

CS290i - Lecture 4

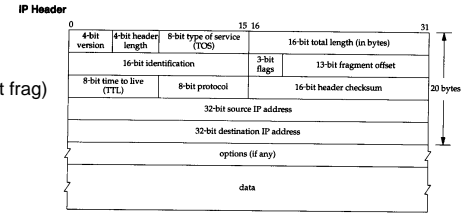
TCP, Meas Web Srvr Perf

Scalable Internet Services and Systems, Spring 2001

Thorsten von Eicken
 Department of Computer Science
 University of California at Santa Barbara

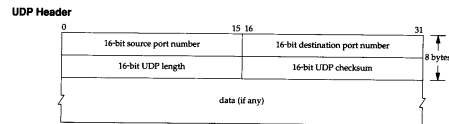
IP recap

- † Source
 - † TCP/IP Illustrated, Vol 1
 - † By Richard Stevens
- † Fragmentation
 - † Flag: more frags
 - † Flag: don't frag
 - † ID: id of packet, (not frag)
- † Protocols
 - † ICMP, UDP, TCP
- † Addresses
 - † I/f on host
 - † Subnets
 - † /19 = ff.ff.e0.00



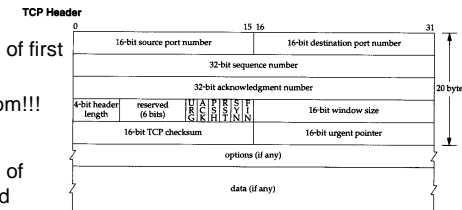
UDP

- † Ports
 - † Process (really?)
- † Details to consider
 - † Reliability
 - † Fragmentation
- † Is UDP useful?
 - † Why not use TCP?
 - † Do firewalls matter?



TCP

- † Seq number
 - † Offset into stream of first byte in packet
 - † Initial value: random!!!
- † Ack number
 - † Offset into stream of next byte expected
- † Flags:
 - † Urgent: packet has urgent data
 - † Ack: ack number valid
 - † Psh: useless
 - † Reset: reset connection
 - † Syn: sync seq nums to start connection
 - † Fin: sender finished sending



TCP: connect/disconnect

† Connect

- † Three packets
- † Note SYN consumes 1 sequence number

† Close

- † Four packets
- † "Half-close"
- † Note FIN consumes 1 sequence number

† "Connection" =

- † <src ip, dest ip, src port, dest port>

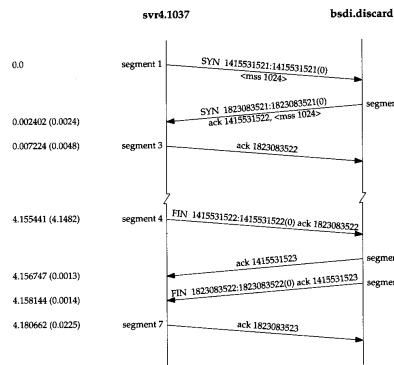


Figure 18.3 Time line of connection establishment and connection termination.

TCP state diagram

† TIME_WAIT

- † Spec
- † Cannot reuse 4-tuple
- † For 2 minutes
- † BSD impl:
- † Cannot reuse 2-tuple (local ip, local port)
- † SO_REUSEADDR:
- † Avoids impl restriction
- † But not TCP's
- † Q: max conn/sec?

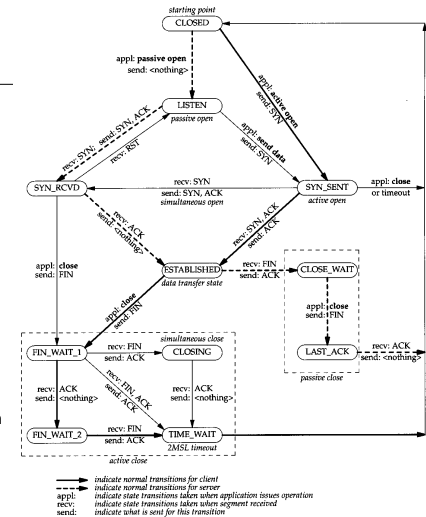


Figure 18.12 TCP state transition diagram.

Measuring Web Server Perf

† Authors

- † Gaurav Banga & Peter Druschel

† What does the paper really talk about?

- † Generating loads that exceed server performance
 - † Break req-resp -> req-resp dependency
 - † Burstiness exceeding server performance
- † Showing how these loads affect performance
 - † Really about connections/second

Connected...

† What do we know after this:

- † # telnet bugatti.cs.ucsb.edu 80
- Trying 128.111.40.111...
- Connected to bugatti.cs.ucsb.edu.
- Escape character is '^]'.
- .

- † What is running?
- † What is responding?

† And if I type a line:

- † # telnet bugatti.cs.ucsb.edu 80
- Trying 128.111.40.111...
- Connected to bugatti.cs.ucsb.edu.
- Escape character is '^]'.
- GET / HTTP/1.0
- † Who is echoing the characters I type? (trick question)

Dynamics of connections

† Avg queue lengths:

- † Syn-recvd:
 - † ~ request rate
 - † ~ r-t delay
- † Accept:
 - † ~ accept rate
 - † ~ req rate

† Ideal queue lengths?

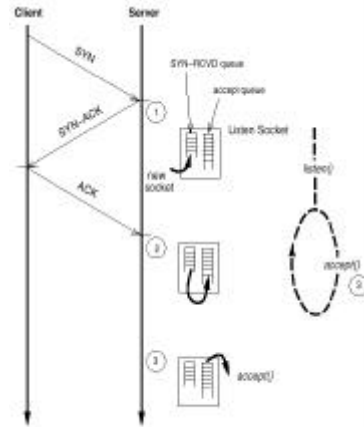


Figure 1: HTTP Connection Establishment Timeline

Load Gen Problems

† Typical:

- † Client Loop:
 - † Connect
 - † Send
 - † Recv
 - † Sleep
- † Slowed down by server

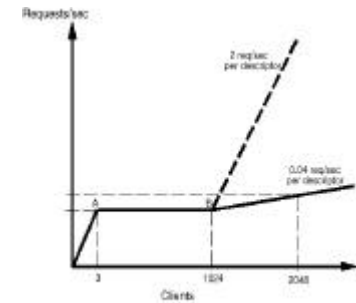


Figure 2: Request Rate versus no. of Clients

New Load Gen Client

† S-Client

- † Loop:
 - † Open connections w/o blocking
 - † Issue requests on connected sockets
 - † Read data & increment stats
- † Why 2 processes?

† Client limits?

- † #processes
- † #threads
- † #file desc
- † Cpu time
- † Disk I/O (traces, results)
- † Memory
- † LAN

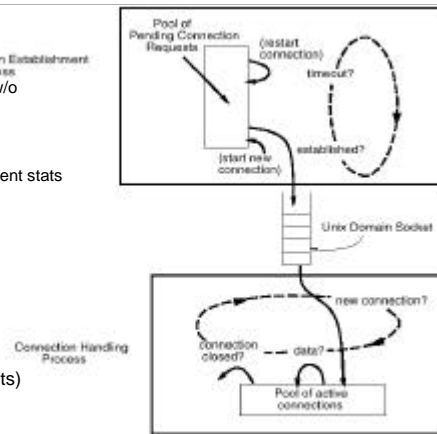


Figure 4: A Scalable Client

Results 1: request rate

HTTP Request Rate (req/sec)

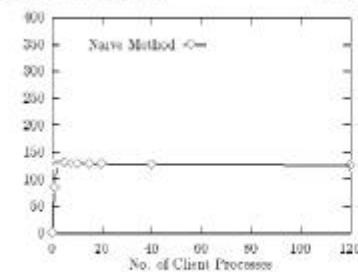


Figure 5: Request rate versus number of clients

HTTP Request Rate (req/sec)

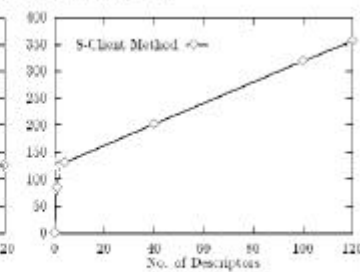


Figure 6: Request rate versus number of descriptors

Results 2: server t-put

- † Cause for decrease?
 - † Processing SYNs that are eventually dropped
- † At extreme:
 - † "receiver livelock": only process SYNs
- † Could it behave more interestingly than this?
 - † E.g. more complex requests?

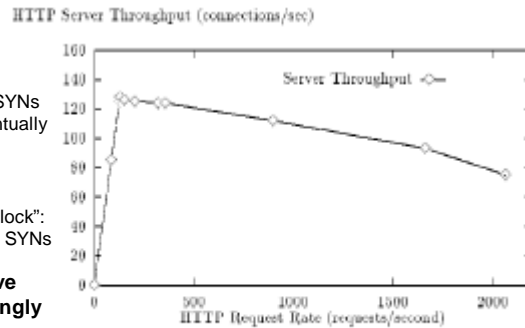


Figure 7: Web server throughput versus request rate

13

Results 3: Burstiness

† Traffic:

- † Period = 100sec
- † Burstiness (?,?)
- † ? = max/avg req rate
- † ? = %time @max / @avg

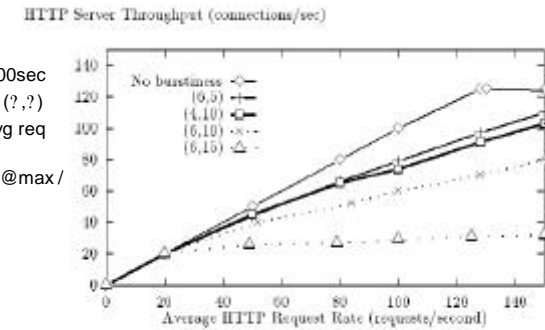


Figure 8: Web server throughput under bursty conditions versus request rate

14

WAN vs. LAN

- † Mentioned but not covered in paper...
- † How to measure?
 - † What params need to be tweaked?
 - † How can that be implemented?
 - † How much resources are required?
- † Expected effects?
- † Additional problems to test?
 - † Clients die/disconnect...
 - † Clients are slow

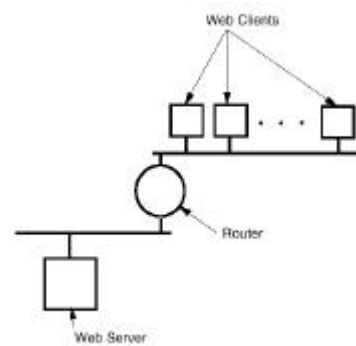


Figure 3: Testbed Architecture

15

Additional questions

- † Effect of HTTP1.1?
 - † On technique & results
 - † On realism of results / generating realistic load
- † How to apply technique to trace/rule based loads?
 - † What to do with dropped connections/requests?
 - † Need intelligent robots?
- † If server is overloaded, what should it do?
 - † Drop connections early
 - † Drop requests selectively
 - † Allow only N users into site

16

IBM Large Scale Web Srvr

† Source

- † Analysis and Characterization of Large Scale Web Server Access Patterns and Performance
- † A. Iyengar, M. Squillante, L. Zhang

† Site:

- † IBM Olympic Web site 1998 Nagano, Japan

† System:

- † SP2s with 10 uniprocessor + 1 MP node
- † 4 in Schaumburg
- † 3 ea in Columbus, Bethesda, Tokyo
- † "Network Dispatcher" routers

17

IBM Olympic web sites

Date: feb 13, 1998
Up to 1000 req/sec

Unique pages → 10Kb avg

25% dynamic →

dynamic →

Request Type	REQUEST		PAGE ID		PAGE SIZE	
	Number	Percent	Number	Percent	Number (bytes)	Percent
html/htm	4546960	8.0	8702	14.4	59035096	8.8
gif	24923318	43.9	4263	7.0	45030983	7.2
jpg	2781487	4.9	6902	10.1	70617034	11.2
txt	666690	1.2	34	0.1	41358	0.0
class	4068382	7.2	333	0.4	812835	0.1
operation mark (*)	1702843	3.0	39941	61.0	469765517	65.2
trovcom	15218947	26.8	1	0.0	1487	0.0
other (data/backdoor)	2922597	5.1	4241	7.0	47603967	7.6
Total	56830964	100.0	60537	100.0	628958077	100.0
GET (http.0)	44388877	78.1				
GET (http.1)	12402936	21.8				
HEAD (http.0)	23055	0.1				
HEAD (http.1)	94	0.0				
HTTP1.0	44426836	78.2				
HTTP1.1	12404308	21.8				
code304 (if-modified-since)	12567258	21.8				
code200 (successful-transfer)	43886936	77.2				
code404 (unknown URL)	566878	1.0				

Table 1: Summary of statistics and data from empirical analysis of the Olympic Web site access logs.

18

Projects

† What to turn in:

- † Check handout!
- † Email to blanquer@cs.ucsb.edu
 - † Subject: cs290i project 1 <your login id>
 - † "1/2-page" description of your solution
 - † Which modules were handy, which were a waste of time
 - † What special problems you encountered, your solutions
 - † Your perl script
 - † Separate files by "---- filename.pl -----"
- † Your database with the data on bugatti
- † Deadline: Thu

19