Welcome

to

Performance Modeling and Simulation
DVA D05
Johan Garcia

Today's Agenda

Before Lunch
- Course Introduction and Overview
- Networking refresher

After Lunch
- Performance Evaluation overview
- Metrics
- Simple statistics (time permitting)

Course Goals

- Apply appropriate statistical techniques for performance evaluation
- Knowledge of basic experimental design
- Know and apply basic queueing theory
- Know and apply basic TCP modelling
- Perform and analyze ns-2 simulations
- Perform and analyze emulation experiments
- Practically oriented course, many labs
- Understand the trade-offs involved in using analytical modeling, simulation and emulation.

Course Approach

- Dual focus
  - General perfromance evaluation
  - Network performance evaluation
- Breadth, not depth
- Theory and practice
- Masters course => student participation and interaction expected

Course Overview

- Four major parts:
  - Performance evaluaiton essentials
  - Analytical perfromance evaluation
  - Simulation-based performance evaluation
  - Emulation-based performance evaluation
- Indepent project work

Overview / Essentials

Lecture 1: Course Introduction
Repetition of basic prerequisites in computer networking.

Lecture 2: Performance evaluation basics
Performance Metrics
Analysis vs. simulation vs. emulation vs. live experiments
Basic statistics

Lecture 3: Statistics
Error models, student-t etc

Lecture 4: Experimental Evaluation and Design
Anova, two factor experiments

Laboratory Exercise 1: Octave
Octave is a free Matlab-like program. In this laboratory excersise we will familiarize ourselves with octave, and how to do statistical analysis using Octave/Matlab.

Overview / Analytical

Lecture 5: Queuing theory introduction
Terminology and basics.

Lecture 6: M/M/c systems

Lecture 7: TCP analytical performance evaluation

Laboratory Exercise 2
Queueing systems using octave

Laboratory Exercise 3
TCP modelling using octave

Overview / Simulation

Lecture 8: Simulation introduction
Terminology and basics.

Lecture 9: ns-2 simulation

Lecture 10: ns-2 continued

Laboratory Exercise 4
ns-2 Introduction, TCP analysis

Laboratory Exercise 5
ns-2 simulation of a more complex scenario

Overview / Emulation

Lecture 11: Emulation introduction
Terminology and basics. How to setup and measure.

Lecture 12: KauNet Emulation

Lecture 13: KauNet Emulation

Laboratory Exercise 6
KauNet Introduction, TCP analysis

Laboratory Exercise 7
KauNet emulation of a more complex scenario

Independent project work

- Corresponding to at least 60 hours per person
- Groups of 1, 2 or 3 student allowed
- Theoretical work (ca 10 pg)
- Practical work (ca 3 pg)
- Possible to combine with Wireless course
- Course grading will be done on this work
- START EARLY

REQUIREMENTS TO PASS

- Pass grade on both examinations
- Pass grade on project work

Two written test will take place

These tests will cover material from the book, the lectures and the laboratory exercises

Laboratory Exercises

- The exercises will be done in SMART-lab
- However, if you have a fast laptop it is suggested you use that instead of the machines in SMART
- Min 512kb / 1.5 GHz / 13 Gb free space
- Two vmware images. Use VMware player

Simple networking refresher

Slides adapted from
Computer Networking: A Top Down Approach,
Jim Kurose, Keith Ross

Protocol Layering

Layers

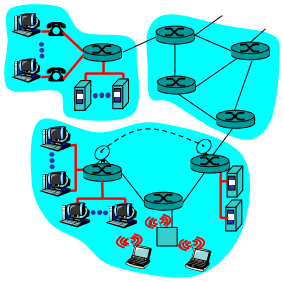
Application

Transport

Network

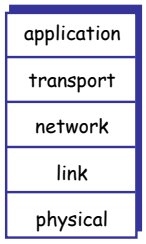
Link

Physical

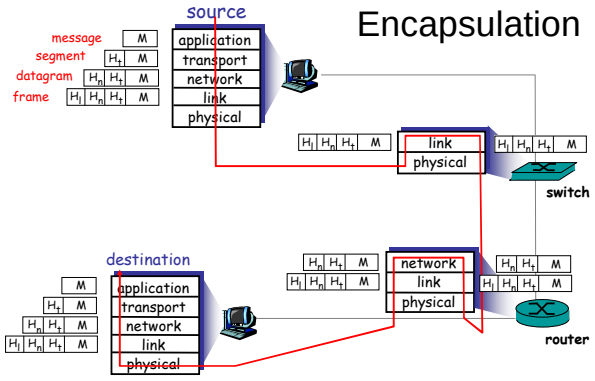


Internet protocol stack

- application: supporting network applications
  - FTP, SMTP, STTP
- 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"

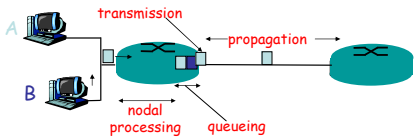


Encapsulation



Four sources of packet delay

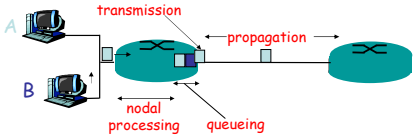
- 1. nodal processing:
  - check bit errors
  - determine output link
- 2. queueing
  - time waiting at output link for transmission
  - depends on congestion level of router



Delay in packet-switched networks

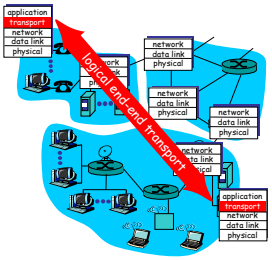
3. Transmission delay:
- R=link bandwidth (bps)
- L=packet length (bits)
- time to send bits into link = L/R

4. Propagation delay:
- d = length of physical link
- s = propagation speed in medium (~$2x10^8$ m/sec)
- propagation delay = d/s



Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Internet transport protocols services

TCP service:

- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum bandwidth guarantees

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Q: why bother? Why is there a UDP?

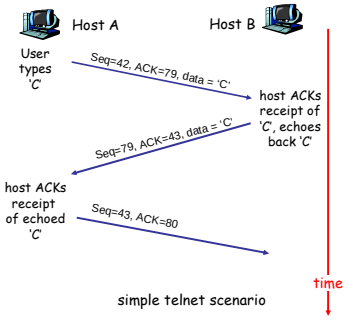
TCP seq. #'s and ACKs

Seq. #'s:

- byte stream "number" of first byte in segment's data

ACKs:

- seq # of next byte expected from other side
- cumulative ACK



simple telnet scenario

TCP Round Trip Time and Timeout

Q: how to set TCP timeout value?

- longer than RTT
  - but RTT varies
- too short: premature timeout
  - unnecessary retransmissions
- too long: slow reaction to segment loss

Q: how to estimate RTT?

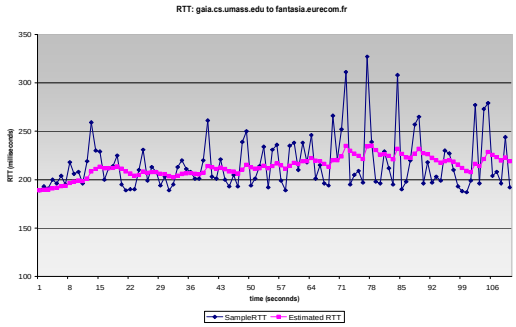
- **SampleRTT**: measured time from segment transmission until ACK receipt
  - ignore retransmissions
- **SampleRTT** will vary, want estimated RTT "smoother"
  - average several recent measurements, not just current **SampleRTT**

TCP Round Trip Time and Timeout

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

- Exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: **α** = 0.125

Example RTT estimation:



TCP Round Trip Time and Timeout

Setting the timeout

- **EstimtedRTT** plus "safety margin"
  - large variation in **EstimatedRTT ->** larger safety margin
- first estimate of how much SampleRTT deviates from EstimatedRTT:

```
DevRTT = (1-β)*DevRTT +
             β*|SampleRTT-EstimatedRTT|

(typically, β = 0.25)
```

Then set timeout interval:

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

TCP reliable data transfer

- TCP creates reliable service on top of IP's unreliable service
- TCP uses single retransmission timer
- Retransmissions are triggered by:
  - timeout events
  - duplicate acks

Fast Retransmit

- Time-out period often relatively long:
  - long delay before resending lost packet
- Detect lost segments via duplicate ACKs.
  - Sender often sends many segments back-to-back
  - If segment is lost, there will likely be many duplicate ACKs.
- If sender receives 3 ACKs for the same data, it supposes that segment after ACKed data was lost:
  - fast retransmit: resend segment before timer expires

TCP Congestion Control

- end-end control (no network assistance)
- sender limits transmission: **LastByteSent-LastByteAcked ≤ CongWin**
- Roughly,

$$\text{rate} = \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

- **CongWin** is dynamic, function of perceived network congestion

How does sender perceive congestion?

- loss event = timeout *or* 3 duplicate acks
- TCP sender reduces rate (**CongWin**) after loss event

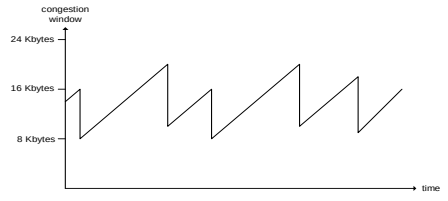
three mechanisms:

- AIMD
- slow start
- conservative after timeout events

TCP AIMD

multiplicative decrease: cut **CongWin** in half after loss event

additive increase: increase **CongWin** by 1 MSS every RTT in the absence of loss events: *probing*



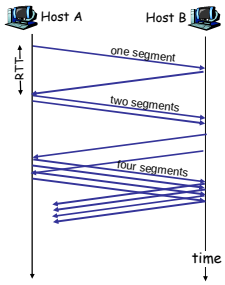
Long-lived TCP connection

TCP Slow Start

- When connection begins, **CongWin** = 1 MSS
  - Example: MSS = 500 bytes & RTT = 200 msec
  - initial rate = 20 kbps
- available bandwidth may be >> MSS/RTT
  - desirable to quickly ramp up to respectable rate
- When connection begins, increase rate exponentially fast until first loss event

TCP Slow Start (more)

- When connection begins, increase rate exponentially until first loss event:
  - double **CongWin** every RTT
  - done by incrementing **CongWin** for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



Refinement

- After 3 dup ACKs:
  - **CongWin** is cut in half
  - window then grows linearly
- But after timeout event:
  - **CongWin** instead set to 1 MSS;
  - window then grows exponentially
  - to a threshold, then grows linearly

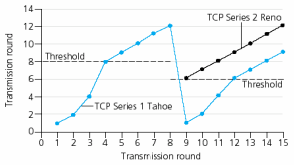
Philosophy:

- 3 dup ACKs indicates network capable of delivering some segments
- timeout before 3 dup ACKs is "more alarming"

Refinement (more)

Q: When should the exponential increase switch to linear?

A: When **CongWin** gets to 1/2 of its value before timeout.



Implementation:

- Variable Threshold
- At loss event, Threshold is set to 1/2 of CongWin just before loss event

Summary: TCP Congestion Control

- When **CongWin** is below **Threshold**, sender in slow-start phase, window grows exponentially.
- When **CongWin** is above **Threshold**, sender is in congestion-avoidance phase, window grows linearly.
- When a triple duplicate ACK occurs, **Threshold** set to **CongWin/2** and **CongWin** set to **Threshold**.
- When timeout occurs, **Threshold** set to **CongWin/2** and **CongWin** is set to 1 MSS.

TCP sender congestion control

Event	State	TCP Sender Action	Commentary
ACK receipt for previously unacked data	Slow Start (SS)	CongWin = CongWin + MSS, If (CongWin > Threshold) set state to "Congestion Avoidance"	Resulting in a doubling of CongWin every RTT
ACK receipt for previously unacked data	Congestion Avoidance (CA)	CongWin = CongWin+MSS * (MSS/CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
Loss event detected by triple duplicate ACK	SS or CA	Threshold = CongWin/2, CongWin = Threshold, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
Timeout	SS or CA	Threshold = CongWin/2, CongWin = 1 MSS, Set state to "Slow Start"	Enter slow start
Duplicate ACK	SS or CA	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed