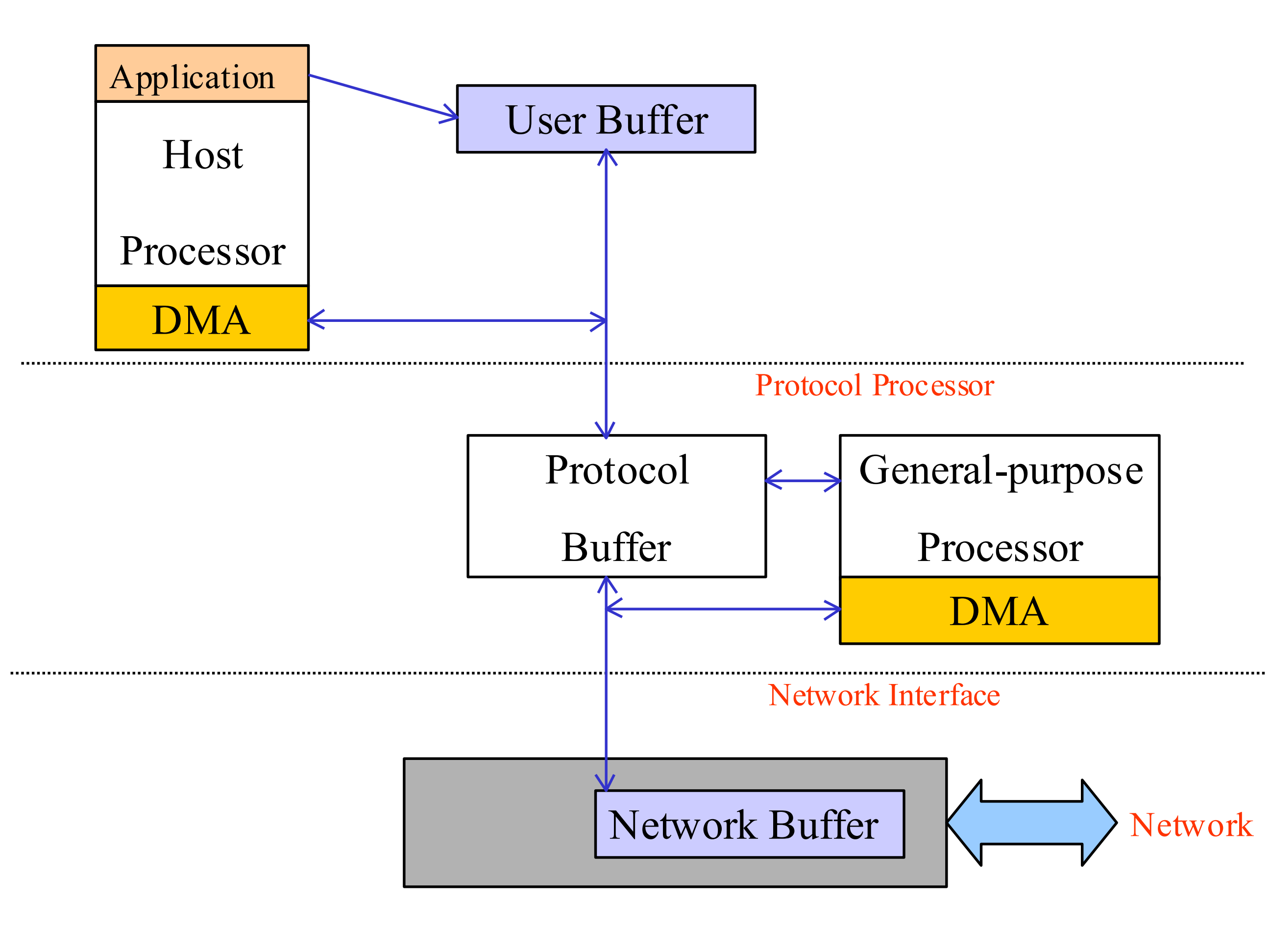
DMA Based Host Network Interface



Zero Copying Host Network Interface



Off loading Protocol Processing

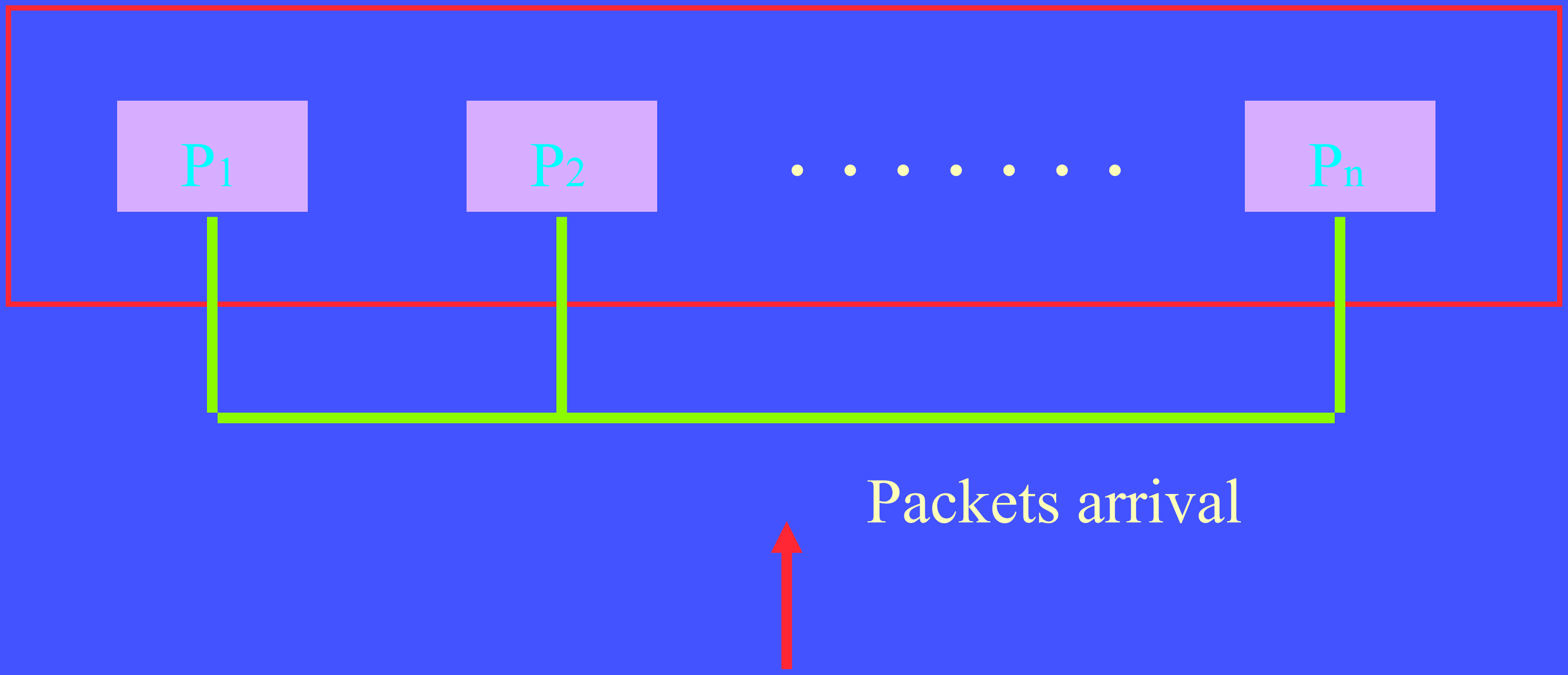


Parallel Processing

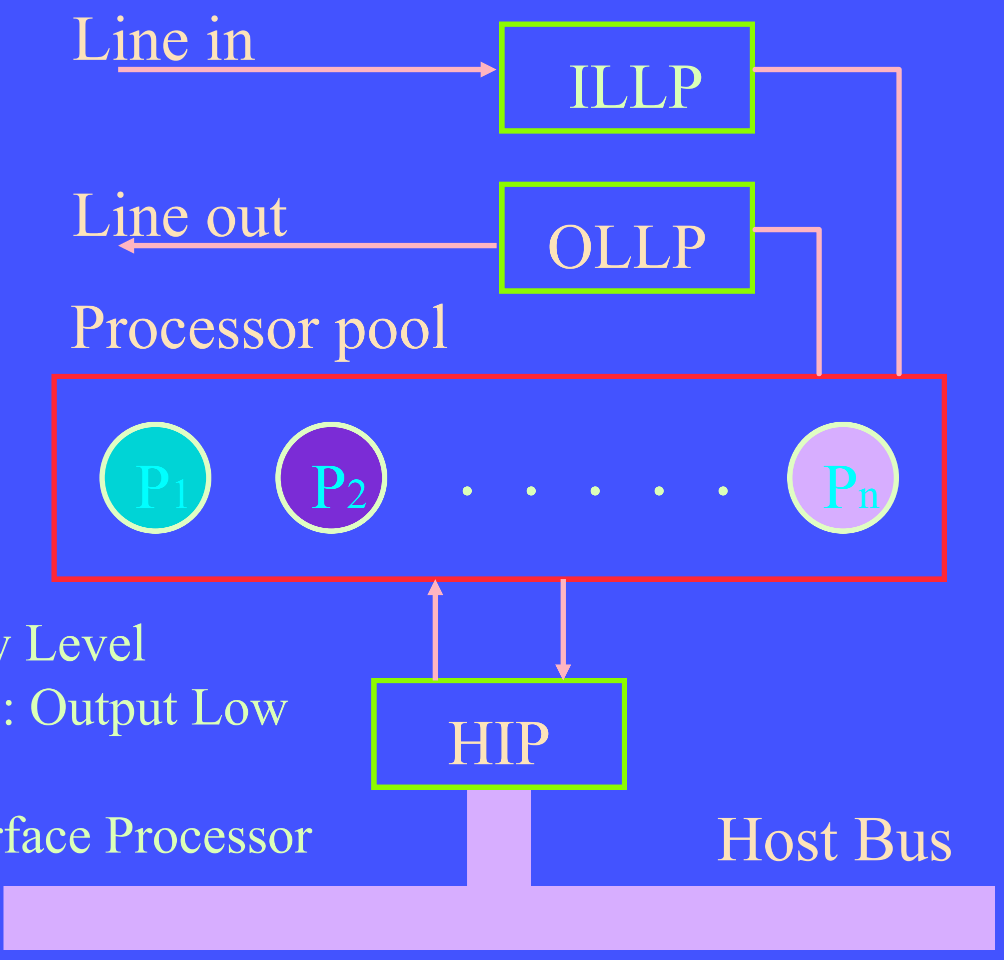
- There are different types of and levels of parallelism that can be applied to implement protocol functions
- Parallelism Unit (coarse, medium, fine)
  - complete stack, protocol entity, protocol function
- Parallelism Type
  - SIMD-Like Parallelism, MIMD, Hybrid

Parallel Processing in Protocol Implementation

- SIMD - like Parallelism :
  - Packet level
  - Connection level



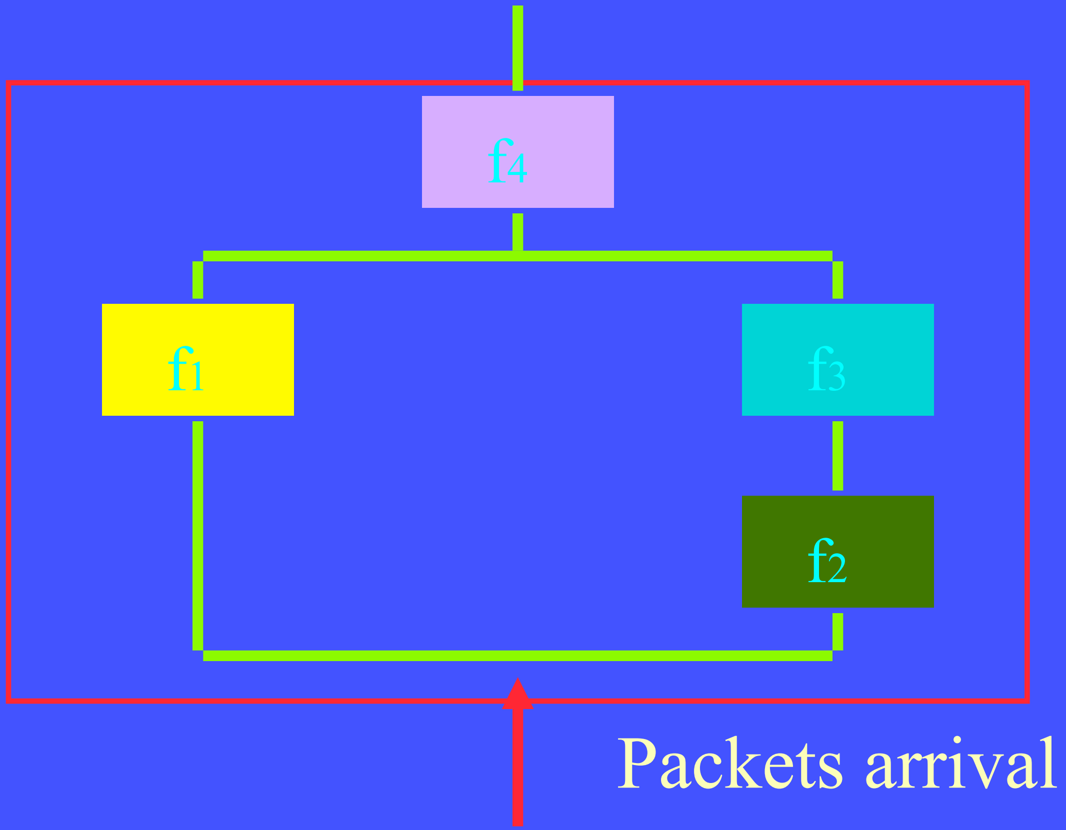
Parallel (Packet - level) Implementation of OSI TP4 Protocol



ILLP : Input Low Level Processor
OLLP : Output Low Level Processor
HIP : Host Interface Processor

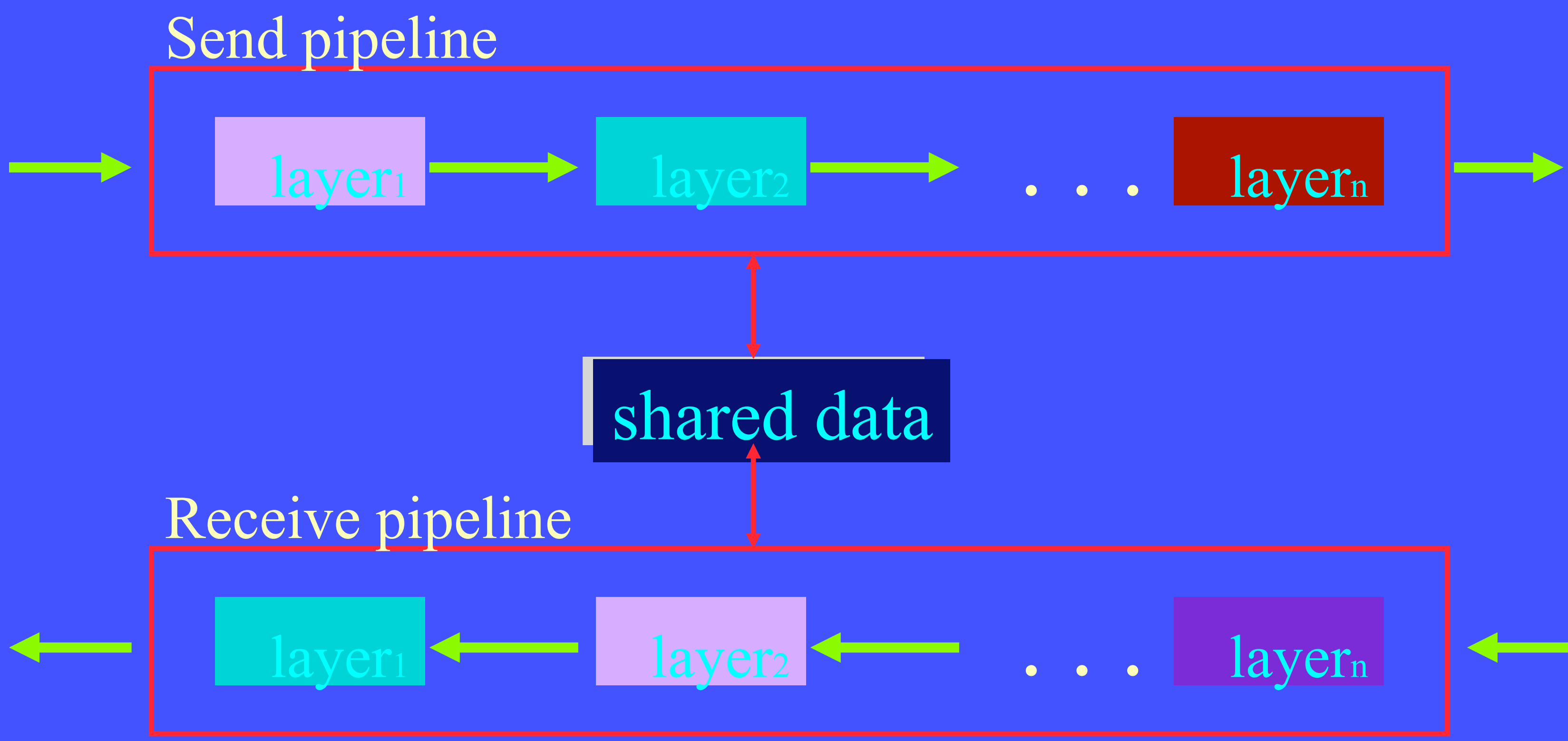
Parallel Processing in Protocol Implementation

- MISD - like Parallelism :
  (function level Parallelism)

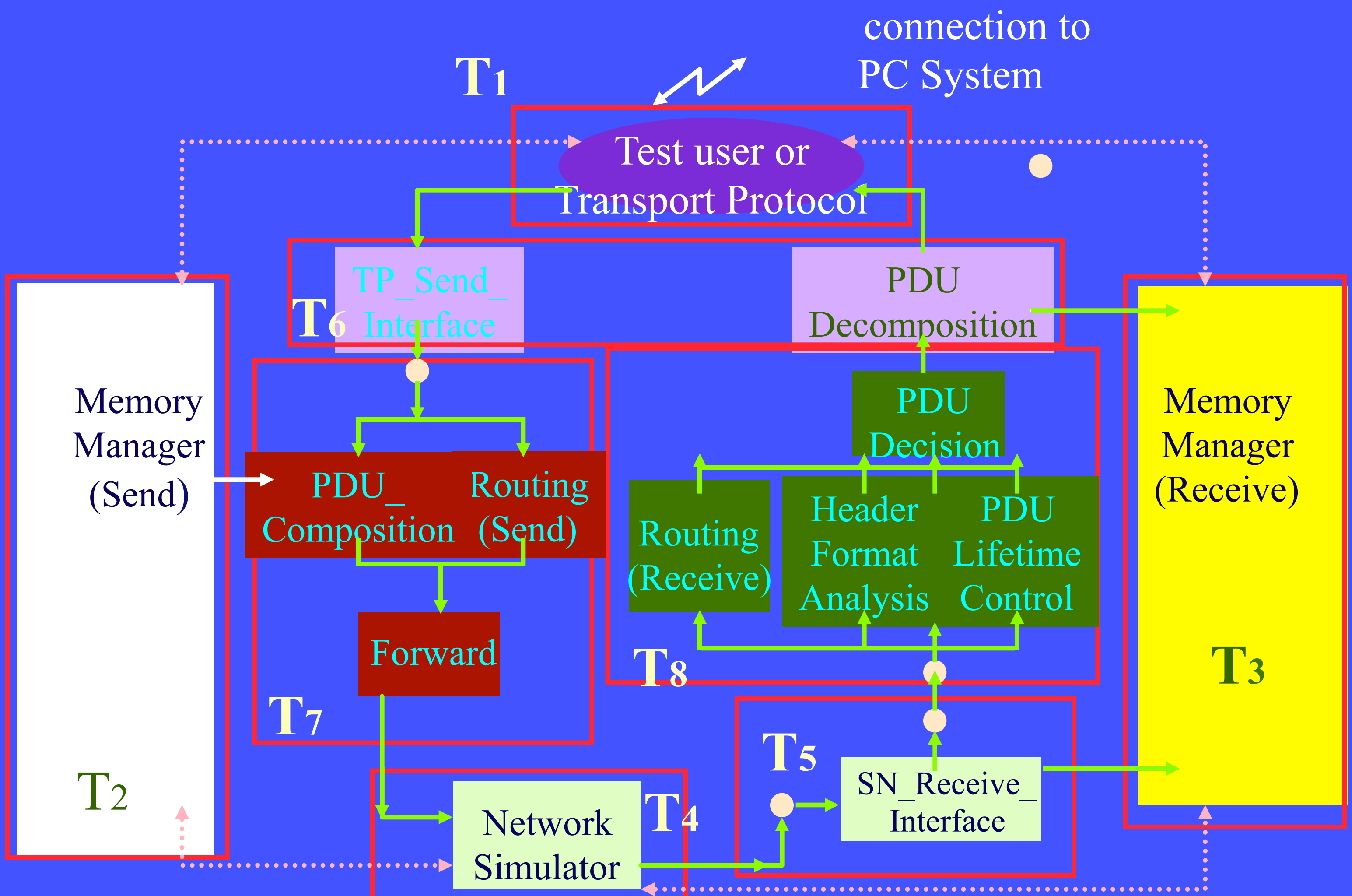


Parallel Processing in Protocol Implementation

- Temporal Parallelism (Pipelining)



Function - level Parallel Implementation of OSI Protocol TP4



HOPS : Horizontally Oriented Protocol Structure

- Horizontal structure as opposed to the vertical structure
- Division of protocols into functions instead of layers
- Functions are mutually independent