

Iperf network measurement results

September 17, 2008

0.1 Introduction

This documents includes the graphical representation of the iperf tests performed between seattle(seattle.krlight.net) and KISTI(daejeon.krlight.net). The tool used was the iperf_tcp_tuning, the tool was used to perform client window sweeps and also parallel stream sweep, to better understand the relation between these parameters and network performance.

0.2 Client Window Sweeps

The aim of this test set was to measure the network performance with respect to the client and server window sizes.

Each graph contains the plotted results of iperf tests with varying client window sizes. The parameters common to all the plotted tests(such as number of threads and server window sizes) are included in the title. The following section will include graphical representation of 2 sets of bidirectional tests.

The tests were repeated to eliminate the possibility of the results being tainted by temporary network conditions.

0.2.1 Client Window sweep run 1

The Figures 1-8 represent the first run

0.2.2 Client Window sweep run 2

The Figures 9-16 represent the second run.

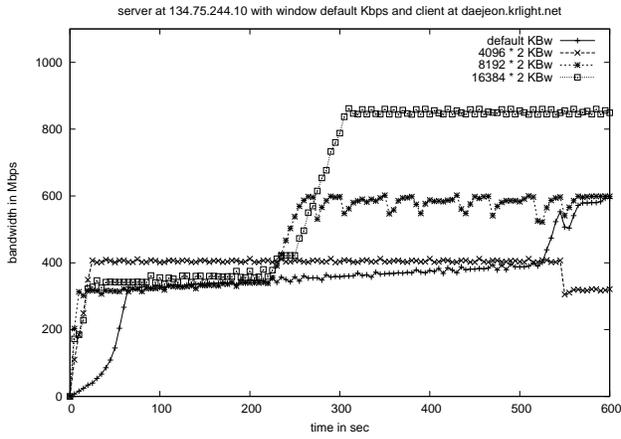


Figure 1: KISTI to Seattle with server window DefaultMb and Single thread.

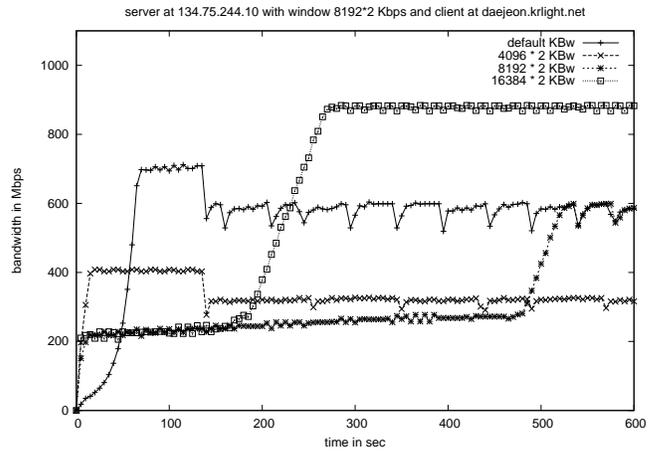


Figure 3: KISTI to Seattle with server window 8*2Mb and Single thread.

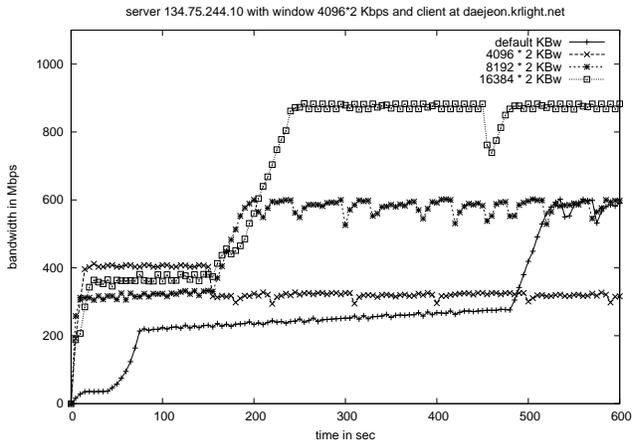


Figure 2: KISTI to Seattle with server window 4*2Mb and Single thread.

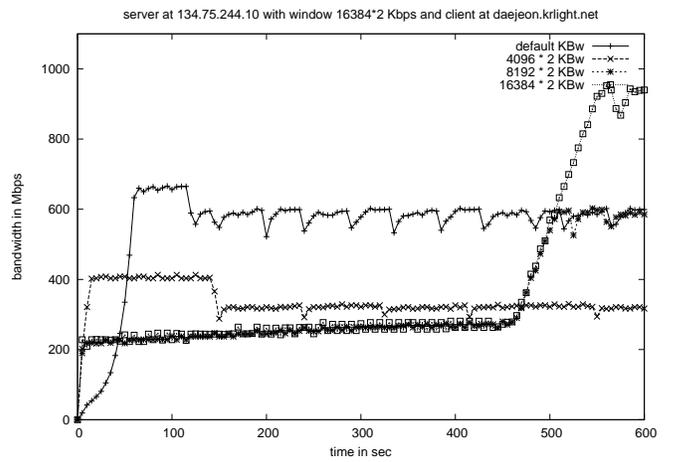


Figure 4: KISTI to Seattle with server window 16*2Mb and Single thread.

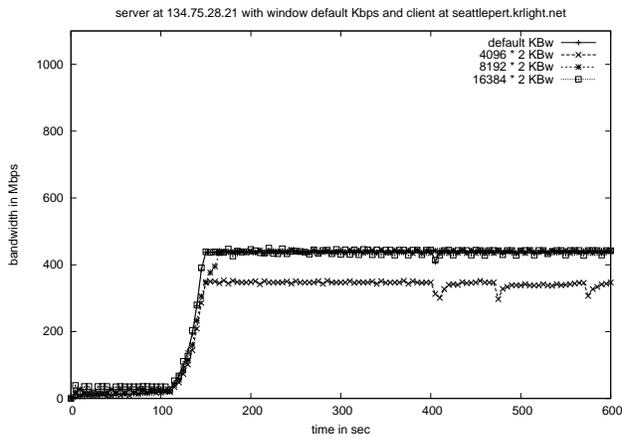


Figure 5: Seattle to KISTI with server window Default Mb and Single thread.

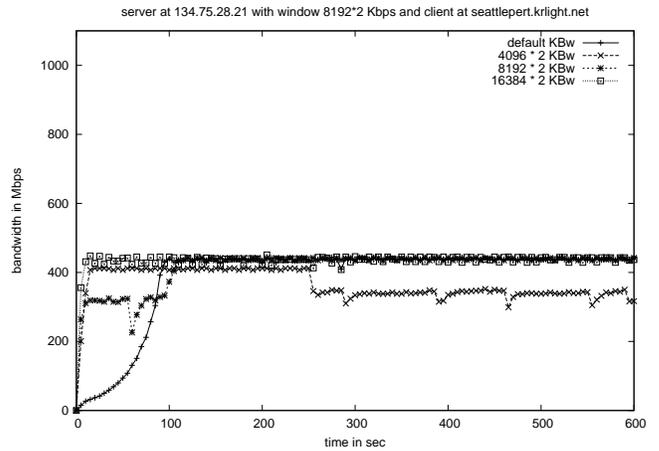


Figure 7: Seattle to KISTI with server window 8*2Mb and Single thread.

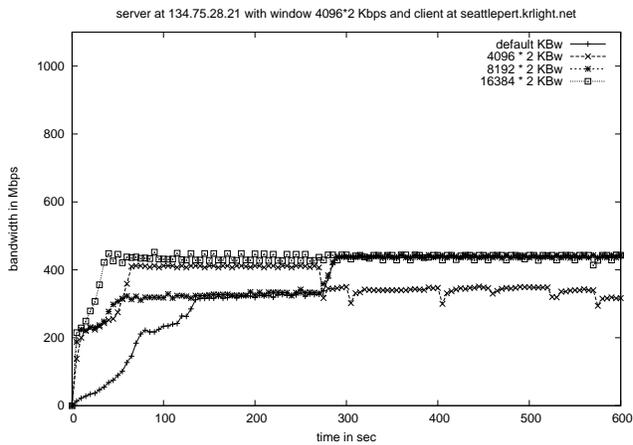


Figure 6: Seattle to KISTI with server window 4*2Mb and Single thread.

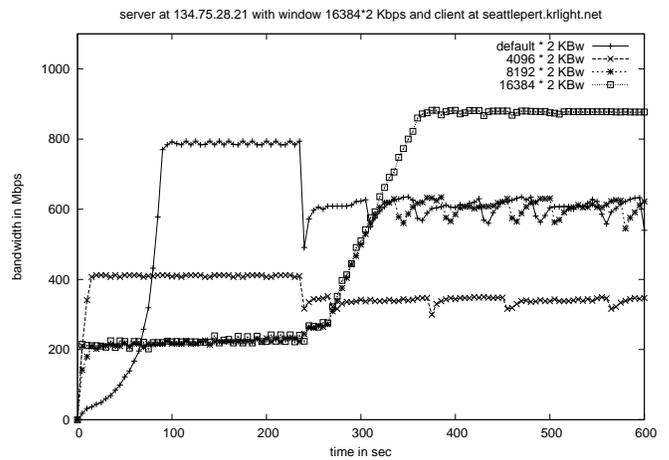


Figure 8: Seattle to KISTI with server window 16*2Mb and Single thread.

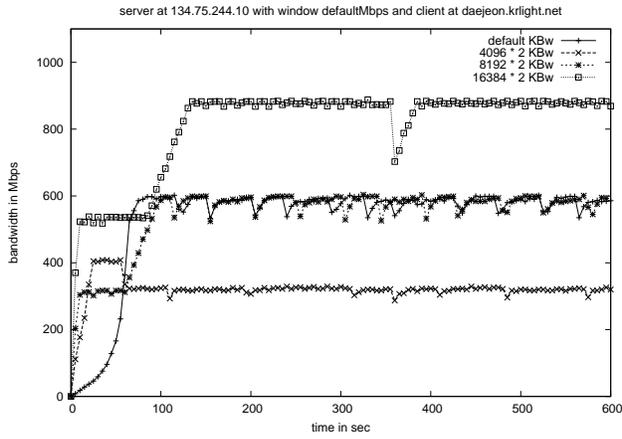


Figure 9: KISTI to Seattle with server window DefaultMb and Single thread.

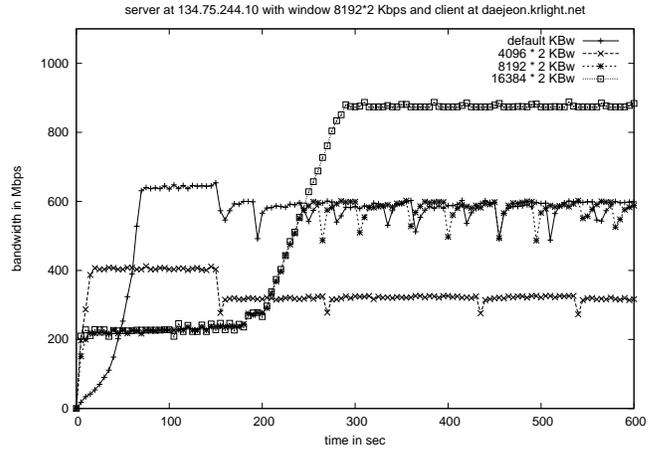


Figure 11: KISTI to Seattle with server window 8*2Mb and Single thread.

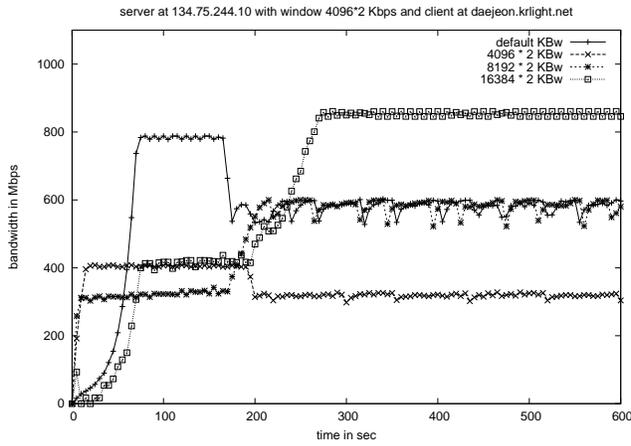


Figure 10: KISTI to Seattle with server window 4*2Mb and Single thread.

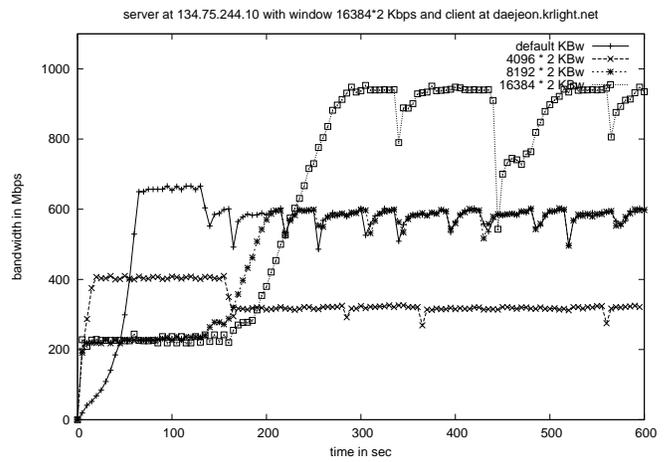


Figure 12: KISTI to Seattle with server window 16*2Mb and Single thread.

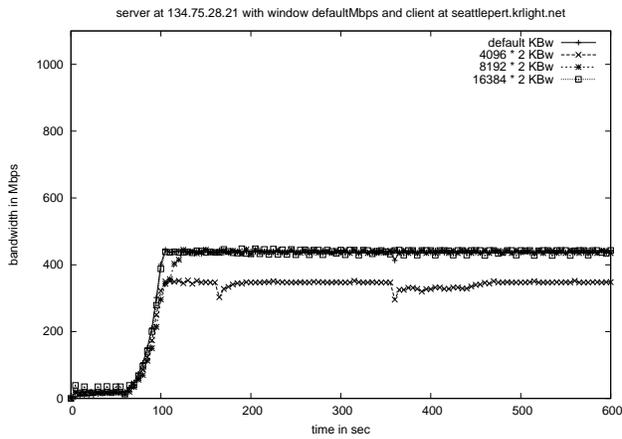


Figure 13: Seattle to KISTI with server window Default Mb and Single thread.

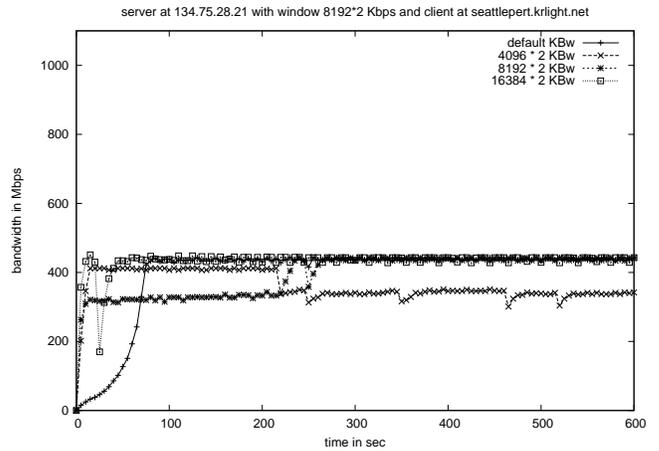


Figure 15: Seattle to KISTI with server window 8*2Mb and Single thread.

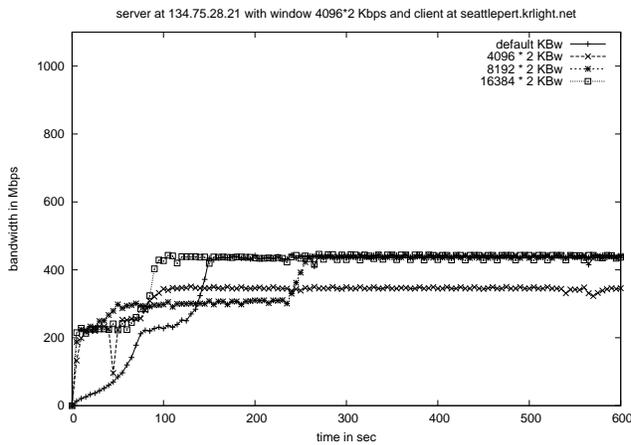


Figure 14: Seattle to KISTI with server window 4*2Mb and Single thread.

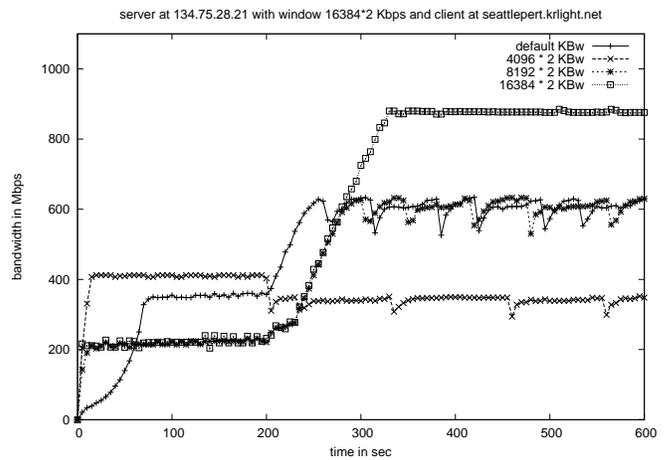


Figure 16: Seattle to KISTI with server window 16*2Mb and Single thread.

0.3 Parallel Stream Sweep

The aim of these tests was to record the network performance with respect to the number of parallel streams used.

Each individual graph represents the data from multiple iperf tests with different combinations of server client windows and number of parallel threads. by comparing the graphs with same server and client window parameters but different number of parallel streams we can analyse TCP performance in terms of number of parallel streams.

e.g. comparing figures 17-20 we can compare the results of iperf tests with 1,2,4 and 8 streams respectively.

0.3.1 Iperf tests from Seattle to KISTI

server with default window

Figures 17 - 20 (pg 7.)

server with window size 4*2 Mbps

Figures 21 - 24 (pg 8.)

server with window size 8*2 Mbps

Figures 25 - 28 (pg 9.)

server with window size 16*2 Mbps

Figures 29 - 32 (pg 10.)

0.3.2 Iperf tests from KISTI to Seattle

server with default window

Figures 33 - 36 (pg 11.)

server with window size = 4*2 Mbps

Figures 37 - 40 (pg 12.)

server with window size = 8*2 Mbps

Figures 41 - 44 (pg 13.)

server with window size = 16*2 Mbps

Figures 45 - 48 (pg 14.)

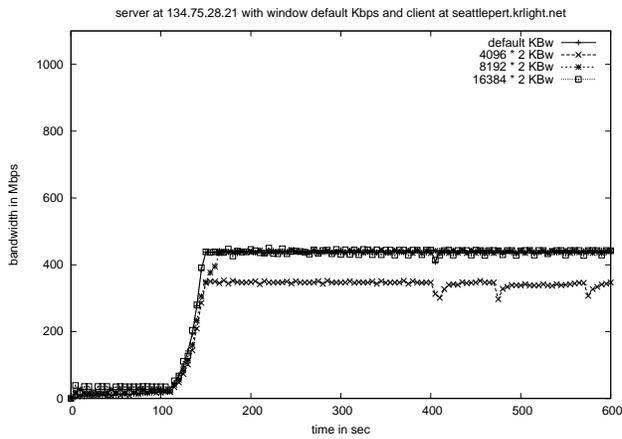


Figure 17: Server at Kisti with default window size and client using single thread

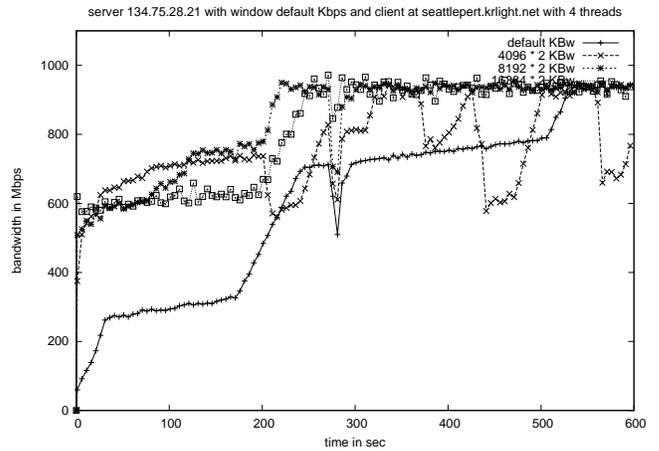


Figure 19: Server at Kisti with default window size and client using 4 parallel threads.

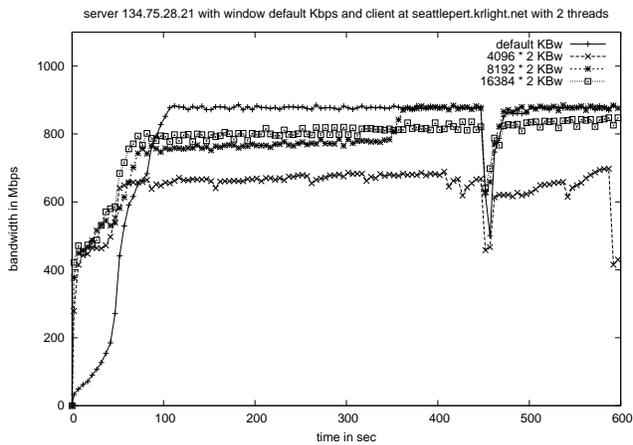


Figure 18: Server at Kisti with default window size and client using 2 parallel threads.

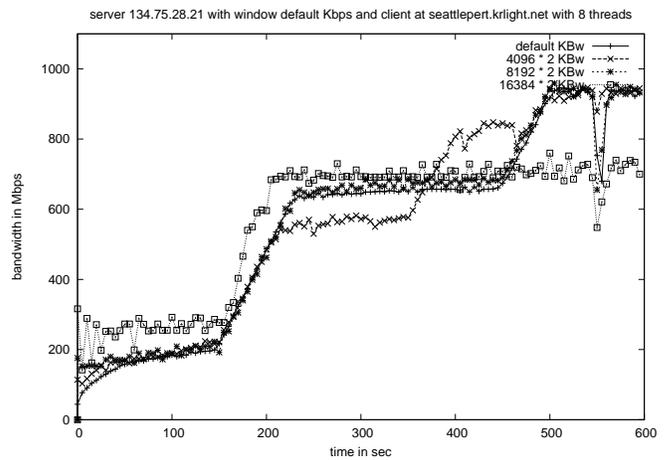


Figure 20: Server at Kisti with default window size and client using 8 parallel threads.

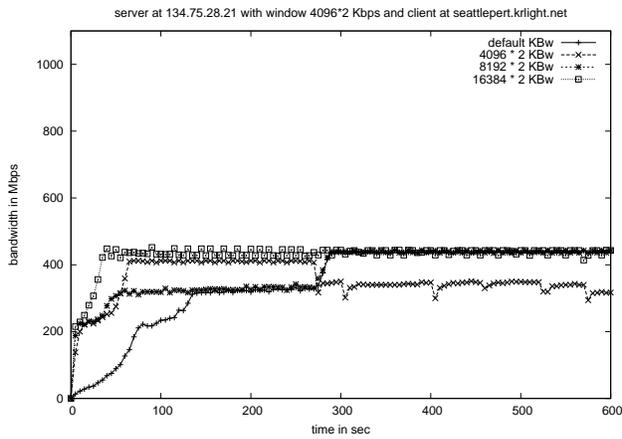


Figure 21: Server at Kisti with window size 4*2Mb and client using single thread

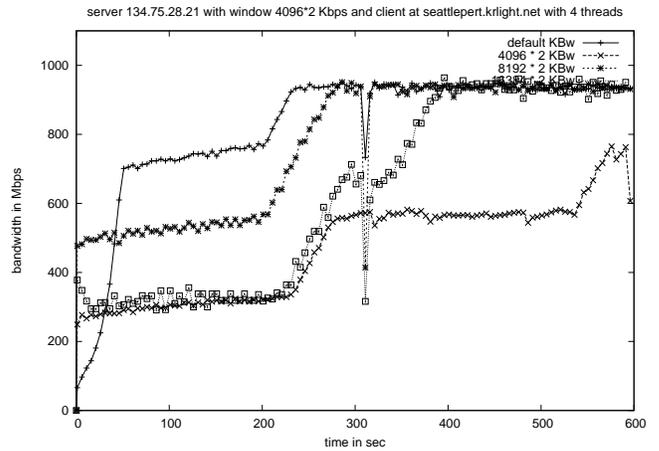


Figure 23: Server at Kisti with window size 4*2Mb and client using 4 parallel threads.

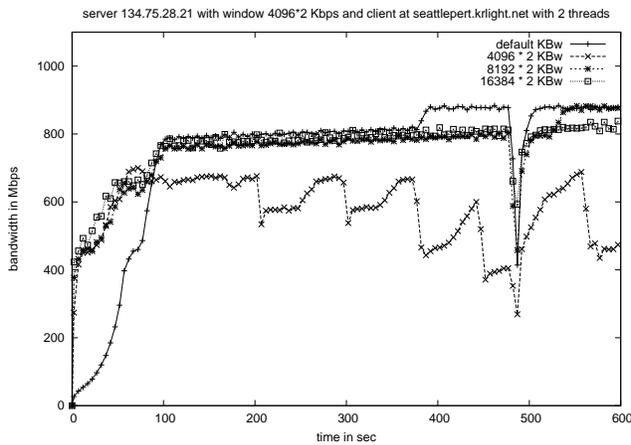


Figure 22: Server at Kisti with window size 4*2Mb and client using 2 parallel threads.

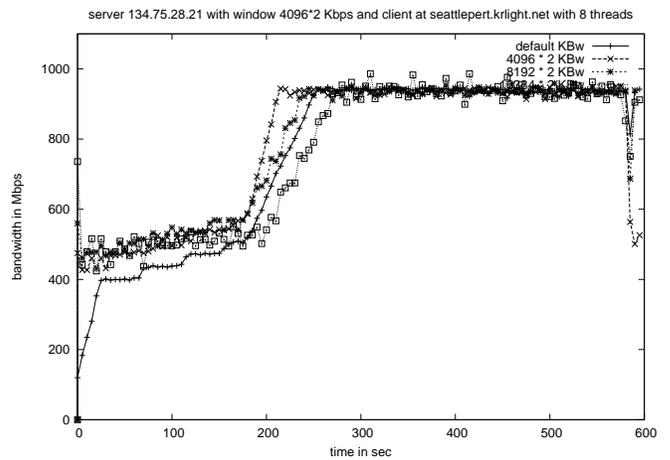


Figure 24: Server at Kisti with window size 4*2Mb and client using 8 parallel threads.

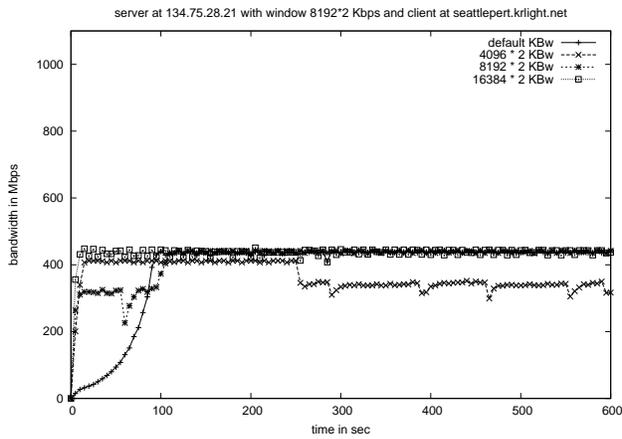


Figure 25: Server at Kisti with window size $8 \times 2m$ and client using single thread

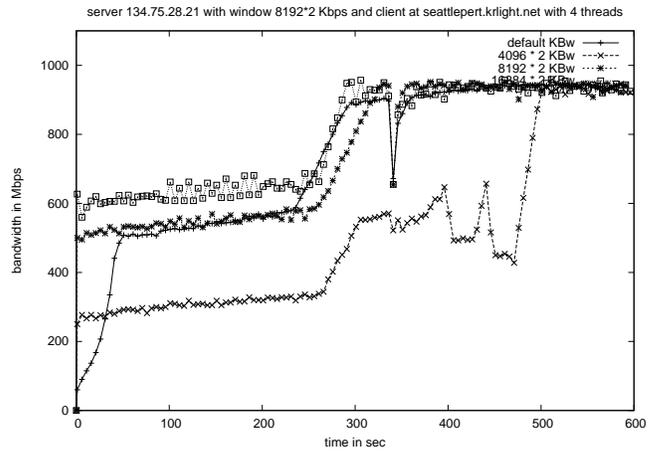


Figure 27: Server at Kisti with window size $8 \times 2m$ and client using 4 parallel threads.

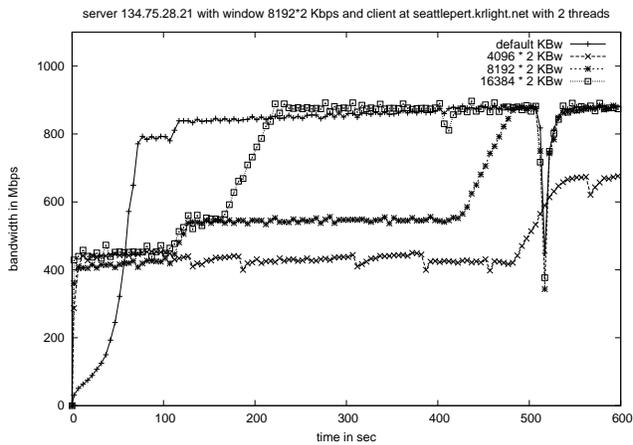


Figure 26: Server at Kisti with window size $8 \times 2m$ and client using 2 parallel threads.

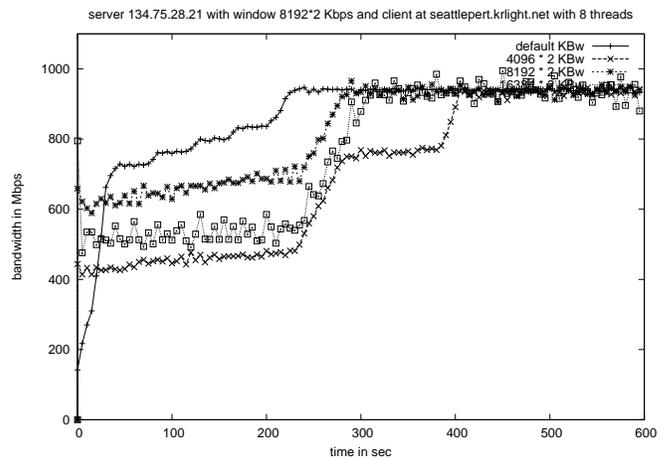


Figure 28: Server at Kisti with window size $8 \times 2m$ and client using 8 parallel threads.

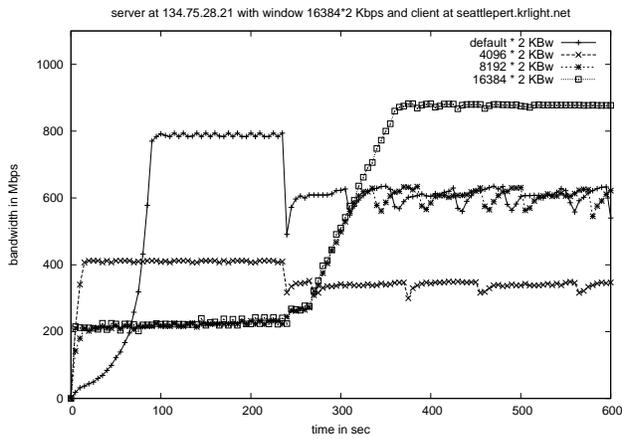


Figure 29: Server at Kisti with window size 16*2m and client using single thread

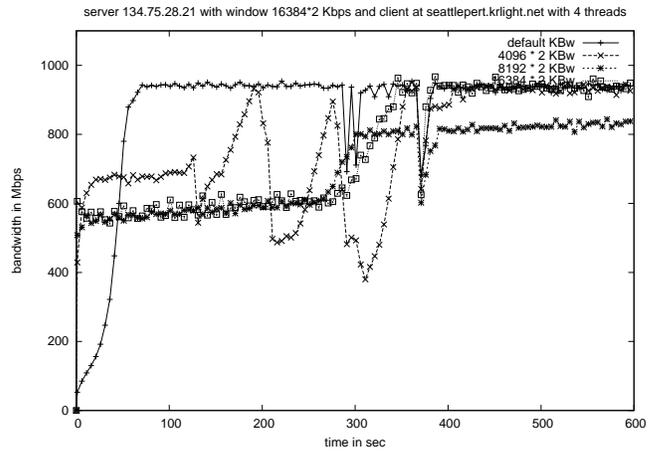


Figure 31: Server at Kisti with window size 16*2m and client using 4 parallel threads.

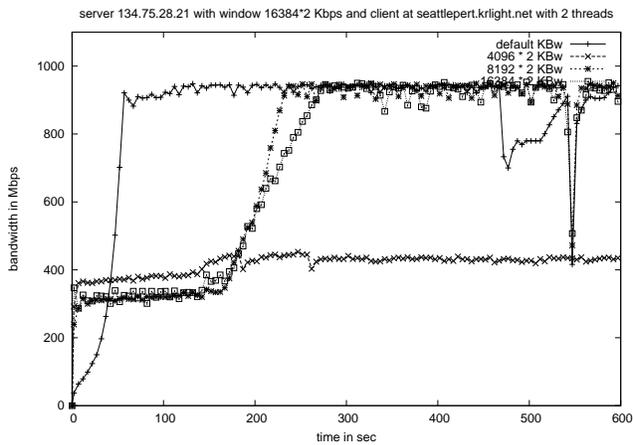


Figure 30: Server at Kisti with window size 16*2m and client using 2 parallel threads.

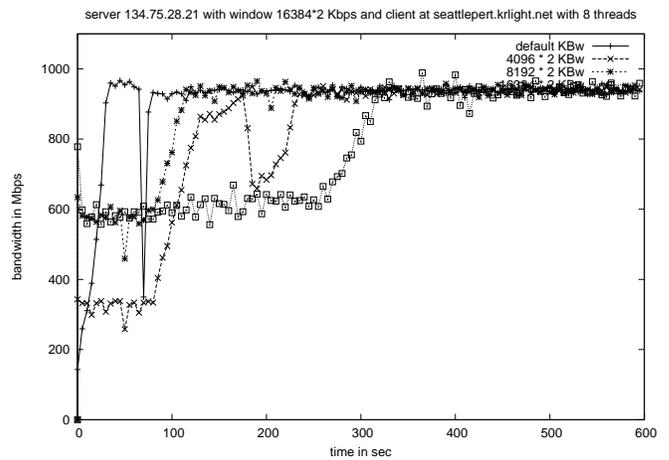


Figure 32: Server at Kisti with window size 16*2m and client using 8 parallel threads.

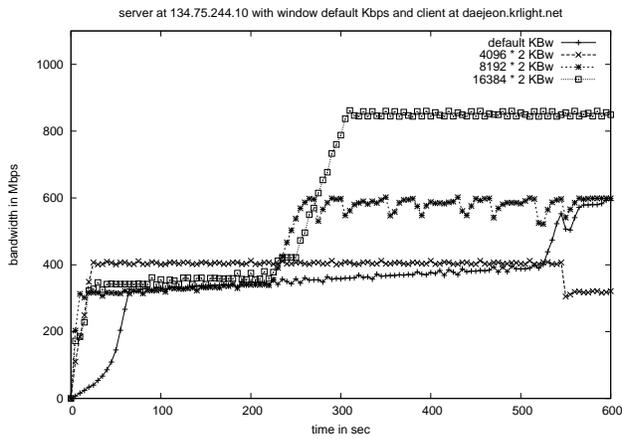


Figure 33: Server at Seattle with default window size and client using single thread

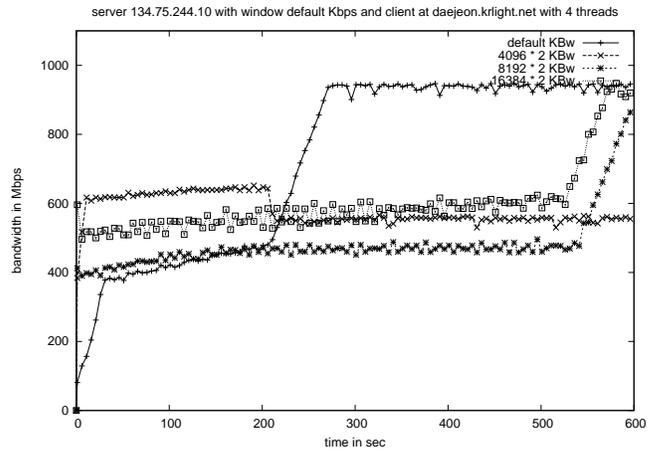


Figure 35: Server at Seattle with default window size and client using 4 parallel threads.

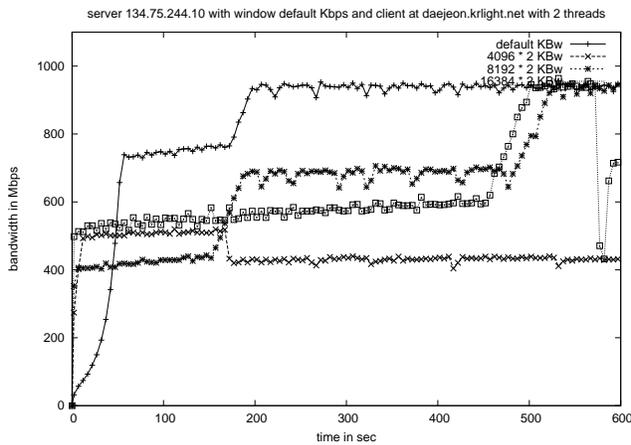


Figure 34: Server at Seattle with default window size and client using 2 parallel threads.

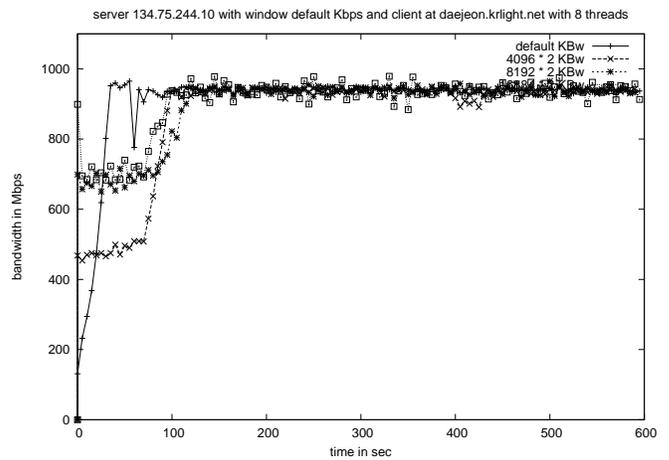


Figure 36: Server at Seattle with default window size and client using 8 parallel threads.

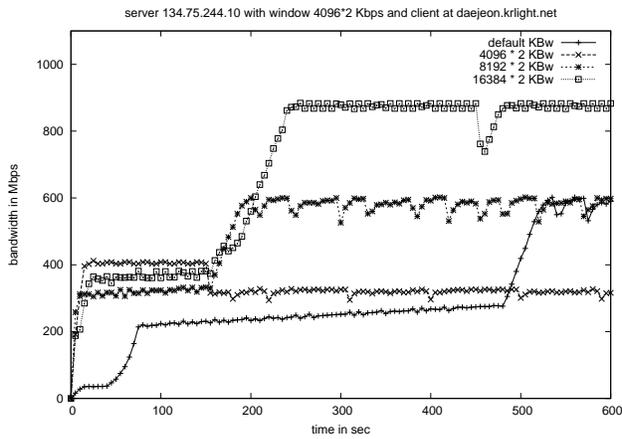


Figure 37: Server at Seattle with 4*2 Mb window size and client using single thread

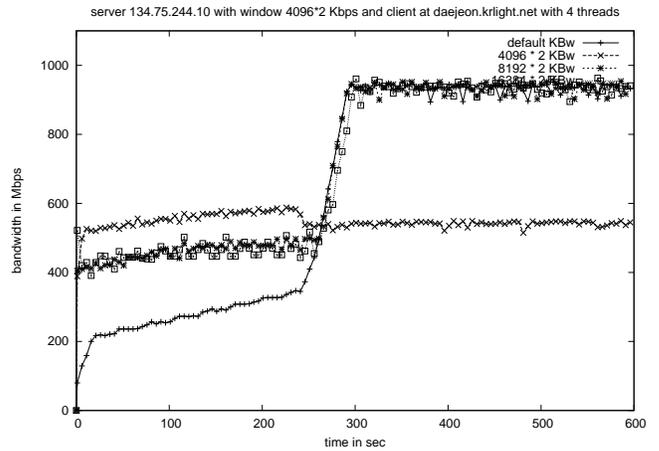


Figure 39: Server at Seattle with 4*2 Mb window size and client using 4 parallel threads.

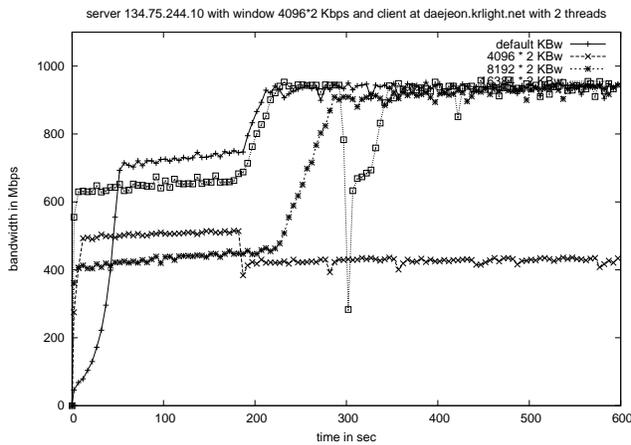


Figure 38: Server at Seattle with 4*2 Mb window size and client using 2 parallel threads.

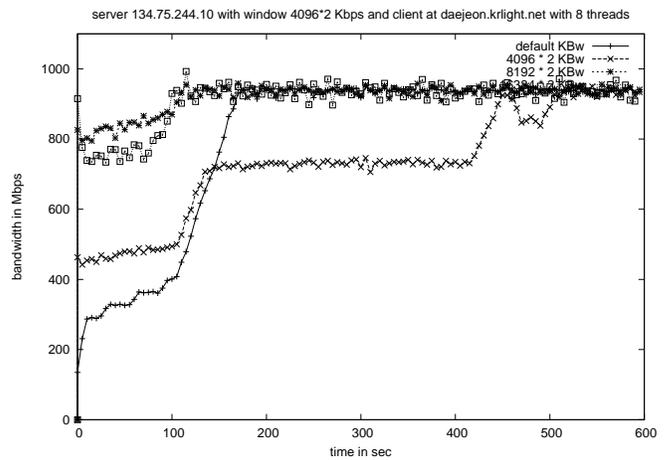


Figure 40: Server at Seattle with 4*2 Mb window size and client using 8 parallel threads.

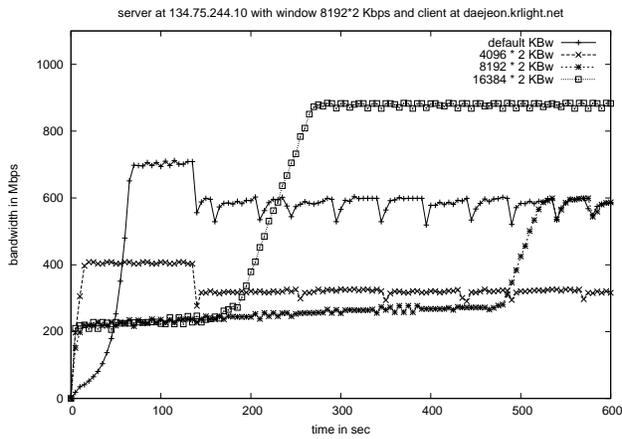


Figure 41: Server at Seattle with $8*2$ Mb window size and client using single thread

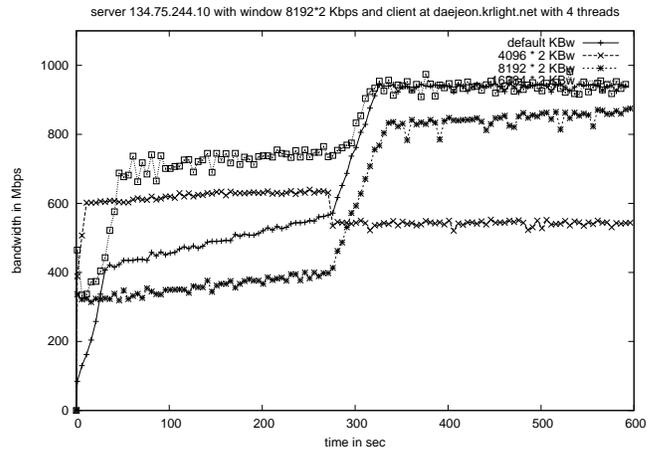


Figure 43: Server at Seattle with $8*2$ Mb window size and client using 4 parallel threads.

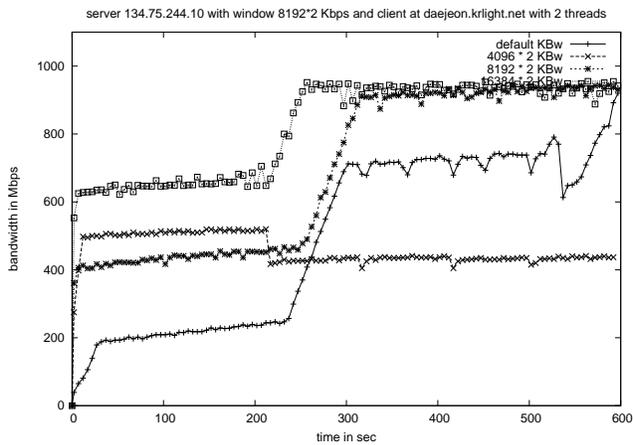


Figure 42: Server at Seattle with $8*2$ Mb window size and client using 2 parallel threads.

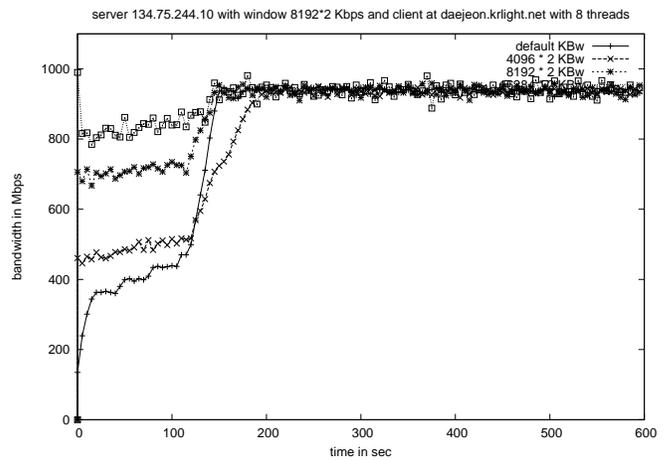


Figure 44: Server at Seattle with $8*2$ Mb window size and client using 8 parallel threads.

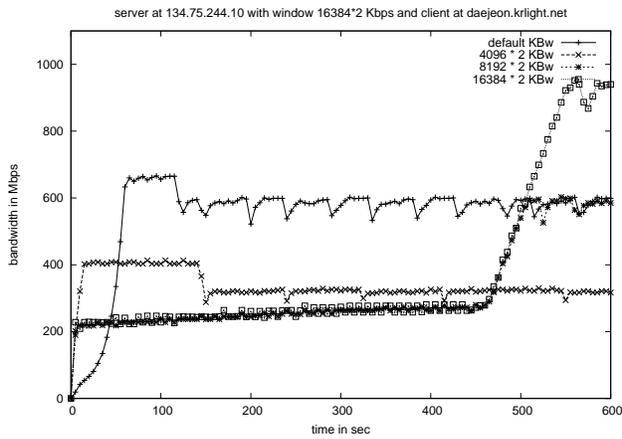


Figure 45: Server at Seattle with 16*2 Mb window size and client using single thread

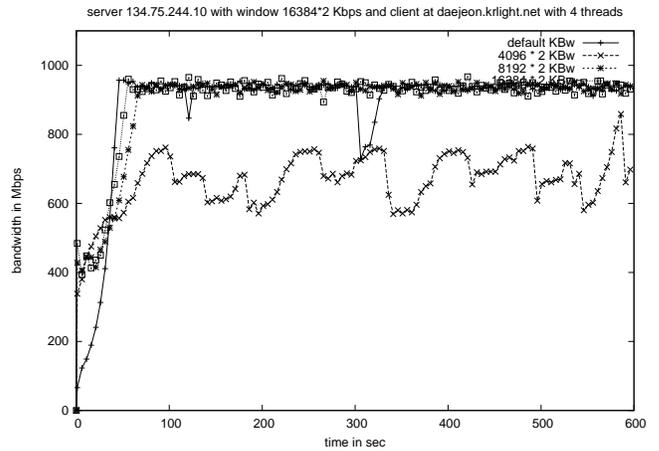


Figure 47: Server at Seattle with 16*2 Mb window size and client using 4 parallel threads.

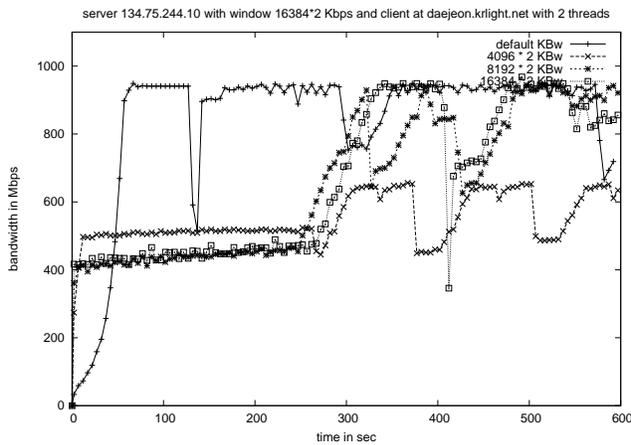


Figure 46: Server at Seattle with 16*2 Mb window size and client using 2 parallel threads.

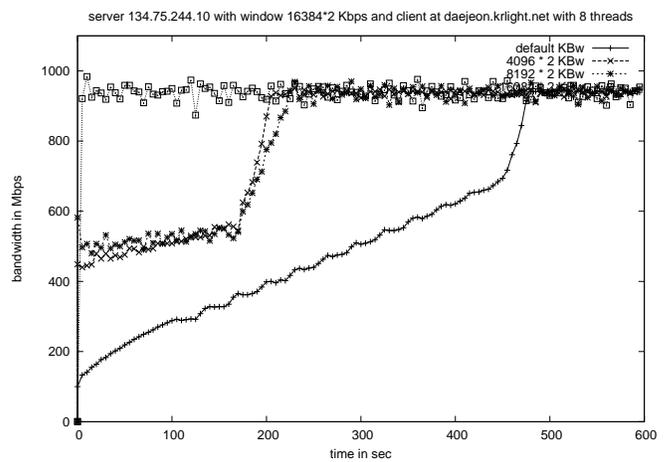


Figure 48: Server at Seattle with 16*2 Mb window size and client using 8 parallel threads.

0.4 Conclusion

Network events.

0.4.1 Delayed ramp-up.

This is seen in almost all the iperf tests, it takes anywhere between 100 to 300 seconds for the iperf flow to reach the maximum bandwidth, the interesting observation is that it does not increase uniformly over the time period but rather stays almost constant, then suddenly ramps up to the maximum. Increasing the window size does seem to have some effect on the delay before the output reaches maximum.

0.4.2 sudden sharp drop in bandwidth within the 400-500s period.