ECSE-4670: Computer Communication Networks (CCN)

Introduction

Shivkumar Kalyanaraman
shivkuma@ecse.rpi.edu

Biplab Sikdar
sikdab@rpi.edu

Rensselaer RSVP

Adapted in part from S.Keshav (Cornell), Peterson (Uarizona)

- Syllabus, administrativia
- Networking: An Overview of Ideas and Issues

Who's Who

- Instructors:
  - Shiv Kalyanaraman; kalyas ; x8979
  - Biplab Sikdar; sikdab ; x6664
- Course secretary: (on-campus)
  - Jeanne Denue-Grady: JEC 6049 ; x6313
- PDE/RSVP Point-of-contact:
  - Kari Lewick; CII 4011; x2347
- TAs:
  - G.Liu, H. Yang, Y. Pei (PDE), S. Raghunath (PDE)

Web Resources

- WebCT Course Web Site:
  - http://webct.rpi.edu
  - (backup) http://www.ecse.rpi.edu/Homepages/shivkuma/teaching/fall2001/index.html
- WebCT: bulletin board, video streams, homework drop-box etc
- Text book Web Site: http://www.awl.com/kurose-ross

Course Description Highlights

- Syllabus:
  - Networking layers: application, transport, network, link
    - Issues: application models, multiplexing, reliability, flow/congestion control, error detection/correction, multiple access etc
  - Network Modeling: Elementary probability, queuing theory, analysis of a router queue, network of queues, LAN performance

Course Description Highlights (Continued)

- Lectures
- Informal quizzes: Every two weeks
- WebCT bulletin board: Post your questions! TAs monitor it daily.
- WebCT: Grades, papers, RFCs, Internet drafts…
- 2 Labs: Transport/Network layers {20 pts}
- 6 Homeworks: {30 pts}
- 3 exams: 15 pts, 15 pts, 20 pts: {50pts}

Prerequisites

- Background in elementary probability
  - *Probability for Engineering Applications, ECSE-4500, Discrete structures, CSCI-4320,* or *Modeling and Analysis of Uncertainity, ENGR-2600*
- Knowledge of basic computer organization
  - *ECSE-2660 Computer Architecture, Networks and Operating Systems or CSCI-2500 Computer Organization*
- C programming knowledge
- If you do not have the required prerequisites, you must drop the course and take it later (next year).

Still trying to get into the course ?

- Do you have the pre-requisites ?
- Please submit course add form to course secretary: Jeanne, JEC 6049 by tomorrow (Wed, Aug 29th), noon time (12 pm).
- Depending upon the number of people who drop the class, space available, TA resources available, we will add more students.
  - Decisions to be emailed to you by Jeanne.
  - Make sure you mention your email address to her.

Answers to FAQ

- All homeworks & labs due at the beginning of the class indicated on the course calendar
  - Up to one late submission: no penalty
  - Beyond that 10% penalty: only if submitted before solutions are posted.
- Exams are open-book and extremely time limited.
- Exams consist of design qns, numerical, true-false, and short answer questions.

Answers to FAQ

- Focus will be on conceptual understanding, and problem-solving skill.
- Labs are based upon the programming assignments suggested in chap 3 and 4 of the textbook
- Informal quizzes will be given for your benefit once in 2-3 weeks to recap/test recently covered material and reading assignments. No grading.

Information, Computers, Networks

- Information: anything that is represented in ***bits***
  - ***Form*** (can be represented as bits) vs
  - ***Substance*** (cannot be represented as bits)
- Properties:
  - Infinitely replicable
  - Computers can *"manipulate"* information
  - Networks create *"access"* to information

Networks

- Potential of networking:
  - move bits *everywhere*, *cheaply*, and with desired *performance characteristics*
  - *Break the space barrier for information*
- Network provides "connectivity"

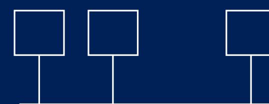
What is "*Connectivity*" ?

- Direct or indirect access to every other node in the network
- Connectivity is the magic needed to communicate if you do not have a direct pt-pt physical link.
  - *Tradeoff: Performance characteristics worse than true physical link!*

Connectivity.

- Building Blocks
  - links: coax cable, optical fiber...
  - nodes: general-purpose workstations...
- *Direct* connectivity:

  - point-to-point
  - multiple access

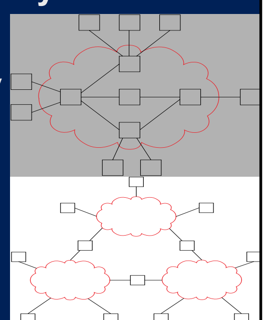


Connectivity..

- *Indirect* Connectivity
  - switched networks

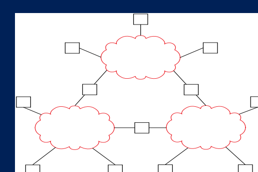
    => *switches*
  - inter-networks

    => *routers*



Connectivity …

- Internet:
  - *Best-effort*
    (no performance guarantees)
  - *Packet-by-packet*



- A pt-pt physical link:
  - *Always-connected*
  - *Fixed bandwidth*
  - *Fixed delay*
  - *Zero-jitter*

Point-to-Point Connectivity

- Physical layer: coding, modulation etc
- Link layer needed if the:
  - link is shared between apps (framing, medium access control, multiplexing)
  - link is unreliable (reliability)
  - link is used sporadically and traffic can flood receivers (flow control)
- *No need for protocol concepts* like addressing, names, routers, hubs, forwarding, filtering …

Connecting N users: *Directly* ...

- *Bus*: broadcast, collisions, media access control
- *Full mesh*: Cost vs simplicity

Bus

Full mesh

- Address concept needed if we want the receiver *alone* to consume the packet!

List of Problems (so far)

- Topologies
- Framing
- Error control
- Flow control
- Multiple access
  - How to share a wire

How to build Scalable Networks?

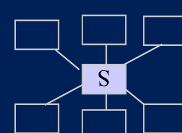
- Scaling: system allows the increase of a key parameter. Eg: let N increase…
  - *Inefficiency limits scaling* …
- Direct connectivity is *inefficient* & hence does not scale
  - Mesh: *inefficient* in terms of # of links
  - Bus architecture: 1 expensive link, N cheap links. *Inefficient* in bandwidth use

Filtering, forwarding …

- Filtering: choose a subset of elements from a set
  - *Filtering is the key to efficiency & scaling*
- Forwarding: actually sending packets to a filtered subset of link/node(s)
  - Packet sent to one link/node => efficient
- Solution: Build nodes which filter/forward and connect indirectly => "switches" & "routers"

Connecting N users: *Indirectly*

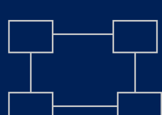
- Star: One-hop path to any node, reliability, forwarding function
- "Switch" S can filter and forward!
  - Switch may forward multiple pkts in parallel for additional efficiency!



Star

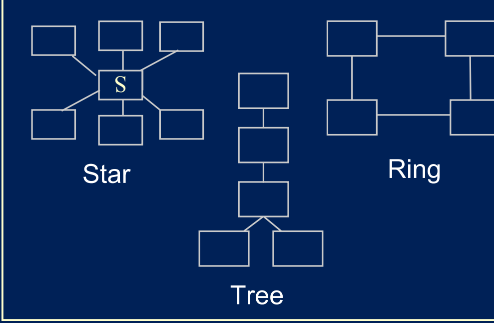
Connecting N users: *Indirectly* …

- Ring: Reliability to link failure, near-minimal links
- All nodes do "forwarding" and "filtering"

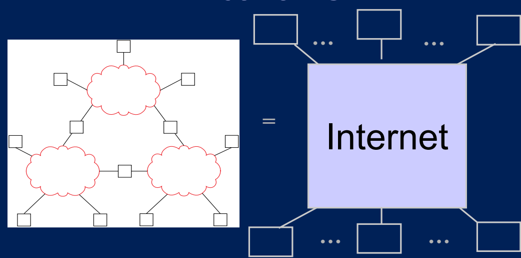


Ring

Topologies: Indirect Connectivity



Inter-Networks: *Networks of Networks*



Our goal is to design this black box on the right

Inter-Networks: Networks of Networks

- Internetworking involves two fundamental problems: *heterogeneity and scale*
- Concepts:
  - Translation, overlays, address & name resolution, fragmentation: to handle *heterogeneity*
  - Hierarchical addressing, routing, naming, address allocation, congestion control: to handle *scaling*
- Covered in more detail in "Internet Protocols" course

Additions to Problem List

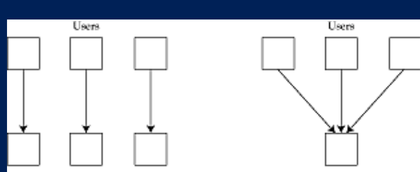
- Fragmentation
- Switching, bridging, routing
- Naming, addressing
- Congestion control, traffic management
- Reliability

How to do system design ?

- Eg goal: Design an Inter-network…
- *Resources:*
  - Space
  - Time
  - Computation
  - Money
  - Labor
- Design: *tradeoff cheaper resources against expensive ones to meet goals.*

Building blocks: *Multiplexing*

- Multiplexing = sharing
  - Trades time and space for money
  - *Cost*: waiting time, buffer space & packet loss
  - *Gain*: Money => Overall system costs less

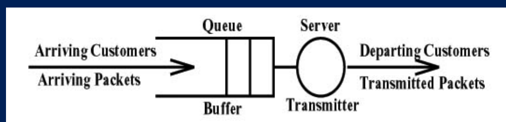


Statistical Multiplexing

- Reduce resource requirements by *exploiting statistical knowledge* of the system.
  - Eg: average rate <= service rate <= peak rate
  - If service rate < average rate, then system becomes unstable!!
    - First design to ensure system stability!!
  - Then, for a stable multiplexed system:
    - *Gain* = peak rate/service rate.
    - *Cost*: buffering, queuing delays, losses.

Stability of a Multiplexed System

Average Input Rate > Average Output Rate
=> system is unstable!



How to ensure stability ?

1. Reserve enough capacity so that demand is less than reserved capacity
2. Dynamically detect overload and adapt either the demand or capacity to resolve overload

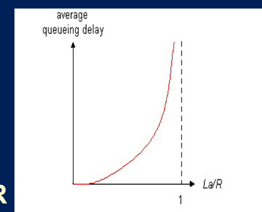
Rensselaer Polytechnic Institute 31
Shivkumar Kalvanaraman, Biplab Sikdar

What's a performance *tradeoff* ?

- A situation where you cannot get something for nothing!
- Also known as a zero-sum game.

- R=link bandwidth (bps)
- L=packet length (bits)
- a=average packet arrival rate

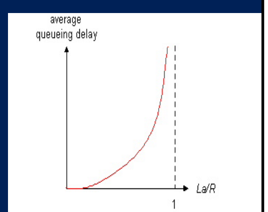
Traffic intensity = La/R



Rensselaer Polytechnic Institute 32
Shivkumar Kalvanaraman, Biplab Sikdar

What's a performance *tradeoff* ?

- La/R ~ 0: average queuing delay small
- La/R -> 1: delays become large
- La/R > 1: average delay infinite (*service degrades unboundedly => instability*)!

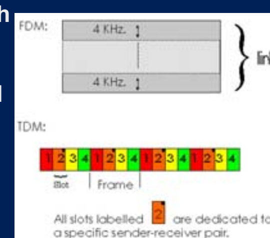


Rensselaer Polytechnic Institute 33
Shivkumar Kalvanaraman, Biplab Sikdar

Example Design: *Circuit-Switching*

Circuit-switching: A form of multiplexing

- Divide link bandwidth into "pieces"
- Reserve pieces on successive links and tie them together to form a "*circuit*"
- Map traffic into the reserved circuits
- Resources wasted if unused: *expensive*.
- Mapping can be done without "headers".
- Everything inferred from timing.



Rensselaer Polytechnic Institute 34
Shivkumar Kalvanaraman, Biplab Sikdar

Example Design: *Packet-Switching*

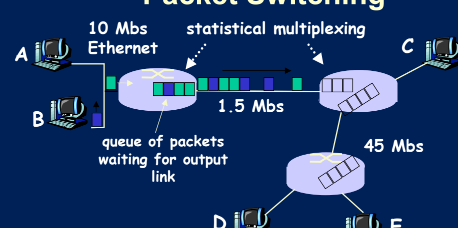
Packet-switching: Another form of multiplexing:

- Chop up data (not links!) into "packets"
  - Packets: data + meta-data (header)
- "Switch" packets at intermediate nodes
  - *Store-and-forward* if bandwidth is not immediately available.

Bandwidth division into "pieces"
Dedicated allocation
Resource reservation

Rensselaer Polytechnic Institute 35
Shivkumar Kalvanaraman, Biplab Sikdar

Packet Switching



- Cost: self-descriptive header per-packet, buffering and delays for applications.
- Need to either reserve resources or dynamically detect/adapt to overload for stability

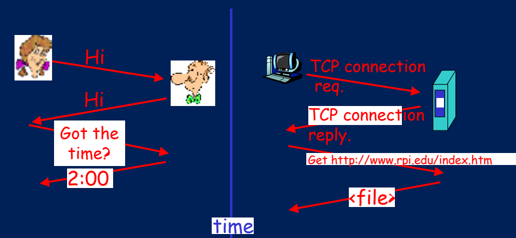
Rensselaer Polytechnic Institute 36
Shivkumar Kalvanaraman, Biplab Sikdar

Summary of System Design Ideas

- Multiplexing
- Statistical Multiplexing
- Stability and performance tradeoffs
- Circuit switching
- Packet switching

What are protocols ?

- Networking software is organized as ***protocols***
- Eg: Human protocol vs network protocol:



Analogy: Organization of air travel

ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing

airplane routing

- Protocols: a series of functions performed at different locations

Organization of air travel: *a different view*

ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing

airplane routing

interface

Layers: each layer implements a ***service***

- via its own internal-layer actions
- relying on services provided by layer below

Layered air travel: *services*

Counter-to-counter delivery of person+bags
baggage-claim-to-baggage-claim delivery
people transfer: loading gate to arrival gate
runway-to-runway delivery of plane
airplane routing from source to destination

Similarly, we organize network protocols into a bunch of layers!

Distributed implementation of layers

Departing airport	arriving airport
ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing

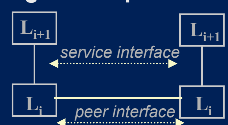
intermediate air traffic sites

airplane routing | airplane routing

airplane routing

Protocol Implementations

- Are building blocks of a network architecture
- Each protocol object has two different interfaces
  - *service interface*: defines operations on this protocol
  - *peer-to-peer interface*: defines messages exchanged with peer



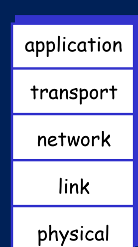
Reference Models for Layering

TCP/IP Model	TCP/IP Protocols	OSI Ref Model
Application	FTP, Telnet, HTTP	Application, Presentation, Session
Transport	TCP, UDP	Transport
Internetwork	IP	Network
Host to Network	Ether net, Packe Radio, Point-to-Point	Datalink, Physical

"Top-down" approach means we will first learn the application layer and then learn about lower layers

Internet protocol stack

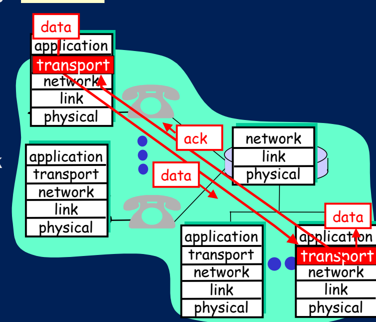
- **application**: supporting network applications
  - ftp, smtp, http
- **transport**: host-host data transfer
  - tcp, udp
- **network**: routing of datagrams from source to destination
  - ip, routing protocols
- **link**: data transfer between neighboring network elements
  - ppp, ethernet
- **physical**: bits "on the wire"



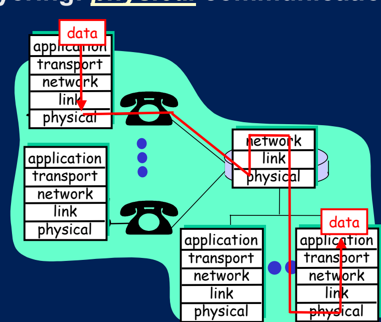
Layering: *logical* communication

E.g.: transport

- take data from app
- add addressing, reliability check info to form "datagram"
- send datagram to peer
- wait for peer to ack receipt
- analogy: post office



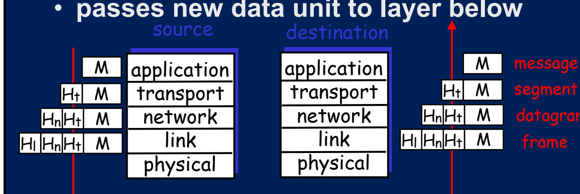
Layering: *physical* communication



Protocol layering and data

Each layer takes data from above

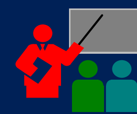
- adds header information to create new data unit ("encapsulation")
- passes new data unit to layer below



Design Perspectives

- *Network users:* **services that their *applications need*, e.g., guarantee that each message it sends will be delivered without error within a certain amount of time**
- *Network designers:* ***cost-effective design* e.g., that network resources are efficiently utilized and fairly allocated to different users**
- *Network providers:* **system that is easy to *administer and manage* e.g., that faults can be easily isolated and it is easy to account for usage**

Summary



- **Administratrivia**
- **Networks, connectivity, topologies …**
- **Pot Pourri of networking concepts and problems to be explored in this course ...**