Computer networks

Networking models, technology, issues and terminology

Network reference models

The ISO OSI reference model

The ARPA TCP/IP reference model

Terminology

Network physical organization

Network layer issues and techniques

Overview

- Networks are complex systems
  - complexity arises from:
    - need to apply sound software engineering principles (black boxes, formal specifications, standards, separation of implementation from protocol specifications, etc. etc.)
    - requirement to anticipate errors and unpredictable environments
    - requirement to handle diversely-configured hosts
- Consider two aspects (at opposite ends of the software design spectrum):
  - layered systems and formal network models
  - “nuts and bolts” of some of the problems and how to deal with them
- Also, introduce some terminology and technology concepts

The big picture – network models

- Networks must be designed and implemented carefully
- Often have conflicting goals
  - reliability vs throughput
- Various abstract models have been designed and implemented
- References:
  - Tanenbaum: Ch 1.3.1, 1.3.2, 1.4
  - Kurose & Ross: Ch 1.7

Basics

- Networks are organized as a series of layers or levels
  - each level is built on its predecessor
- Typically depicted as top-to-bottom stack of layers:
  - closer to the top is more abstract (user-oriented)
  - closer to the bottom is more “real”
- The number, name, content and functionality of each layer varies from network to network
  - each layer provides services to next-higher level, hiding details of how services are implemented
- Inter-machine communication occurs when layer *N* on a machine converses with layer *N* on another machine

…basics, 2

- The rules used in this conversation are known as the *Layer N protocol*
- The entities forming the corresponding layers on the different machines are called *peer processes*
- Peer processes communicate using their protocol
- Peer process communication is an abstraction: data is **not** actually transferred directly between peers
  - a layer *N* process passes data and control information to layer immediately below (*N-1*), until lowest level is reached
    - the uppermost layer is the application: its logic is just a protocol like any other
  - below the lowest level is the wire (fiber, radio-wave, carrier-pigeon, etc.)

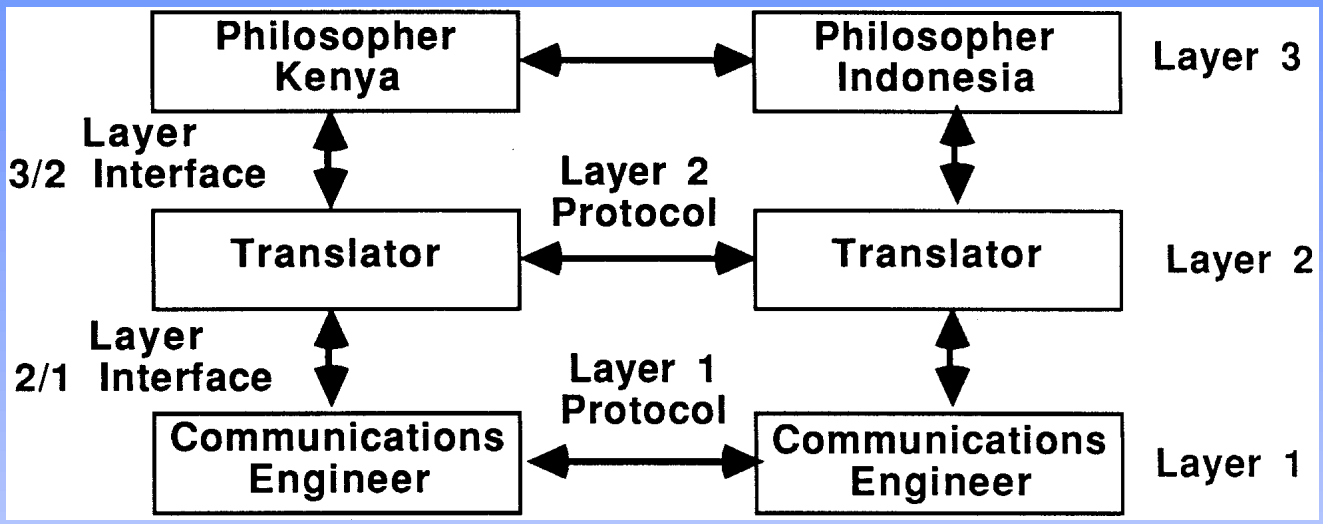
Layer-to-layer interfaces

- Layer *N* to layer *N-1* (adjacent layers) communication:
  - called the “LayerN to LayerN-1” interface specification
  - defines the services offered by layer *N-1* (layer *N* is a user of the services)
  - defines the primitive operations available in the lower layer
- Interface specifications must define interfaces and services clearly
  - services must be well-understood and specific
- Well-defined interfaces are critical to allow layer-by-layer interchangeability

Network architecture

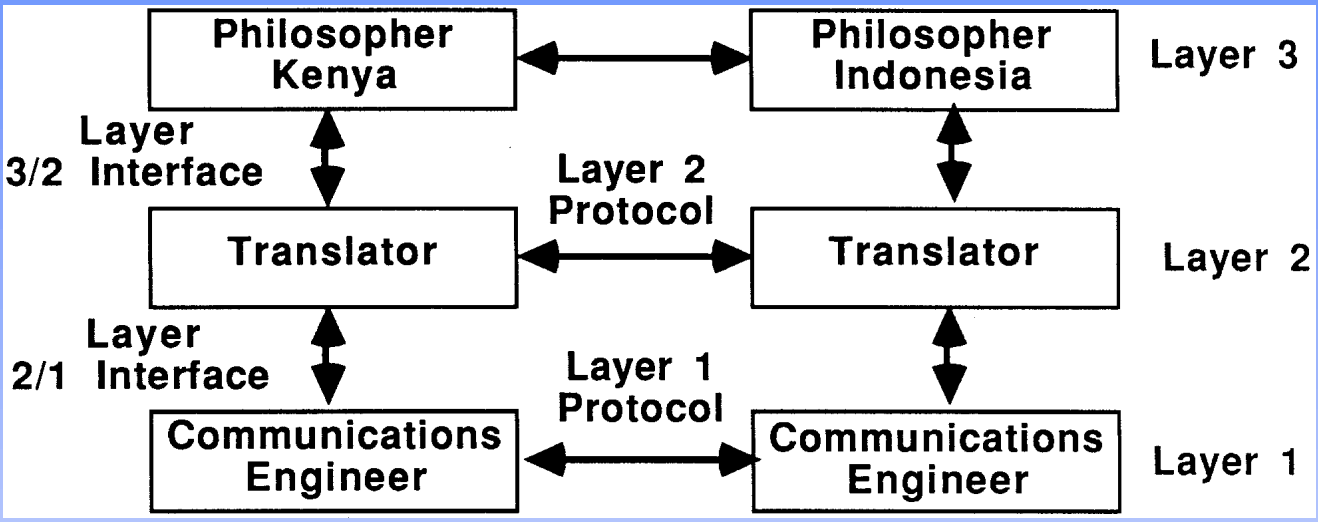
- The set of layers is called the *network architecture*
  - set of peer protocols
  - set of layer service definitions
- The architecture specification must be detailed enough to allow an implementor to write code and define interfaces
- Neither implementation details nor interface specification are part of the architecture

Another analogy



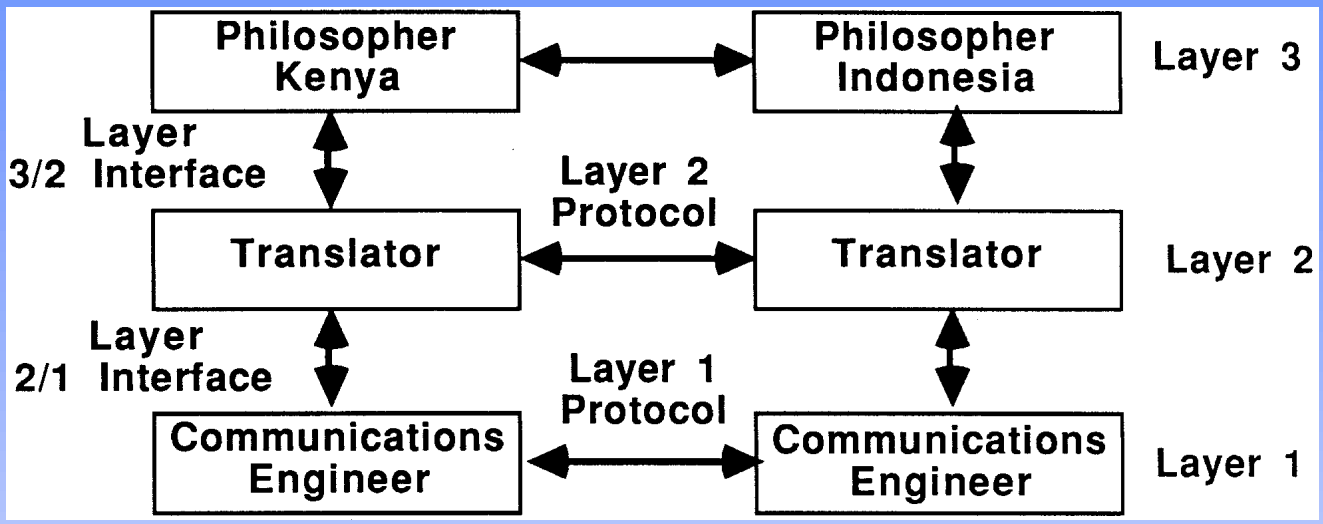
- (From Tanenbaum, p18)
- Consider two philosophers (peer processes on Layer 3) who wish to communicate
  - one in Kenya (P_K), the other in Indonesia (P_I)
  - they do not speak a common language, so each hires a translator (peer processes on Layer 2)
  - translators each contract with a communications engineer (peer processes on Layer 1)

continued...



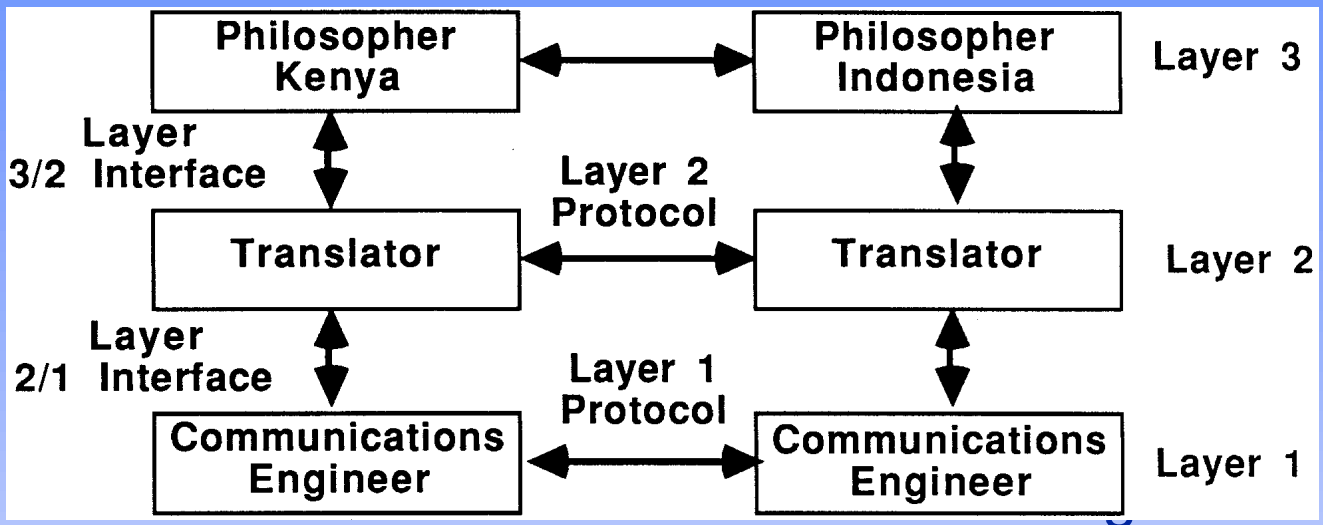
- P_K wishes to convey, in philosophical terms, affection for *oryctolagus cuniculus* to peer (P_l)
- P_K passes a message (in Swahili) across Layer Interface 3/2 to the translator
  - translator changes the message to "I like rabbits" or "J'aime des lapins" or "Me gustan los conejos", depending on Layer 2 protocol
  - both translators must agree in advance on what the "interchange language" will be

continued...



- Translator gives the message to the engineer (via the Layer 2/1 Interface) for transmission by telephone, telegram, letter or any other means
  - engineers must agree in advance on the transmission method (i.e. the Layer 1 protocol)
- When the message arrives in Indonesia, the engineer gives it to the Indonesian translator (via the Layer 1/2 Interface)

continued...



into Indonesian and passes the translated message (via the Layer 2/3 interface) to P_I

Observations

- Each protocol is independent of the others
- Translators could change the Layer 2 Protocol to Dutch or Italian
  - both peers must agree
  - no other protocol or interface need change
- Layer interfaces in Kenya do not have to be the same as those in Indonesia

- The peer abstraction is crucial for all network design
  - divide the task of network design into smaller, isolated, manageable design problems: i.e. the design of the layers

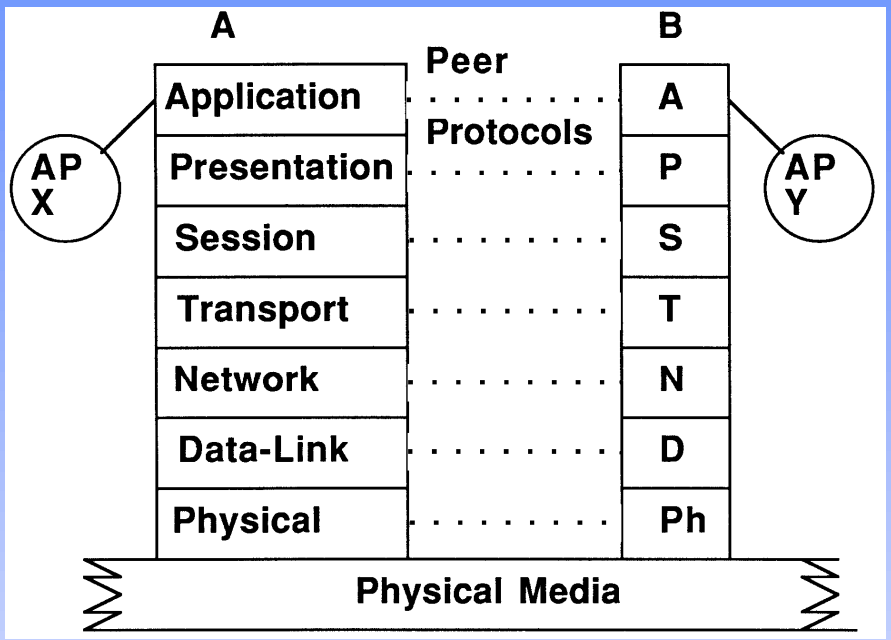
The ISO OSI reference model

- A layer model proposed by ISO
  - OSI: Open System Interconnect
  - generally called the OSI reference model
- Seven layers
  - generally, a layer represents a change in abstraction
  - each layer function is clearly defined
  - layer function should be conducive to protocol standardization (for existing standards or new ones)
  - layer function should minimize data-flow across layer interfaces

Layer definitions
(top-to-bottom)

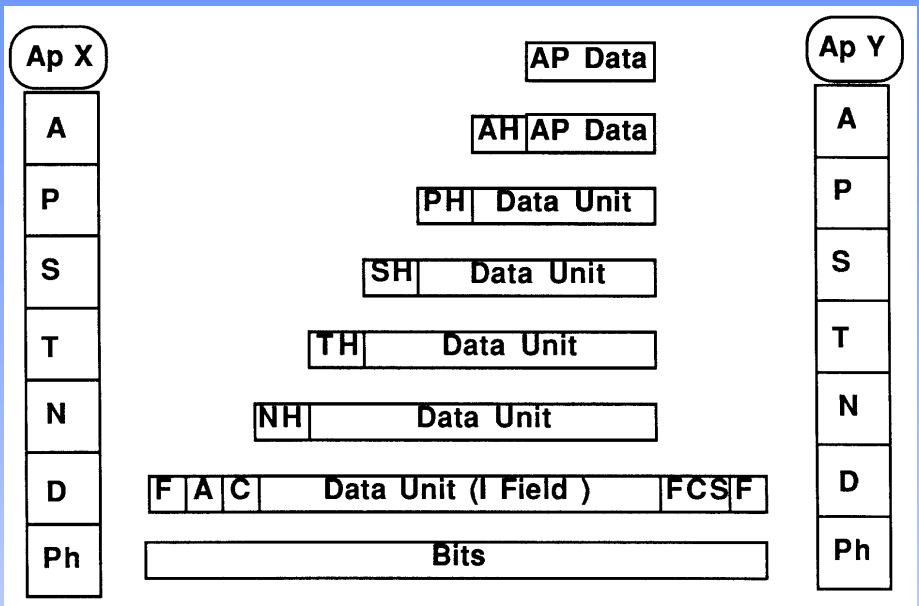
- **Application**: end-user tasks like e-mail, file transfer, remote login
- **Presentation**: character-set, language, formatting functions
- **Session**: for establish long-term connections between hosts
- **Transport**: error-free host-to-host communication; quality of service functions; multiplexing/aggregation
- **Network**: getting data through subnet (routing)
- **Data Link**: error-free frame delivery
- **Physical**: transmission of raw bits on a communications medium

continued...



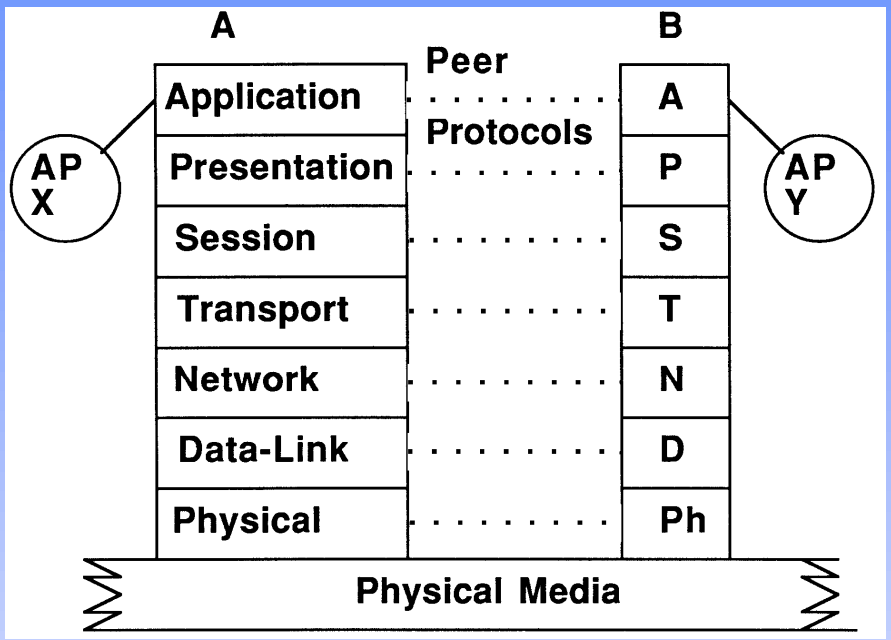
- Why seven layers?
  - enough to separate function, but not too much overhead
- The OSI reference model is not a complete network architecture:
  - no specification of service interfaces and layer protocols
  - ISO has produced and published separately a set of (independent) standards for each of the layers

Data transmission in the OSI model



- Sender process *X* sends to receiver *Y*
- *X*'s data given to Application Layer; application header added
- Augmented data given to Presentation Layer; presentation header added
- Augmented[2] data given to Session Layer
- Etc. until the physical layer, where bits are transmitted to *Y*
- Augmented[7] data percolates up through *Y*'s layers, removing headers along the way

Services and interfaces



- Active parts of layers are *entities*
  - software process
  - hardware (I/O chip, etc.)
- Entities in the same layer are peer entities: application entities, presentation entities, etc.
- Entities in layer N implement a service used by layer N+1
  - layer N is the service provider, layer N+1 is the service user

continued...

- Services are available at Service Access Points (SAPs)
  - every SAP is uniquely identified with an address
  - users and application processes use the SAPs at the top layer
  - layer *N* entities uses SAPs at layer *N-1*
- Adjacent layers must agree on the rules about using the interface

Encapsulation and fragmentation

- Datagrams are a *protocol data unit* (PDU); may need to be encapsulated by lower level PDUs
- Each level may add checksums, addresses and other information
- Each level can choose its PDU size, and can fragment the higher-level PDU
- Conversely, a level could decide to buffer many PDUs
- Each level must keep track of its own sequencing and error-checking information
- For example: a layer 3 datagram might be segmented and packed into layer 2 frames by the sender, and unpacked and reassembled by the receiver

The TCP/IP reference model

- Developed by DARPA (Defense Advanced Projects Research Agency) branch of U.S. DoD
- Practical (military) applications
  - diversely configurable
  - robust
- Layered organization analogous to OSI
- Four layers (top to bottom):
  - application
  - transport
  - internet
  - host-to-network (Tanenbaum); “link layer & LANs” (Kurose & Ross)
- Note that the reference model layer names are not protocol names

continued…

- The reference model layers have well-known protocols defined for them:
  - application layer:
    - ftp, smtp, telnet, pop, …
  - transport layer:
    - tcp, udp
  - internet layer:
    - ip
  - host-to-network:
    - ? – not defined by the reference model, use protocols like CSMA/CD, token-ring etc.

Commentary

- OSI was supposed to rule, TCP/IP was a stopgap until OSI fully deployed
  - didn't happen!
- OSI does a good job of specifying the reference model (although some layers seem thin, and others are overloaded)
  - defines explicit separation between services, interfaces and protocols
  - good software engineering principles
  - otherwise, unwieldy and inefficient?
- TCP/IP has good protocols, but the reference model was retrofitted and cannot realistically describe anything else
- See Tanenbaum Ch 1.4.3, 1.4.4 and 1.4.5

The little picture

Or, now that we've seen the big picture frameworks, look at some of the details of the issues and problems to be handled

Design issues for layers

- For all layers:
  - definition of peer protocol
  - addressing scheme (how to identify the peer)
  - conversation management (initialize, finalize)
  - definition of data unit:
    - packet, datagram, file, frame, etc
- For only some layers:
  - error detection/correction
  - message ordering
  - flow control
  - data fragmentation/aggregation between layers
  - multiplexing

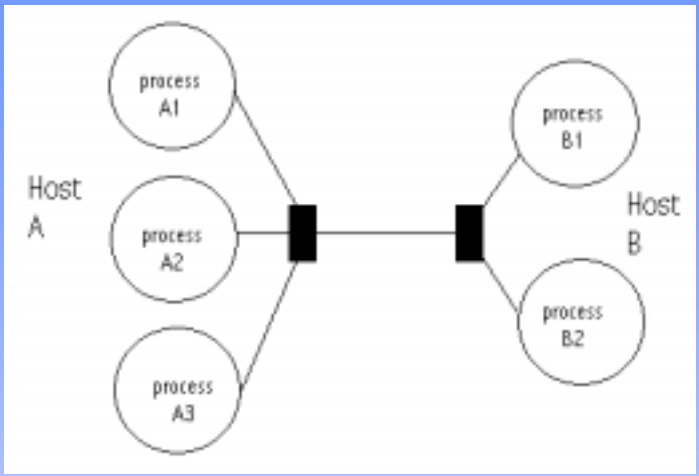
Terminology and technology definitions

- Analog vs digital information and signalling
  - modems vs codecs
  - carrier modulation vs pulse-coded modulation
- Multiplexing
- Baseband vs broadband
- Broadcasting vs point-to-point
- Full and half duplex
- Circuit-switched vs packet-switched again
- Datagrams vs virtual circuit
- Network physical organization & topology
- Routers, bridges and subnets
- References:
  - Tanenbaum: Ch 1.3.4, 2.4.5, 5.1.1
  - Kurose & Ross: Ch 1.1, 1.3.2, 1.4, 1.5.2, 5.5.1, 5.6.2

Analog vs digital

- Analog information: waveforms e.g. voice, audio tape, radio, video
- Digital information: binary-represented
- Analog signalling: encoding the information into an analog *carrier wave* by modulation (frequency, amplitude or phase)
  - digital information: use a modem to convert to analog
  - analog information:
    - voice telephone; AM and FM radio
- Digital signalling: represent binary information with low-high voltages (e.g. 0,+5 or ±5)
  - analog information: use a codec to convert to digital (pulse-coded modulation)
  - digital information: Manchester encoding

Multiplexing



- Sending multiple messages through a single communications channel
- Frequency-division multiplexing (FDM):
  - all A_i messages sent simultaneously
  - each A_i uses a different electrical / radio frequency (ie analog)
- Time-division multiplexing (TDM):
  - each process A_i has a turn using the full capacity of the path
  - path is divided into slots or equal duration; each process can use its slot or not (*synchronous* usage based on clocking)

...multiplexing, 2

- Statistical multiplexing (SMUX)
  - like TDM, based on taking turns to use entire capacity of path
  - not synchronous; processes “queue” messages at multiplexer
  - messages are sent according queue-processing rules (first-come-first serve, priority, etc.)
  - if messages arrive at multiplexer faster than sending rate, queue builds and throughput decreases, otherwise each process has perception of a private path
- FDM implies broadband, most applicable with analog signalling
- TDM takes turns using medium, useful for baseband
- SMUX is a form of TDM (irregular slot usage), same applicability

Baseband vs broadband

- Baseband:
  - digital signalling
  - signal uses small portion of available electromagnetic spectrum
- Broadband:
  - signalling using large part of available spectrum
    - historically referred to analog systems, also applicable to digital systems
  - many simultaneous channels (uses FDM to put each channel at different frequency)
  - cable TV, satellite
  - does ***not*** simply mean high-capacity, despite popular-press usage

Half-duplex vs. full-duplex

- *Duplex* refers to the capability of which hosts can transmit and when
  - *half-duplex* means that a host cannot send and receive at the same time
  - *full-duplex* means that a host can transmit which it is receiving (and v.v.)
- A full-duplex transmission is one in which both ends can be transmitting and receiving simultaneously
- In a half-duplex transmission, one host is receiving and the other is transmitting
  - the roles change, perhaps often
- Duplex can refer to physical media and also higher-layer communication
  - e.g. at the application layer, “who sends and receives when” is a significant part of the app. protocol

Broadcast vs point-to-point transmissions

- Broadcasting:
  - shared medium, not secure
  - data must be addressed
  - recipients must be listening for their data
  - multicasting: subgroups of recipients
- Point-to-point:
  - private connection
  - path must established before use

Circuit- vs packet-switched

- Circuit-switched:
  - a point-to-point path constructed from smaller circuits
  - circuit is dedicated, may be wasted
  - circuit must be established and disconnected
  - circuit can contain be any combination of (analog or digital) X (signals or data)
- Packed-switched:
  - digital only
    - items that are switched are PDUs: packets, frames, datagrams, …
  - data string is divided into packets
  - packets are sent individually
  - “store-and-forward”

Datagrams vs virtual circuits

- In a packet-switched network, different kinds of services may be available:
  - datagram service
  - virtual circuit
- Datagram service:
  - form of connectionless service
  - datagrams are independent units
  - datagrams are routed independently
  - arrival order not guaranteed
  - choice in quality of service:
    - reliable – datagrams are ACKed
      - higher cost, delays possible
    - unacknowledged service
      - lower cost, less protocol interaction

… datagrams vs virtual circuits, 2

- Virtual circuits (“VC”):
  - a connection-oriented service
  - application requests a route or fixed connection
    - similar to circuit establishment in circuit-switched networks
  - virtual circuit is identified by circuit number
    - subsequent datagrams are addressed via the circuit number
  - all datagrams follow same route
    - usually, network conditions
  - high-quality service – datagrams are guaranteed to arrive in order
  - circuit functionality distinct from raw datagram delivery (different layer)
  - bi-directional
  - PVC: permanent virtual circuit; SVC: switched virtual circuit

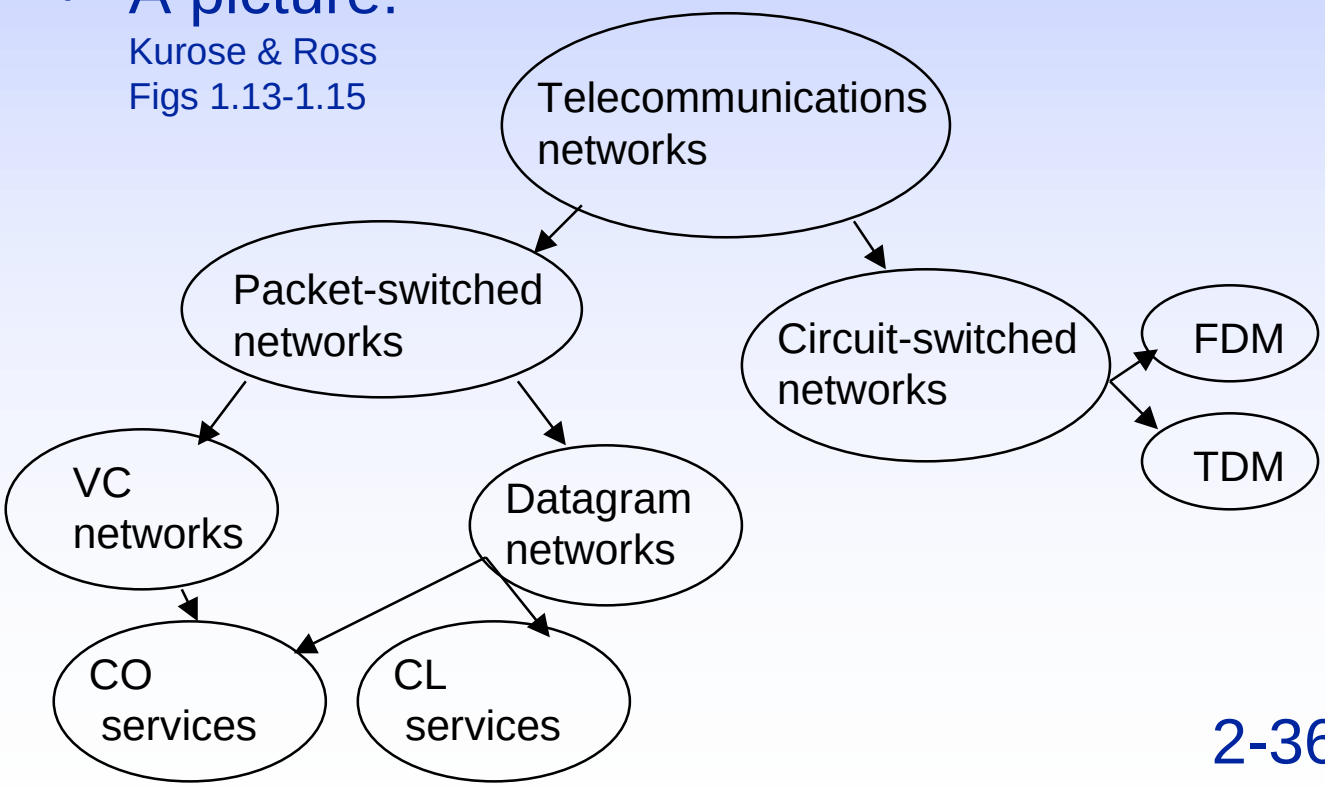
Connectionless vs. connection-oriented services

- Applications have two basics services available to implement application logic & protocols:
  - connect-oriented
  - connectionless
- Connection-oriented (“CO”):
  - requires initial protocol “handshake”
    - each end system is aware of the other
  - generally higher level of service
  - e.g. Internet CO is reliable, guaranteed order, flow control, congestion control
- Connectionless (“CL”):
  - no handshake, no awareness
  - generally lower service level

Circuit, packet, CO, CL, datagram, VC – ???

- Lots of terminology – somehow related
  - circuit-switching vs. packet-switching refers to basic network plumbing
  - CO Vs CL characterizes services available to applications
  - datagram vs. VC characterizes the way data moves through network
    - refer to network as “datagram network” or “VC network”
- A picture:

Kurose & Ross
Figs 1.13-1.15



Reliable vs. unreliable data transfer

- “Reliability” has specific meaning:
  - correct sequence (of datagrams)
  - based on acknowledgements and automatic retransmissions
  - most applications (presumably) need reliable transmission, the question is which layer(s) provide the reliability
  - virtual circuit networks are usually reliable
- “Unreliable” means:
  - datagrams might arrive, might not
  - if they arrive, might be in the wrong order
  - datagram networks are often unreliable

Network physical organization

- Historically, computer networking divided the network world into:
  - wide-area networks (WANs)
  - metropolitan-area networks (MANs)
  - local-area networks (LANs)
- These were differentiated by scale and connectivity technology:
  - all are packet-switching networks
  - generally, LANs use broadcasting and WANs use point-to-point
- LANs (>1Mbits/sec) using Ethernet™ or Token-ring, etc
  - used in geographically-restricted areas because of technology limitations
  - cheap
  - used in office / classroom / commercial environments, forming groups of related usage
  - before Ethernet/TokenRing, LANs were created using other technologies, including serial, parallel

…network physical organization, 2

- WANs based on continental backbones (>100Mbits/sec)
  - collections of point-to-point circuits
  - big players: telcos, satellite, microwave
  - used over geographically large areas to connect LANs
  - “fractional” services available from providers for private leasing
  - various commercial products defined
- MANs had been a curiosity (special-purpose hybrids) but are becoming more relevant:
  - availability of “dark fiber” from many sources
  - use of cable TV infrastructure
  - both can provide cost-effective LAN-like speeds (or better) over municipal-sized areas

…network physical organization, 3

- Terms like LAN, WAN are less important in the context of the Internet
  - the Internet is a multi-tier infrastructure of connections
  - its fundamental purpose is to connect smaller networks together; i.e. a network of networks
    - connectivity between the networks is the “Internet”
  - what is of interest is how to connect to that infrastructure
    - via LAN (e.g. corporate, education)
    - via dialup (standard POTS, ISDN or dedicated private links) via dedicated MAN-sized technologies like ADSL (Bell high-speed) or cable (@Home)
    - via wireless
    - etc.
- Kurose & Ross Ch 1.5.1 & 1.8

Network topologies

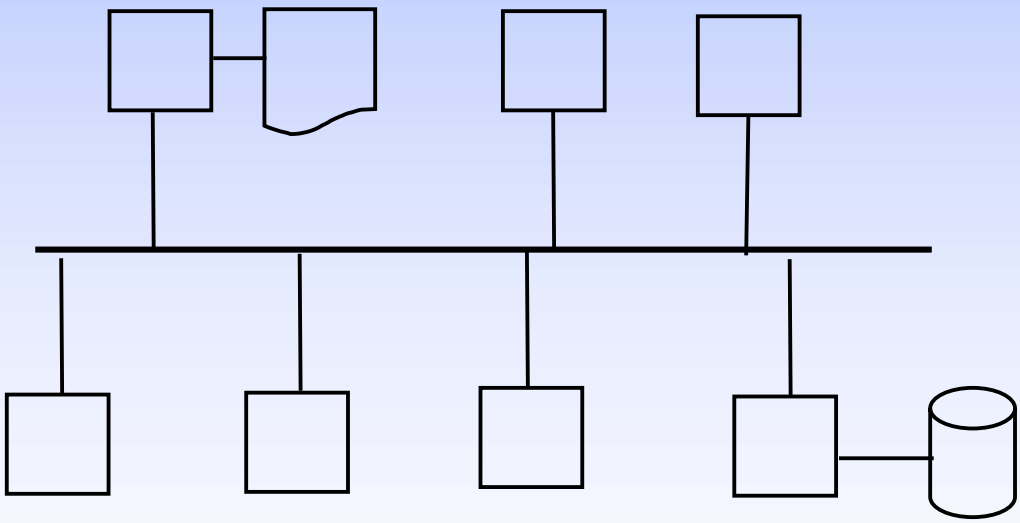
- The Internet and its constituent networks have specific topologies (organization)
- The continental infrastructure is formed from point-to-point links (optical fiber technology typically)
  - major links are constructed and operated by NSP (national service providers)
  - NSPs connect to each other at NAPs (network access points)
  - each connection is made according to performance and capacity requirements
  - every NAP does **not** connect to each other, hence the topology graph of the Internet is not complete
  - NSPs provide connections for regional and local ISPs in a multi-tier fashion
- Kurose & Ross Fig. 1.26

…network topologies, 2

- LANs are organized quite differently
- LANs attempt to provide a complete graph of connectivity
  - every host in a LAN is connected to every other host (conceptually a complete graph)
  - referred to as “shared medium”
  - physical topology is a LAN need not be the same as the conceptual connectivity topology
    - Ethernet LANs can be implemented with a star-shaped cabling system or a continuous loop cable or wireless
    - Token-ring LANs are implemented electrically as a sequence of point-to-point cables, but may be cabled as a star
- Topology of ISP connectivity varies
  - star for dialup, ADSL and other telco
  - shared for cableco (bus)

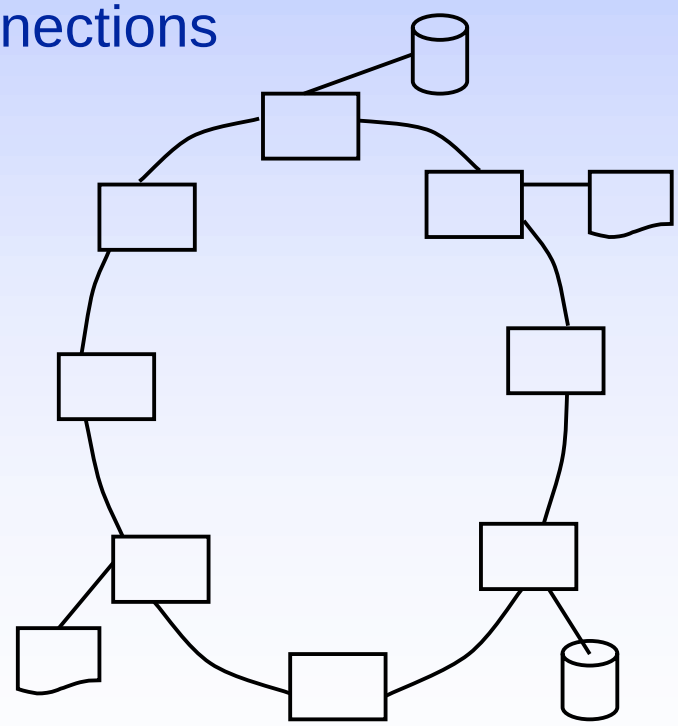
LAN topology: bus

- Analogous to antenna; any host can transmit at any time; all hosts receive all traffic
- Contention: hosts might transmit simultaneously
  - Contention resolution via CSMA/CD or Ethernet™



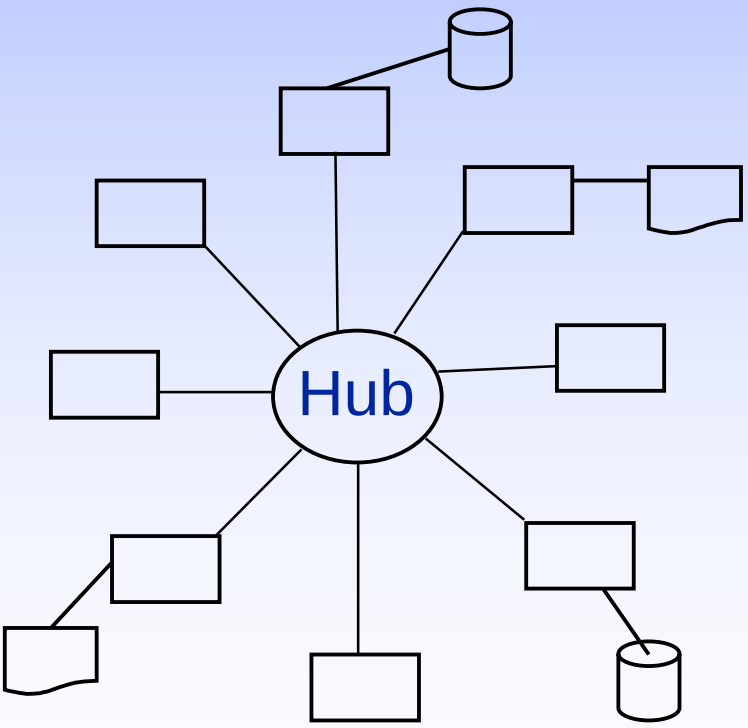
LAN topology: ring

- Only one host can transmit at any time
- Host must possess the “token” to transmit, hence “token ring”
- Token circulates continuously until a host takes possession to begin transmitting:
  - idle hosts must propagate the token
  - token flow is unidirectional over the sequence of host-to-host (point-to-point) connections



LAN topology: star

- Any station can transmit at any time
- Central hub or switch handles all traffic, contention
- Star might be just a wiring convenience for bus or ring, not a different conceptual topology
  - depends on the hardware in the middle



Routers, bridges and subnets

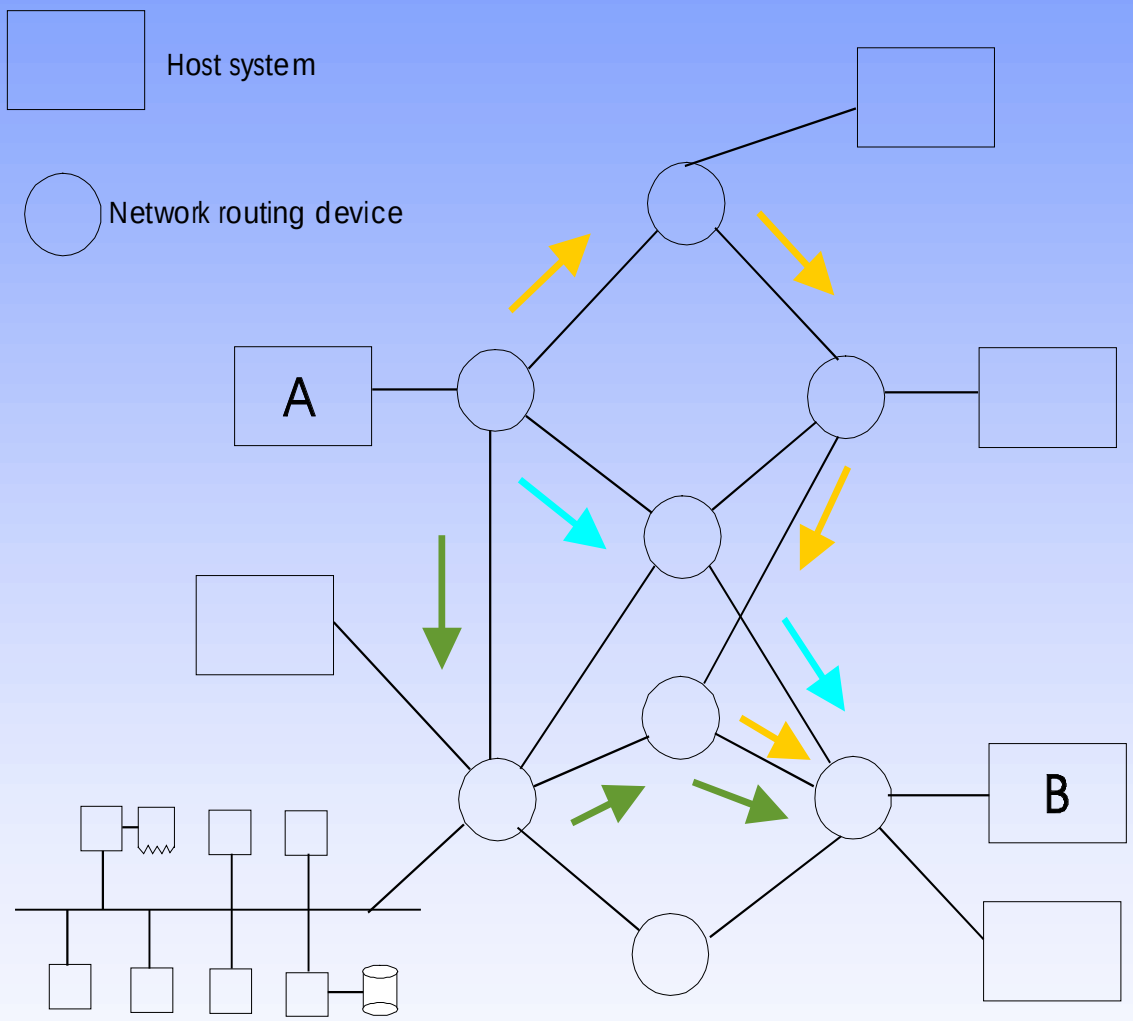
- Router: any computer or node with a primary function to facilitate data transmission.
  - connects LANS of possibly differing type; form WANs with point-to-point connections
  - also called: packet-switching node, data-switching exchange, intermediate system
- Bridge: connects LANs. LANs may be similar or differing types; may be physically close or remote
- Subnet: collection of routers and communications channels
  - a network that is connected to other networks (to form an internet)
  - any identifiable portion of a network
  - subnetting: the process of subdividing a network

Layer design – issues to be handled

- Computer networks and communications systems are complex
  - complexity largely due to error detection
  - techniques/algorithms exist to solve these problems
  - not all issues are solved in each layer

- References:
  - Kurose & Ross Ch 5.2, 3.4, 3.5.2, 1.6
  - Tanenbaum, Ch 1.3.3, Chapter 3

The problem



…the problem, 2

- How to send a large document to someone on another computer on the network, from A to B
- Assume that the receiver has secondary storage (disk) capable of storing the document
- Standard method: divide document into small units
  - units are usually equally-sized
  - units are packets or *datagrams*
    - datagrams is the preferred term in general discussions about application data
    - packet is usually reserved for a specific class of data structure

Why datagrams?

- Errors will not cause re-transmission of entire document (better *granularity*)
- Efficient use of medium:
  - sender may not be able to send continuously
  - dedicated use of medium for entire document may not be possible (or expensive)
  - datagrams from many senders (or sender's processes) can be multiplexed on a single medium

Transmission issues

- Acknowledgement:
  - how do we know a datagram has arrived at the destination?
- Quality and flow control:
  - do all datagrams need to be acknowledged?
  - how many datagrams can and should be sent (be in transit at the same time)?
  - what if the receiver is too slow or does not have enough memory?
  - how does the sender not overwhelm the receiver?

… transmission issues, 2

- Errors:
  - how do we detect and correct errors in datagrams?
- Making datagrams:
  - how do we divide the document into datagrams?
- Addressing:
  - how do the datagrams arrive at the correct destination?
  - (“correct destination” implies computer and application program on that computer)
- Routing:
  - how is the “correct” route chosen for a datagram?
  - is there a correct route?

… transmission issues, 3

- Re-assembly:
  - how do we re-assemble the document from the sequence of datagrams?
  - Note: datagrams may not (probably won't) arrive in order
- Decisions to be made:
  - how and when do all these decisions get made?
  - what is the address of a computer?
  - what is the datagram size?
  - what is the route through the network?
  - when is a datagram correct?
  - when has a datagram arrived?
  - …

Acknowledgement

- How do we know if a datagram has arrived?
  - receiving computer sends back a datagram indicating correct arrival. Called an acknowledgement (ACK)
  - ACK datagram contains sequence number of received datagram
  - receiving computer can also send a datagram indicating the received datagram has a content error – called a negative acknowledgement (NAK)

…acknowledgement, 2

- What happens if a datagram does not arrive?
  - receiving computer has no knowledge it was sent
  - sending computer waits a while for an ACK
  - if it does not arrive, sender re-sends the datagram
  - waiting time called a *time-out* or *time-out period*
- What if the first (NAKed) datagram then arrives?
  - duplicate message at receiver
- What if…
- What if...

… acknowledgement, 3

- Is an ACK always necessary?
  - an ACK adds to the cost of transmission
  - does every datagram need an ACK?
- For example:
  - electronic junk-mail
    - does the sender care if there are a few errors that are undetected?
    - probably not: want large coverage at low cost, not complete coverage at high cost
  - streaming media
    - does loss of a few datagrams have a material effect on the overall media presentation?
    - probably not: perceived as a tiny “glitch” in the content

Document size and datagrams

- Problem: how to divide a document into datagrams
- Example:
  - 100,000 character (byte) document; arbitrarily choose datagram size of 1000 bytes
  - assume 10,000 characters/sec transmission facility
  - one second for a signal to travel from the source to the destination (propagation delay)
- Observations:
  - document can leave sender in 100,000/10,000 = 10 sec
    - transmission delay or store & forward delay
  - entire document arrives at receiver 1 second after last character leaves sender

… document size and datagrams, 2

- Total transmission time for non-datagram document:
  - transmit time: 10 + 1 = 11 seconds
  - plus time for ACK: transit time + ACK time = 11 + 1 = 12 seconds
  - (assume ACK is so small that time to send is negligible)
- Datagram transmission:
  - 100,000 byte document / 1,000 bytes per datagram = 100 datagrams
  - transmit time for one datagram:
    1,000 B per dg / transmission speed + signal delay
    1/10 + 1 = 1.1 seconds
  - plus ACK time: 1.1 + 1 = 2.1 seconds
  - time for 100 datagrams: 2.1. x 100 = 210 seconds (!)

… document size and datagrams, 3

- Performance penalty is huge: factor of ~17
- Compromise solution: send *n* datagrams, then stop and wait for an ACK
- Example 1: send 10 datagrams before waiting for ACK
  - total transit time for 10 datagrams: 10 x (1,000/10,000) + 1 = 2 seconds
  - plus time for one ACK: 2 + 1 = 3 seconds
  - time for 10 groups of 10 datagrams: 10 x 3 = 30 seconds
- Example 2: 20-datagram groups
  - total transit time for 20 datagrams: 20 x (1,000/10,000) + 1 = 3 seconds
  - plus time for one ACK: 3 + 1 = 4 seconds
  - time for 5 groups of 20 datagrams: 5 x 4 = 20 seconds

Discussion

- In an error-free environment:
  - best case: 12 seconds
  - worst case: 210 seconds
- Datagram grouping has a significant effect
- What about datagram size?

Flow control

- Problem: the sending computer is sending a sequence of datagrams
- It is possible that the receiving computer will run out of buffer space
  - must be able to stop sending computer
  - must be able to restart sending computer
- Called “Flow Control”

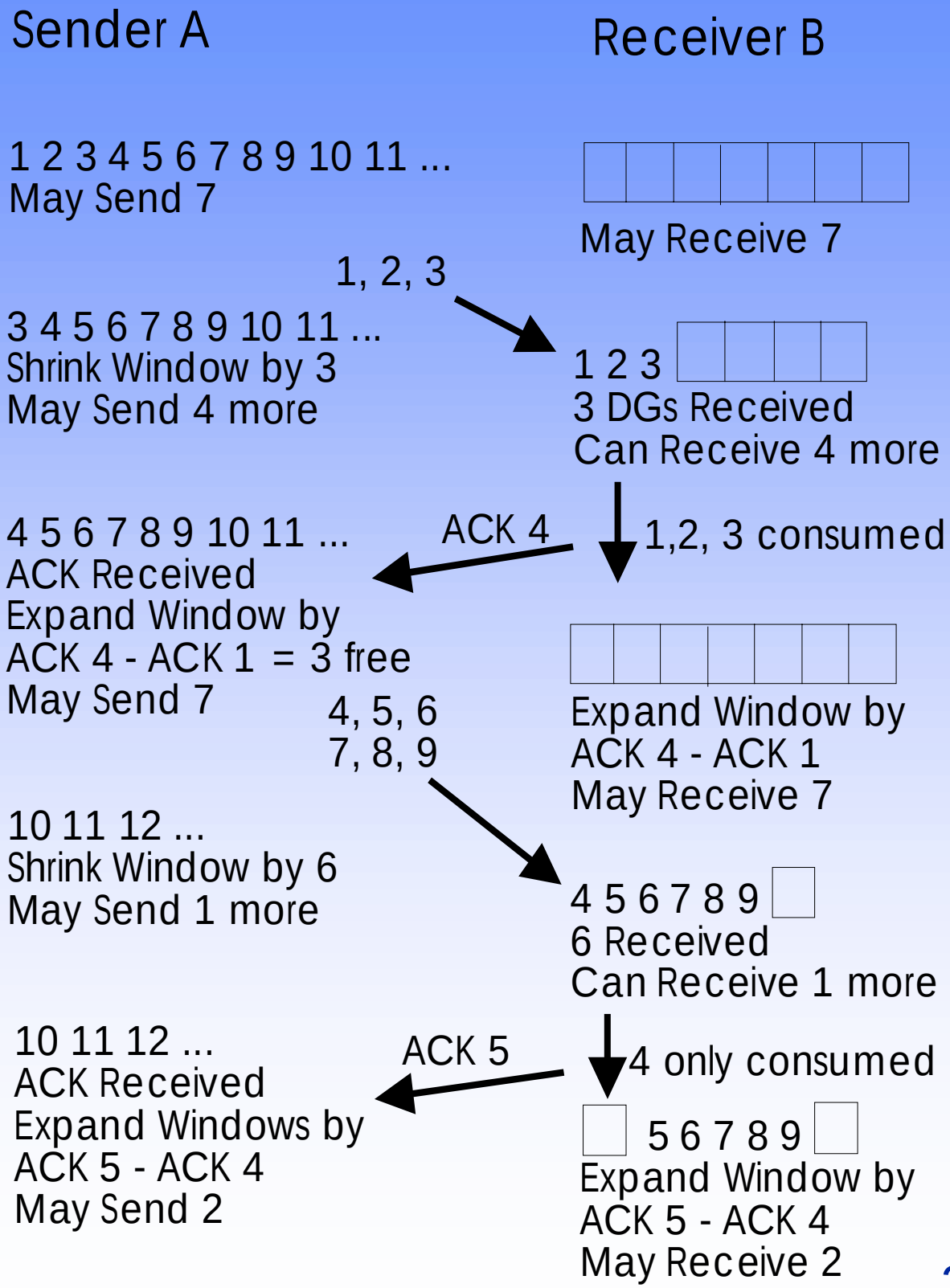
A flow-control protocol

- Number each datagram for the document sequentially from 1 to N
  - datagrams are labelled D:i (the i^{th} datagram)
- The receiving computer tells the sending computer the maximum number of datagrams that the receiver can handle – the so-called window size W
- The sending computer sends datagrams numbered from D:i up to D:i+(W-1)
  - eg for W=4 and i=1 initially, can send D:1 to D:4
- Assume that datagrams are delivered in order
- Receiver ACKs a sequence of datagrams by indicated the number of the next datagram expected in the sequence to maintain sequential order

… a flow-control protocol, 2

- If:
  - the last ACK was for D:i (or D:1 to start)
  - the last sender sequence stopped at D:j
  - an ACK:k (request for D:k) is received
- Then:
  - Transit=j-i+1 datagrams were in transit before the ACK:k, starting at D:i
  - Available=W-(j-i+1) slots were left over once all the in-transit datagrams arrived
  - Freed=k-i slots have now been freed (datagrams D:i to D:k-1 have been consumed)
  - there are (j-k+1) DGs still in transit (unconsumed); W-(j-k+1) datagrams can be put in transit now, starting at D:j+1
- Sending computer is implicitly controlled: if W frames are sent without acknowledgement, the sender must stop
- This protocol is the Sliding-Window Protocol or sliding-window flow-control

Sliding-window protocol



Errors

- Regardless of design, there will always be errors in a data-transmission system
- Any practical system must be able to detect errors
- Basic idea:
  - consider data to be transmitted as a sequence of bits, divided into *frames* (short sequences e.g. 8, 16, 32 or 128 bits)
    - frame: another term analogous to packet, datagram
  - add extra, redundant bits to frame to detect errors
  - extra bits are calculated as a function of the bits in the frame
- original bits plus extras are transmitted

continued...

- Receiver:
  - receives augmented frame and separates original frame from extra
  - repeats the extra-bit calculation and compares “received extra” with “calculated extra”
  - if same, frame is error-free
  - if error, notify for retransmission (NAK)
- Several standard methods:
  - parity check
  - longitudinal redundancy check (LRC)
  - cyclical redundancy check (CRC)

Parity check error detection

- Simple method for small amount of data
- Count number of one-bits; if odd, add one more one-bit to make even:

P_{even}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
1	0	1	1	0	0	1	0	0

P_{even}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
0	0	1	1	1	0	1	0	0

  - this is “even parity”

continued...

- Or, if the number of one-bits is even, add one more one-bit to make odd:

P_{odd}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
0	0	1	1	0	0	1	0	0

P_{odd}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
1	0	1	1	1	0	1	0	0

  - this is “odd parity”
- Parity-checking works only for single-bit errors; errors in two or more bits may not be detected
- In principle, could count zero-bits, too

Longitudinal redundancy check

- Also called “block-sum check”
- Extension of parity checks to blocks of data
- Will handle some two-bit errors
- Basic idea:
  - arrange a group of frames to form a rectangular block
  - add parity to rows, as usual
  - add a new row after the existing rows, composed of parity bits for the columns
- One bit will be ambiguous (parity of parity) – choose meaning arbitrarily

continued...

P_{odd}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	
1	0	0	0	0	0	0	0	0	
1	0	1	0	1	0	0	0	0	
0	1	0	0	0	1	1	0	0	
0	0	1	0	0	0	0	0	0	
1	0	1	0	1	1	0	1	0	
0	1	0	0	0	0	0	0	0	
1	1	1	0	0	0	1	1	0	
1	1	0	0	1	0	0	0	0	
									Parity
0	0	0	0	1	0	0	0	0	← row

- Rows are odd parity, columns are even parity (or vice versa)
- Bottom-left is ambiguous, arbitrarily choose row representation (odd)

Cyclic redundancy check (CRC)

- Given a message frame M that is *k* bits in length, add a checksum (error-detecting code) of length *n* bits to the end of the frame
- Formally called a Frame Check Sequence (FCS)
- The checksummed frame M' now has *k*+*n* bits (M' is 2^nM + FCS; + is modulo-2 (XOR))
- Basic idea: choose FCS such that M' is divisible (no remainder) by some divisor D (modulo-2 long division)
  - sender and receiver agree in advance on D
  - sender computes FCS and constructs M' and sends
  - receiver computes M' / D; remainder = 0 $\Rightarrow$ no error

continued...

- How is the FCS defined?
  - remainder($2^n M$ / D)
- How is D chosen?
  - FCS must be no more than n bits
  - length(D) < length(M)
  - FCS is a remainder, so choose D with n+1 bits and high-order bit 1
  - low-order bit 1
  - prime (?)
- Concatenate M and FCS to form M' : $2^n M$ - FCS
  - Assert: M' is divisible by D

Choices for CRC divisor D

- Sender and receiver must agree in advance
- Notation: D is a **generator polynomial** *G(x)*
  - 0 or 1 bits in D denote the presence or absence of terms in *G(x)*
  - Example:
    G(x) = $x^5 + x^4 + 1$ denotes D = 110001
- Some international standards for choices for *G(x)*:
- CRC-12 is $x^{12}+x^{11}+x^3+x^2+x^1+1$ (1100000001111)
- CRC-16 is $x^{16}+x^{15}+x^2+1$ (1100000000000101)
- CRC-CCITT is $x^{16}+x^{12}+x^5+1$ (1000100000100001)

CRC effectiveness

- Representing M, M' and D as generator polynomials allow detailed error analysis
  - Examples: probabilities of detecting single-, double-bit errors, burst errors, multiple scattered errors, etc.
  - (ref: Tanenbaum)
- Speed & space considerations:
  - calculations can be done with shift-register hardware
  - 1000-bit frame with CRC-16 incurs only 1.7% space overhead

continued...

Error type	CRC-16 Probability of detection	CRC-32 Probability of detection
single bit	1	1
two bits not necessarily together	1	1
odd number errors	1	1
burst error < CRC size	1	1
burst error = CRC size	$1 - 1/2^{15}$	$1 - 1/2^{31}$
burst error > CRC size	$1 - 1/2^{16}$	$1 - 1/2^{32}$

Error correction

- Many potential errors:
  - CRC/checksum failure
  - timeout
  - flow-control/sequencing (missing data)
- Recovery must be automatic: typically requires retransmission of some data
- Called ARQ mechanisms: *Automatic Repeat reQuest*
- “stop-and-wait” ARQ: for one datagram at a time flow-control
- “go-back-N” ARQ: for sliding-window flow-control

Names, addresses & ports

- Computers have names:
  - csg.uwaterloo.ca
  - www.microsoft.com
- Names only denote/identify the computer: they say nothing about where the computer is, or how to get to it
- Naming scheme governed by central authorities

continued...

- Networks have addresses; computers connected to the network have addresses
- The address defines the location
- Addresses are typically numeric
  - E.g. internet (tcp/ip): 129.97.208.19
    - 129.97 is the network address
    - 208.19 is the computer address on that network
- Networks know how to reach other networks; an individual network knows how to reach its computers
- Will require name-to-address mapping (to be discussed later)

continued...

- Programs are also addressable
- Program addresses are *ports* or *sockets* or *service access points* (SAPs)
- Every SAP has a unique address on a computer
- SAPs are the endpoint of all services
  - all communications in both directions is done via a SAP
  - a datagram arrives at the computer and is sent to a SAP, where a program can process it
  - programs send a datagram through a SAP to another computer

Datagram contents

- Messages (documents) for transfer are divided into fixed-size units: datagrams, packets, frames, segments, …
  - "official" generic term: Protocol Data Unit (PDU)
- PDUs must contain all necessary protocol information:
  - addresses: source and destination
  - addresses: computer, SAP
  - sequence number
  - error detection/correction (checksums)

Example PDU

Source port #	Destination port #
Sequence number	
Acknowledgment number	
Header length & flags	Receiver window size
Checksum	Urgent pointer
Options (optional, variable size)	
Data segment (variable size)	

- TCP
- In this example, datagrams are program-to-program, hence port addresses are used