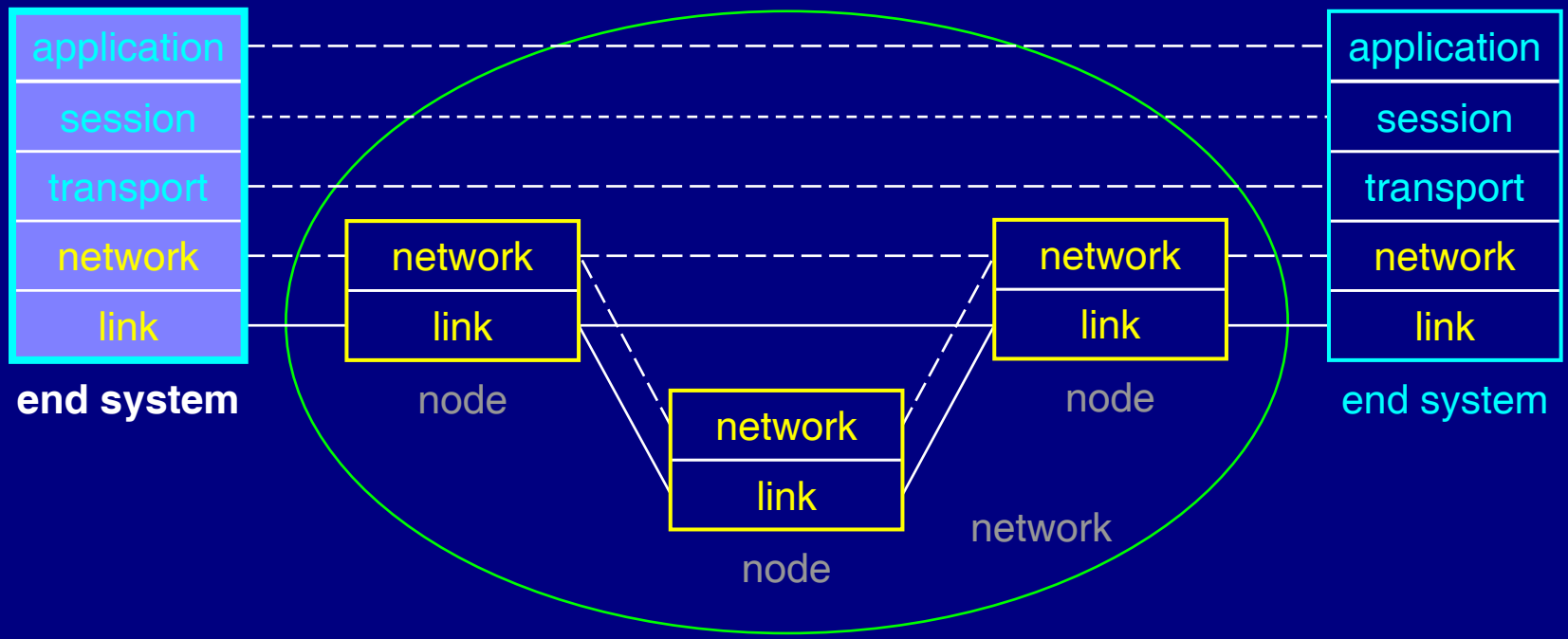
End Systems

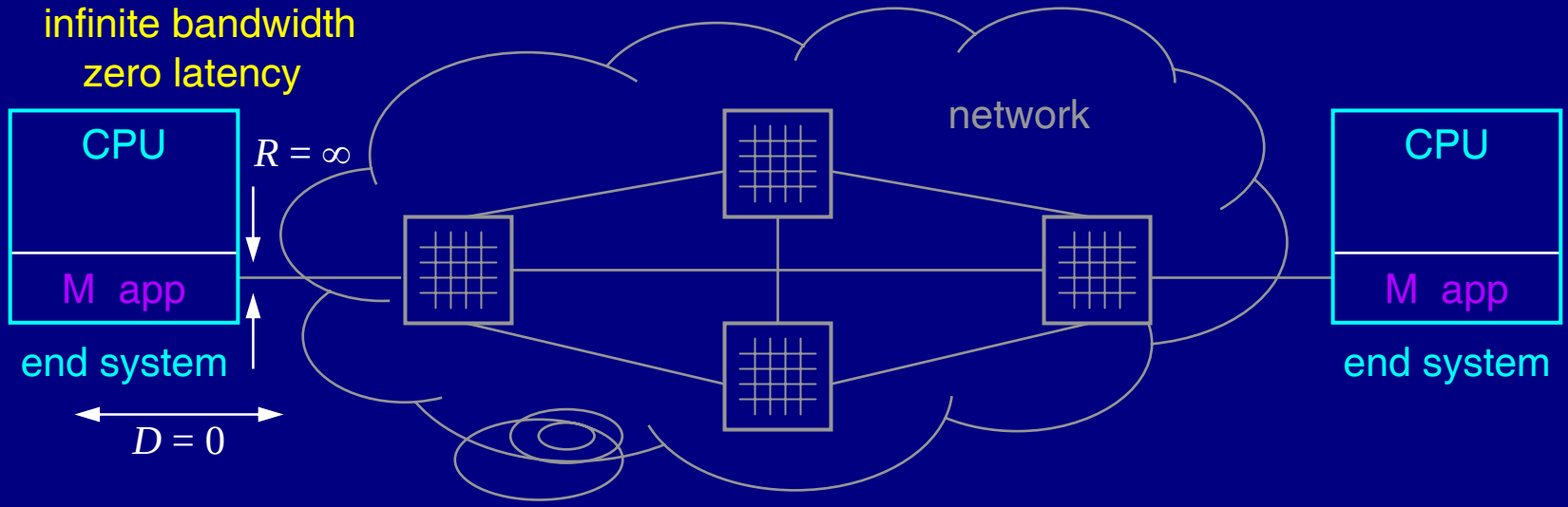
1. Introduction
2. Fundamentals and design principles
3. Network architecture and topology
4. Network control and signalling
5. Network components
   - 5.1 links
   - 5.2 switches and routers
6. End systems
7. End-to-end protocols
8. Networked applications
9. Future directions

End Systems



6.1. End system components
6.2. Protocols and OS software
6.3. End system organisation
6.4. Host–network interface

Ideal Network
End System Principle



End System Principle E-II

The communicating end systems are a critical component in end-to-end communications and must provide a low-latency, high-bandwidth path between the network interface and application memory.

End Systems
Application Primacy

- End systems have limited resources to be used by
  - applications
  - inter-application communication
- Protocol processing
  - must not significantly interfere with applications themselves
  - protocol benchmarks must consider this

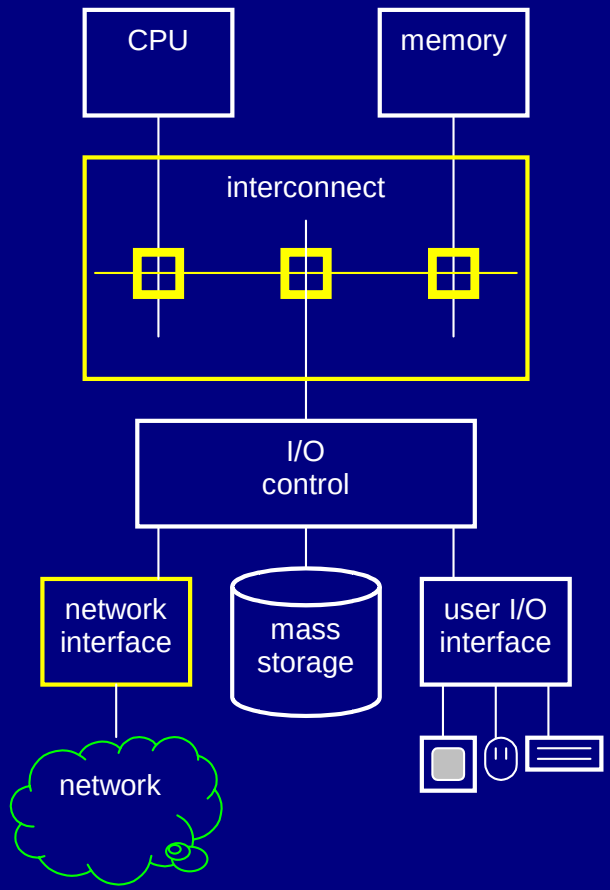
Application Primacy E-I

Optimisation of communications processing in the end system must not degrade the performance of the applications using the network.

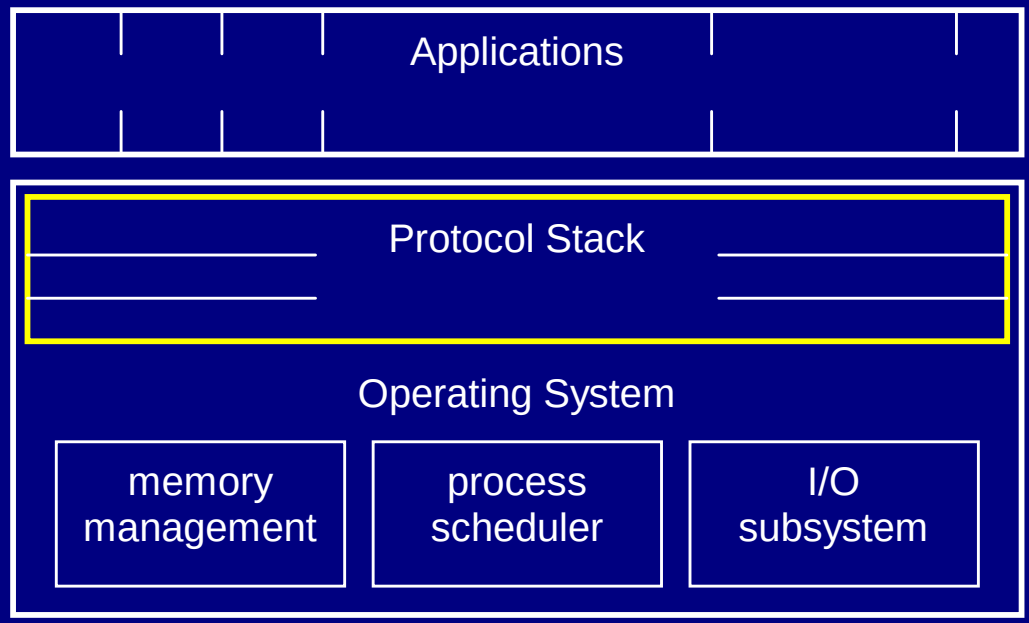
End Systems
End System Components

- 6.1 End system components
  - 6.1.1 End system hardware
  - 6.1.2 End system software
  - 6.1.3 End system bottlenecks
  - 6.1.4 Traditional end system implementation
  - 6.1.5 Ideal end system implementation
- 6.2 Protocol and OS software
- 6.3 End system organisation
- 6.4 Host–network interface

End System Components
Hardware

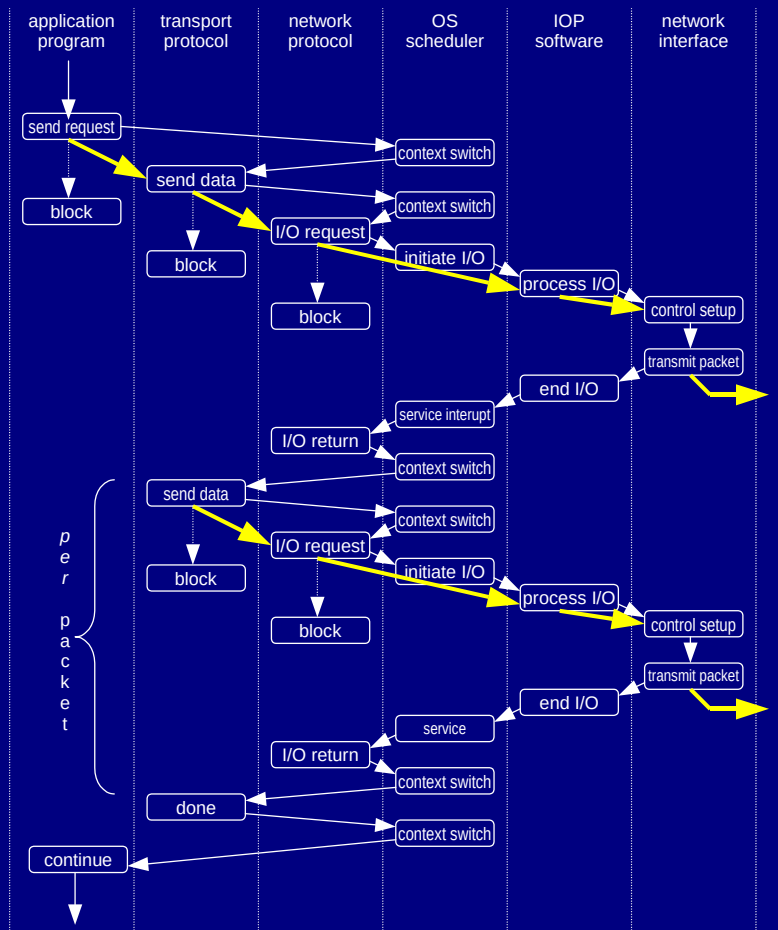


End System Components
Software



End System Components
Traditional End System Implementation

- Communication
  - handled as I/O
    - I/O mechanisms not optimised communication
  - transfer per packet
    - multiple per ADU
- Protocol implementation
  - process per layer
  - multiple copies of data
  - many context switches



End System Components
End System Bottlenecks

- *Systemic* elimination of bottlenecks is necessary
  - host organisation
  - memory subsystem
  - processor–memory interconnect
  - operating system
  - protocol stack

Systemic Elimination of End System Bottlenecks E-IV

The host organisation, processor–memory interconnect, memory subsystem, operating system, protocol stack, and host–network interface are all critical components in end system performance, and must be optimised in concert with one another.

End System Components
End System Bottlenecks

- More efficient protocol implementation
  - not process per layer
    - reduce context switches
    - reduce copies
  - don't treat communications like I/O

End System Layering Principle E-4A

Layered protocol architecture does not depend on a layered process implementation in the end system.

End System Components
Importance of Networking

- Importance of networking in the end system
  - networking should be considered a first class citizen
    - in system design
    - in performance specifications
    - in purchase decisions
      - what do users do with their PCs? Web surf. P2P file sharing.

Importance of Networking in the End System E-I.4

Networking should be considered a first-class citizen of the end system computing architecture, on a par with memory of high-performance graphics subsystems.

End System Components
Protocol Constraints

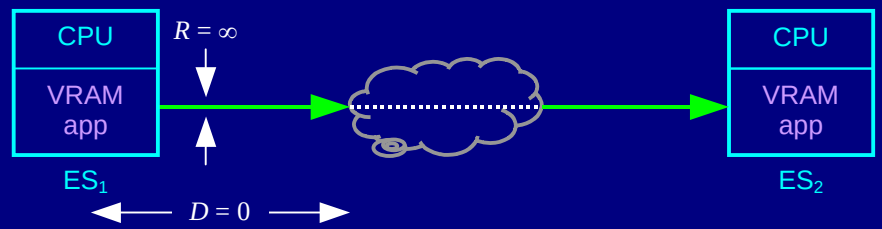
- Widely deployed protocols are difficult to replace
  - important to optimise existing protocols
  - add backward-compatible enhancements for interoperability
- Replace with new protocols only when necessary

Optimise and Enhance Widely Deployed Protocols E-III.7

The practical difficulty in replacing protocols widely deployed on end systems indicates that it is important to optimise existing protocol implementations and add backward-compatible enhancements, rather than only trying to replace them with new protocols.

Ideal End System Model

- Data shifted directly between application memory
- But
  - non-trivial latency
    - processor can't block
  - where to put data
  - channel not reliable
- Need transport protocol



Copy Minimisation Principle E-II.3

Data copying, or any operation that involves a separate sequential per byte touch of the data, should be avoided. In the ideal case, a host–network interface should be zero copy.

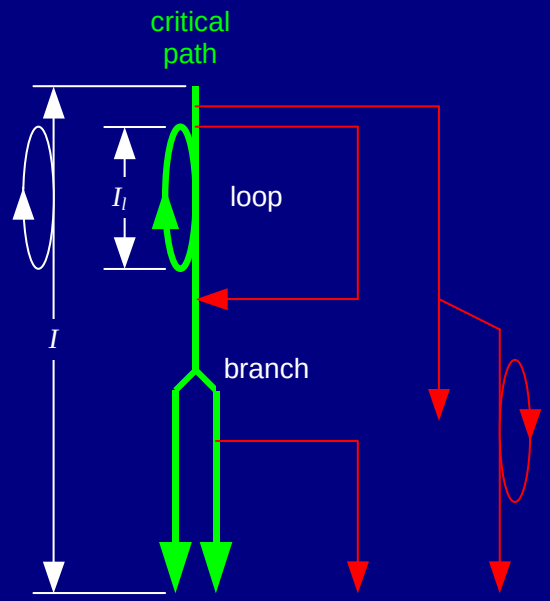
End Systems
Protocol and OS Software

6.1 End system components
6.2 Protocol and OS software
- 6.2.1 Protocol software
- 6.2.2 Operating systems
- 6.2.3 Protocol software optimisations

6.3 End system organisation
6.4 Host–network interface

Protocol and OS Software
Critical Path

- Critical path
  - operations required for data transfer
    - bottlenecks
  - operations that happen frequently have greater overall impact



Critical Path Principle

E-1B

Optimise end system critical path protocol processing software and hardware, consisting of normal data path movement and the control functions on which it depends.

Protocol and OS Software
Protocol Processing Classes

- Data manipulation
  - Data movement (to/from network and intra-host)
  - bit error detection and correction
  - buffering for retransmission
  - encryption/decryption
  - presentation formatting (e.g. ASN.1 or XDR)

+ These functions are part of the critical path

Protocol and OS Software
Protocol Processing Classes

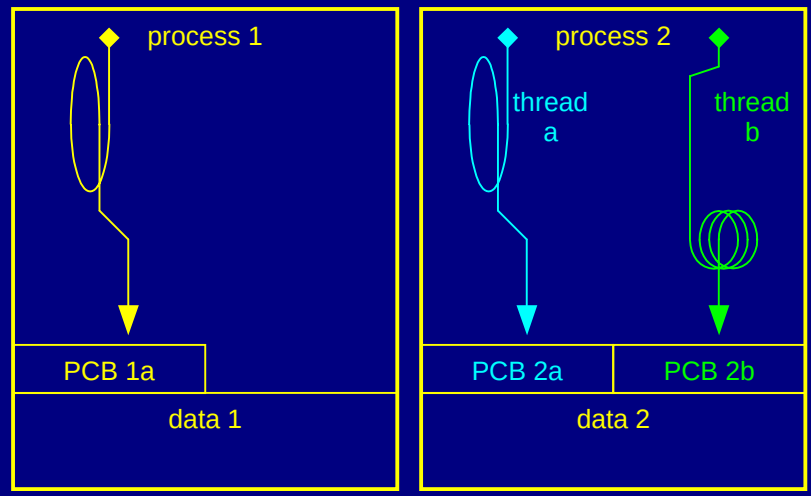
- Transfer control
  - flow and congestion control
  - lost and mis-sequenced packet detection
  - acknowledgements
  - multiplexing/demultiplexing flows
  - time stamping and clock recovery of real-time packets
  - formatting
    - framing/delineation
    - encapsulation/decapsulation
    - fragmentation/reassembly
- These functions *may* be part of the critical path
  - analysis is needed to determine dependency

Protocol and OS Software
Protocol Processing Classes

- Asynchronous control
  - connection setup and modification
  - per connection granularity flow and congestion control
  - routing algorithms and link state updates
  - session control
- These functions are not part of the critical path

Protocol and OS Software
Context Switch Avoidance

- Context switches
  - transmission of packets
  - process per layer
- Avoidance
  - thread per layer
  - ILP



Context Switch Avoidance E-II.6*a*

The number of context switches should be minimised, and approach one per application data unit.

Protocol and OS Software
Polling vs. Interrupts

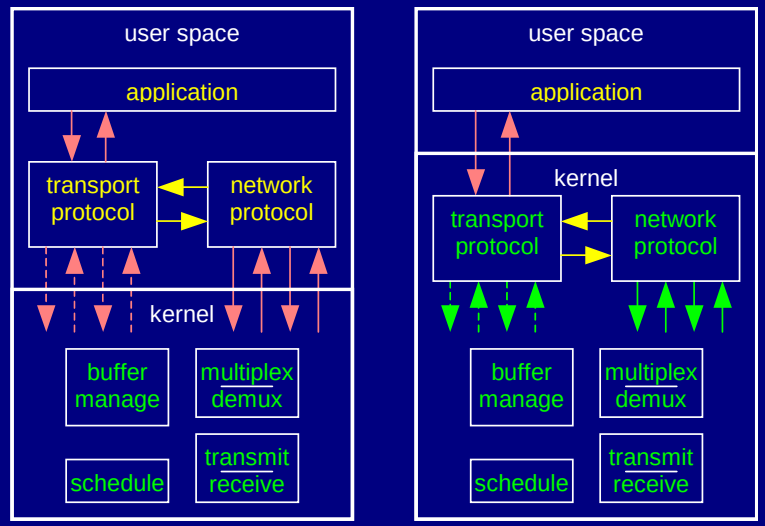
- Interrupts incur significant overhead
  - force context switch to OS
- Polling
  - avoids overhead of context switch
  - requires knowledge of when information arrives
    - polling interval critical to avoid wasted cycles

Interrupt vs. Polling E-4*h*

Interrupts provide the ability to react to asynchronous events, but are expensive operations. Polling can be used when a protocol has knowledge of when information arrives.

Protocol and OS Software
Kernel Crossing Avoidance

- User state
  - unprivileged
  - significant overhead
    - authorisation
    - parameter checks
    - context switch
- Kernel: trusted

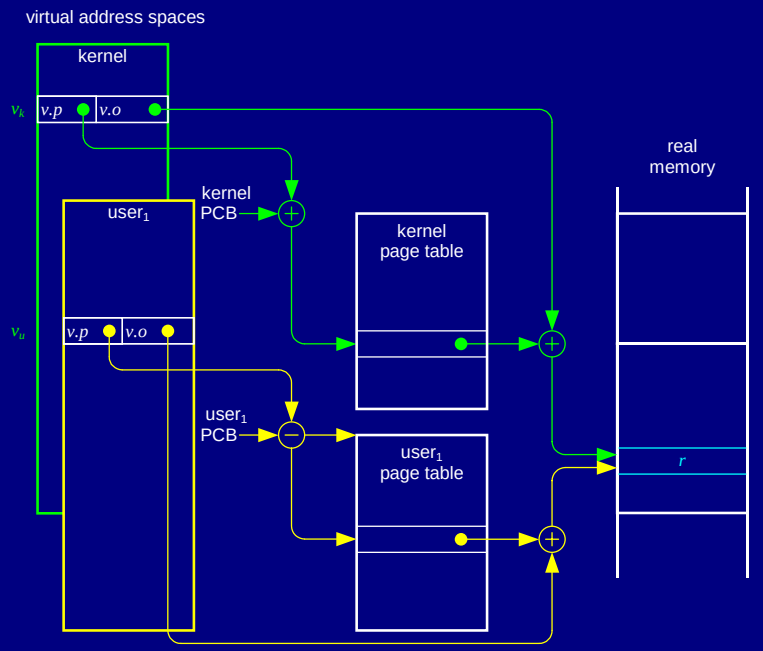


User/Kernel Crossing Avoidance E-II.6*k*

The number of user space calls to the kernel should be minimised due to the overhead of authorisation and security checks, the copying of buffers, and the inability to directly invoke needed kernel functions.

Protocol and OS Software
Memory Management and Remapping

- Virtual memory
  - translates addresses
  - avoids user ↔ kernel copy
    - map both to same



Avoid Data Copies by Remapping E-II.3*m*

Use memory and buffer remapping techniques to avoid the overhead of copying and moving blocks of data.

Protocol and OS Software
Resource Reservation

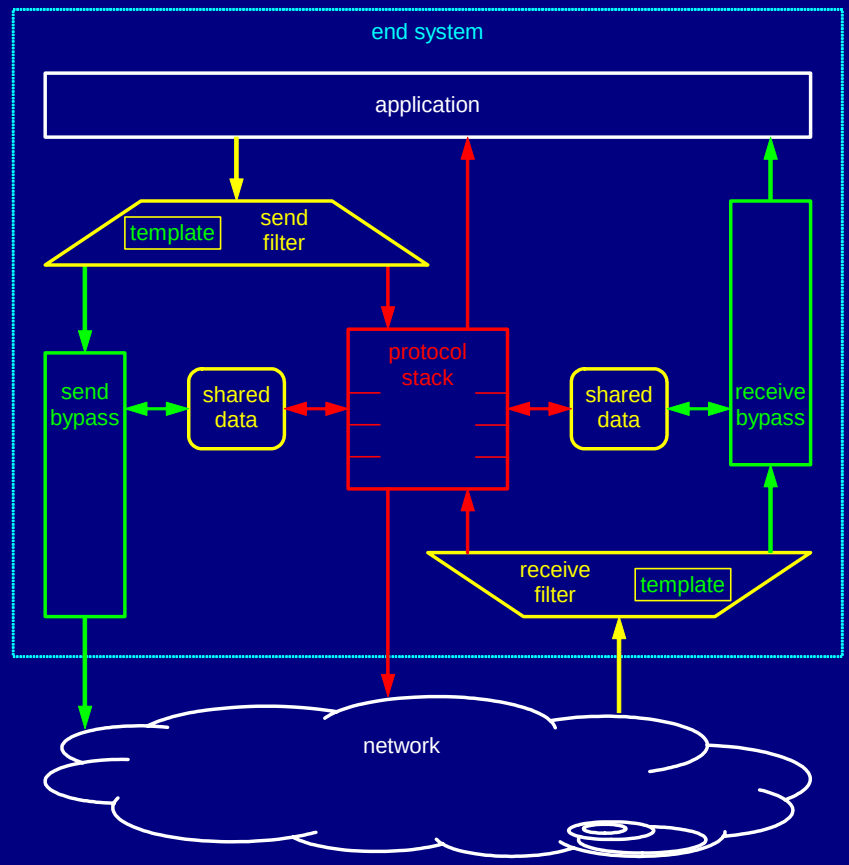
- Application-to-application QOS requires
  - network over-provisioning or reservations
  - end system over-capacity or reservations
    - CPU cycles
    - memory
    - bus or interconnect bandwidth

Path Protection Corollary E-II.2

In a resource constrained host, mechanisms must exist to reserve processing and memory resources needed to provide the high-performance path between application memory and the network interface and to support the required rate of protocol processing.

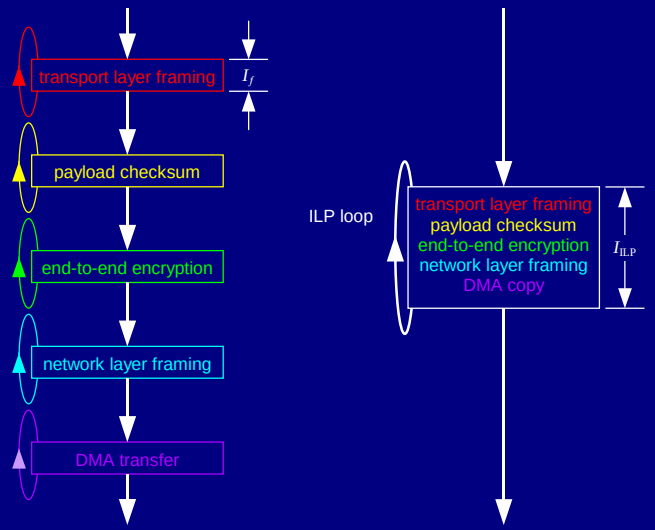
Protocol and OS Software
Optimisations: Protocol Bypass

- Protocol bypass
  - critical path optimisation
  - receive and send bypass
    - data manipulation
    - critical transfer control
    - shared data with stack
  - normal protocol stack
    - non-critical path



Protocol and OS Software
Optimisations: Integrated Layer Processing

- Operations in single ILP loop
  - software or hardware
- Avoids overhead
  - inter process or thread
  - eliminates copy
- Side effects
  - big cache miss penalty



ILP Principle

E-4E

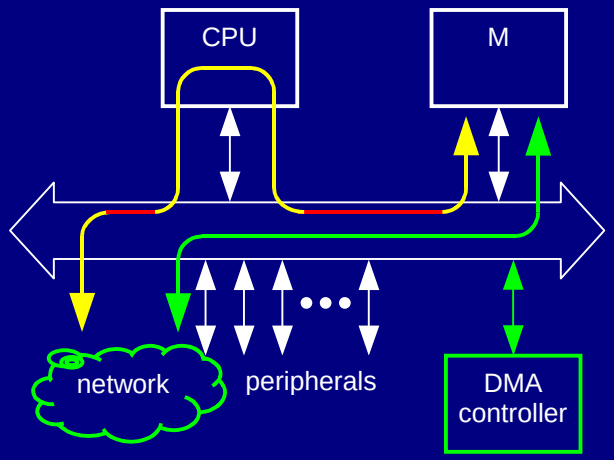
All passes over the protocol data units (including layer encapsulations/decapsulations) that take place in a particular component of the end system (CPU, network processor, or network interface hardware) should be done at the same time.

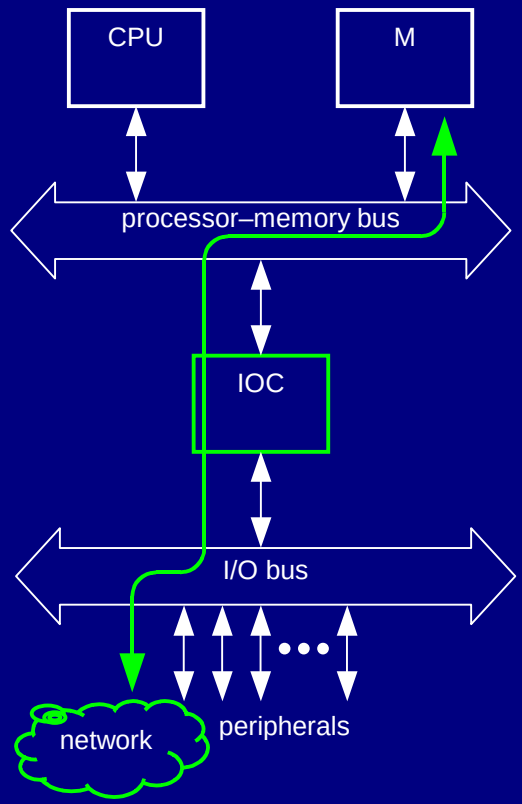
End Systems
End System Organisation

6.1 End system components
6.2 Protocol and OS Software
6.3 End system organisation
- 6.3.1 Host interconnects
- 6.3.2 Host–network interconnection alternatives
- 6.2.3 Host–network interface issues

6.4 Host–network interface

End System Organisation
Host Interconnects

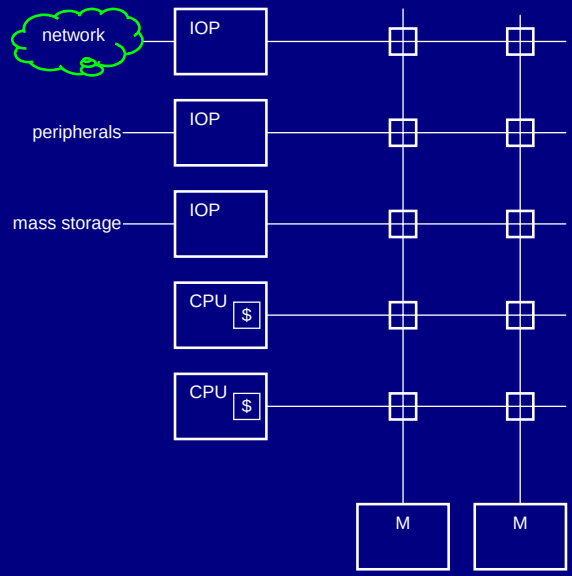




- PIO
  - via CPU
  - 2 bus transfers
- DMA reduces contention
- Separate P–M and I/O bus helps isolate I/O effects

End System Organisation
Nonblocking Host Interconnects

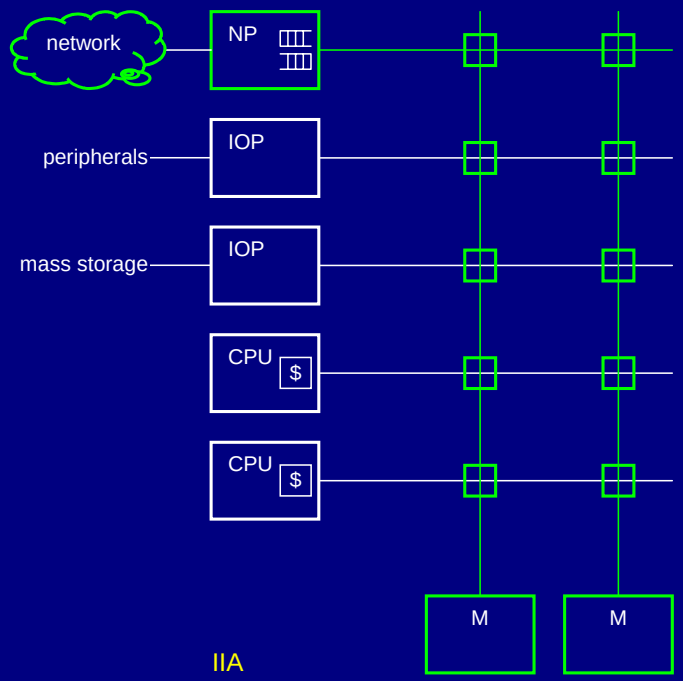
- Scalable host-interconnects
  - when bus interconnects saturate
  - used in high-performance systems
  - crossbar: $O(n^2)$ good for small n
  - $n \log(n)$ for large n

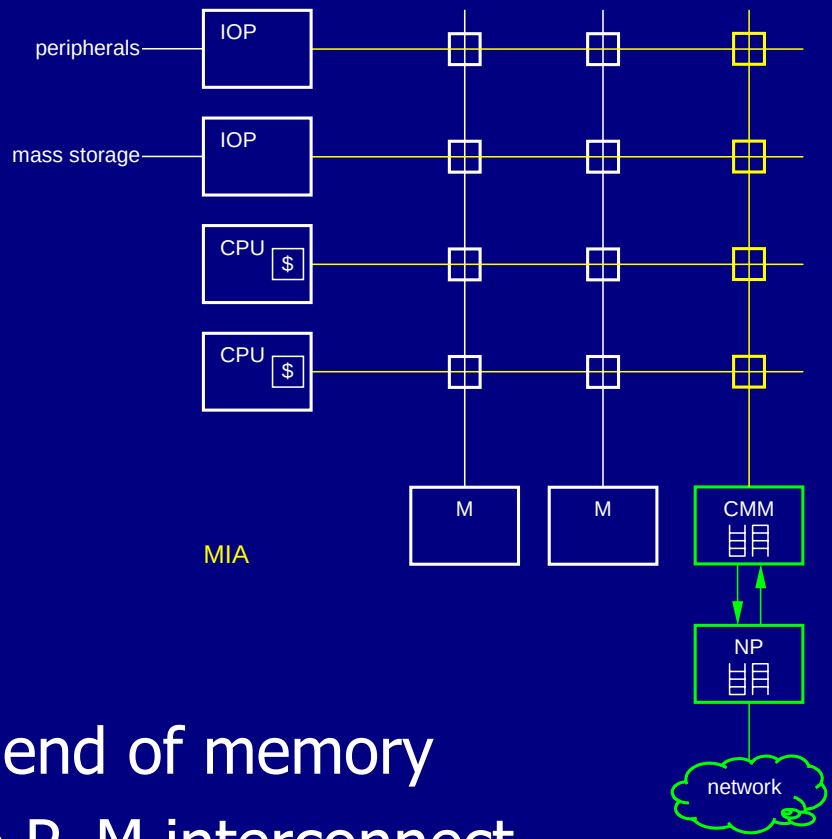


Nonblocking Host–Network Interconnect E-II.4

The interconnect between the end system memory and the network interface should be nonblocking, and not interfere with peripheral I/O, and CPU–memory data transfer.

End System Organisation
Nonblocking Host-Network Interconnects





- MIA – NP access to back end of memory
- IIA – NP direct access to P–M interconnect

End System Organisation
System Area Networks

- Unification of host interconnect and network:
  - bringing the network into the end system
  - spreading the end system across the network
- System area networks (SAN)
  - ideas based on 1960s/1970s mainframe architectures
  - switched inter-CPU and I/O communication
  - technologies
    - ESCON/FICON (enterprise systems / fiber connection)
      - originally extension of IBM sys/370, sys/390 channels
    - FC switching: fibre channel
    - IBA: InfiniBand architecture

End System Organisation
Example$_{6.1}$ InfiniBand Architecture

- Infiniband SAN
- Communications architecture for:
  - IPC: HCA – host channel adapter
  - I/O: TCA – target channel adapter
- Switched interconnection
  - intra-subnet switches
  - inter-subnet routers

End System Organisation

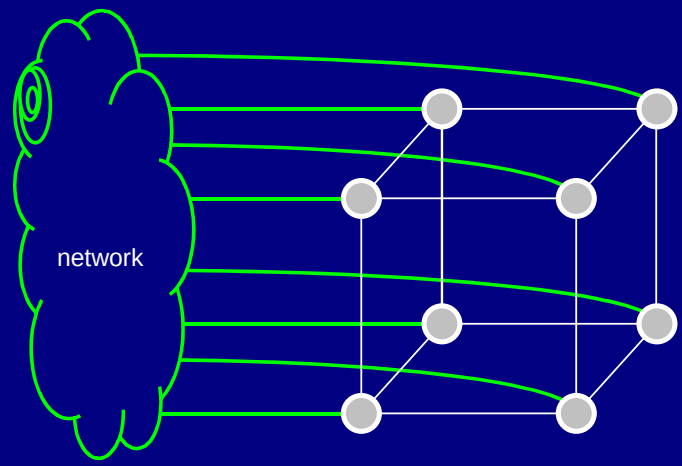
Example 6.1 InfiniBand Protocols

End System Organisation

Example 6.1 InfiniBand Packets

End System Organisation
Parallel Host–Network Interfaces

- Limited value in uniprocessors
  - protocols don't parallelise well
- Useful for NUMA systems
  - e.g. hypercubes



Nonuniform Memory Multiprocessor–Network Interconnect E-II.4*m*

Message passing multiprocessors need sufficient network interfaces to allow data to flow between the network and processor memory without interfering with the multiprocessing applications.

End Systems
Host–Network Interface

6.1 End system components
6.2 Protocol and OS software
6.3 End system organisation
6.4 Host–network interface
6.4.1 Offloading of communication processing
6.4.2 Network interface design

Application Layer to Network Interface Synergy and Functional Division E-1C*i*

Carefully determine what functionality should be implemented on the network interface rather than in end system software

Host–Network Interface
Offloading Functionality

- Determine which functionality to implement in NI
  - trend in 1980's to offload everything and put in hardware
  - but systemic analysis required
- Candidate processing to offload
  - best done between NI and memory
  - done efficiently in specialised hardware (esp. commodity)
  - places significant burden on host (e.g. per bit/byte)

Host–Network Interface Functional Partitioning and Assignment E-4C

Carefully determine what functionality should be implemented on the network interface rather than in end system software

Host–Network Interface
Offloading Functionality

- Determine which functionality to implement host
  - implementing in hardware may not increase performance
  - some processing *should* take place in host
    - ALF
    - part of ILP loop

Application Layer to Network Interface Synergy and Functional Division E-4C

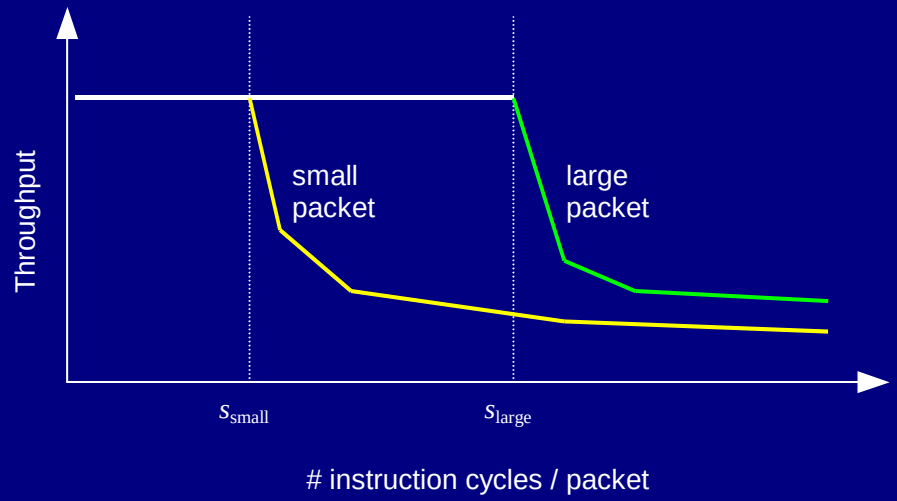
Application and lower-layer data unit formats and control mechanisms should not interfere with one another , and the division of functionality between host software and the network interface should minimise this interference.

Host–Network Interface
Offloading TCP/IP Functionality

- Partial datapath offload to network interface
  - TCP segmentation offload / large send offload
    - large VMTUs (jumbograms) moved host ↔ network interface
  - TCP checksum offload
- TOE: TCP offload engines
  - datapath (partial) TOE
    - reduced copies or RDMA (remote DMA) for zero copy
    - only beneficial for long flows
  - full TOE
    - control and datapath
- Many emerging products: jury still out

Host–Network Interface
Functional Partitioning

- Functional partitioning
  - hardware
    - custom, ASIC, gate array
  - software
    - network processor, embedded controller



Network Interface Hardware Functional Partitioning and Assignment E-1C*h*

Carefully determine what functionality should be implemented in network interface custom hardware, rather then on an embedded controller. Packet interarrival time driven by packet size is a critical determinant of this decision.

Host–Network Interface
NP Instruction Budgets

functionality: significant must be optimised infeasible

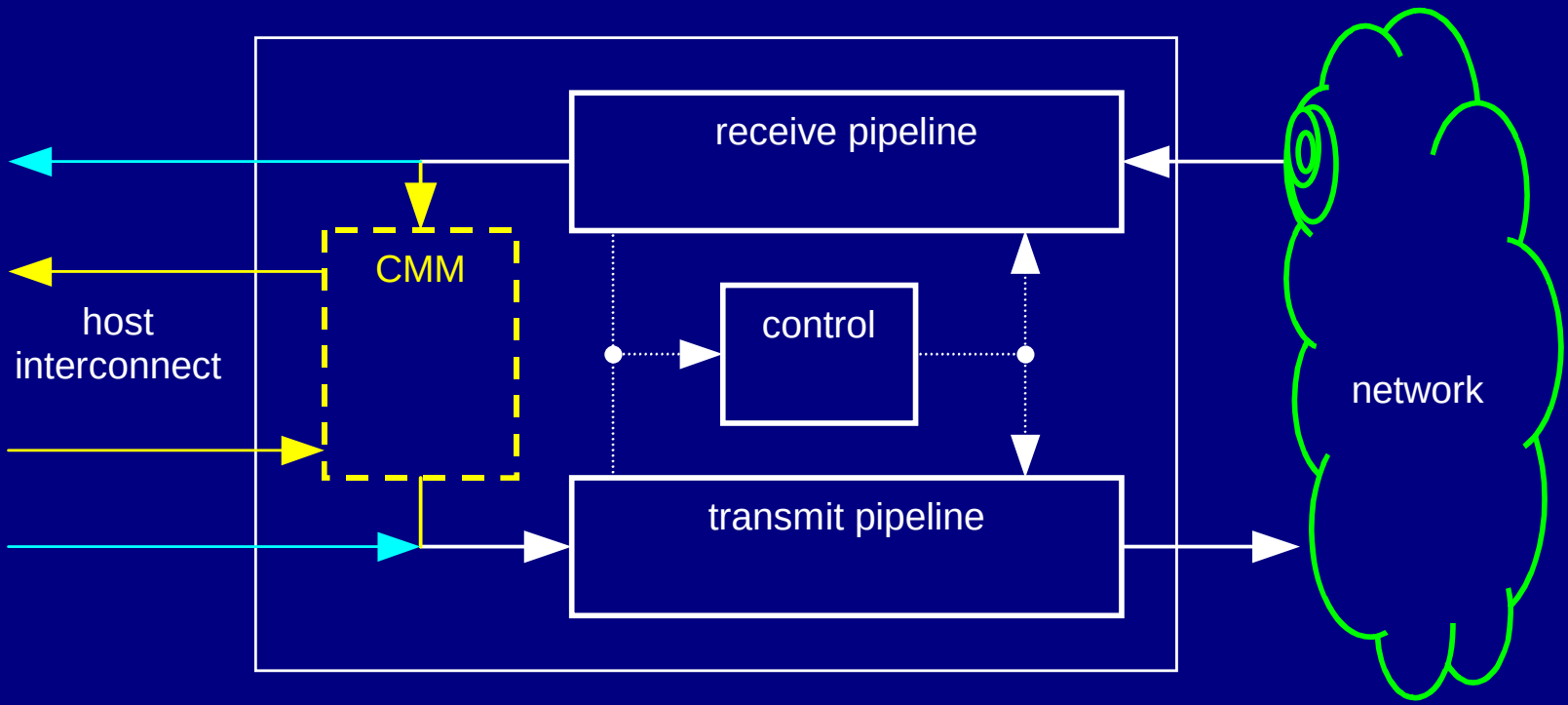
size / rate	1B			32B			128B			1KB		
	t	*i*		*t*	*i*		*t*	*i*		*t*	*i*	
		100MHz	1GHz		100MHz	1GHz		100MHz	1GHz		100MHz	1GHz
1 Mb/s	8μs	800	8000	250μs	25k	250k	1ms	100k	1M	8ms	800k	8M
10 Mb/s	800ns	80	800	250μs	2500	25k	100μs	10k	100k	800μs	80k	800k
100 Mb/s	80ns	8	80	250μs	250	2500	10μs	1000	10k	80μs	8000	80k
1 Gb/s	8ns	0	8	250ns	25	250	1μs	100	1000	8μs	800	8000
10 Gb/s	800ps	0	0	25ns	2	25	100ns	10	100	800ns	80	800

Host–Network Interface
Design Parameters

- Bandwidth
  - line rate / 8 determines required clock frequency
- Latency
  - latency budget needed by application
    - interactive ≈ 100 ms
    - real time process control significantly lower
  - fraction of end-to-end latency
    - LAN ≈ 10 μs for 1 km diameter
- Granularity
  - pipeline major cycle and buffer size

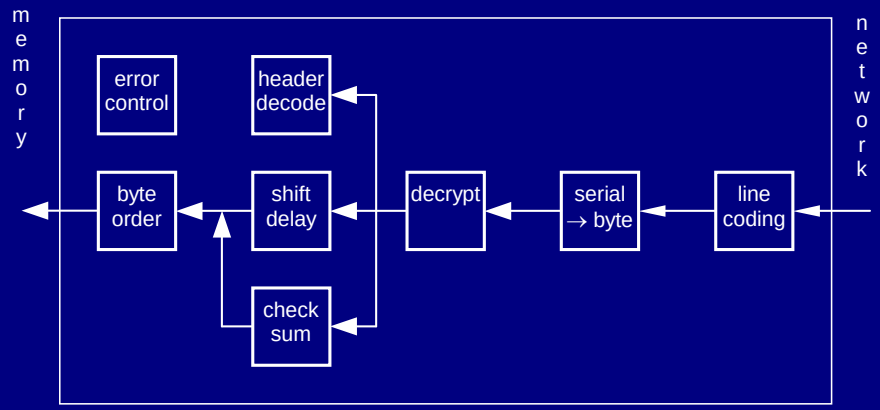
Host–Network Interface

Network Interface Design

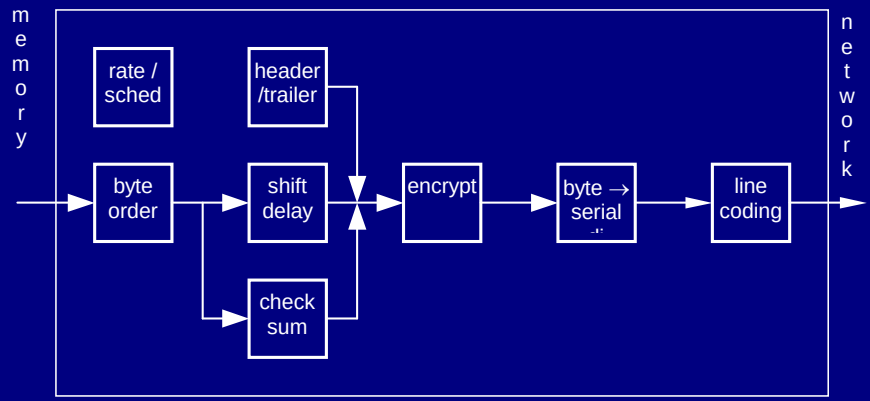


Host–Network Interface
Network Interface Design

- Receive pipeline



- Transmit pipeline



Host–Network Interface
High-Speed Encryption

- Cipher types
  - stream: bit stream
  - block
- Encryption modes

ECB	electronic codebook – single block
CBC	cipher block chaining – parallelisation possible with mux
CFB	cipher feedback
OFB	output feedback
CTR	counter – fully parallelisable since blocks independent

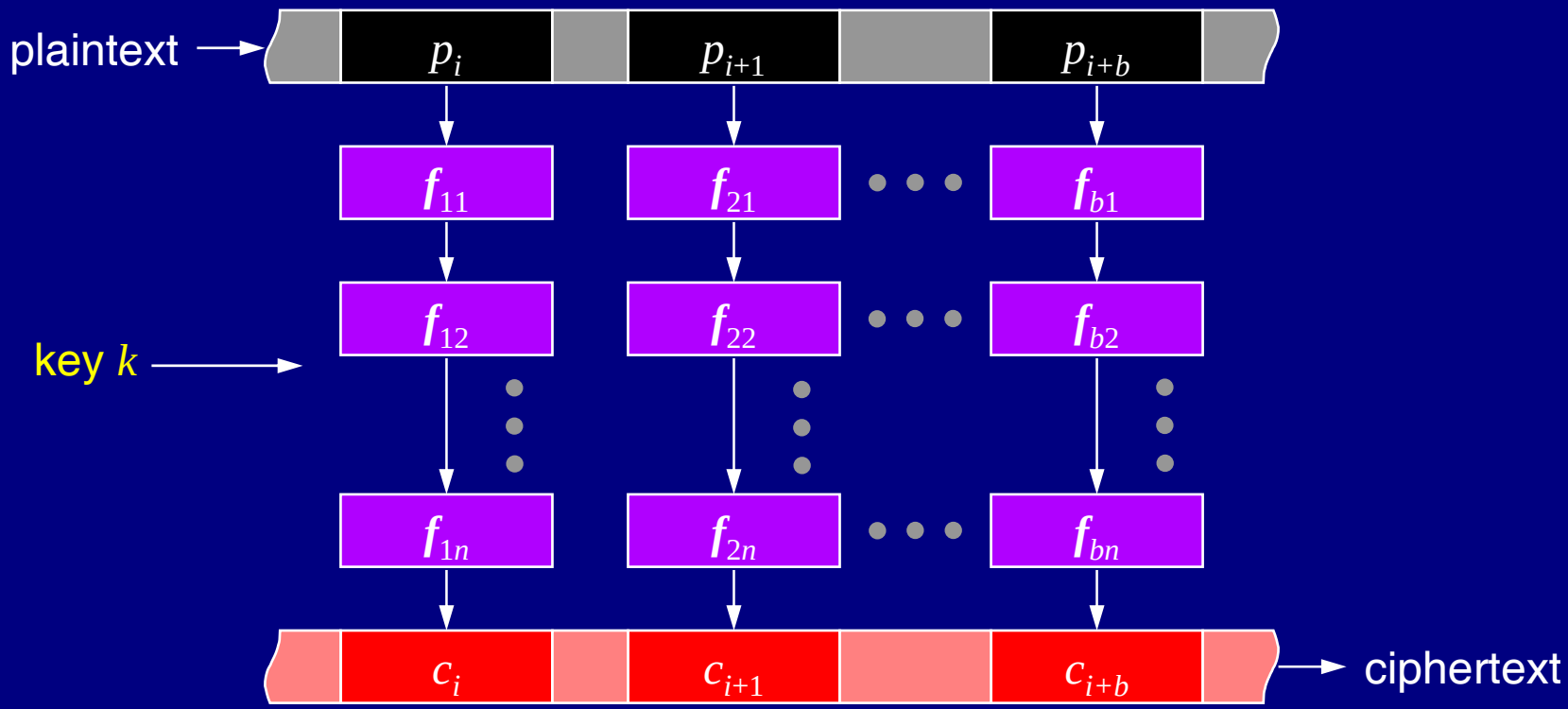
Host–Network Interface
High-Speed Encryption

- Desirable characteristics
  - pipelinable: no feedback dependencies
    - loop unrolling for multiple encryption rounds
  - parallelisable: no interblock dependencies
    - CTR mode only needs block id
- Challenges
  - maintaining cryptographic synchronization
    - out-of-band block-id for CTR mode

Critical Path Optimisation of Security Operations T-6D*c*

Encryption and per packet authentication operations must be optimised for the critical path.

Host–Network Interface
High-Speed Encryption



- Encryption functions *f*
  - *n* pipeline stage delays over *b* blocks (parallel speedup)

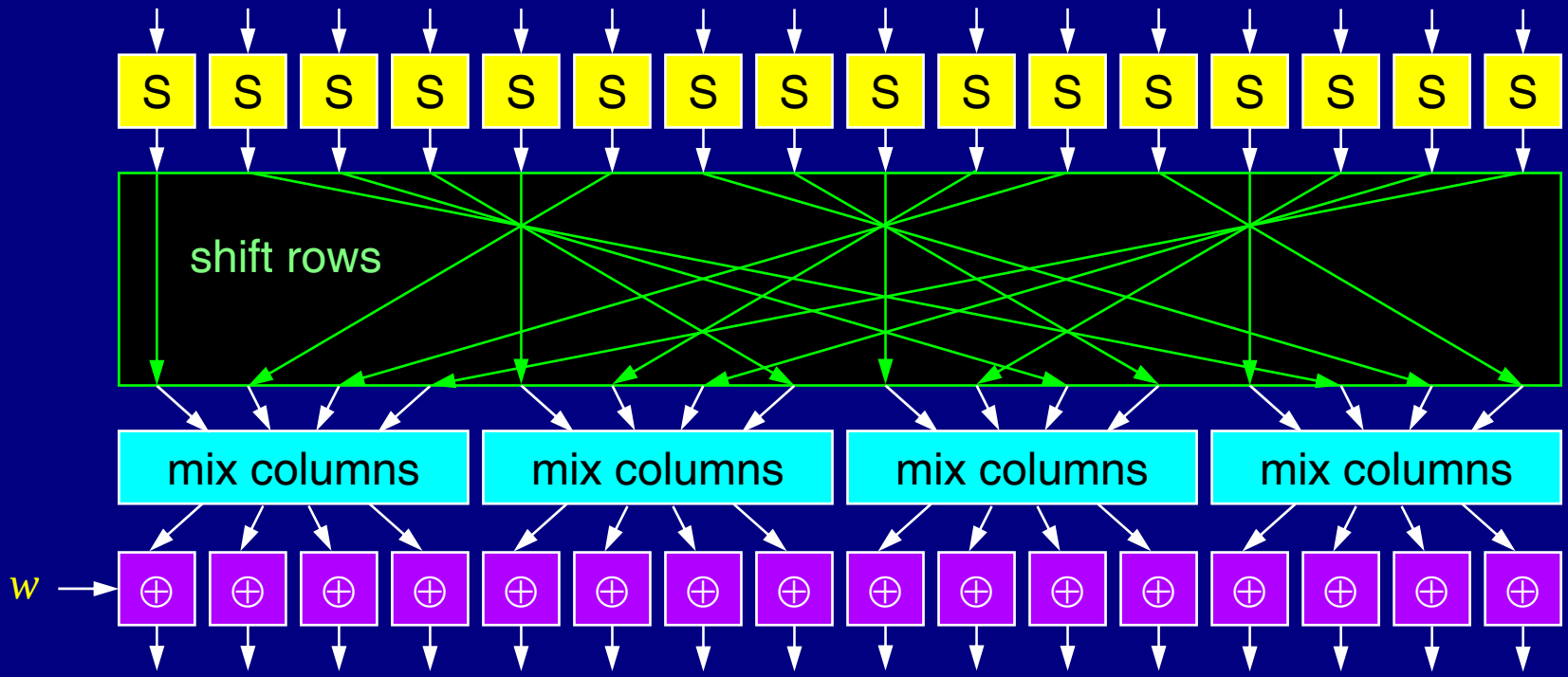
Host–Network Interface

Example$_{6.2}$ Advanced Encryption Standard

- AES: advanced encryption standard [NIST FIPS-197]
  - replacement for DES for commercial/consumer encryption
- Rijndael algorithm chosen by competition
  - high-speed implementation was one criteria
- Designed for high-performance implementation
  - pipelinable sequence of rounds (internally pipelinable)
  - parallalisable in CTR mode

Host–Network Interface

Example$_{6.2}$ AES Encryption Round

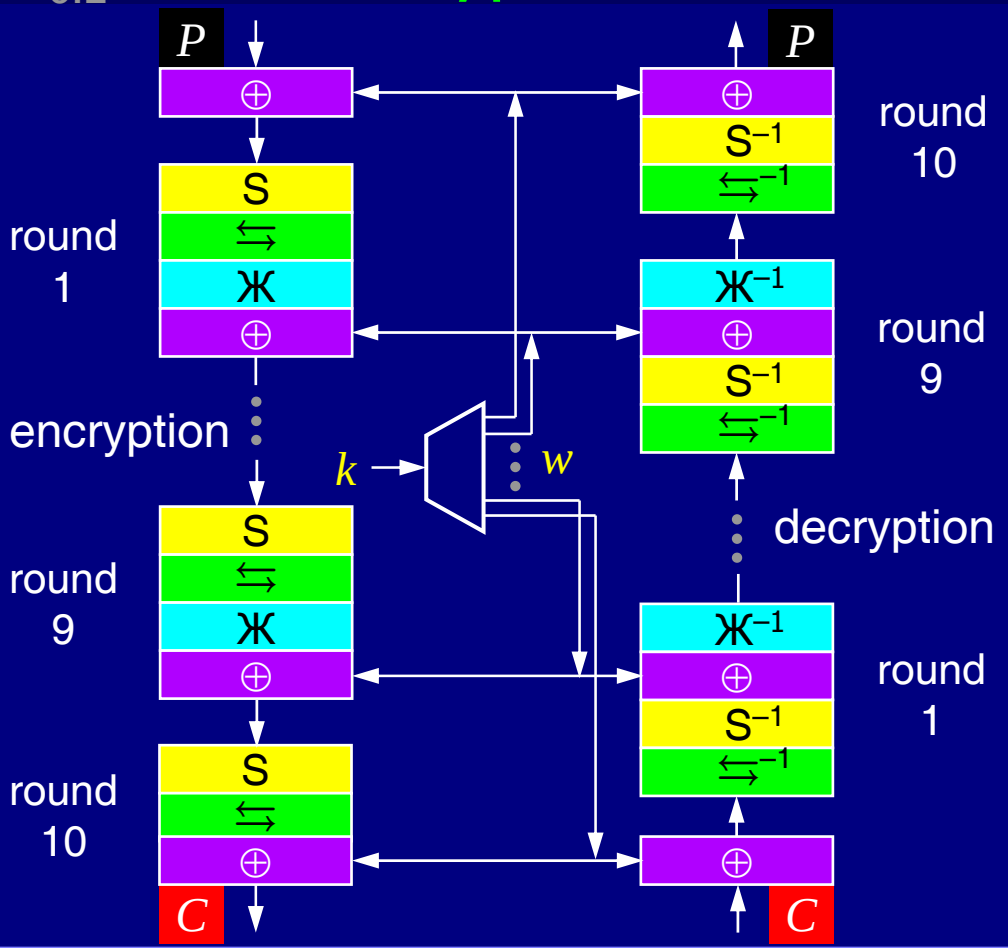


- S: substitute bytes (table)
- ⇆: shift rows (permute)
- Ж: mix columns (matrix ×)
- ⊕: add (xor) round key *w*

Host–Network Interface

Example 6.2 AES Encryption

- 128b blocks
- 128/192/256b key k expanded to 1408/1664/1920b round key w; 128b/round
- 10/12/14 reversible encryption rounds
- fully pipelinable (no feedback)



Host–Network Boundary Blurring
Distributed Storage Area Networks

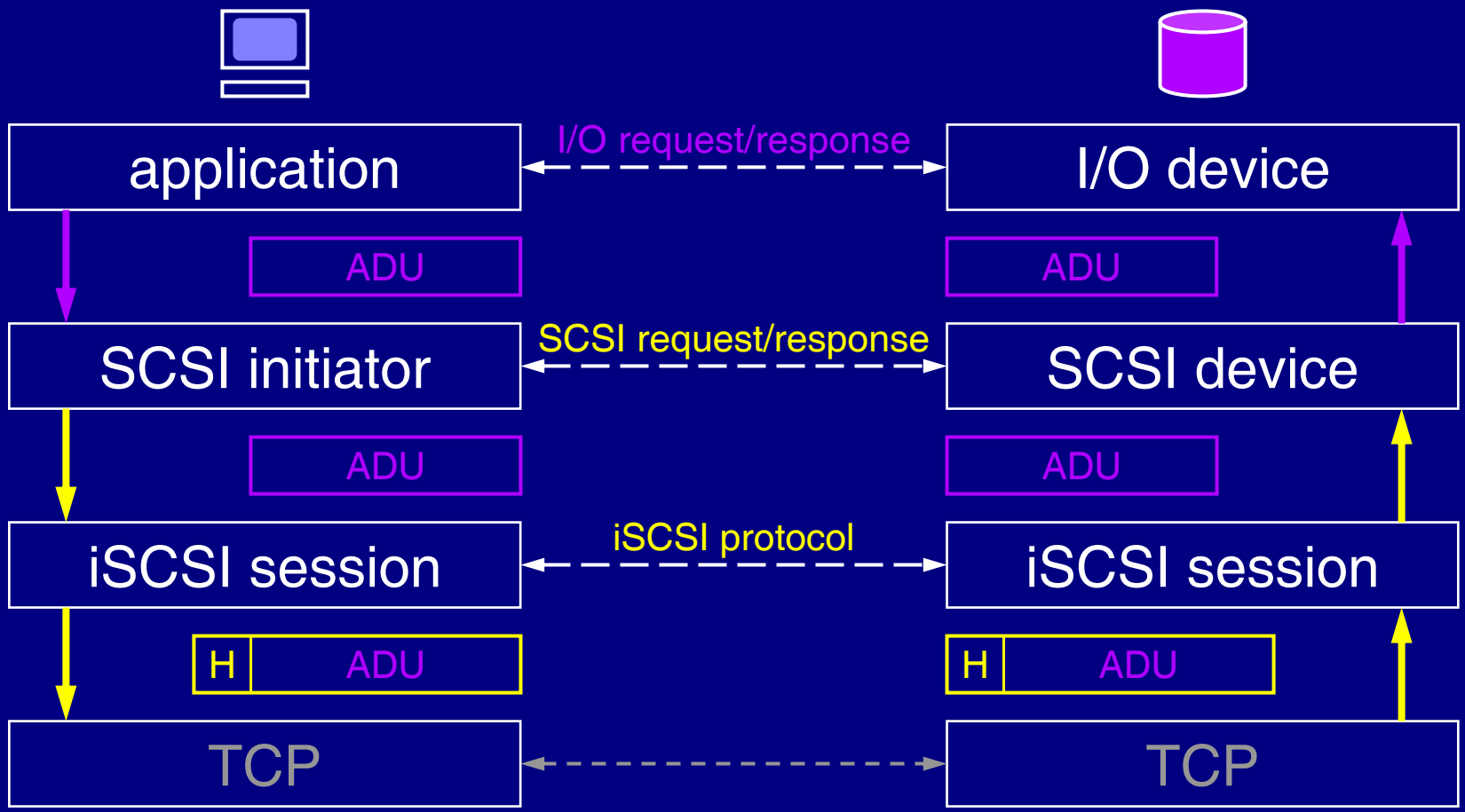
- System area network for CPU access to disk storage
  - using SAN network architectures and protocols
- Remote access to storage over long distance
  - LAN, MAN, WAN access over IP
    - iSCSI: Internet SCSI
    - FCIP: fibre channel over TCP/IP

Storage Area Networks
Example$_{6.i}$ iSCSI Background

- Internet distributed storage
  - based on SCSI (small computer systems interface)
    - T10 reference
    - standard interface for storage devices
- iSCSI (Internet small computer systems interface)
  - [RFC 3347, 3720]
- Session layer protocol

Storage Area Networks

Example$_{6.i}$ iSCSI Protocol Stack



Storage Area Networks
Example$_{6.i}$ iSCSI PDU Format Overview

- BHS Header [48b]
  - basic header segment
- AHS (optional)
  - additional header segment
  - requests only
- Header digest (optional)
- Data segment
- Data digest