# Fundamentals and Design Principles

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

# Fundamental Axioms
## for High-Speed Networking Research

2.1 Axioms for high-speed networking research

- Ø Know the past, present, and future
- I. Application primacy
- II. High-performance paths goal
- III. Limiting constraints
- IV. Systemic optimisation principle

2.2 Drivers and constraints

2.3 Design principles and tradeoffs

2.4 Design techniques

# Fundamental Axioms
## Know the Past

- It is essential to know the past
  - to know what is *really* new
  - to not waste time reinventing the past
  - to not repeat past mistakes

**Know the Past** $Ø_1$

*Genuinely new ideas are rare. Almost every "new" idea has a past full of lessons that can either be learned or ignored.*

# Fundamental Axioms
## Know the Past, Present, and Future

- It is essential to have a broad current understanding
  - to understand how to reapply past ideas
  - to understand how to avoid past mistakes
  - to know what new needs to be done

### Know the Present $Ø_2$

*"Old" ideas look different in the present because the context in which they have reappeared is different. Understanding the difference tells us which lessons to learn from the past, and which to ignore.*

# Fundamental Axioms
## Know the Past, Present, and Future

- The future *will* contain surprises
- We can't predict the future ...

  ... but we can prepare for it
  - understand historical progression of technology
  - not design systems only current assumptions ...

    ... that will break in the future

**Know the Future** $Ø_3$

*The future hasn't happened yet, and is guaranteed to contain at least one completely unexpected discovery that changes everything.*

# Fundamentals and Design Principles
## A Brief History of Networking

2.1 A brief history of networking
- 2.1.1 First generation: emergence
- 2.1.2 Second generation: the Internet
- 2.1.3 Third generation: convergence and the Web
- 2.1.4 Fourth generation: scale, ubiquity, and mobility

2.2 Drivers and constraints

2.3 Design principles and tradeoffs

2.4 Design techniques

# History: First Generation (≤1970s)
## Emergence

- Voice – widely deployed
  - circuit switched PSTN over copper
  - RF two-way radios
- Entertainment – widely deployed
  - broadcast RF for radio and television
- Data – limited pervasiveness
  - serial link over copper
  - modem remote terminal access

*Distinct infrastructure for the three*

# History: Second Generation (1980s)
## Internet

- Voice – widely deployed
  - digital switched PSTN over copper
  - cellular mobile telephony (late)
- Entertainment – significant deployment
  - CATV over copper coax starts to supplement broadcast
- Data – research / corporate enterprise networks
  - packet switched store-and-forward: Internet, X.25
  - gatewayed subnetworks: SNA, DECnet, BNA, DCNA, etc

*High-speed networking emerges as a discipline*

# History: Second Generation (1980s)
## Unix and Internet

- Dominant systems emerged in the second generation
  - operating systems: BSD and System V Unix
  - networking: TCP/IP
- Pioneering earlier work shouldn't be forgotten ...

  ... or reinvented
  - e.g. OS/360, MVS, Multics, TSS/360, CP-67, MCP, ...
  - e.g. SNA, DECnet, X.25, ...

**Not Invented Here Corollary [1980s version]** Ø-A

*Operating systems didn't begin with Unix, and networking didn't begin with TCP/IP.*

# History: Third Generation (1990s)
## Convergence and the Web

- Beginnings of converged IP-based infrastructure
  - IP-based global Internet subsuming enterprise networks
  - multimedia streaming
  - voice and video-conferencing over IP
- Web
  - replaces all other information access
    - e.g. FTP, gopher, archie
- Fast packet switching over fiber
- Significant 1st world deployment

# History: Third Generation (1990s)
## Windows and PC

- Dominant systems emerged in the third generation
  - operating systems: Windows
  - host architecture: PC based on x86
- Pioneering earlier work shouldn't be forgotten ...

  ... or reinvented
  - e.g. Unix, OS/360, Multics, TSS/360, CP-67, MCP
  - e.g. mainframe and superminicomputer architectures

**Not Invented Here Corollary [1990s version]** Ø-A

*Operating systems didn't begin with Windows, and host architectures didn't begin with the PC and x86 architecture.*

# History: Fourth Generation (2000s+)

## Scale, Ubiquity, Mobility

- Global infrastructure
  - optical core networks
  - wireless access networks
  - (hopefully) significant 2nd and 3rd world pervasiveness
- Ubiquitous and pervasive computing
  - personal area networks
  - smart spaces
- Tera- and Peta-node networks
- Autonomic control and management

# Fundamentals and Design Principles
## Drivers and Constraints

# Fundamental Axioms
## Application Primacy

**Application Primacy**

*The sole and entire point of building a high-performance network infrastructure is to support the distributed applications that need it.*

- Corollaries
  1. field of dreams vs. killer app dilemma
  2. interapplication delay
  3. network bandwidth and latency
  4. networking importance in system design

# Application Primacy
## Field of Dreams vs. Killer App Dilemma

- Applications need infrastructure on which to build
  - *field of dreams*
- Infrastructure deployment needs to justify expense
  - *killer app*
- Difficult to resolve without government funding
  - e.g. ARPANET, NSFNET, BSD Unix

**Field of Dreams vs. Killer App Dilemma** I.1

*The emergence of the next "killer application" is difficult without sufficient network infrastructure. The incentive to build network infrastructure is viewed as a "field of dreams" without concrete projections of user demand.*

# Application Primacy
## Interapplication Delay

- Users and Applications care about *delay*
  - not bandwidth! (directly)
  - users: interapplication delay
  - applications: end-to-end network and end system delay
- If delay is zero, all data is available instantaneously
  - no difference between distributed and local

**Interapplication Delay** I.2

*The performance metric of primary interest to communicating applications is the total delay in transferring data. The metric of interest to the users includes the delay through the application.*

# Fundamental Axioms
## Application Primacy

- Interapplication delay $D$ consists of two components
  - path latency $d_{path}$
  - transmission delay $d_{transmission} = b$ [bit] / $r$ [bit/sec]
  - $D = d_{path} + d_{transmission}$
- Low latency directly needed
- High bandwidth needed based on object size

**Network Bandwidth and Latency** I.3

*Bandwidth and latency are the primary performance metrics important to interapplication delay.*

# Application Primacy
## Importance of Networking

- Components that have significant networking role
  - communication performance should drive design
- Obvious for network components
- Has not been driving consumer PC architecture
  - even though Web browsing is primary application for many

**Networking Importance in System Design** I.4

*Communication is the defining characteristic of networked applications, and thus support for communication must be an integral part of systems supporting distributed applications.*

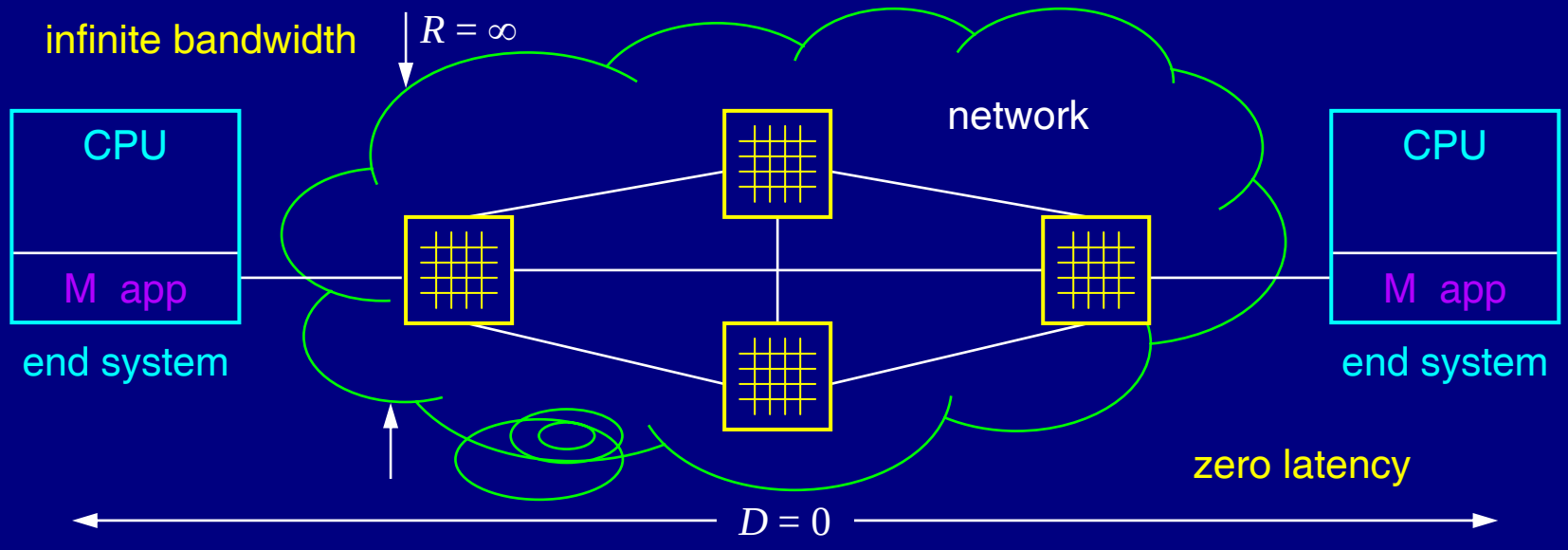
# Fundamental Axioms
## High-Performance Paths Goal

**High-Performance Paths Goal** II

*The network and end systems must provide a low-latency, high-bandwidth path between applications to support low interapplication delay.*

- Corollaries
  1. path establishment – signalling, routing, and control
  2. path protection – resource reservation or overprovisioning
  3. store-and-forward and copy avoidance
  4. blocking avoidance
  5. contention avoidance
  6. path information assurance tradeoff (against performance)

# Fundamental Axioms
## High-Performance Paths in Ideal Network




- Unlimited bandwidth
- Zero delay

# High-Performance Paths
## Path Establishment

- High performance paths need to be *established*
  - signalling is needed to establish paths
  - routing algorithms are need to determine path route
  - forwarding mechanisms move data along path
- Paths may need to be reconfigured
  - routing around faults, congestion, opponents
  - in response to path dynamics: mobility or multipoint

**Path Establishment Corollary** II.1

*Signalling and routing mechanisms must exist to discover, establish, and forward data along the high performance paths.*

# High-Performance Paths
## Path Protection

- High performance paths need to be *protected*
  - by overprovisioning
  - in a resource constrained environment
    - resource reservation
    - congestion avoidance and control

### Path Protection Corollary II.2

*In a resource constrained environment, mechanisms must exist to arbitrate and reserve the resources needed to provide the high-performance path and prevent other applications from interfering by congesting the network.*

# High-Performance Paths
## Store-and-Forward and Copy

- Per hop per byte delays are significant
  - store-and-forward
  - packet copying between memory buffers
  - per byte touching operations
- Avoid store-and-forward
  - cut-through
  - zero copy mechanisms

**Store-and-Forward Avoidance** II.3

*Store-and-forward and copy operations on data have such a significant latency penalty that they should be avoided whenever possible.*

# High-Performance Paths
## Blocking

- Blocking delays packets
  - interfering paths
    - non-blocking switches
    - alternate path routing
  - queueing
    - steady-state queue length should approach zero

**Blocking Avoidance** II.4

*Blocking along paths should be avoided, whether due to that overlap of interfering paths, or due to the building of queues.*

# High-Performance Paths
## Contention

- Contention delays packets
  - MAC delays for shared medium
  - spatial reuse to reduce contention
    - parallel waveguides
    - direction antennæ
    - transmission power control

### Contention Avoidance II.5

*Channel contention due to a shared medium should be avoided. Spatial reuse techniques including parallel waveguide, directional antennæ, and transmission power limitations can mitigate contention.*

# High-Performance Paths
## Transfer of Control

- Transfer control operations assist data movement
  - critical path analysis needed
    - granularity and implementation for line-rate performance
  - efficient transfer of control between protocol components

### Efficient Transfer of Control II.6

*Control mechanisms on which the critical path depends should be efficient. High overhead transfer of control between protocol processing modules should be avoided.*

# High-Performance Paths
## Information Assurance

- Information assurance requires resources
  - control processing delays: authentication and keying
  - data path delays: encryption/decryption, security headers
  - processing/memory that could be used for packet processing
- Trade IA vs. performance requirements

**Path Information Assurance Tradeoff** II.7

*Paths have application-driven reliability and security requirements, which may have to be traded against performance.*

# Fundamental Axioms
## Limiting Constraints

**Limiting Constraints** III

*Real-world constraints make it difficult to provide high-performance path to applications.*

- Corollaries
  1. speed of light
  2. channel capacity
  9. attenuation and transmission power
  3. switching speed
  4. cost and feasibility
  5. heterogeneity
  6. policy and administration
  7. backward compatibility inhibits real change
  8. standards both facilitate and impede dilemma

# Limiting Constraints
## Speed of Light

- Propagation velocity through a medium
  - ~ 0.66–1.0 *c* = 3–5 μs/km
  - dictates fundamental limit on latency over a distance
  - techniques can mask, but not eliminate
    - caching
    - prediction

**Speed of Light** III.1

*The latency suffered by propagating signals due to the speed of light is a fundamental law of physics, and is not susceptible to direct optimisation.*

# Limiting Constraints
## Channel Capacity

- Bandwidth of a medium
  - dictates fundamental limit on data rate
  - techniques to efficiently utilise available channel bandwidth
    - multiplexing
    - spatial reuse

### Channel Capacity III.2

*The capacity of communication channels is limited by physics. Clever multiplexing and spatial reuse can reduce the impact, but not eliminate the constraint.*

# Limiting Constraints
## Attenuation and Transmission Power

- Attenuation limits the range of transmission
  - guided: logarithmic dependent on medium (~0.1–10 dB/km)
  - wireless: square law $1/r^2$ – $1/r^4$ (with multipath)
- Transmission energy needed to compensate
  - limited by transmitter power
  - constrained by channel design parameters
    - interference with other channels

**Attenuation and Power** III.9

*Attenuation of signals limits their propagation distance for a given transmission energy. Transmission energy is limited by the power available of the transmitter, and may be constrained by the design parameters of the transmission medium*

# Limiting Constraints
## Switching Speed

- Switching rate of electronic & photonic components
  - dictates fundamental limit on data rate
  - Moore's law keeps reducing the constraint
  - impact on data rate dependent on transistor complexity
    - transmission rate typically 4–10× > packet processing rate
      - e.g. OC-768 vs. OC-192

**Switching Speed** III.3

*There are limits on switching frequency of components, constrained by process technology at a given time, and ultimately limited by physics.*

# Limiting Constraints
## Cost and Feasibility

- Competing solutions have different costs
  - $C = f + f(v)$: fixed plus variable (scaling) cost
  - cost of deployment is significant determinant

**Cost and Feasibility** III.4

*The relative cost and scaling complexity of competing architectures and designs must be considered in choosing alternatives to deploy.*

# Limiting Constraints
## Heterogeneity

- Networks serve to interconnect heterogeneous
  - users, end systems, applications

  using heterogeneous
  - technologies, network infrastructure
- Significant overhead needed to support heterogeneity
  - transcoding, format conversion, control interworking

**Heterogeneity** III.5

*The network is a heterogeneous world, which contains the applications and end systems that networks tie together, and the node and link infrastructure from which networks are built.*

# Limiting Constraints
## Policy and Administration

- Policies & administrative concerns constrain networks
  - economics and business models
  - intellectual property and legal issues
  - government regulation
  - social dynamics

**Policy and Administration** III.6

*Policies and administrative concerns frustrate the deployment of optimal high-speed network topologies, constrain the paths through with applications can communicate, and may dictate how application functionality is distributed.*

# Limiting Constraints
## Backward Compatibility

- Backward compatibility is very important
  - the Internet cannot be simply upgraded to a new version
    - network infrastructure
    - end systems
- Change is
  - incremental
  - hacks extend the life of existing non-extensible protocols

**Backward Compatibility Inhibits Radical Change** III.7

*The difficulty of completely replacing widely deployed network protocols means that improvements must be backward compatible and incremental. Hacks are used and institutionalised to extend the life of network protocols.*

# Limiting Constraints
## Standards

- Standards are a very difficult dilemma
  - necessary for interoperation
  - timing is critical
    - too early or over specific: impedes progress
    - too late or general: useless
- De facto standards may be better than official ones
  - monopolies threaten both (e.g. MS-Windows)

**Backward Compatibility Inhibits Radical Change** III.8

*Standards are critical to facilitate interoperability, but standards that are specified too early or are overly specific can impede progress. Standards that are specified too late or are not specific enough are useless.*

# Fundamentals and Design Principles
## Design Principles and Tradeoffs

# Fundamental Axioms
## Systemic Optimisation

**Systemic Optimisation Principle** IV

*Networks are systems of systems with complex compositions and interactions at multiple levels of hardware and software. These pieces must be analysed and optimised in concert with one another.*

- Corollaries
  1. consider side effects
  2. keep it simple and open
  3. system partitioning
  4. flexibility and workaround

# Systemic Optimisation
## Granularity

- Many orders of magnitude time difference: $>10^{18}$
  - protocol and architecture standardisation
  - protocol deployment
  - network configuration
  - network management
  - session establishment and lifetime
  - connection/flow establishment and lifetime
  - packet/frame transfer
  - cell transfer
  - byte transfer
  - bit/photon transfer
- Principles help isolate/mitigate concerns...
  ...but not completely!

# Systemic Optimisation
## Side Effects

- Optimisations frequently have side effects
  - may reduce effectiveness of optimisation
  - may reduce *overall* performance
- Careful systemic analysis needed

**Consider Side Effects** $IV_1$

*Optimisations frequently have unintended side effects to the detriment of overall performance. It is important to consider, analyse, and understand the consequences of optimisations.*

# Systemic Optimisation
## Simple and Open

- Difficult to understand and optimise complex systems
- Virtually impossible for closed systems
  - open source highly desirable
  - open interfaces essential with sufficient functionality

**Keep it Simple and Open** $IV_2$

*It is difficult to understand and optimise complex systems, and virtually impossible to understand closed systems, which do not have open published interfaces.*

# Systemic Optimisation
## System Partitioning

- Essential to analyse partitioning of functionality
  - across network
  - among components
    - switch node central or line interface
    - host CPU or network interface
- Improper partitioning
  - decreases overall performance
  - may increases overall cost

**System Partitioning Corollary** $IV_3$

*Carefully determine how functionality is distributed across a network. Improper partitioning of a function can dramatically reduce overall performance, and increase overall cost.*

# Systemic Optimisation
## Flexibility and Workaround

- Impossible to perfectly future-proof systems
- Provide mechanisms to allow future enhancements
  - protocol fields
  - control mechanism
  - software hooks

**Flexibility and Workaround Corollary** $IV_4$

*Provide protocol fields, control mechanisms, and software hooks to allow graceful enhancements and workarounds when fundamental tradeoffs change.*

# Systemic Optimisation
## Design Principles

- Systemic optimisation supported by design principles
  1. Selective optimisation
  2. Resource tradeoffs
  3. End-to-end arguments
  4. Protocol layering
  5. State management
  6. Control mechanism latency
  7. Distributed data
  8. Protocol data units

# Selective Optimisation
## Selective Optimisation

- Corollaries
  - second order effects
  - critical path
  - functional partitioning and assignment

### Selective Optimisation Principle 1

*It is neither practical nor feasible to optimise everything. Spend implementation time and system cost on the most important constituents of performance.*

# Selective Optimisation
## Second-Order Effects

- Impact of optimisations should be understood
- Optimising second-order effects is not useful
  - e.g. optimising link that is not bottleneck
  - e.g. optimising LAN latency for a WAN application
  - e.g. optimising operations not in critical path

### Second-Order Effect Corollary 1A

*The impact of spatially local or piecewise optimisations on the overall performance must be understood; components with only a second-order effect on performance should not be the target of optimisation.*

# Selective Optimisation
## Critical Path

- Optimise *critical path* on which high speed depends
  - data transfer operations
  - transfer control operations that assist in data transfer
    - occur frequently
    - cause serial delay to subsequent data transfer

**Critical Path Corollary** 1B

*Optimise implementations for the critical path, in both control and data flow.*

# Selective Optimisation
## Functional Partitioning and Assignment

- Essential to analyse partitioning of functionality
  - between hardware and
    - electronic vs. photonic
    - CMOS vs. GaAs
    - DRAM vs. SRAM vs. CAM
  - software
    - compiled vs. hand optimised
    - main memory vs. cache
- Improper partitioning
  - increases overall cost
  - may decreases overall performance

**Functional Partitioning and Assignment Corollary** 1C

*Carefully determine what functionality is implemented in scarce or expensive technology. Improper partitioning of a function can dramatically increase overall cost and reduce performance.*

# Resource Tradeoffs

## Resource Tradeoffs

- Network is collection of resources: P, M, B, E
  and constraints: L
- Balance relative composition to optimise
  - performance and cost
- Corollaries
  - resource tradeoffs change
  - optimal resource utilisation vs. overengineering tradeoffs
  - support for multicast

### Resource Tradeoff Principle 2

*Networks are collections of resources. The relative composition of these resources must be balanced to optimise cost and performance.*

# Resource Tradeoffs
## Resource Tradeoffs Change

- Relative cost of resources change over time
  - non-uniform advances in technology
  - changing application constraints
- Design and engineering for future
  - track trends and shifts in resource tradeoffs
  - be prepared for significant and disruptive shifts

**Resource Tradeoffs Change** 2A

*The relative cost of resources and the impact of constraints change over time, due to non-uniform advances in different aspects in technology.*

# Resource Tradeoffs
## Optimality vs. Overengineering

- Optimal use of a resource costs others
  - e.g. optimal bandwidth reservation and utilisation costs
    - memory for link state and topology databases
    - processing for admission control and resource reservation
- Optimise entire system cost
  - overengineering of a resource may reduce cost of another

**Optimal Resource Utilisation vs. Overengineering** 2B

Tradeoff *Balance the benefit of optimal resource utilisation against the costs of the algorithms that attempt to achieve optimal solutions.*

# Resource Tradeoffs
## Multicast

- Multicast needed by network control protocols
  - e.g. address resolution
- Multicast group communication model for apps
  - e.g. audio and video distribution
  - L3 multicast less bandwidth than $n^2$ point-to-point mesh
    - requires network layer protocol support e.g. IGMP, ATM
    - requires native switch multicast

**Support for Multicast** 2C

*Multicast is an important mechanism for conserving bandwidth and supporting network control protocols, and should be supported by the network.*

# End-to-End vs. Hop-by-Hop
## End-to-End Argument

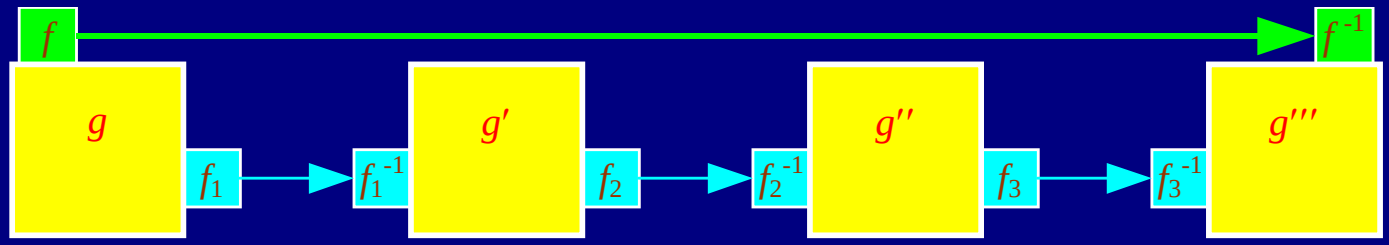
- End-to-End Arguments *rephrased from [Saltzer 1984]
  1. end-to-end argument
  2. hop-by-hop performance enhancement

### End-to-End Argument* 3

*Functions required by communication applications can be correctly and completely implemented only with the knowledge and help of the applications themselves. Providing these functions as features within the network itself is not possible.*

# End-to-End vs. Hop-by-Hop

## End-to-End Argument




- Hop-by-Hop functions do not compose end-to-end
  - between HBH boundaries, function *f* is defeated (*g*)
    - e.g. error control: errors may occur *within* switches
    - e.g. encryption: cleartext *within* switches may be snooped
  - these functions *must* be done E2E
    - doing them HBH is *redundant*, and may lower performance
- Corollaries
  - hop-by-hop performance enhancement
  - endpoint recursion

# End-to-End vs. Hop-by-Hop
## Hop-by-Hop Performance Enhancement

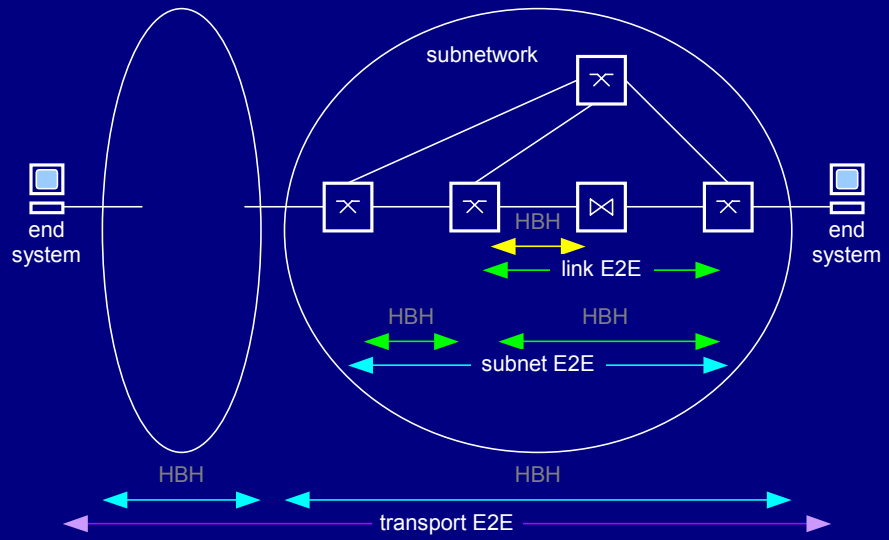
- E2E Argument (1st half) says what *must* be E2E
- HBH Performance enhancement (2nd half)
  - functions *should* duplicated HBH if *overall* E2E benefit
    - e.g. wireless link error control reduces E2E control overhead shorter control loop frequent; longer control loop infrequent
- Analysis is required to determine cost/benefit

**Hop-by-Hop Performance Enhancement Corollary*** 3A

*It may be beneficial to duplicate an end-to-end function hop-by-hop, if doing so results in an overall (end-to-end) improvement in performance.*

# End-to-End vs. Hop-by-Hop
## Endpoint Recursion

- End-to-end arguments
  - apply edge-to-edge
  - subnet-to-subnet
  - link-to-link
- Overall E2E concerns must not be forgotten

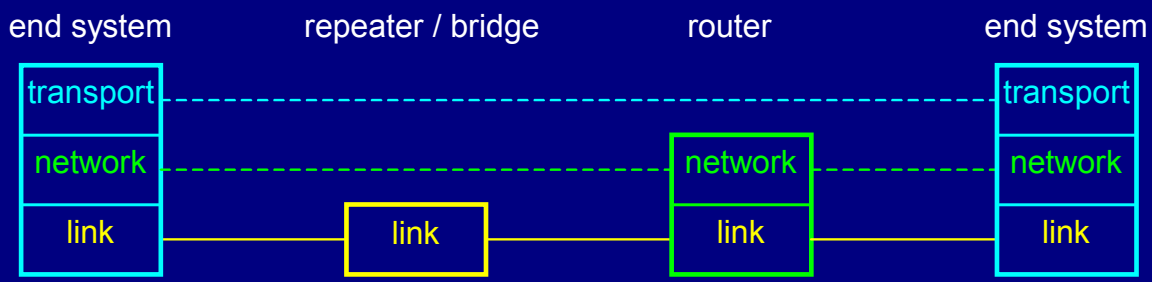



### Endpoint Recursion Corollary 3B

*What is hop-by-hop in one context may be end-to-end in another. The End-to-End Arguments can be applied recursively to any sequence of nodes in the network, or layers in the protocol stack.*

# Protocol Layering
## Protocol Layering

- Layering is a useful abstraction
  - role-based
    2. link
    3. switch
    4. end system
    5. session (control plane)
    7. application

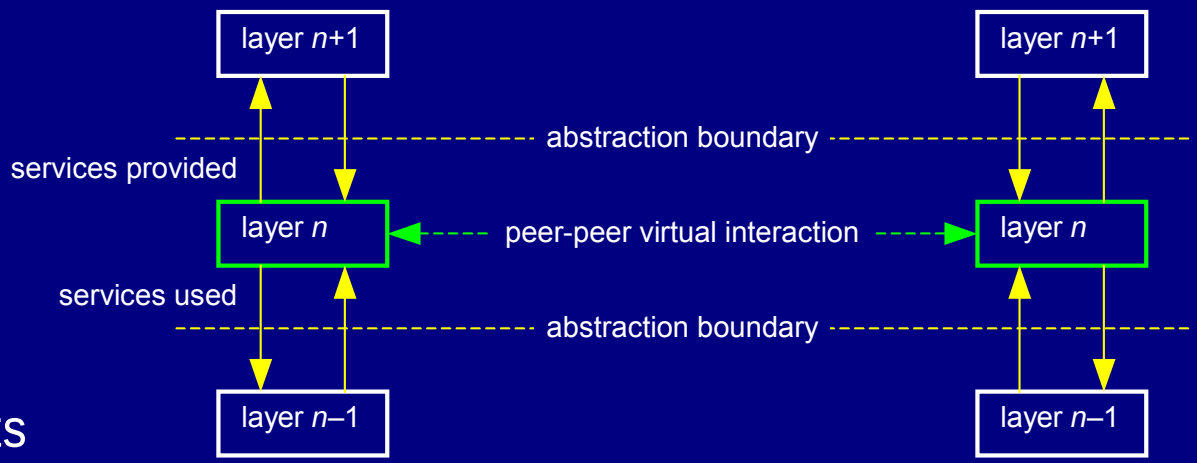



### Protocol Layering Principle 4

*Layering is a useful abstraction for thinking about networking system architecture and for organising protocols based on network structure.*

# Protocol Layering
## Protocol Layering

- Layering provides service abstraction
  - isolate: protocols components technology
    - any transport layer over IP
    - IP over any link layer
    - commodity link layer chip evolution, e.g.
      - 10BASE-T → 100BASE-T → 1000BASE-X → 802.11b → 802.11a

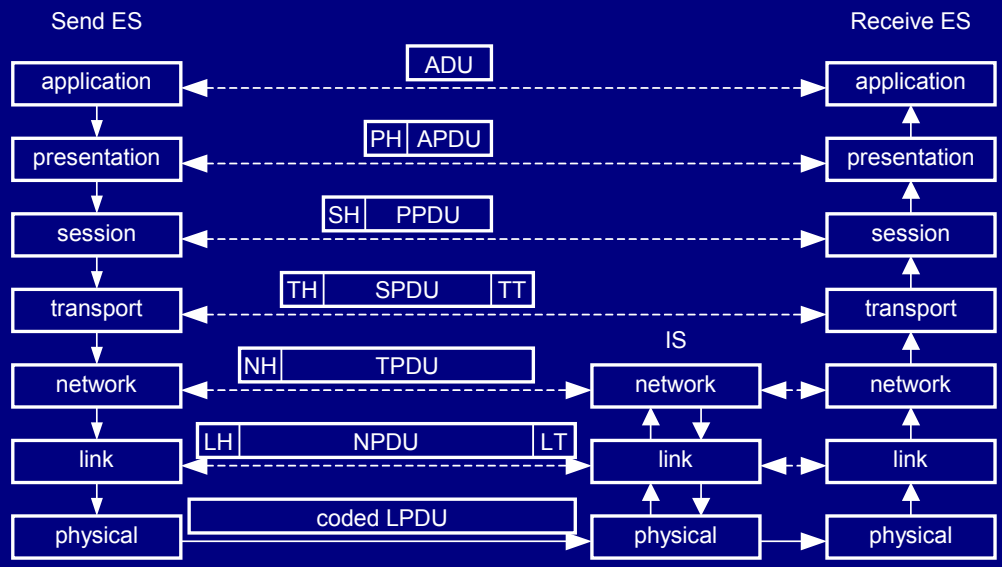


# Protocol Layering
## Protocol Layering

- Corollaries
  - laying as an implementation technique performs poorly
  - redundant layer functionality
  - layer synergy
  - hourglass
  - integrated layer processing
  - balance transparency and abstraction vs. hiding
  - support a variety of interface mechanisms
  - interrupt vs. polling
  - interface scalability

# Protocol Layering
## Layering as Implementation Technique

- Layered model
  - specifies service abstraction
  - doesn't specify implementation
  - process/layer implementation terribly inefficient




**Layering as an Implementation Technique Performs** 4A

**Poorly** *Layered protocol architecture should not be confused with inefficient layer implementation techniques.*

# Protocol Layering
## Redundant Functionality

- Functionality should not be included in a layer
  - that must be located at a higher layer (E2E argument)
  - unless an overall performance benefit (HBH corollary)
- E2E vs. A2A (application-to-application) functionality

**Redundant Layer Functionality Corollary** 4B

*Functionality should not be included at a layer that must be duplicated at a higher layer, unless there is a performance benefit in doing so.*

# Protocol Layering
## Layer Synergy

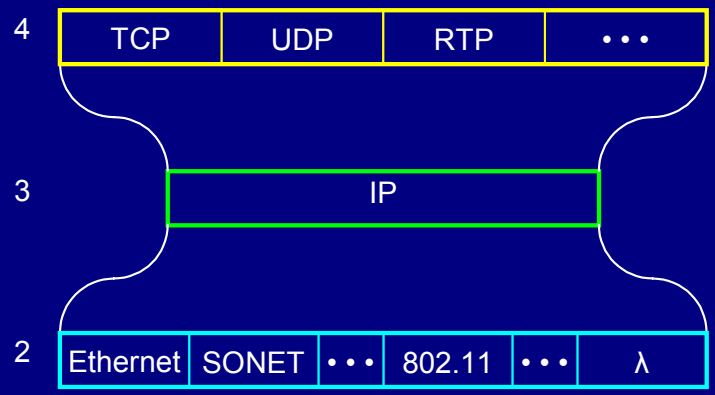
- Inter-layer transfers involve non-trivial overhead
  - encapsulation/decapsulation of PDUs
  - inter-layer control transfer
  - effects of overlapping intra-layer control mechanisms
- Protocol layers should be designed with this in mind
  - antithesis of layering to isolate protocols and technology

**Layer Synergy Corollary** 4C

*When layering is used as a means of protocol division, allowing asynchronous protocol processing and independent data encapsulations, the processing and control mechanisms should not interfere with one another. Protocol data units should translate efficiently between layers.*

# Protocol Layering
## Hourglass

- Common network layer
  - common addressing essential for seamless interworking
  - compatible routing & signalling
- Active networking
  - reduces constraint

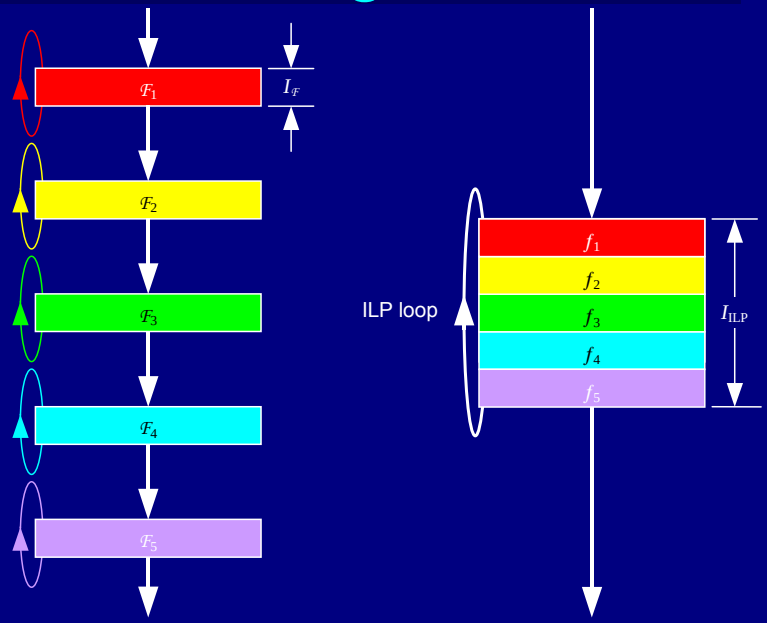



### Hourglass Corollary 4D

*The network layer provides the convergence of addressing, routing, and signalling that ties the global Internet together. It is essential that addressing be common and that routing and signalling protocols be highly compatible.*

# Protocol Layering
## Integrated Layer Processing

- Process multiple layers simultaneously
  - en/decapsulations
  - per byte operations
    - checksums
    - copies
- Software or hardware




**Integrated Layer Processing (ILP) Corollary** 4E

*When multiple layers are generated or terminated in a single component, all encapsulations/decapsulations should be done at once, if possible.*

# Protocol Layering
## Transparency and Hiding

- Layering abstracts interface below
  - simpler representation of complex interface
- Hiding needed properties or parameters Is bad
- Translucency is better then transparency

**Balance Transparency and Abstraction vs. Hiding** 4F

*Layering is designed around abstraction, providing a simpler representation of a complicated interface. Abstraction can hide necessary property or parameter, which is not a desirable property of layering.*

# Protocol Layering
## Variety of Interface Mechanisms

- Interface mechanisms are best in different situations
  - synchronous vs. asynchronous
    - synchronised interlayer interfaces
    - asynchronous nondeterministic interlayer interfaces
  - interrupt vs. polled
- Interlayer interfaces should provide necessary variety

**Support a Variety of Interface Mechanisms** 4G

*A range of interlayer interface mechanisms should be provided as appropriate for performance optimisation: synchronous and asynchronous, as well as interrupt-driven and polled.*

# Protocol Layering
## Interrupt vs. Polled

- Polling: more efficient mechanism than interrupts
  - when event timing is known a priori
  - otherwise extra polling and spin locks *less* efficient
- Interrupts: significant context switch overhead
  - more efficient *overall* when event timing not known a priori
- Hybrid
  - interrupt for first event
  - then polled for sequence of expected events

**Interrupt vs. Polling** 4H

*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 Layering
## Interface Scalability

- Interlayer interfaces relate to performance and scale
  - parameter values: max values and range
  - control fields (e.g. hierarchy and source routes)
- Interfaces need to balance
  - efficient encoding for current and near future networks vs.
  - scalability for future
    - base value and multiplier e.g. TCP window scaling option
    - concatenation of fields e.g. MPLS label stack

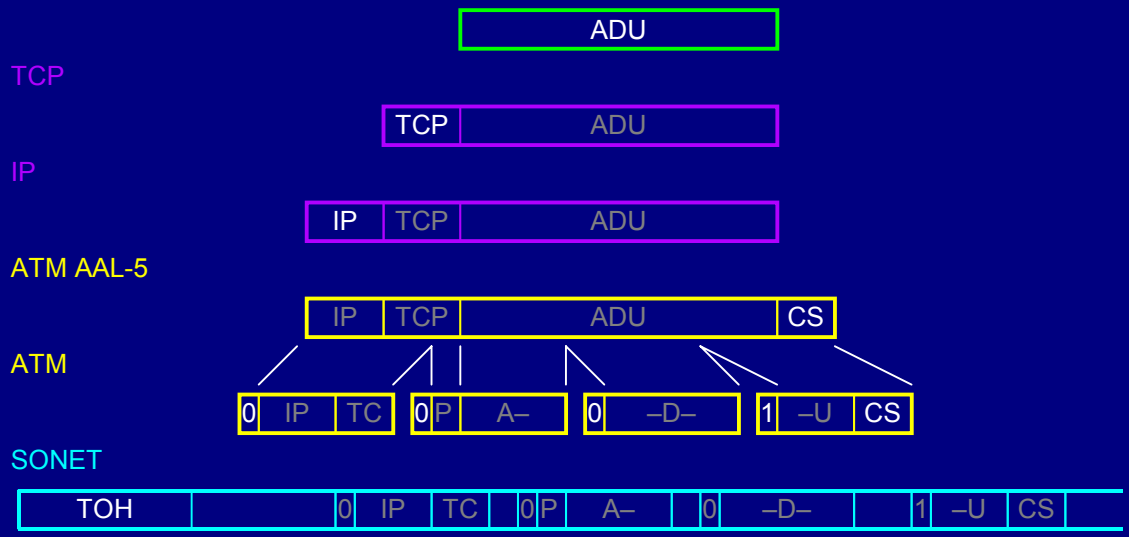
**Interface Scalability Corollary** 41

*Interlayer interfaces should support the scalability of the network and parameters transferred among the application, protocol stack, and network components.*

# Protocol Layering

## Example$_{2.1}$ ATM and the Revenge of Layering

- 5-6 layers:
  - TCP over
  - IP over
  - AAL-5 over
  - ATM over
  - SONET
  - OTN




- ATM supposed to be fast packet switching but
  - IP/ATM interface *completely* incompatible
  - 40B TCP ACK takes *two* 48B ATM cells

# State and Hierarchy
## State Management

- Corollaries
  - hard state vs. soft state vs. stateless tradeoff
  - aggregation and reduction of date transfer
  - hierarchy corollary
  - scope of information tradeoff
  - assumed initial conditions
  - minimise control overhead

**State Management Principle** 5

*The mechanisms for installation and management of state should be carefully chosen to balance fast, approximate, and coarse-grained against slow, accurate and fine-grained.*

# State and Hierarchy
## Hard vs. Soft State

- Hard state
  - predictable & deterministic
  - latency to establish
- Stateless
  - resilient to failure
  - overhead per data unit
- Soft state
  - intermediate mechanism
  - state accumulation without latency
  - resilience to failures

**Hard State vs. Soft State vs. Stateless Tradeoff** 5A

*Balance the tradeoff between the latency to set up hard state on a per connection basis versus. the per data unit overhead of making stateless decisions or of establishing and maintaining soft state.*

# State and Hierarchy
## Aggregation of State

- State aggregation benefits
  - reduces amount of information stored
  - reduces bandwidth used for state transfer
- State aggregation costs
  - loss of precision and fine-grained control
  - state shared is fate shared

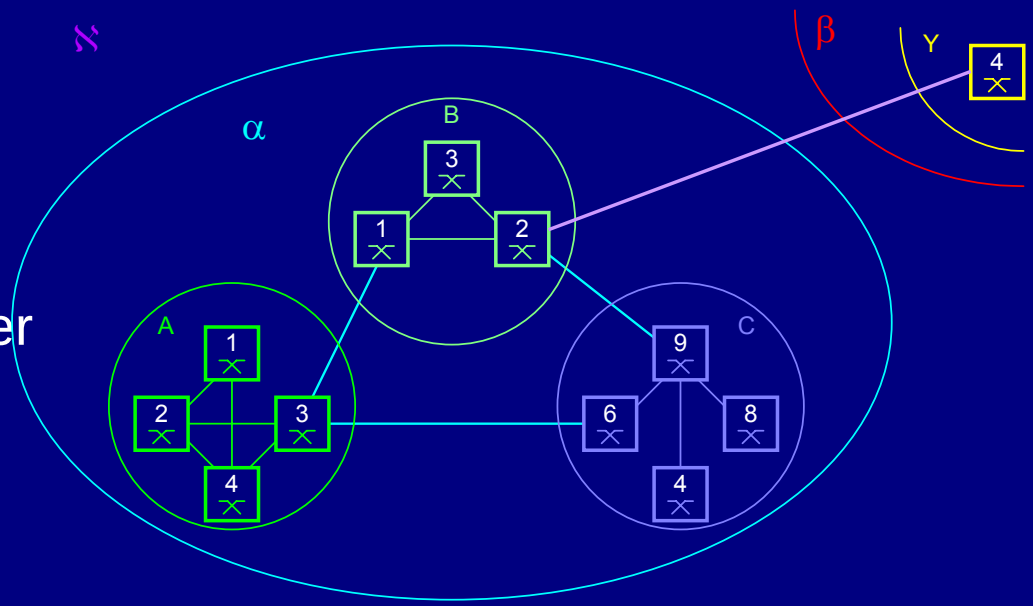
### Aggregation and Reduction of State Transfer 5B

*Aggregation of state reduces the amount of information stored. Reducing the rate ate which state information is propagated through the network reduces bandwidth and processing at network nodes, which comes at the expense of finer-grained control with more precise information.*

# State and Hierarchy

## Hierarchy

- Hierarchy aggregates & abstracts
  - full info intracluster
  - abstracted below
- Scalability




### Hierarchy Corollary 5C

*Use hierarchy and clustering to manage complexity by abstracting and aggregating information to higher levels and to isolate the effects of changes within clusters.*

# State and Hierarchy
## Scope of Information

- Scope and accuracy of information tradeoff
  - local information: less accurate – important quick decisions
    - quickly accessible
  - global scope: more accurate – only when needed
    - more overhead
      - delay to access or
      - overhead in keeping globally synchronised

### Scope of Information Tradeoff 5D

*Make quick decisions based on local information when possible. Even if you try to make a better decision with more detailed global state, by the time information is collected and filtered, the state of the network may have changed.*

# State and Hierarchy
## Scope of Information

- Assumed initial conditions based on
  - past history
  - related associations
- Allows fast initialisation or state installation
- Fine tune later

### Assumed Initial Conditions 5E

*Use reasonable assumptions based on past history or related associations to establish initial conditions for the installation of new state*

# State and Hierarchy
## Control Overhead

- Signalling and control overhead is necessary
- Keep low to maximise ability to transport data

### Minimise Control Overhead 5F

*The purpose of a network is to carry application data. The processing and transmission overhead introduced by control mechanisms should be kept as low as possible to maximise the fraction of network resources available for carrying application data.*

# Control Mechanisms
## Latency

- Low latency control mechanisms
- Corollaries
  - minimise round trips
  - exploit local knowledge
  - anticipate future state
  - separate control mechanisms

### Control Mechanism Latency Principle 6

*Effective network control depends on the availability of accurate and current information Control mechanisms must operate within convergence bounds that are matched to the rate of change in the network, as well as latency bounds to provide low interapplication delay.*

# Control Mechanisms
## Round Trips

- Techniques to minimise control round trips
  - hop-by-hop acknowledgements
  - parameter ranges
    - desired – minimum acceptable
    - desired, alternate
  - overlap of control and data

**Minimise Round Trips** 6A

*Structure control messages and the information they convey to minimise the number of round trips required to accomplish data transfer.*

# Control Mechanisms
## Exploit Local Knowledge

- Exert control from entity that has knowledge
  - reduces number of E2E control message signalling latencies
  - delegate control to entity with knowledge
    - e.g. fast resource reservation by transaction server
  - move knowledge to control entity
    - e.g. server push

**Exploit Local Knowledge** 6B

*Control should be exerted by the entity that has the knowledge needed to do so directly and efficiently.*

# Control Mechanisms
## Anticipate Future State

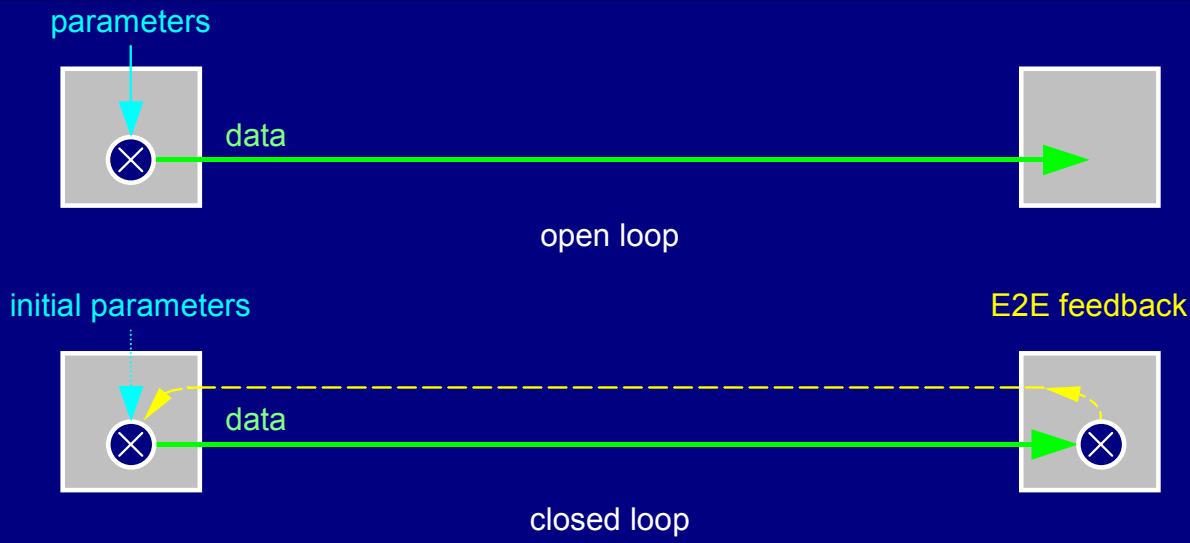
- Anticipate future state
  - proactive control before needed
  - quick reactive control without E2E control signalling
  - e.g. predictive algorithm with periodic E2E convergence

**Anticipate Future State** 6C

*Anticipate future state so that actions can be taken proactively before before repair needs to be performed on the network that affects application performance.*

# Control Mechanisms

## Open vs. Closed Loop Control




### Open- vs. Closed Loop Control 6D

*Use open-loop control based on knowledge of the network path to reduce the delay in closed loop convergence. Use closed-loop control to react to dynamic network and application behaviour.*

# Control Mechanisms
## Separate Control Mechanisms

- Combined control mechanisms frequently suboptimal
  - e.g. combined TCP error/flow/congestion control bad for wireless links
- Control mechanisms should be
  - distinct
  - if combined, distinct information explicitly conveyed
    - e.g. explicit congestion and error notification

**Separate Control Mechanisms** 6E

*Control mechanisms should be distinct, or the information conveyed explicit enough, so that the proper action is taken.*

# Distribution of Application Data
## Distributed Data

- Distribute data among applications to
  - minimise latency and amount of data exchanged
  - allow incremental processing of data as it arrives
- Corollaries
  - partitioning and structuring of data
  - location of data

**Distributed Data Principle** 7

*Distributed applications should select and organise the data they exchange to minimise the amount of data transferred and the latency o transfer and allow incremental processing of data.*

# Distribution of Application Data
## Partitioning and Structuring

- Partition & structure data distributed application data
  - to minimise amount of data transferred
  - to minimise latency of transfer

**Partitioning and Structuring of Data** 7A

*Whenever possible, data should be partitioned and structured to minimise the amount of data transferred and the latency in communication.*

# Distribution of Application Data
## Location

- Locate data near application that uses it
  - replicate data when possible (trade M vs. L)
    - e.g. mirroring and caching
  - anticipate transfer request in advance (reduce L and peak B)
    - e.g. prefetching

**Location of Data** 7B

*Data should be positioned close to the application that needs it to minimise the latency of access. Replication of data help to accomplish this.*

# Protocol Data Units
## Protocol Data Units

- PDU size and structure critical to performance
- Corollaries
  - PDU size and granularity
  - PDU control field structure
  - control field scalability

**Protocol Data Unit Principle** 8

*The size and structure of PDUs are critical to high-bandwidth, low-latency communication.*

# Protocol Data Units
## Size and Granularity

- Small vs. large packets
  - statistical multiplexing efficiency vs. more time to process
- Fixed vs. variable size
  - easier to process vs. more flexible
  - granularity: byte, word, power-of-2, end system buffer

**PDU Size and Granularity** 8A

*The size of PDUs is a balance of a number of parameters that affect performance. Trade the statistical multiplexing benefits of small packets against the efficiency of large packets.*

# Protocol Data Units
## Control Field Structure

- Header/trailer structure has performance impact
  - simple encoding (bit vector vs. code points)
  - byte/octet granularity and alignment
  - fixed length fields when possible
    - offset value when variable length necessary

### PDU Control Field Structure 8B

*Optimise PDU header and trailer fields for efficient processing. Fields should be simply encoded, byte aligned, and fixed length when possible. Variable length fields should be prepended with their length.*

# Protocol Data Units
## Control Field Scalability

- Scalability of control fields
  - large linear range requires many bits
  - value and multiplier allows scaling over orders of magnitude

**Scalability of Control Fields** 8C

*PDU header and trailer fields and structure should be scalable with data rate and bandwidth-×-delay product.*

# Fundamentals and Design Principles
## Design Techniques