

**GRS CS 655**

**Graduate Intro to Computer Networks**

**Programming Assignment 1**

Name: Haoxuan Sun

BUID: U58198360

Data: 10/6/2022

## How to run pa1part1 program?

Run program *EchoServer* on csa1, csa2, csa3.bu.edu.

0. Enter the directory where these files are located.
1. On command line: *javac EchoServer.java*
2. On command line: *java EchoServer 58989* (any ports from 58000 to 58999)

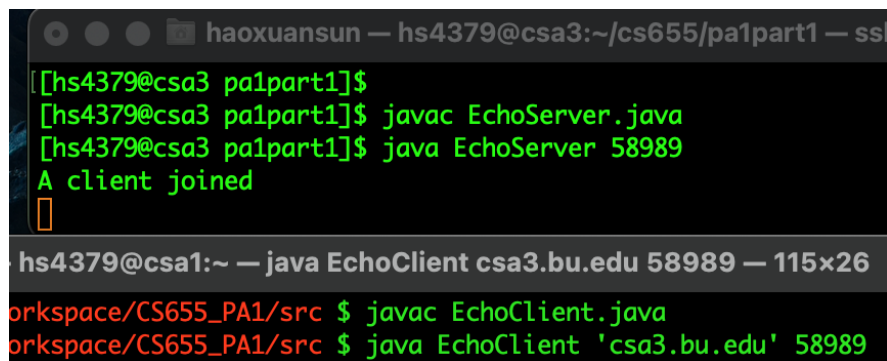
Now the **EchoServer** is working and waiting for a client connection.

Run program *EchoClient* on local machine.

3. On command line: *javac EchoClient.java*
4. On command line: *java EchoClient 'csa1.bu.edu' 58989* (csa1 or csa2 or csa3 depends on what address you run server on, and the same port number the server is listening)

Now the *EchoClient* is connecting to the server.

Use Control + C to exit EchoClient program



```
haoxuansun — hs4379@csa3:~/cs655/pa1part1 — ssh
[hs4379@csa3 pa1part1]$
[hs4379@csa3 pa1part1]$ javac EchoServer.java
[hs4379@csa3 pa1part1]$ java EchoServer 58989
A client joined
[hs4379@csa3 pa1part1]$

hs4379@csa1:~ — java EchoClient csa3.bu.edu 58989 — 115x26
orkspace/CS655_PA1/src $ javac EchoClient.java
orkspace/CS655_PA1/src $ java EchoClient 'csa3.bu.edu' 58989
```

Figure 1 Server and Client ready to communicate

## pa1part1 tradeoff and extension

Server program can be implemented to accept multi clients at the same time (multi thread).

## pa1part1 Testing Documents

Case no.	Address of Server	Port of Server	EchoClient argument [0]	EchoClient argument [1]	Terminal Output	Client Input	Echo Output	Expected?
1	csa3.bu.edu	58989	csa3.bu.edu	58989	-	Hello	Hello	Yes
2	csa3.bu.edu	58989	csa3.bu.edu	58989	-	DLROW OLLEH	DLROW OLLEH	Yes
3	csa3.bu.edu	58989	csa3.bu.edu	58988	Couldn't get I/O for the connection to: csa3.bu.edu	-	-	Yes
4	csa3.bu.edu	58989	csa1.bu.edu	58989	Couldn't get I/O for the connection to: csa1.bu.edu	-	-	Yes
5	csa3.bu.edu	58989	csa4.bu.edu	58989	Don't know host:csa4.bu.edu	-	-	Yes

Table 1 Testing Record

```

^C[haoxuansun@SHxMBP:] ~/eclipse-workspace/CS655_PA1/src $ java EchoClient 'csa3.bu.edu' 58988
Couldn't get I/O for the connection to: csa3.bu.edu
[haoxuansun@SHxMBP:] ~/eclipse-workspace/CS655_PA1/src $ java EchoClient 'csa1.bu.edu' 58989
Couldn't get I/O for the connection to: csa1.bu.edu
[haoxuansun@SHxMBP:] ~/eclipse-workspace/CS655_PA1/src $ java EchoClient 'csa4.bu.edu' 58989
Don't know host:csa4.bu.edu

```

Figure 3 Test Case 1 ~ 2

```

[haoxuansun@SHxMBP:] ~/eclipse-workspace/CS655_PA1/src $ java EchoClient 'csa3.bu.edu' 58989
Hello
echo: Hello
DLROW OLLEH
echo: DLROW OLLEH
^C[haoxuansun@SHxMBP:] ~/eclipse-workspace/CS655_PA1/src $

```

Figure 2 Test Case 3 ~ 5

5 tests are run to check the correctness of both server and client program. All the outcomes indicate that the program have achieved the design objectives. The first 2 cases are normal client input. The third case checks what if port number is inconsistent. The fourth case checks what if server address is incorrect. The fifth case check what if the server address in no real.

## How to run pa1part2 program?

Run program **Server** on csa1, csa2, csa3.bu.edu.

0. Enter the directory where these files are located.
5. On command line: **javac Server.java**
6. On command line: **javac MultiThread.java**
7. On command line: **java Server 58989** (any ports from 58000 to 58999)

Now the **Server** is working and waiting for a client connection.

Run program **Client** on local machine.

8. On command line: **javac Client.java**
9. On command line: **java Client 'csa1.bu.edu' 58989** (csa1 or csa2 or csa3 depends on what address you run server on, and the same port number the server is listening)

Now the **Client** is connecting to the server.

Use Control + C to exit EchoClient program

Output: Please enter measure type ('rtt' for RTT or 'tput' for throughput):

Input: 'rtt' or 'tput' (taking rtt as an example here)

Output: Please enter the number of measurement probes that the server should expect to receive:

Input: 20

Output: Please enter the number of bytes in the probe's payload:

Input: 1000

Output: Please enter the amount of time that the server should wait before echoing the message back to the client (in ms):

Input: 0

Output: ##### (Client sending and receiving logs)

Output: Mean RTT of 20 probes: 282 ms

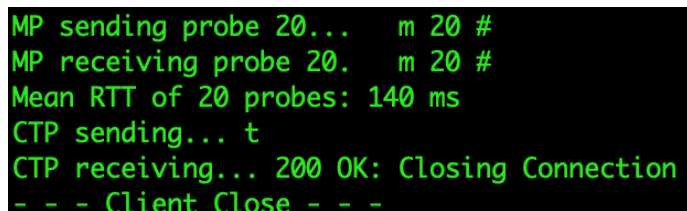
CTP sending... t

CTP receiving... 200 OK: Closing Connection

- - - Client Close - - -

## Description

The location of my client machine is at home. My laptop installs MacOS and home WiFi is used to access the network. The server part in whole part 2 assignment is done on cs1.bu.edu or 'pcvm3-2.geni.uchicago.edu @ 50000'. For every averaged RTT or throughout, it is averaged by at least 3 measurements (detailed can be found later). And before the average is taken, 20 probes of a certain payload's size is sent and received in every measurement. The line charts are produced using office software: Numbers. Both data file and line chart are provided in the attached data files.

A screenshot of a terminal window with a black background and green text. The text displays the results of a network test, including the number of probes sent and received, the mean Round Trip Time (RTT), and the status of the connection.

```
MP sending probe 20... m 20 #  
MP receiving probe 20. m 20 #  
Mean RTT of 20 probes: 140 ms  
CTP sending... t  
CTP receiving... 200 OK: Closing Connection  
- - - Client Close - - -
```

Screenshot of a measured RTT (probe number = 20; message size = 1 byte; server delay = 0ms; measure no. 3)

## pa1part2 Testing Documents

Case no.	Address of Server	Port of Server	Client argument [0]	Client argument [1]	Terminal Output	Client Input	Echo Output	Expected?
1	csa1.bu.edu	58989	csa1.bu.edu	58989	-	Hello	Hello	Yes
2	csa1.bu.edu	58989	csa1.bu.edu	58989	-	DLROW OLLEH	DLROW OLLEH	Yes
3	csa1.bu.edu	58989	csa1.bu.edu	58988	Couldn't get I/O for the connection to: csa1.bu.edu	-	-	Yes
4	csa1.bu.edu	58989	csa3.bu.edu	58989	Couldn't get I/O for the connection to: csa3.bu.edu	-	-	Yes
5	csa1.bu.edu	58989	csa4.bu.edu	58989	Don't know host:csa4.bu.edu	-	-	Yes

Case no.	Get Measurement type	Get probe number	Get message size	Get server delay	Excepted output	output	Expected?
1	"rt"	-	-	-	Ask again	Ask again	Yes
2	"rtt"	-	-	-	No error	No error	Yes
3	-	-1	-	-	Ask again	Ask again	Yes

4	-	0	-	-	Ask again	Ask again	Yes
5	-	1	-	-	No error	No error	Yes
6	-	-	-1	-	Ask again	Ask again	Yes
7	-	-	0	-	Ask again	Ask again	Yes
8	-	-	1	-	No error	No error	Yes
9	-	-	-	-1	Ask again	Ask again	Yes
10	-	-	-	0	Ask again	Ask again	Yes
11	-	-	-	1	No error	No error	Yes

Test Case	Terminal Input	Excepted output	output	Expected?
1	java Client csa1.bu.edu	Wrong Input Format! 'java Client <host name> <port number>'	Wrong Input Format! 'java Client <host name> <port number>'	Yes
2	Java Server	Wrong Input Format! 'java Server <port number>'	Wrong Input Format! 'java Server <port number>'	Yes

## Summary of Results

The first two figures show the result obtained through running experiment on csa1.bu.edu. Figure 1 shows how the size of message affect TCP's RTT. In the graph, each RTT value is averaged by 3 measured RTT and each measurement contains 20 probes (See attached file csa1\_data\_file for detailed dataset).

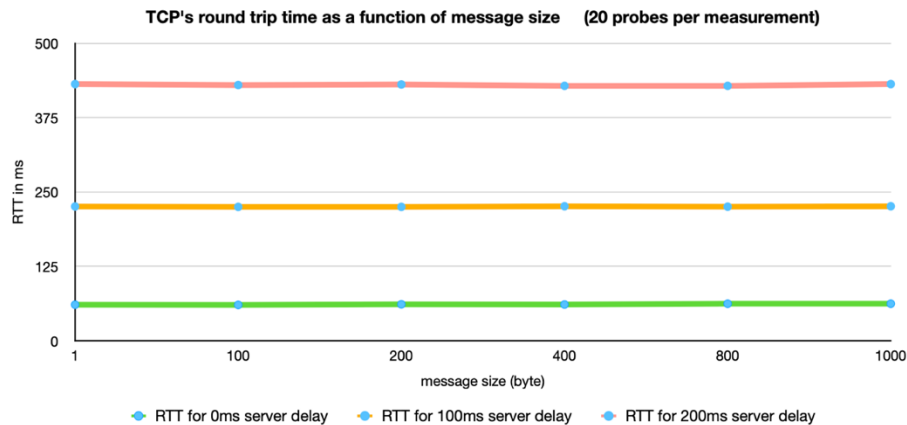


Figure 4 TCP's RTT as a function of message size (20 probe per measurement)

We can find that, no matter what the size of the message is, the RTT for a group of TCP remains the same (for a fixed server delay). The RTT for size 1 byte is almost the same as the RTT for size 1000 bytes when delay is 0ms. When change the server delay, the previous statement still stands. The RTT for size 1 byte is almost the same as the RTT for size 1000 bytes when delay is 100ms and 200ms.

However, if we focus on how RTT changes when set different server delay while keeping message size unchanged, we'll find that the bigger server delay is, the bigger RTT is. It is easy to understand because server delay is to emulate paths with propagation delays. Longer path causes bigger RTT.

The difference between average RTT and 2 times of server delay are about 61ms, 25ms and 30ms respectively when server delay are 0ms, 100ms and 200ms.

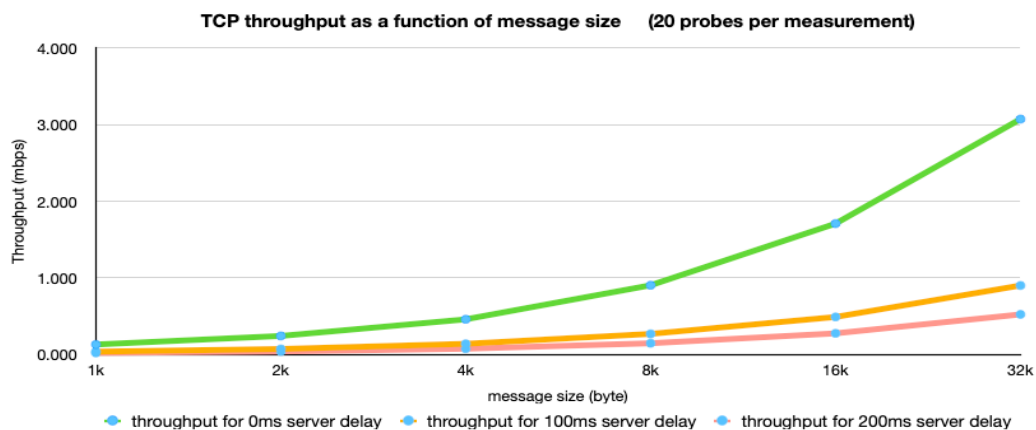


Figure 5 TCP throughput as a function of message size (20 probes per measurement)



Figure 2 shows how the size of message affect TCP's throughput (megabits per second). In the graph, each throughput value is averaged by 3 measured throughput and each measurement contains 20 probes (See attached file `csal_data_file` for detailed dataset).

We can see that, as message size increases, the throughput is also increasing, and throughput grows a little slower than the times of message grows. For example, when server delay is 0ms, throughput of 1000 byte is 0.128, throughput of 2000 byte is 0.240 (twice of 0.128 is 0.256); throughput of 4000 byte is 0.464 (twice of 0.240 is 0.480).

If we focus on how throughput changes when set different server delay while keeping message size unchanged, we'll find that the bigger server delay is, the smaller throughput is. It is also easy to understand because server delay is like a traffic jam. More time jammed on road, less average speed a vehicle has.

The following two figures show the result obtained through running experiment on 'pcvm3-2.geni.uchicago.edu @ 50000'.

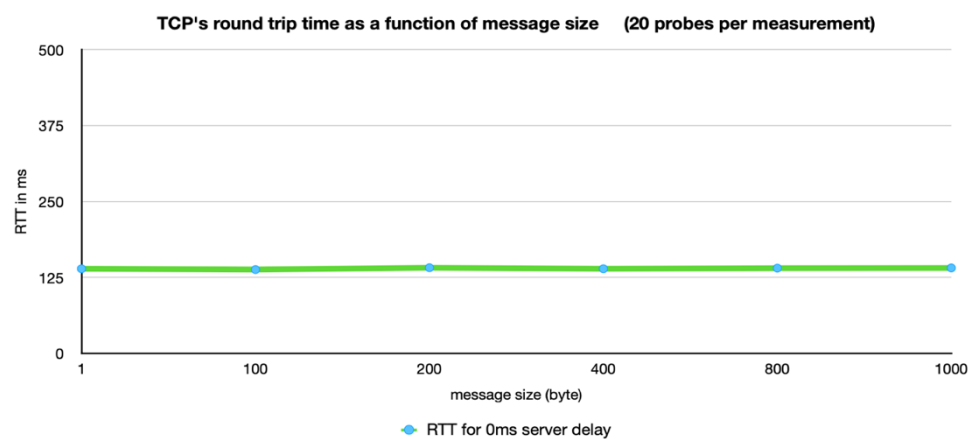


Figure 6 TCP's RTT as a function of message size (20 probes per measurement)

In the figure 3, each RTT value is averaged by 3 measured throughput and each measurement contains 20 probes (See attached file `UChicago_data_file` for detailed dataset).

Like figure 1, figure 3 shows how the size of message affect TCP's RTT and the server delay is only required to be 0ms. The TCP trend conclusion still stands. The RTT remains the same (almost) no matter how the size of message varies. The RTT on this server is much bigger that the RTT in figure 1, the physical location of the server and the length of path for messages travel from local machine to server could be the reason.

In the figure 4, each RTT value is averaged by 5 measured throughput and each measurement contains 20 probes (See attached file `UChicago_data_file` for detailed dataset).

Like figure 2, figure 4 shows how the size of message affect TCP's throughput (megabits per second) and the server delay is only required to be 0ms. The graph shows the similar trend. As message size increases, the throughput is also increasing, but this time, throughput grows much slower than the times of message grows, and it is growing slower and slower. For example, throughput of 4000 byte is 0.246 mbps, throughput of 8000 byte is 0.374 mbps (message size doubles but throughput increases 55%); throughput of 16000 byte is 0.489 (message size doubles but throughput increases 31%).

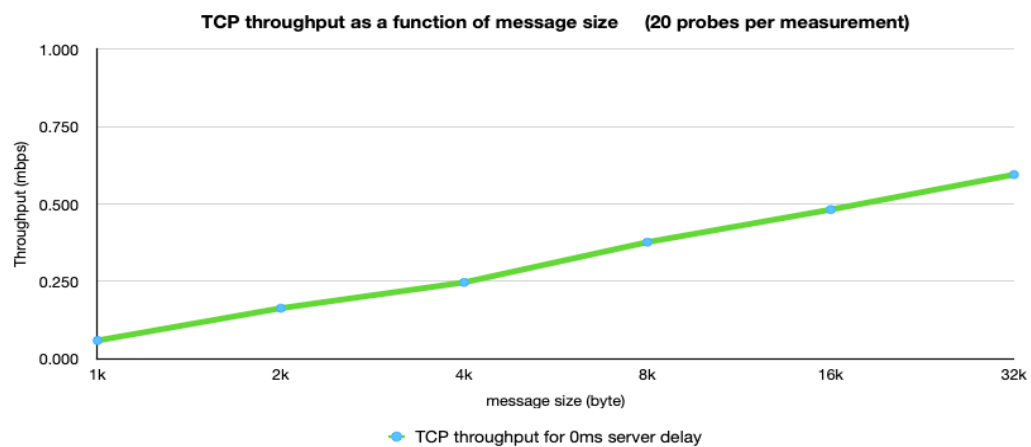
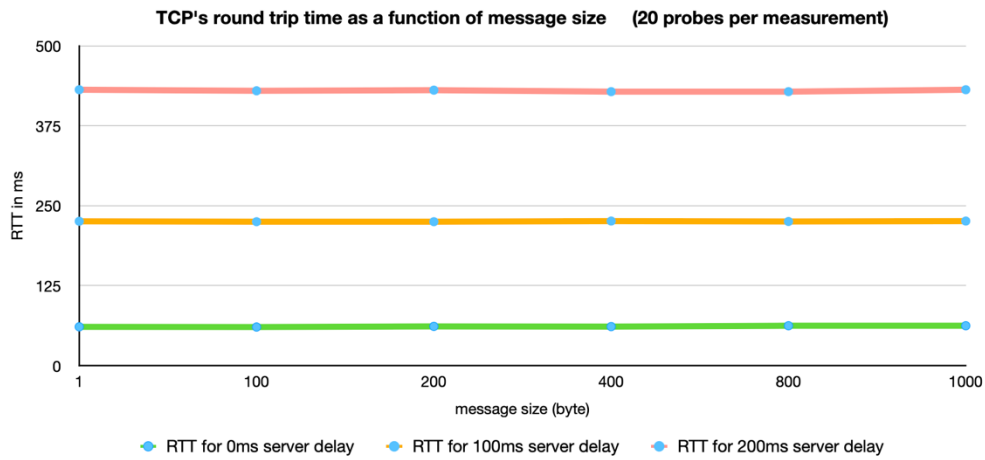


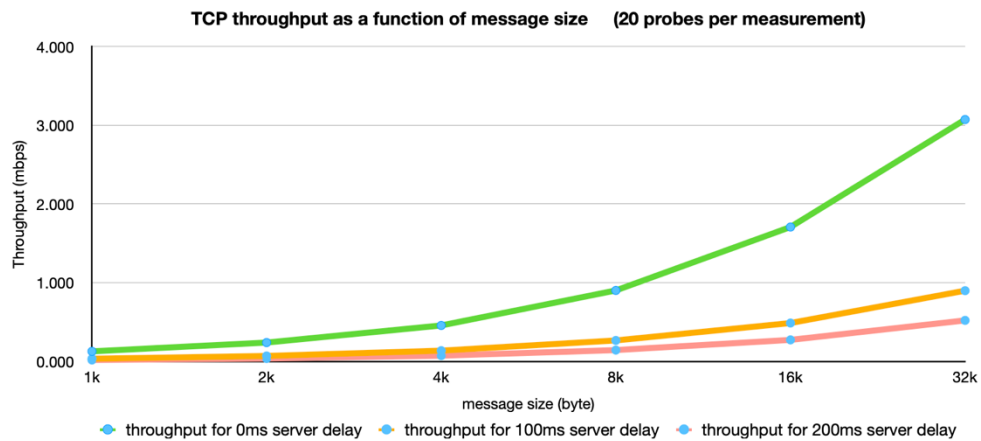
Figure 7 TCP throughput as a function of message size (20 probes per measurement)

## csa1\_data\_file



**TCP's round trip time as a function of message size (20 probes per measurement)**

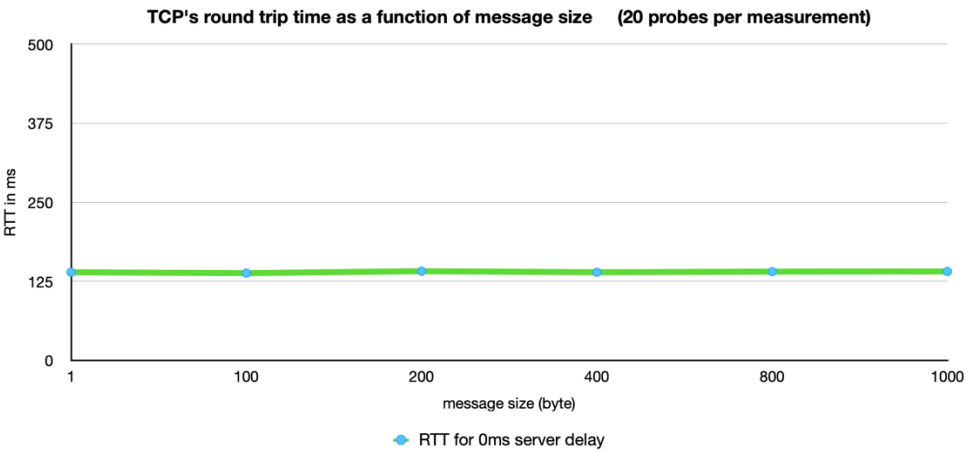
Server Delay (ms)	0				100				200			
size (bytes) / ms	mea sure 1	mea sure 2	mea sure 3	Ave rage	mea sure 1	mea sure 2	mea sure 3	Ave rage	mea sure 1	mea sure 2	mea sure 3	Ave rage
1	60	61	61	61	225	226	226	226	429	439	426	431
100	61	60	60	60	226	224	225	225	428	429	432	430
200	61	62	61	61	225	225	225	225	433	428	431	431
400	62	61	60	61	226	227	225	226	429	429	427	428
800	61	63	63	62	225	225	226	225	429	426	430	428
1000	62	62	63	62	226	227	225	226	427	437	430	431



**TCP throughput as a function of message size (20 probes per measurement)**

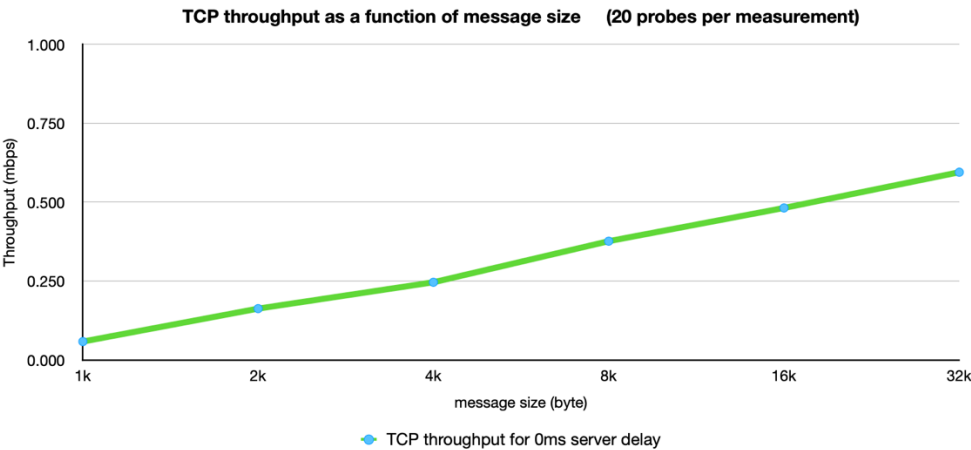
Server Delay (ms)	0				100				200			
size (bytes) / mbps	measu re1	measu re2	measu re3	Avera ge	measu re1	measu re2	measu re3	Avera ge	measu re1	measu re2	measu re3	Avera ge
1k	0.129	0.127	0.129	0.128	0.035	0.035	0.035	0.035	0.019	0.018	0.019	0.019
2k	0.235	0.239	0.246	0.240	0.069	0.071	0.070	0.070	0.037	0.037	0.037	0.037
4k	0.464	0.444	0.464	0.457	0.137	0.137	0.138	0.137	0.074	0.074	0.073	0.074
8k	0.941	0.888	0.877	0.902	0.267	0.266	0.268	0.267	0.145	0.145	0.144	0.145
16k	1.707	1.662	1.753	1.707	0.492	0.487	0.485	0.488	0.274	0.274	0.274	0.274
32k	3.084	3.122	3.011	3.072	0.898	0.895	0.904	0.899	0.522	0.518	0.525	0.522

UChicago\_data\_file



TCP's round trip time as a function of message size (20 probes per measurement)

Server Delay (ms)	0			
size (bytes) / ms	measure1	measure2	measure3	Average
1	139	139	140	139
100	141	138	135	138
200	143	140	140	141
400	139	140	139	139
800	139	141	141	140
1000	140	141	141	141



TCP throughput as a function of message size (20 probes per measurement)						
Server Delay (ms)	0					
size (bytes) / mbps	measu re1	measu re2	measu re3	measu re4	measu re5	Avera ge
1k	0.058	0.057	0.058	0.059	0.060	0.058
2k	0.165	0.162	0.157	0.163	0.167	0.163
4k	0.241	0.250	0.252	0.246	0.242	0.246
8k	0.370	0.390	0.374	0.374	0.374	0.376
16k	0.485	0.474	0.479	0.481	0.489	0.482
32k	0.620	0.595	0.579	0.583	0.597	0.595