TCP Performance ment

Assignment 2

Sweta Dwivedi
Taskinul Haque
Timon Dressler

2/14/2011

TCP Performance measurement using application/ protocol parameters

Experiments 1 - Changing window size and throughput

In this experiment, the window size was varied to see the effects of the received window buffer size on throughput.

Network Technology: 1-Gbit ethernet card

Methodology:
(1) Run iperf to test default values to get the maximum throughput
(2) Set the time interval (-t 30) and port number (-p 5555) to test fairly
(3) Change the window buffer size (-w) at the receiver using iperf (sudo iperf -s -p 5555 -w 500 [KB] -t 30)*

*The OS always sets the window buffer size to the double of the value specified.

Parameters: Time interval (0-30 ms, constant), Client window buffer size (16 KB, constant) , Server window buffer size (10 – 256 KB).

Result: As the window buffer size at the receiver kept increasing the throughput also increased, while reduced buffer size showed a decrease in the throughput.

Table 1.1 - Effect of Window Buffer Size on Throughput

	Throughput (Mb/s)			
Window Buffer Size (KB)	Trial 1	Trial 2	Trial 3	Average
9.77	138	186	151	158.333333
19.5	218	216	216	216.666667
39.1	383	385	386	384.666667
58.6	665	666	666	665.666667
85.3	941	937	936	938
97.7	903	901	904	902.666667
117	932	930	932	931.333333
137	935	933	935	934.333333
195	936	935	935	935.333333
256	936	936	936	936

PLEASE NOTE: For all Tables refer to Chart for visualization with Corresponding Table number (i.e. Table 1.1 = Chart 1.1)

Explanation
Although by performing the test with the default window buffer size at the receiver, we could achieve throughputs of nearly 936 Mbits/s, by reducing our buffer size we could achieve only lower values that is due the fact that even though the capacity of our network maybe higher, we are limited by the window buffer size as it tells us the amount of data or size of data that the receiver can accept at a given time. Hence the higher the buffer size the higher throughput achievable whereas lower the buffer size lower the throughput.

Experiment 2 – Read / Write length at buffer and throughput

In this experiment, the read/write length at the buffer on the receiver was varied to see the effect on the throughput.

Methodology:

(1) Change the window buffer size to default 85.3 KB
(2) Run iperf to test default values to get the maximum throughput
(3) Set the time interval (-t 30) and port number (-p 5555) to test fairly
(4) Change the length of read/write (-l) at the receiver using iperf (sudo iperf -s -p 5555 -l 8 [KB] -t 30)

Parameters: Time interval (0-30 ms, constant), Client window buffer size (16 Kb, constant), Server window buffer size (85.3 KB, constant), Read/write length at buffer receiver (1B – 10KB).

Result: Increasing the length of read/write at the buffer on the receiver increased the throughput whereas decreasing the length of read/write at the buffer decreased the throughput.

Table 2 - Effect of Read Length on Throughput

	Throughput (Mb/s)			
Read Length (KB)	Trial 1	Trial 2	Trail 3	Average
10	937	937	937	937
8	**936**	**937**	**937**	936.667
4	937	937	937	937
1	937	937	937	937
0.256	936	937	936	936.333
0.128	937	937	936	936.667
0.064	793	832	831	818.667
0.032	423	413	446	427.333
0.016	221	225	220	222
0.008	112	115	111	112.667
0.004	53.3	53.1	52.2	52.8667
0.001	15.1	14.9	14.8	14.9333

Explanation
The default buffer read/write length at the receiver is 8KB which explains that the data or packets can be read by the application in 8KB segments. By increasing the read/write length of the buffer, we increase the amount/size of data read per segment which means the buffer gets cleared faster and we get more space in the buffer to receive more packets and since tcp uses sliding window flow control, even if we get an ACK for even a single frame, the sender can send the next frame without waiting.

Although varying the value of the read/write buffer size by a few KB doesn't show any significant changes unless they are halved because the read length is still very fast to make a difference in throughput, until the length value is set to around 64B and halved further to show an effect on the throughput.

TCP Performance measurement using network/link conditions (tc)

Experiment 3 – Link Data rate and Throughput

In this experiment, the link data rate is being varied to measure the effects on the throughput.

Methodology:

(1) Change the length read/write back to the default value 8 [KB]

(2) Set the Link Data rate using: sudo tc qdisc add dev etho root tbf rate 500 Kbit latency 50ms
where the latency is the maximum wait (delay) for the packet

(3) Replace the Link Data rate by using replace instead of the command add above

(4) To get more accurate results set the time interval in iperf to be 60 secs

Parameters: Time interval (0-30ms, constant), Data Rate (100Kb - 500Mb)

Result: Increasing the data rate increases the value of throughput and decreasing the value of throughput decreases the throughput.

Table 3 - Effect of Data Rate on Throughput

Data Rate (Mb/s)	Throughput (Mb/s)
0.1	0.0964
0.3	0.291
0.5	0.478
1	0.954
5	4.76
10	9.53
50	37.9
100	61.1

Explanation

The data rate tells us about the capacity of the network, so by setting the data rate we are also defining the upper limit at which the data can be sent. In our experiment, we manually specify the data rate at the client and observe an increasing trend starting from lower values but however the throughput will stabilize after a certain data rate because it would reach the maximum throughput achievable.

Experiment 4 – Link Delay and Throughput

In this experiment, the link delay was tested using 2 methods:

(1) Using the random function to pick one value from the desired range to stimulate real network delays (Eg. 100Ms +/ - 10)

(2) Setting the delay manually without specifying the range parameter. (Eg. 100ms)

Methodology:

(1) Delete the previously set data rate by using the same command as adding data rate by replacing add with del option

(2) For using a random delay with a range (client), use command:
sudo tc qdisc add dev eth0 root netem delay 100ms 10ms (10ms adds range eg. 90 – 110)

(3) Setting the delay parameter without a range, use same command as above without specifying 10 ms of range.

(4) Test desired delay is being used by ping.

Parameters: Time interval (0-60ms, constant), Link Delay (0-100ms), RTT

Result: Increasing the link delay decreases the throughput and decreasing the link delay increases the throughput.

Table 4.1 - Effect of Added Delay on Throughput

Delay (ms)	Throughput (Mb/s)
0	940
1	927
2	888
5	715
6	695
7	678
8	670
9	657
10	649
20	754
30	512
40	386
50	361
100	180
0ms Delay RTT = 0.157	

Table 4.2 - Using Random Range of delays and its affect on throughput

		Throughput (Mb/s)			
Range of Delays (ms)	RTT(ms)	Trial 1	Trial 2	Trial 3	Average
0-2	0.919	930	928	920	926
0-5	2.087	412	417	412	413.666667
0-10	6.101	205	201	202	202.666667
0-20	10.431	95.8	92.2	97	95
20-40	32.374	89.8	87.5	96.8	91.3666667
40-60	50.23	92	79.2	82.9	84.7
60-80	70.556	66.9	72.6	75.9	71.8
80-100	89.541	69.4	66.8	70.6	68.9333333

Explanation

Changing the link delay parameter in our experiment specifies adding different delay in between the packets that are being sent from the client. By adding a fixed delay between each packet being send we are actually increasing the Round Trip Time (RTT) which is the time taken for the frame to be sent and the ACK of that frame to be received. As the RTT increases the throughput decreases because the throughput is the measurement of the amount of time spent in sending useful data and since here we're actually spending the time waiting between packets our throughput gets lower with increased delays.

Experiment 5– Packet Drop % and Throughput

In this experiment, the packet drop % is being varied to see the effect on the throughput to demonstrate the effect of packet loss on throughput in the real network.
The experiment could be demonstrated using tc or iptables.

Iptables: iptables acts as a firewall, that can contain rules like how many packets to drop or allow for incoming or outgoing packets.

Methodology (iptables):

(1) Remove the delay using the same command as adding delay, replace add with del
(2) Input a rule to drop a certain percentage of incoming packets using command:
sudo iptables -A INPUT -m statistic --mode random --probability 0.03 -j DROP
(3) Check if the rule gets added to input table: sudo iptables -L
(4) Checking whether the packets are being dropped using iperf: iperf -c 192.168.1.5 -u -t 60
(5) Use iperf to test with normal tcp packets remove -u option from the command above

Parameters: Time interval (0-60ms, constant), Packet drop % (0.5 – 70)

Result: The increase in the packet drop % decreases the throughput whereas less packet drop % results in a higher throughput

Table 5.0 - Dropping Packets and Its effect on Throughput

Packet Drop probability (%)	Throughput (Kb/s)
0.1	454
0.5	243
1	158
5	17.3
10	3.12
20	0.54
30	0.231
40	0.115
50	0.0627
60	0.00313
70	0.000897

Explanation
In real networks there are always few packet drop occurring in the routers due to the congestion in the network. Congestion is caused by many sources trying to send data at a high rate. Here the packet loss or packet drop is configured manually to stimulate the real networks, the packet loss reduces the throughput due to the retransmissions of the packets which keeps increasing with the increasing packet loss. Retransmissions also increases the RTT due to the wait for time out.

Experiment 6 – Multiple TCP sessions and Throughput

In this experiment, multiple tcp sessions are started to see the effect of parallel tcp sessions on throughput to demonstrate the real life situation of running multiple applications which uses tcp

Methodology:

(1) Remove the packet loss % to test the effect of multiple sessions without the effect of packet loss using same command as adding packet loss but replacing -A with -D (delete)

(2) Check the rules for input table is gone (iptables -L)

(3) Set the multiple sessions in iperf (-P option) at the client using: iperf -c [IP address] -P 2 -t 60

Parameters: Time interval (0-60s, constant), Multiple session (-P 1 – 5)

Result: The bandwidth was divided equally amongst all sessions as long as no packet loss was added

Table 6.0 - Multiple Session running Simultaneously and its effect on Throughput

	Throughput (Mb/s)						
Session Type	**Session 1**	**Session 2**	**Session 3**	**Session 4**	**Session 5**	**Total**	**Average**
1 TCP	941					941	188.2
2 TCP	475	468				942	188.6
3 TCP	316	315	312			943	188.6
4 TCP	233	239	238	234		943	188.8
5 TCP	193	194	183	183	190	942	188.6
Same Tests as above changing Packet loss to 0.5%							
1 TCP	252					252	50.4
2 TCP	199	195				394	78.8
3 TCP	190	190	188			568	113.6
4 TCP	182	182	181	182		728	145.4
5 TCP							

Explanation

In real life networks there might be multiple tcp sessions created by different applications and it's important that all get to share the bandwidth equally, although iperf used different data sizes but not very different from each other. Hence the bandwidth was almost the same for all the sessions and since all other connections were turned off there wasn't any other Background traffic either.

Nonetheless, when packet loss values were added the average throughput of each session decreased dramatically refer to Chart 6.0. addding packet loss gave us a negative trend; so when more sessions existed more packets were lost thus decreasing the average throughput of each session, where as without loss all packets shared the BW equally.

Experiment 8 – Effects of Multiple TCP & UDP sessions on Throughput

This experiment is somewhat similar to the multiple tcp sessions except that there are udp sessions running in parallel as well.

Methodology:

(1) The server or receiver has to listen to a specific port number and listen to both udp and tcp packets at the same port number using: iperf -s -u -p 5zzz & iperf -s -p 5zzz

(2) The client has to send both udp and tcp packets to the same port number using: iperf -c [IP address] -u -t 30 -P [# sessions] (udp) & iperf -c [IP address] -t 30 -P [# sessions]

(3) The client needs to keep varying the number of tcp and udp sessions to see the effects on the throughput

Parameters: Time interval [0-30s, constant], -P [1-5]

Result: The TCP sessions get more bandwidth compared to the udp sessions

Table 6.1 - Simultaneous UDP & TCP Sessions and its Effect on Throughput

	Throughput (Mb/s)		
	TCP Sessions	UDP Sessions	UDP Packet Loss
1TCP, 1UDP	941	1.05	
2TCP, 1UDP	471	1.05	
	469		
3TCP, 1UDP	317	1.05	
	316		
	308		
1TCP, 2UDP	940	1.05	
		1.05	0.04%
2TCP, 2UDP	464	1.05	
	474	1.05	0.02%
3TCP, 3UDP	321	1.05	
	368	1.05	0.04%
	310	1.05	

Table 6.2 - Increased UDP and TCP Sessions

	Throughput (Mb/s)		
Dramatic Increase	TCP Session Sum	UDP Session (min)	UDP Max Packet Loss
5TCP, 5UDP	937	1.05	
10TCP, 5UDP	941	1.04	1.30%
15TCP, 5UDP	939	1.02	2.00%

Explanation

The UDP packets are much smaller in size due to less overheads and no retransmission schemes involved compared to the TCP sessions

Experiment 9 – Constant BDP vs Changing window buffer size vs Throughput

The effects of bandwidth delay product and window size on the throughput

Methodology:
(1) Keeping the Bandwidth delay product constant by choosing a fixed delay 10ms, add the delay of 10 ms
(2) Changing the window buffer size each time using iperf
(3) Calculate Bandwidth delay product using the data rate and delay and observe the effects of bandwidth delay product to window size and throughput

Parameters: Time interval (0-60ms, constant), Delay (10ms), Window buffer size (9.77 – 256 KB)

Result: As the window buffer size increases so does the throughput but the throughput isn't reaching the maximum value due to the window buffer size limitations even though the BDP is so much larger.

Table 1.2 - Varying Window Buffer Size with Constant BDP and its affect on Throughput

BDP (KB)	Window Buffer Size (KB)	Throughput (Mb/s)
1250	9.77	2.29
	19.5	9.13
	39.1	19.6
	58.6	18.6
	78.1	26.2
	97.7	43.4
	195	88.7
	256	95.6

Explanation
The Bandwidth delay product (BDP) = data rate * delay
BDP = 1000000 Kb/s * 0.01s
BDP = 10000 Kb
BDP = 1250 KB

Since a delay of 10 ms was chosen to keep the BDP constant our BDP in KB is 1250 KB and our window buffer size is increased gradually to see the effects on the throughput.
The size of the BDP delay product is actually larger than that of the maximum window buffer size allowed. So here even though we are allowing our client to send a data of 1250 KB before receiving an ACK but since our buffer size is smaller, we are limiting the amount of data that can be sent therefore achieving lower throughputs.

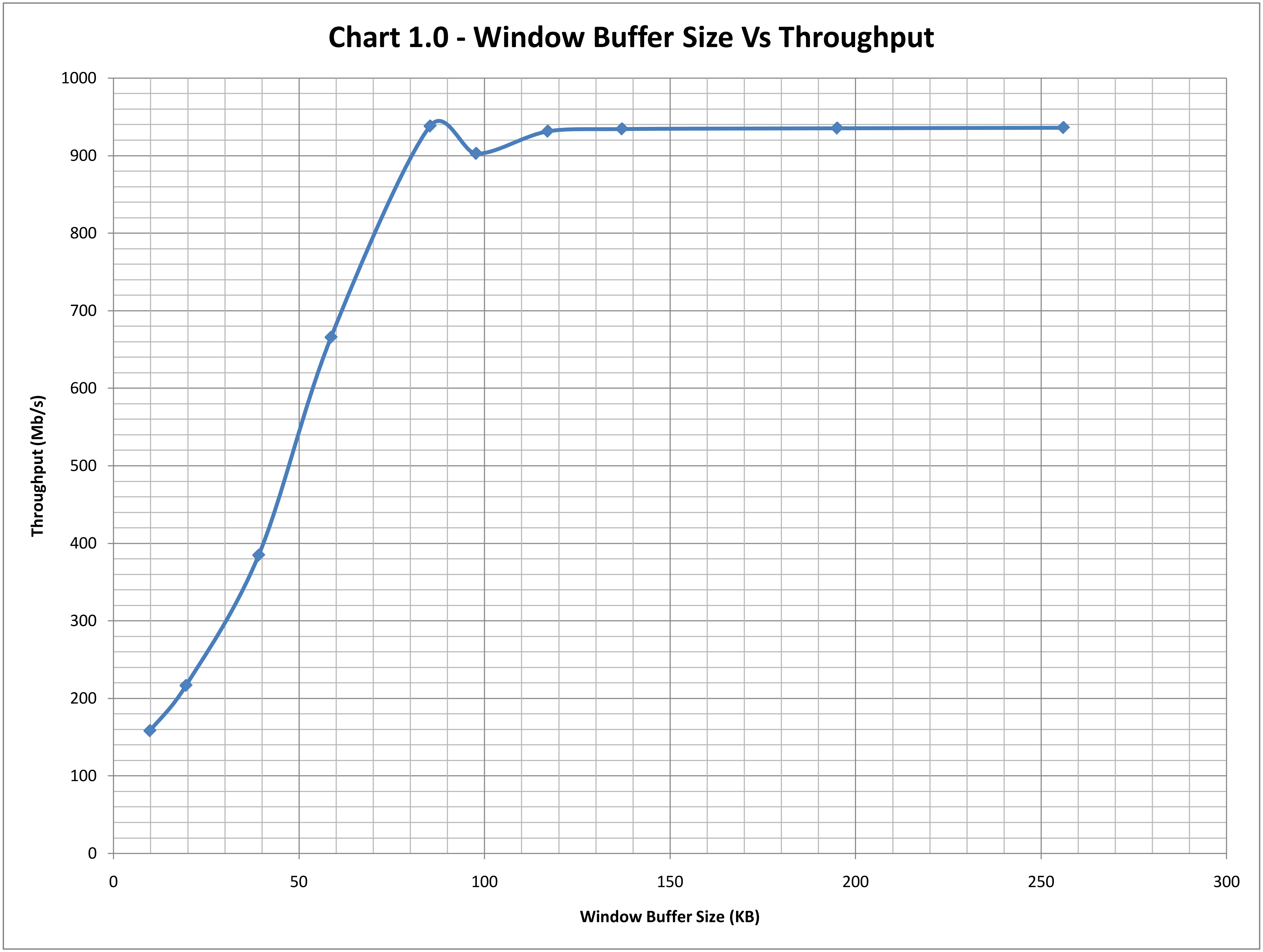

Chart 1.0 - Window Buffer Size Vs Throughput
Throughput (Mb/s)
Window Buffer Size (KB)
0
100
200
300
400
500
600
700
800
900
1000
0
50
100
150
200
250
300

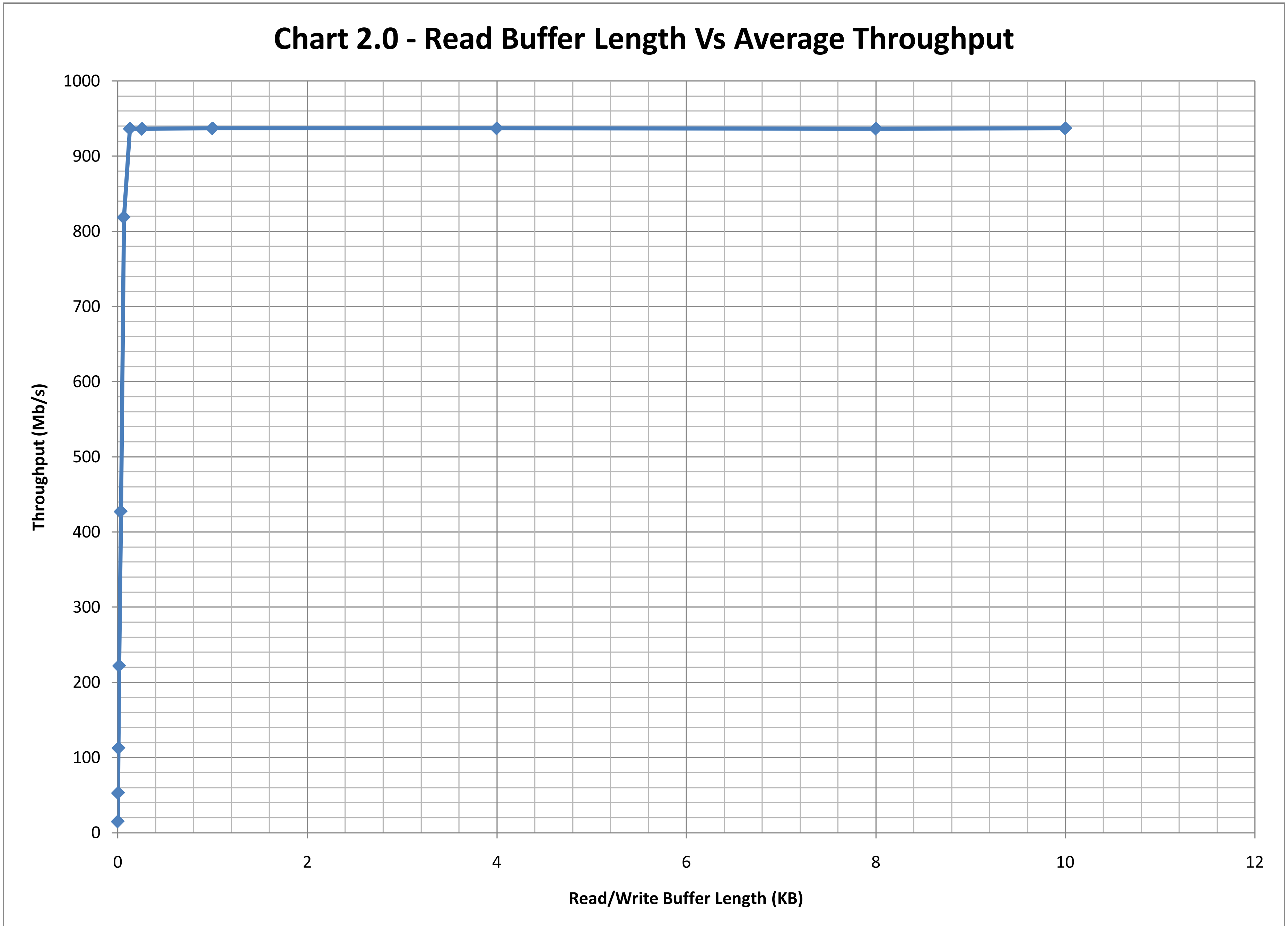
Chart 2.0 - Read Buffer Length Vs Average Throughput
Throughput (Mb/s)
1000
900
800
700
600
500
400
300
200
100
0
0
2
4
6
8
10
12
Read/Write Buffer Length (KB)

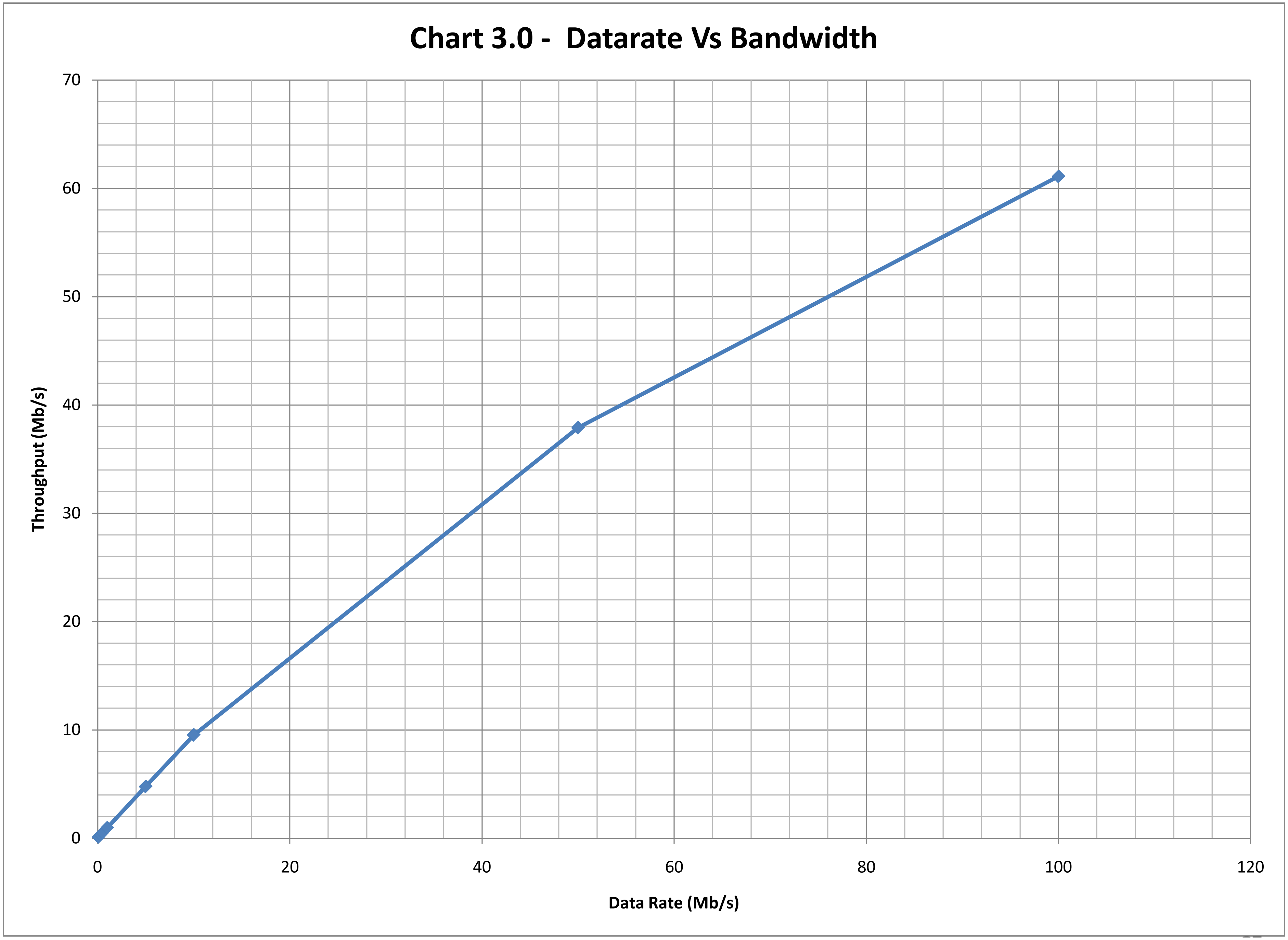
Chart 3.0 - Datarate Vs Bandwidth
Throughput (Mb/s)
Data Rate (Mb/s)
0
10
20
30
40
50
60
70
0
20
40
60
80
100
120

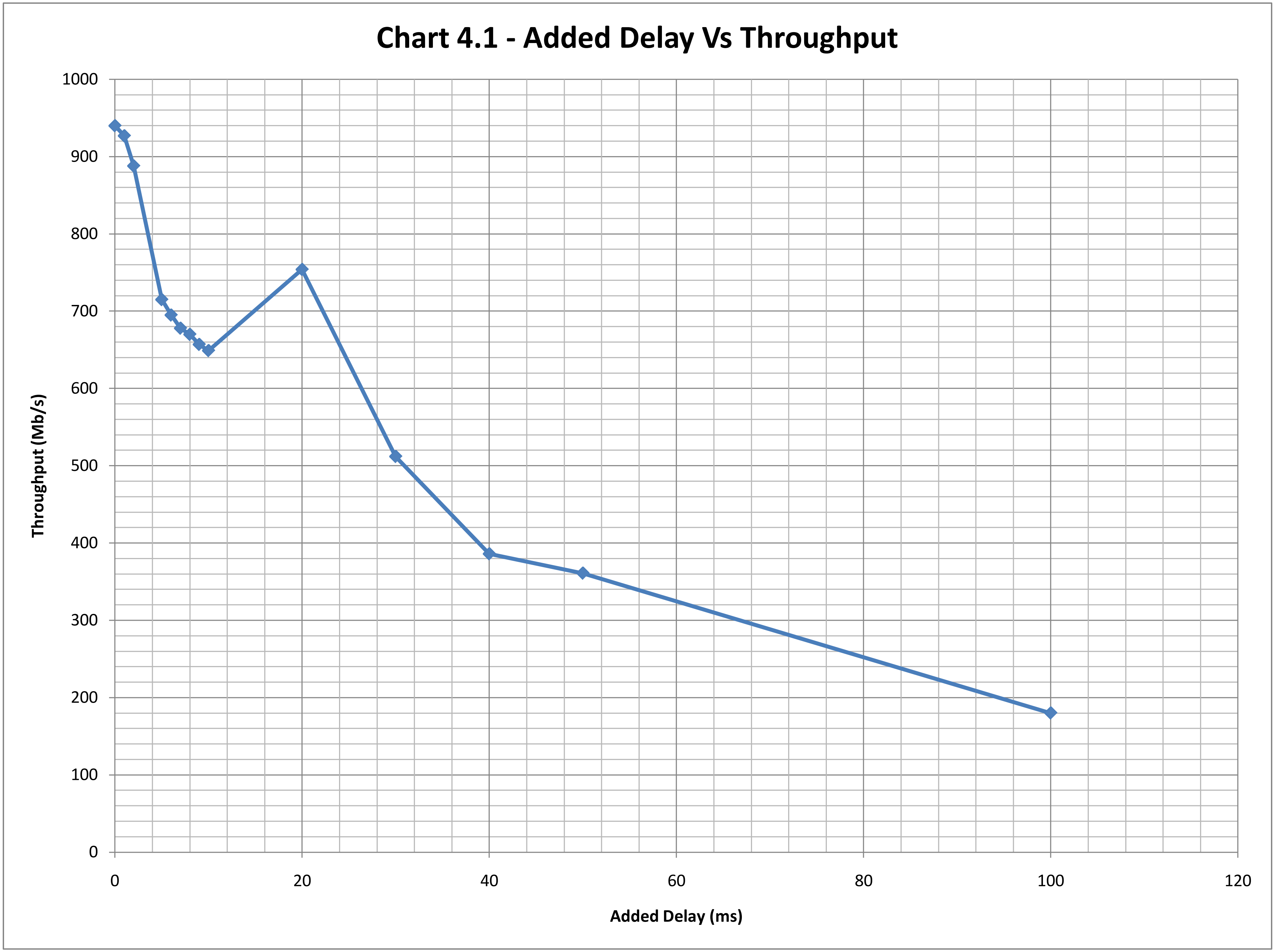
Chart 4.1 - Added Delay Vs Throughput
Throughput (Mb/s)
1000
900
800
700
600
500
400
300
200
100
0
0
20
40
60
80
100
120
Added Delay (ms)

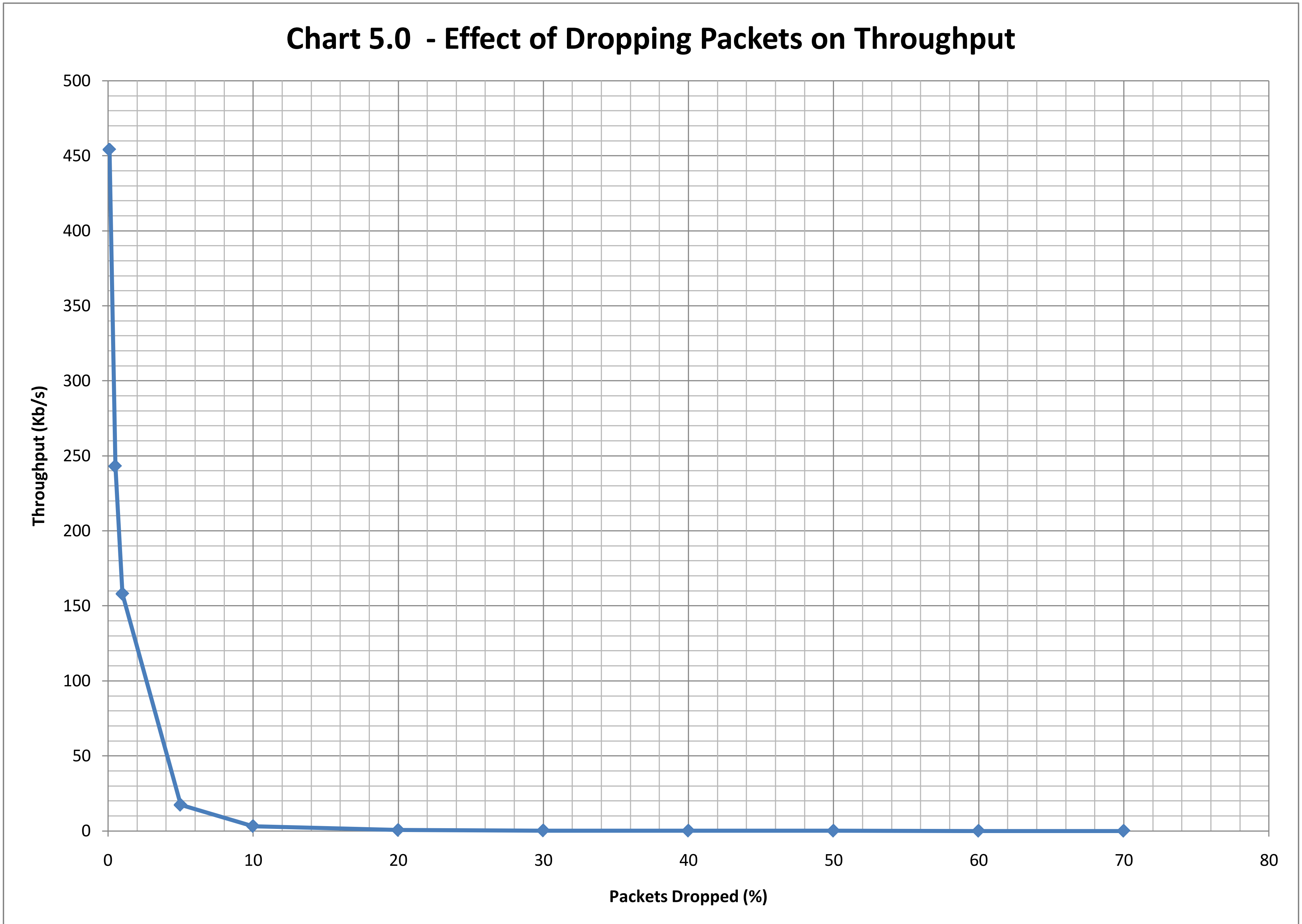
Chart 5.0 - Effect of Dropping Packets on Throughput
Throughput (Kb/s)
500
450
400
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
80
Packets Dropped (%)

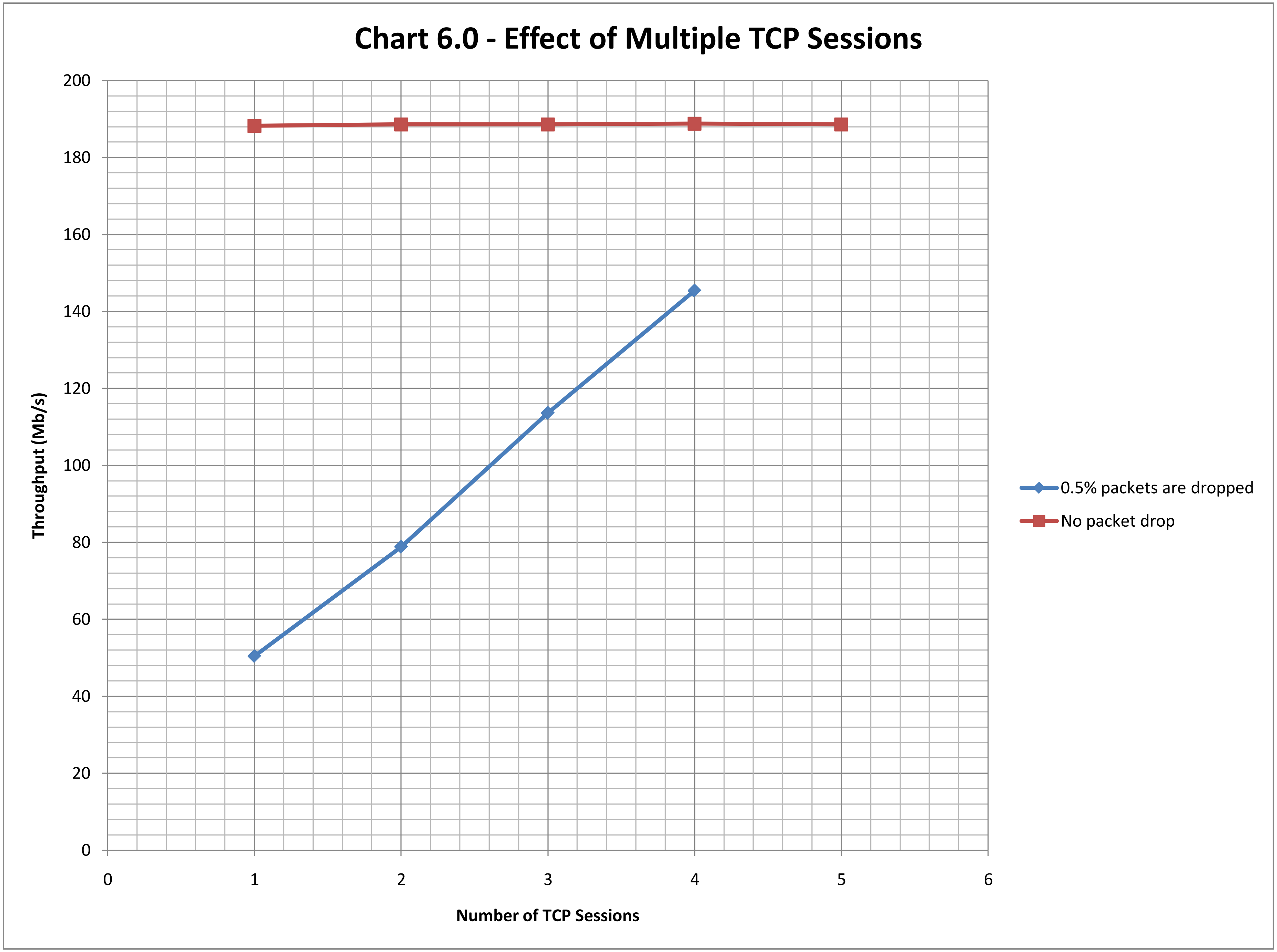
Chart 6.0 - Effect of Multiple TCP Sessions
Throughput (Mb/s)
200
180
160
140
120
100
80
60
40
20
0
0
1
2
3
4
5
6
Number of TCP Sessions
0.5% packets are dropped
No packet drop