Architecture de protocoles haute performance

Bloc 5, INF 586

Walid Dabbous

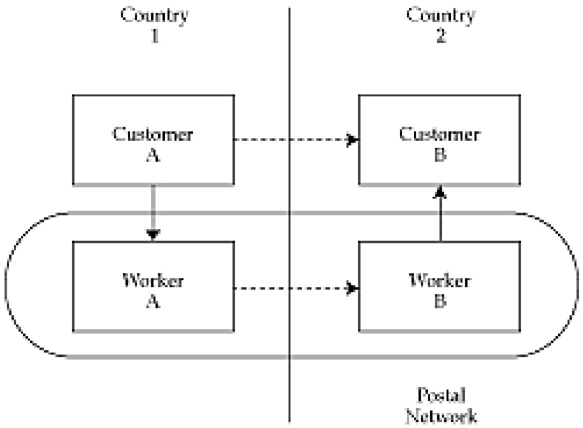
INRIA Sophia Antipolis

Outline

- Protocol Layering
- Layering considered harmful
- Enhancing Protocol performance
- Adaptive applications
- Application Level Framing

Protocol Layering

Peer entities



- Customer A and B are *peers*
- Postal worker A and B are *peers*

Protocols

- A *protocol* is a set of rules and formats that govern the communication between communicating peers
  - set of valid messages
  - meaning of each message
- A protocol is necessary for any function that requires cooperation between peers

Example

- Exchange a file over a network that corrupts packets
  - but doesn’t lose or reorder them
- A simple protocol
  - send file as a series of packets
  - send a *checksum*
  - receiver sends OK or not-OK message
  - sender waits for OK message
  - if no response, resends entire file
- Problems
  - single bit corruption requires retransmission of entire file
  - what if link goes down?
  - what if not-OK message itself is corrupted?

What does a protocol tell us?

- *Syntax* of a message
  - what fields does it contain?
  - in what format?
- *Semantics* of a message
  - what does a message mean?
  - for example, not-OK message means receiver got a corrupted file
- *Actions* to take on receipt of a message
  - for example, on receiving not-OK message, retransmit the entire file

- The three above called: protocol specification

Another way to view a protocol

- As providing a *service*
- The example protocol provides *reliable file transfer service*
- Peer entities use a protocol to provide a service to a higher-level peer entity
  - for example, postal workers use a protocol to present customers with the abstraction of an *unreliable letter transfer* service

Protocol layering

- A network that provides many services needs many protocols
- Turns out that some services are independent
- But others depend on each other
- Protocol A may use protocol B as a *step* in its execution
  - for example, packet transfer is one step in the execution of the example reliable file transfer protocol
- This form of dependency is called *layering*
  - reliable file transfer is *layered* above packet transfer protocol
  - like a subroutine

Some terminology

- *Service access point (SAP)*
  - interface between an upper layer and a lower layer
- *Protocol data units (PDUs)*
  - packets exchanged between peer entities
- *Service data units (SDUs)*
  - packets handed to a layer by an upper layer
- PDU = SDU + optional header or trailer
- Example
  - letter transfer service
  - protocol data unit between customers = letter
  - service data unit for postal service = letter
  - protocol data unit = mailbag (aggregation of letters)

Protocol stack

- A set of protocol layers
- Each layer uses the layer below and provides a service to the layer above
- Key idea
  - once we define a service provided by a layer, we need know nothing more about the details of *how* the layer actually implements the service
  - information hiding
  - decouples changes
  - but reduces system performance!

The importance of being layered

- Breaks up a complex problem into smaller manageable pieces
  - can compose simple service to provide complex ones
  - for example, WWW (HTTP) is Java layered over TCP over IP (and uses DNS, ARP, DHCP, RIP, OSPF, BGP, PPP, ICMP)
- Abstraction of implementation details
  - separation of implementation and specification
  - can change implementation as long as service interface is maintained (long distance telephone migration from copper to fiber)
- Can reuse functionality
  - upper layers can share lower layer functionality
  - example: WinSock on Microsoft Windows

Problems with layering

- Layering hides information
  - if it didn't then changes to one layer could require changes everywhere
    - *layering violation*
- But sometimes hidden information can be used to improve performance
  - for example, flow control protocol may think packet loss is always because of network congestion
  - if it is, instead, due to a lossy link, the flow control breaks
  - this is because we hid information about reason of packet loss from flow control protocol

Layering

- There is a tension between information-hiding (abstraction) and achieving good performance
- Art of protocol design is to leak enough information to allow good performance
  - but not so much that small changes in one layer need changes to other layers

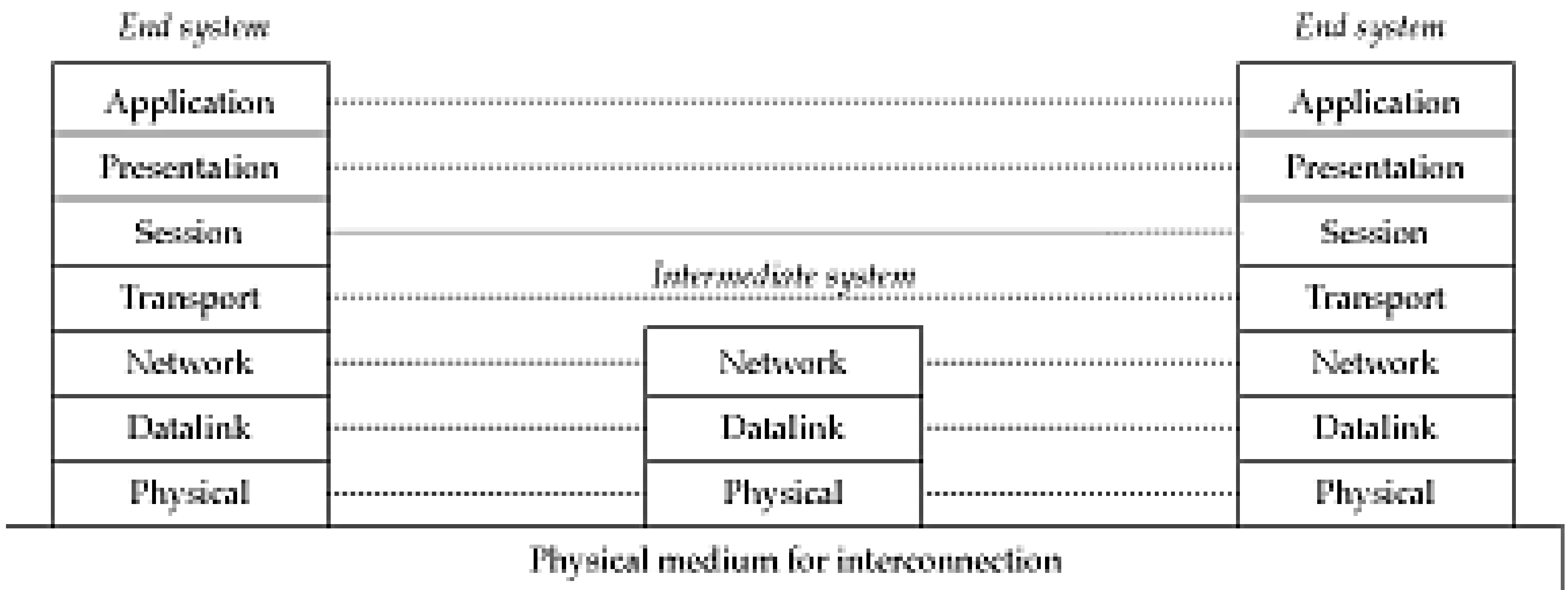
ISO OSI reference model

- A set of protocols is *open* if
  - protocol details are publicly available
  - changes are managed by an organization whose membership and transactions are open to the public
- A system that implements open protocols is called an *open system*
- Any vendor can implement open standard compliant systems
- International Organization for Standards (ISO) prescribes a standard to connect open systems
  - *open system interconnect (OSI)*
- Has greatly influenced thinking on protocol stacks

ISO OSI

- *Reference model*
  - formally defines what is meant by a layer, a service etc.
- *Service architecture*
  - describes the services provided by each layer and the service access point
- *Protocol architecture*
  - set of protocols that implement the service architecture
  - compliant service architectures may still use non-compliant protocol architectures

The seven layers



Physical layer

- Moves bits between physically connected end-systems
- Standard prescribes
  - coding scheme to represent a bit
  - shapes and sizes of connectors
  - bit-level synchronization
- Postal network
  - technology for moving letters from one point to another (trains, planes, vans, bicycles, ships…)
- Internet
  - technology to move bits on a wire, wireless link, satellite channel etc.

Datalink layer

- Introduces the notion of a *frame*
  - set of bits that belong together
- *Idle* markers tell us that a link is not carrying a frame
- *Begin* and *end* markers delimit a frame
- On a broadcast link (such as Ethernet)
  - end-system must receive only bits meant for it
  - need datalink-layer address
  - also need to decide who gets to speak next
  - these functions are provided by *Medium Access sublayer (MAC)*
- Some data links also retransmit corrupted packets and pace the rate at which frames are placed on a link
  - part of *logical link control sublayer*
  - layered over MAC sublayer

Datalink layer (contd.)

- Datalink layer protocols are the first layer of *software*
- Very dependent on underlying physical link properties
- Usually bundle both physical and datalink layer on *host adaptor card*
  - example: Ethernet
- Postal service
  - mail bag 'frames' letters
- Internet
  - a variety of datalink layer protocols
  - most common is Ethernet
  - others are FDDI, SONET, HDLC

Network layer

- Logically concatenates a set of links to form the abstraction of an end-to-end link
- Allows an end-system to communicate with any other end-system by computing a route between them
- Hides specificities of datalink layer
- Provides unique network-wide addresses
- Found both in end-systems and in intermediate systems
- At end-systems primarily hides details of datalink layer
  - segmentation and reassembly
  - some error detection (e.g header check)

Network layer (contd.)

- At intermediate systems
  - participates in routing protocol to build routing tables
  - responsible for forwarding packets
  - scheduling the transmission order of packets (Not implemented)
  - choosing which packets to drop (Not deployed)

- IP only provides “best effort”
  - it runs an “all” underlying technologies

Transport layer

- Network provides a ‘raw’ end-to-end service
- Transport layer provides the abstraction of an *error-controlled*, *flow-controlled* and *multiplexed* end-to-end link
- Error control
  - message will reach destination despite packet loss, corruption and duplication
  - retransmit lost packets; detect, discard, and retransmit corrupted packets; detect and discard duplicated packets
- Flow control
  - match transmission rate to rate currently sustainable on the path to destination, and at the destination itself

Transport layer (contd.)

- Multiplexes multiple applications to the same end-to-end connection
  - adds an application-specific identifier (*port number)* so that receiving end-system can hand in incoming packet to the correct application
- Some transport layers provide fewer services
  - e.g. simple error detection, no flow control, and no retransmission
  - *lightweight transport layer*

Transport layer (contd.)

- Postal system
  - doesn’t have a transport layer
  - transport level functionality is implemented, if at all, by customers
  - detect lost letters (how?) and retransmit them
- Internet
  - two popular protocols are TCP and UDP
  - TCP provides error control, flow control, multiplexing
  - UDP provides only multiplexing

Session layer

- Not common
- Provides *full-duplex service, expedited data delivery*, and *session synchronization*
- Duplex
  - if transport layer is simplex, concatenates two transport endpoints together
- Expedited data delivery
  - allows some messages to skip ahead in end-system queues, by using a separate low-delay transport layer endpoint
- Synchronization
  - allows users to place marks in data stream and to roll back to a pre-specified mark

Presentation layer

- Unlike other layers which deal with *headers* presentation layer touches the application data
- Hides data representation differences between applications
  - e.g. *endian-ness*
- Can also encrypt data
- Usually *ad hoc*
- Internet
  - no standard presentation layer
  - only defines network byte order for 2- and 4-byte integers

Application layer

- The set of applications that use the network
  - File transfer
  - E-mail,
  - Web access,
  - Audio and video conferencing
  - Distributed games
  - Shared virtual environments
- Doesn't provide services to any other layer

Layering

- We have broken a complex problem into smaller, simpler pieces
- Provides the application with *sophisticated* services
- Each layer provides a clean abstraction to the layer above

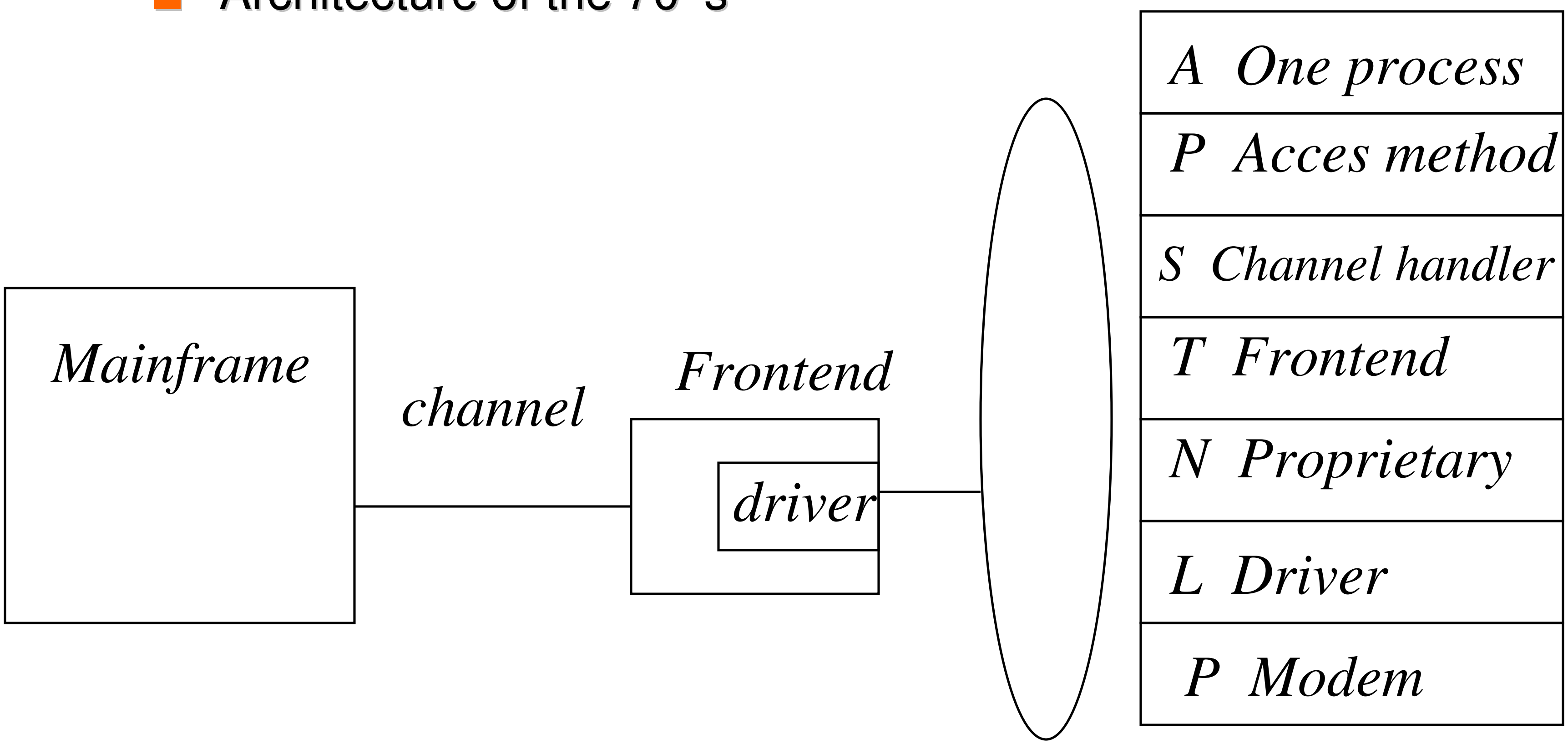
Layering considered harmful

Why seven layers?

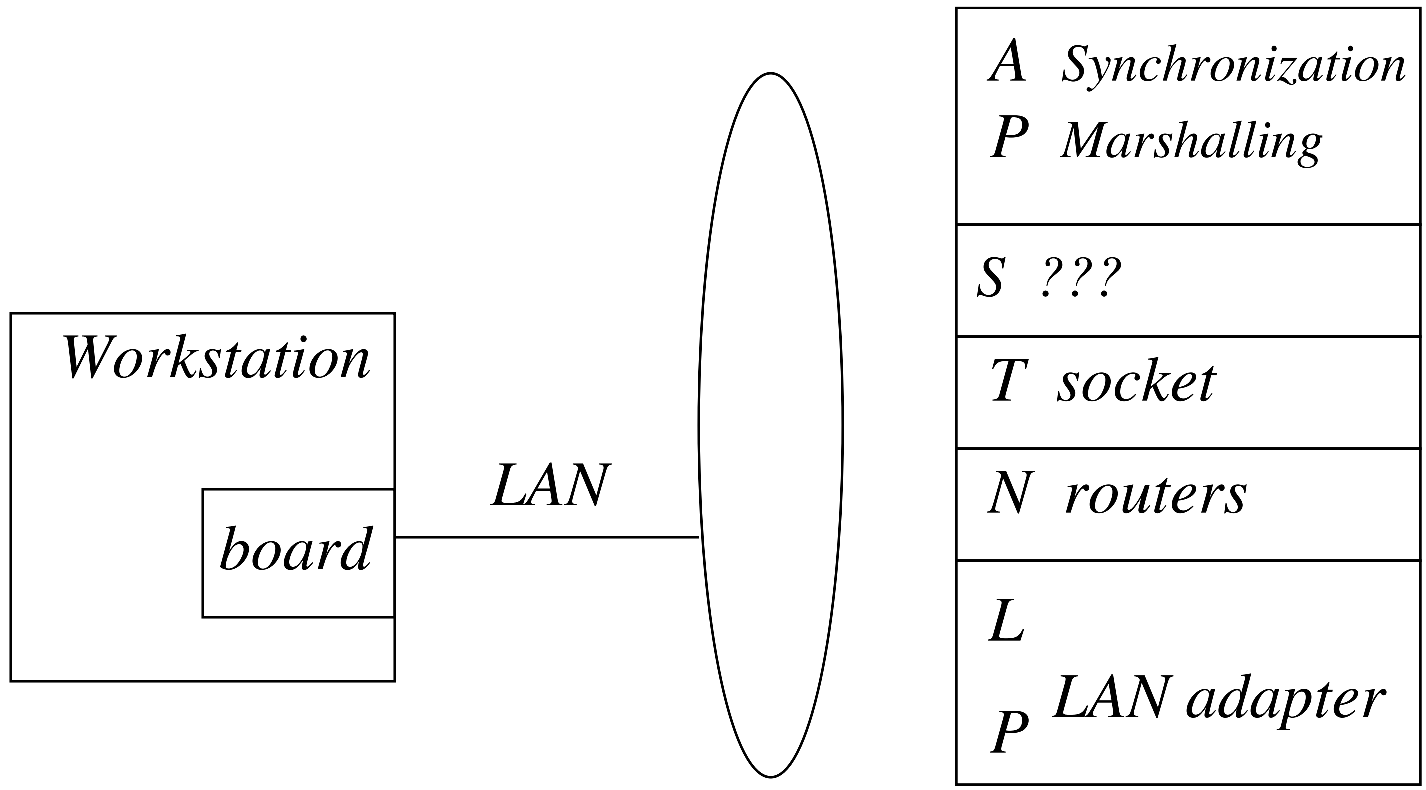
- Need a top and a bottom -- 2
- Need to hide physical link, so need datalink -- 3
- Need both end-to-end and hop-by-hop actions; so need at least the network and transport layers -- 5
- Session and presentation layers are not so important, and are often ignored
- So, we need at least 5, and 7 seems to be excessive
- Note that we can place functions in different layers
- Will study the impact on performance

Layering considered harmful

- Architecture of the 70 's



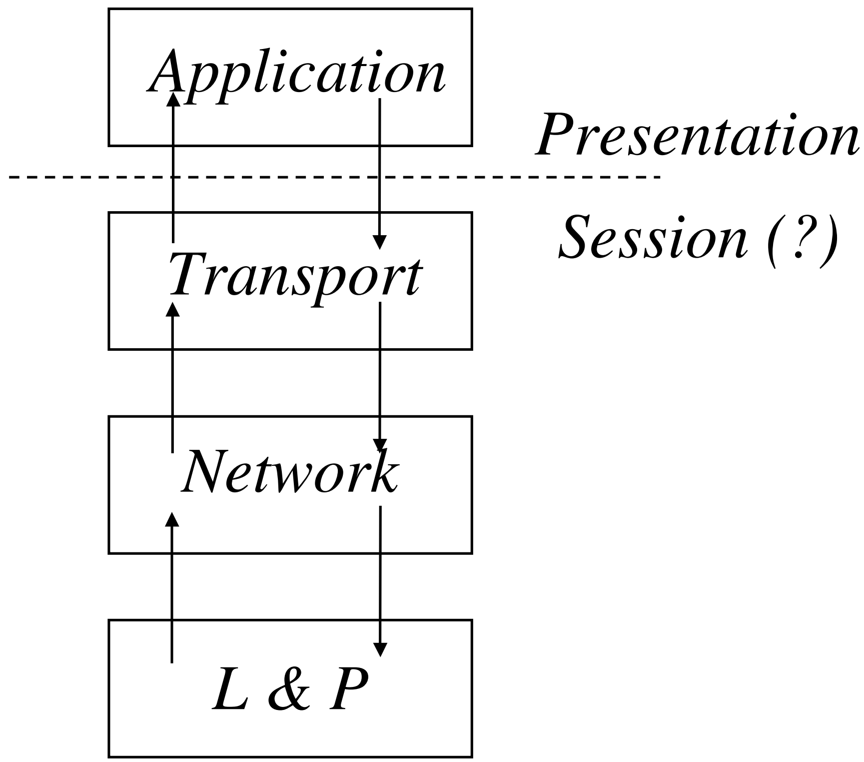
Architecture of the 90 's



Layering considered harmful

- A layer is considered as an asynchronous entity
- Independent « vendors » to develop layers hw/sw
- How to implement an asynchronous entity on a workstation
  - A subroutine?
    - No asynchrony
  - A process
    - several kind of overhead
      - process scheduling
      - asynchronous interfacing
      - passing control and data

What about persentation and session?



How to pass application control?

- « Transport » should be aware of the « Application »
  - e.g. VMTP replies serving as ACKs
- This avoids « bad » decisions that may be taken by the protocol
  - ACK a packet when a reply is waiting
  - After a packet loss during a video transfer
    - close the window
    - retransmit the packet

Layering, the lessons

- Asynchronous interfaces result in reduced efficiency
  - reduce asynchrony

- Avoid « artificial separation » in layers
- Have such interfaces only when necessary i.e. between
  - transmission networks
  - end system hardware and operating system
  - end system communications stack

- Specific case for OSI failure:
  - slow standardization process (political done by the “goers”)
  - done before technology was mature

Enhancing protocol performance

“Enhancing Protocol Performance”

- Impact of the environment
- Parameter tuning & adaptive algorithms
- Special purpose protocols
- Can applications “share” performance?

Impact of the environment

- Implementing protocols in software inside the OS.
- The OS executes the protocol code
- For each packet
  - take an interrupt or two,
  - reset a timer or two
  - allocate a buffer
  - schedule a process

Operating system support

- An O.S. Wish List:
  - Shared memory among processes
  - Blocking on multiple events
  - Good I/O buffer management
  - Good resource management scheduler
  - Low overhead timers
  - High resolution clocks
- Over the last years, it has gotten much better

Generic protocol enhancements

- Protocol parameter tuning
  - Acknowledgments
    - ACK grouping
    - Nacks
  - Adaptive timer values
- New adaptation algorithms
  - TCP slow-start

Special purpose protocols

- Target a specific application
  - File transfer (NETBLT)
  - (distributed) Intra-system communication (VMTP)
- Multiple communication modules overhead
- Provide a toolkit
  - XTP
  - merges layers 3 and 4
  - not a good choice

Can we share performance?

- Two views of performance :
- The explicit approach -- classical telecommunications design.
  - Voice channels shall have a digital bandwidth of 64Kb/s
  - Direct match to application
- The implicit approach -- classical computer design.
  - Applications don’t seem to have real requirements
  - Live in a « virtual » world of performance.

The « virtual » world of computers

- Most computer systems attempt to hide the real performance limits of the hardware.
- Real memory limits -> virtual memory.
  - The bigger the program, the slower it runs.
- One CPU -> multiple processes.
  - The more processes, the slower each runs.
- One disk -> multiple files.
  - Can run out of disk space, but not file space
- In the virtual world:
  - Performance gets subdivided
  - Logical entities are not bounded a priori

Connectivity

- In the network world, connectivity is the logical entity
- Degree of connectivity is critical parameter
  - Not like telephone, but many conversations at once
  - Even a small workstation (a client) may have many simultaneous conversations
  - Patterns of connectivity are highly variable
- The computer is a programmable device
  - Network designers should live with this!
- Connectivity has nothing to do with bandwidth
  - many computer applications have very minimal demand for bandwidth. Especially high-connectivity applications

The « virtual » network

- The most natural model of a network (to a computer programmer)
- Bandwidth is a performance parameter, and just gets subdivided.
- Connectivity is a a logical construct, and should be unbounded

- A bit of a shock to the voice specialists
- Voice nets are exactly backwards from this
- Video nets as well.

Two examples

- Ethernet: a success
  - Very high connectivity (no set-up)
  - Fixed total bandwidth, arbitrary allocation (No allocation).
- ISDN circuits: Not a success
  - One logical connection per physical
  - Fixed bandwidth per connection
- Ethernet, with very poor bandwidth allocations tools, has supported effectively a wide range of applications

Can networks be virtual?

- In the world of virtual performance:
  - Add performance in needed quantity by system configuration
- Do networks work this way?
- In the past: we have not built network technology with « scalable » performance
  - Initial solution: gross over-design
  - Problem: it does not last.
- Current products are addressing this need
  - Switching hubs

Can protocols be virtual?

- Protocols like TCP are from the virtual school of performance. They assume that performance will be added *later*, in the proper quantities
- But when is « later »?
  - Protocol implementation time may be « too soon ». Performance gets frozen before the application is defined
  - the telephone system was designed for ONE application. Computer systems are general; we don't know the application
- Can we build software that scales?

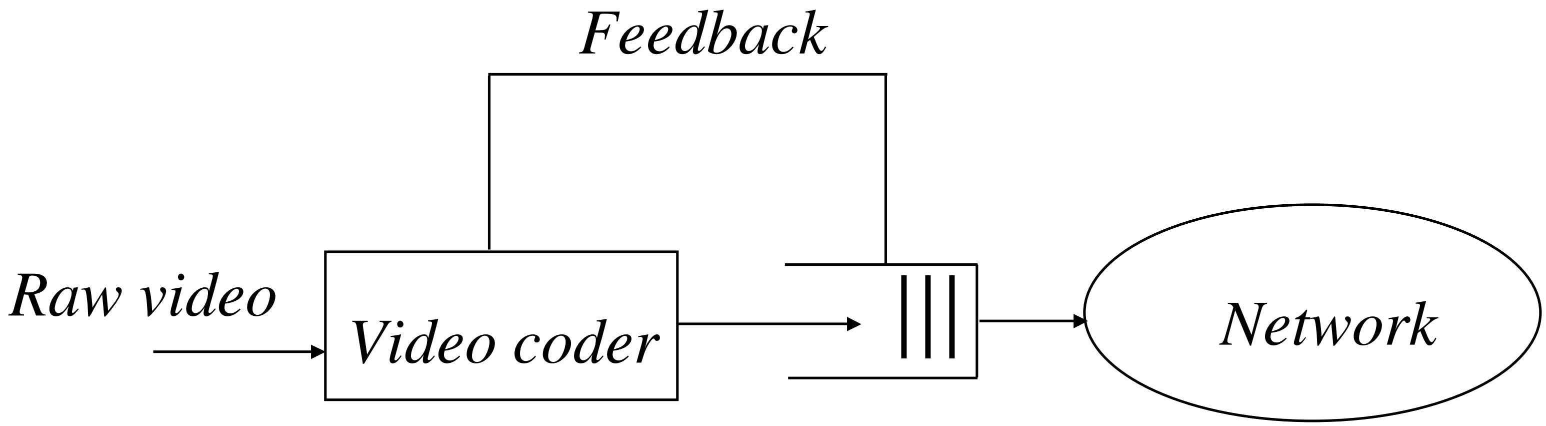
Adaptive applications

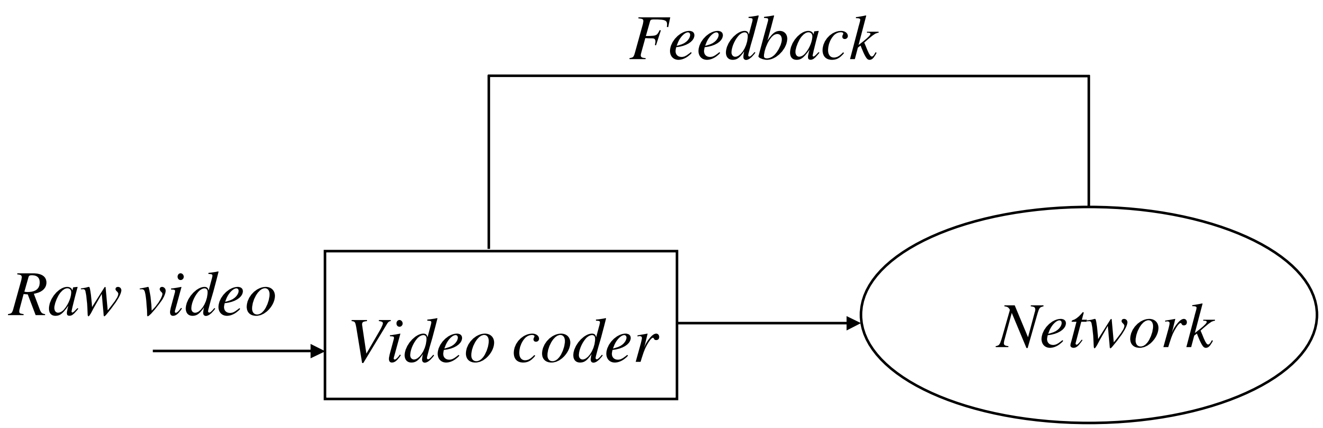
Adaptive applications

- Can network applications *adapt?*

- Should network applications adapt?

Video over the Internet





A/Video applications can adapt to bandwidth

- Video codec is controllable
  - Control the frame rate
  - Or the frame quality
    - quantization granularity
    - movement detection threshold

- What about audio?
  - Use of multiple codec
  - covers a « wide » range of adaptability

A/Video application can adapt to packet losses

- Add redundant information in packets
- Each packet may contain a sample of the previous packet
  - on a combination of the *k* previous packets
- Increases bandwidth!
- Redundancy should be added within a total fixed budget
  - how to best allocate the channel capacity

Applications « should » adapt

- Underlying networks are heterogeneous
- Bandwidth is not free
- No resource reservation and QOS routing supported
- Graceful degradation
- High fidelity
- Efficiency (optimal use of the available resource)
- Lessons
  - share performance
  - use a connection less packet switching network interface
  - a communication subsystem integrated within the application

Application Level Framing

A new communication architecture

- We need to reduce asynchrony
- more involve the application in transmission
- Analyze protocol functions
  - data manipulation functions
    - (copy, checksum, buffer allocation, data alignment, marshalling, byte swap, compression, encryption)
  - control functions
    - (sequence numbers, ACKs, window, etc…)
- Example of the transport:
  - Control: demultiplex, seq. numbers, update W, 10s of instructions
  - Data manipulation: copy and checksum
    - better performance if these two operations were *combined*

What is the bottleneck

- Heavy manipulation functions are the bottleneck
  - Presentation encoding/decoding, encryption
- How to best use the slowest part in the chain
  - remove transport level resequencing
  - let the application decide
  - need for autonomous « Application Data Units »

Application Level Framing

- An ADU is :
- Unit of error and flow control = Unit of transmission
  - avoid transmission inefficiencies in case of error
- Unit of processing = Unit of transmission
  - avoid idle waits
- Unit of processing = Unit of error and flow control
  - simplify adaptive multimedia applications design
- Can be processed as soon as it is received
  - augment the « message » structure
  - adequate size (ADU size discovery)

Conclusion

- ALF results in more complex application design
  - No code re-use
  - specific protocol mechanisms integrated within the application
- But, hopefully scalable
  - if carefully designed
- Research oriented audio and video conference applications are based on this architecture