Large MTUs and Internet Performance

David Murray, Terry Koziniec, Kevin Lee, Michael Dixon
School of Information Technology, Murdoch University,
Murdoch 6150, Western Australia, Australia
{D.Murray, T.Koziniec, Kevin.Lee, M.Dixon}@murdoch.edu.au

***Abstract*—Ethernet data rates have increased many orders of magnitudes since standardisation in 1982. Despite these continual data rates increases, the 1500 byte Maximum Transmission Unit (MTU) of Ethernet remains unchanged. Experiments with varying latencies, loss rates and transaction lengths are performed to investigate the potential benefits of Jumboframes on the Internet. This study reveals that large MTUs offer throughputs much larger than a simplistic overhead analysis might suggest. The reasons for these higher throughputs are explored and discussed.**

***Index Terms*—Jumboframe, MTU, TCP, Experiment, Performance**

I. Introduction

Over the last 30 years, data rates on the Internet have progressively increased. Table I shows the evolution of Ethernet standards with increasing rates from 10Mb/s to 100Gb/s and decreasing serialization delay. Ethernet is the de facto Local Area Networking (LAN) technology and is increasingly being used in carrier networks with half of all North American enterprises now deploying carrier Ethernet services [1].

The Maximum Transmission Unit (MTU) specifies the largest packet size, including headers and data payload, that can be transmitted by the link-layer technology. If an end to end connection uses a MTU larger than the link MTU, the packet will either be fragmented, or dropped. To prevent this from occurring, mechanisms exist to discover the MTU of a path over the Internet.

Table I shows that Ethernet speed increases and stagnant MTUs, have reduced the serialization delays of standard 1500 byte frames from $1200\mu s$ to $0.12\mu s$. The large numbers of frames produced by modern Ethernet increases CPU loads [2], [3], [4] and overheads [5]. The CPU and overhead benefits of jumboframes over 1500 byte frames are well understood.

The originality of this study is the experimental exploration of Jumboframes over Wide Area Network (WAN) links. This research is particularly relevant given the deployment of +100Mb/s high speed broadband and FTTN (Fibre To The Node) broadband in many developed nations. The experiments discussed in this paper use default and tuned 1500 and 9000 byte MTU connections in a laboratory testbed.

The remainder of this paper is structured as follows. Section 2 describes the background of Jumboframes. Section 3 performs experiments to evaluate the impact of Jumboframes on Internet links. The implications of the results of these experiments are discussed in section 4. Section 5 discusses why Ethernet MTUs have not scaled as desired.

Technology	Rate	Year	MTU	Serialization Delay
Ethernet	10	1982	1500	$1200\mu s$
Fast Ethernet	100	1995	1500	$120\mu s$
Gig Ethernet	1000	1998	1500	$12\mu s$
10-Gig Ethernet	10,000	2002	1500	$1.2\mu s$
100-Gig Ethernet	100,000	2010	1500	$0.12\mu s$

TABLE I
Ethernet Standards and MTUs

II. Background

This section describes the traditional arguments for and against the use of Jumboframes on the Internet and reviews prior work investigating Jumboframes on WAN links.

A. Arguments for Jumboframes

1) Lower Overheads: Jumboframes lower network overheads because fewer packets, and therefore fewer headers, are required to move data from the source to the destination. Table II shows the size of Physical, Data-link, Network and Transport layer headers in a TCP/IP transaction over Gigabit Ethernet using 1500 byte MTUs and 9000 byte MTUs. Physical layer Inter-Frame Gaps, Start Frame Delimiters and Preambles as well as higher layer Ethernet, IP and TCP headers are included in the calculation.

The Percentage row in Table II compares the time spent transmitting payload data, with the time spent transmitting overheads in Gigabit Ethernet. Using 9000 byte frames, 99% of the time is spent transmitting payload data, with only 1% used for headers. With regular 1500 byte frames, 94.3% of the time is used for transmitting data. Based on this simplistic analysis of overheads, 9000 byte frames should be approximately 4%-5% faster than 1500 byte frames. As IPv6 headers are larger than the IPv4 header, the transferable payload would be further reduced in IPv6 transactions; exacerbating this overhead margin.

2) CPU Loads: With every Ethernet speed increase; routers and end devices must process 10 times more packets, substantially increasing CPU loads [2], [3], [4]. Given today's processor speeds, high speed 10Gb/s links impose a significant burden on CPUs [6]. As IPv6 is more computationally expensive to process than IPv4, this problem may worsen in the future [5].

TOEs (TCP/IP Offload Engines) can reduce CPU loads; offloading packet processing onto the Network Interface Card (NIC). The use of TOE is problematic because changes within

TABLE II
OVERHEAD OF TCP/IP TRANSACTIONS OVER GBIT ETHERNET

	1500 MTU		9000 MTU	
Frame Component	Bytes	Time(μs)	Bytes	Time(μs)
IFG	12	0.096	12	0.096
SFD	1	0.008	1	0.008
Preamble	7	0.056	7	0.056
Eth Hdr	14	0.112	14	0.112
IP Hdr	20	0.16	20	0.16
TCP Hdr	32	0.256	32	0.256
Payload	1448	11.58	8948	71.5
Percentage	-	94.3	-	99.0

TCP, such as congestion control or security amendments, require the NIC (Network Interface Card) firmware to be updated [7]. Consequently, attempts to integrate TOEs into the Linux kernel have been rejected for reasons including security, performance and a lack of IETF RFC compliance [7]. Comparatively, if processing was performed in software, new security or performance extensions will be updated with the Operating System (OS).

Given that TOEs cannot be integrated into the Linux kernel, Jumboframes are an attractive alternative. Numerous studies show that CPU usage is dramatically reduced with Jumboframes [5], [8], [2]. Jumboframes reduce packet overheads, routing decisions, protocol processing and device interrupts [9].

B. Arguments Against the Use of Jumboframes

1) CRC Effectiveness: Despite the overhead and CPU benefits of Jumboframes, drawbacks exist. It has been suggested that the effectiveness of Ethernet's CRC-32 mechanism degrades as the packet size increases beyond 12,000 bytes [2]. This is the subject of debate, with numerous studies suggesting different MTUs for CRC-32's effectiveness [10], [11]. The exact size at which CRC-32 becomes ineffective is beyond the scope of this paper. The widespread use of 9000 byte frames in high performance LAN environments is demonstrative of CRC-32's effectiveness at this size. Only significantly larger packet sizes will need a more robust checksum [11], [12], and thus, further discussion it is beyond the scope of this study.

2) Jitter in Delay Sensitive Applications: Another argument against the use of Jumboframes is that they increase delay and jitter [10]. As 9000 byte packets take six times longer to serialize, they can negatively affect smaller, time sensitive packets queued behind them. For low speed networks, this is a valid argument but for 100Mb/s links and above this argument ceases to be applicable.

A 9000 byte frame, transmitted at 100 Mb/s, is serialized 66% faster than a 1500 byte packet transmitted at 10 Mb/s. Thus, any argument suggesting that voice and video are negatively affected by the transmission of 9000 byte frames over 100 Mb/s links, must also suggest that 1500 byte frames are too big for 10Mb/s links. 1500 byte MTUs are standard for Ethernet, ADSL, DOCSIS links; which frequently carry voice and video.

C. Fragmentation and PMTU Discovery

Using larger MTUs is highly detrimental to performance when it causes fragmentation because every fragment must have an additional set of headers [13]. Given the scenario where 1024 byte segments are being sent and fragmented into 488 byte fragments, an additional set of headers will be added to each of the three fragments. Additional headers are inefficient, however, the argument that large MTUs cause fragmentation is only historically relevant. In 2007 the IETF released a new Path MTU (PMTU) Discovery [14] mechanism to dynamically discover the path's MTU. This new mechanism specifies an alternative way to probe end-to-end MTUs by sending progressively larger packets. This new PMTU Discovery mechanism is considered robust, effective and is enabled by default in the Linux kernel.

D. Jumboframe Performance

Prior research has found that Jumboframes significantly outperform standard 1500 byte frames in WAN environments [6], [15]. Two studies have produced different results.

Makineni et al [6] found that, in tests on Microsoft Windows machines, Jumboframes yield throughput increases of 22%. However, CPU limitations were a factor in this experiment, possibly explaining the superior performance of Jumboframes.

Ravot et al [15] found that jumboframes outperformed standard 1500 byte frames by a factor of fourteen when competing over the same link. When 1500 byte frames and Jumboframes are competing, it is possible that a Random Early Detection (RED) gateway dropping packets could have caused this large performance difference. Ravot et al [15] also concluded that Jumboframes accelerated the congestion window increase by a factor of six compared with standard 1500 byte MTUs [15].

These throughputs are significantly different from the 4%-5% overhead differences; motivating a search to discover why TCP performance is greater than an overhead analysis might suggest. The remainder of this paper investigates these issues experimentally.

III. EXPERIMENTS EVALUATING THE EFFECT OF MTU SIZE ON TCP PERFORMANCE

This paper performs a series of experiments comparing 1500 byte and 9000 byte frames. After the setup is described, the performance of large file transfers under a range of latency and loss conditions is evaluated. Small Internet transactions are also investigated.

A. Experimental Setup

An experiment was performed to investigate the real world performance characteristics of different MTUs over a range of conditions. The real Internet cannot be used because many devices do not support Jumboframes. The experimental setup is shown in Fig 1. All machines were Core 2 series Intel processors running Ubuntu 10.04. The high speed of these PCs kept CPU utilization below 3% and thus the results should not be CPU bound. The network cards were PCI-E based Intel

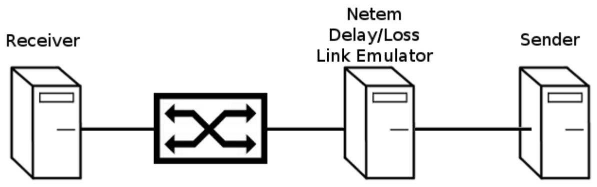

Fig. 1. Experimental Setup

TABLE III
TCP PARAMETERS USED IN THE EXPERIMENT

Setting	Default	Tuned
ifconfig eth0 txqueuelen	100 packets	1000 packets
net.dev_max_backlog	1000 packets	10000 packets
net.ipv4.tcp_sack	1 (On)	1 (On)
tcp_mem	4k 87k 4194k	80530k 80530k 80530k
tcp_rmem	4k 87k 4194k	80530k 80530k 80530k
tcp_wmem	4k 87k 4194k	80530k 80530k 80530k
tcp_congestion_control	cubic	cubic
Kernel Version	2.6.32-28-server	2.6.32-28-server

PRO/1000's. The switch was a Buffalo WZR-HP-G300NH Gigabit switch capable of handling 9000 byte frames. Netem was used to replicate varying levels of delay and loss. All results are based on the throughput from an Apache 2.2.14 server.

Connections utilized 1500 and 9000 byte MTUs. In addition, these downloads were also performed under default conditions, and under 'tuned' conditions. TCP tuning is recommended [15], [16] because a large BDP (Bandwidth Delay Product) is required to to "fill the pipe" of a 1Gb/s Internet link. TCP grows a sliding window which specifies how many packets can be released unacknowledged from the sender at any one time. This window must be able to grow to the size of the BDP. The BDP can be calculated using Eq 1.

$$BDP = Speed(bytespersec) * RTT \qquad (1)$$

The maximum window size and other TCP variable are shown in Table III.

B. Large File Transfer Results

1) Latency - Results: The initial experiment investigated the performance of Jumboframes over a range of latencies. These results were obtained using the topology shown in Fig 1. With 0ms of added latency, no difference was discernible between default and tuned TCP connections. The primary reason is because the TCP window is not the limiting factor. At 0ms, the Jumboframe connection was transferring data at 994Mb/s. Comparatively, standard 1500 byte frames were transferring data at 896 Mb/s. In this scenario, Jumboframes are performing approx 11% faster than standard 1500 byte frames. This is faster than the 4-5% projected by overhead analysis.

As the latency increases above 10ms, the TCP window size becomes the limiting factor. The default 9000 byte and 1500 byte connections are identical in terms of TCP performance. The tuned TCP connections are able to maintain their peak

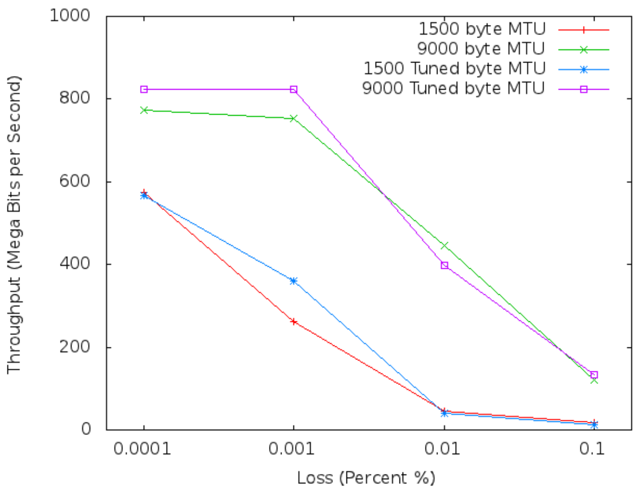

Fig. 2. Throughput achieved using default/tuned and 1500/9000 byte MTUs, 10ms RTTs and loss rates of 0.0001%-0.1%

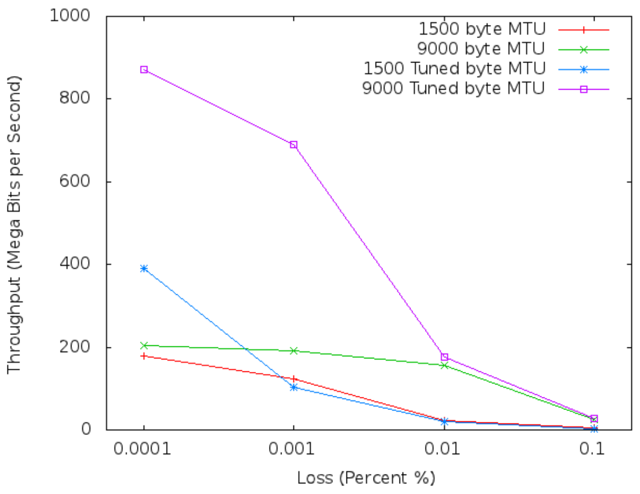

Fig. 3. Throughput achieved using default/tuned and 1500/9000 byte MTUs, 100ms RTTs and loss rates of 0.0001%-0.1%

throughputs until 100ms. Above 100ms the 9000 byte transfer is less affected by latency.

2) Loss - Results: The previous experiment assumed zero congestion or delay based losses over the Internet. This is an unrealistic assumption as real Internet links drop packets due to congestion.

Figs 2, 3 and 4 show the performance of different MTUs and levels of tuning. A number of trends are notable. Firstly, as the loss rate increases, throughputs decrease. This is because every lost packet is interpreted by TCP as a congestion event, dramatically reducing the congestion window [17]. High latencies, coupled with high loss rates, have huge impacts on performance due to the duration required to recover lost packets and begin rebuilding the TCP window [18].

Compared with the previous, purely latency based experiment, the introduction of packet loss widens the performance margin between 1500 byte and 9000 byte frames. In many

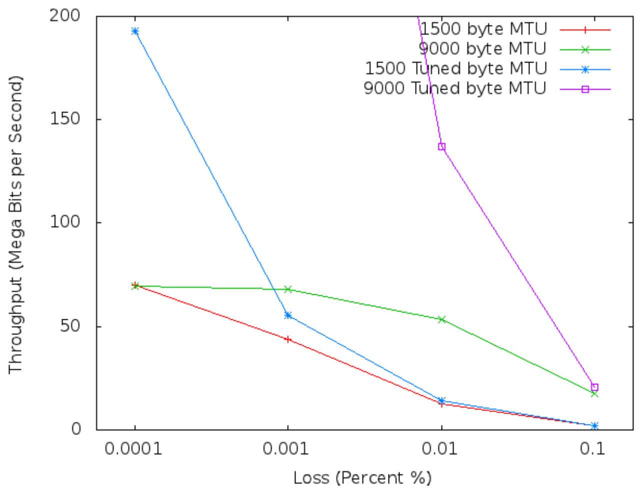

Fig. 4. Throughput achieved using default/tuned and 1500/9000 byte MTUs, 300ms RTTs and loss rates of 0.0001%-0.1%. The scale in this figure has been adjusted to demonstrate the performance advantages under default conditions

of the tested scenarios, throughputs attained by Jumboframes were double the throughputs achieved with 1500 byte frames.

At very low loss rates, the affect of TCP tuning plays a large role in the performance. Evidently, 9000 byte packets, combined with a tuned TCP connection, offer huge performance benefits. This is seen in Figs 2, and 3. The extent of this performance gap hides the advantages of jumboframes seen even under default conditions. In Fig 4, the scale has been adjusted to more clearly demonstrate the performance benefits of 9000 byte frames under default conditions.

C. *Short Lived Transfer - Results*

The experiments above demonstrate the advantage of Jumboframes for large file transfers, however, many flows on the Internet are short and transfer only a small amount of data. Applications such as web servers often open many small flows for every host. Email attachments are also usually limited to a few Megabytes. Therefore, it is important to investigate how Jumboframes perform over small transactions. To better define the size of Internet TCP flows, an experiment was performed on google.com.au, au.yahoo.com, facebook.com and wikipedia.org. Wireshark was used to capture HTTP sessions to each of these websites.

Table IV shows the number of TCP flows opened in one page hit. The min, max, mean and sum of the flows are also shown in Kilobytes. The Facebook and Google pages are quite sparse compared with the significantly more media rich Yahoo and Wikipedia (the English page) sites. Evidently, the size and number of transactions varied between webpages, but the average flow size is very small. Radhakrishnan [19] et al, concurs with this finding, stating that most web objects are relatively small with mean and median sizes of 7.3 KB and 2.4 KB respectively.

These results were used as a basis for the size of our short lived transfer tests. The average TCP transaction size, across

TABLE IV
AN ANALYSIS OF TCP FLOWS FOR THE MOST POPULAR WEBSITES ON THE INTERNET. RESULTS SPECIFIED IN KB.

Website	Num flows	Min Size	Max Size	Sum	Mean
Google	4	0.25	55.53	103.28	25.82
Yahoo	17	0.51	285.63	778.63	45.80
Facebook	7	4.97	59.81	185.98	26.57
Wikipedia	20	0.74	36.22	242.06	12.10

the four web sites was 31371 Bytes or 30.6 Kilobytes. The largest transaction size was 315670 Bytes or 308 Kilobytes. The transfer time of a 1048576 Byte or 1 Megabyte file was also tested. In all of these scenarios, the TCP window would still be growing at the completion of the file transfer. Thus, this experiment would test the growth or acceleration of the TCP's congestion window. The results of these experiments, performed over a range of latencies, are shown in Fig 6 and 7.

Numerous studies [20], [19] propose mechanisms to speed up short TCP transactions. The experimental results in this paper reveal that large MTUs are effective for this purpose. In Fig 6 and 7, Jumboframes completed the transfer in less than half the time of the 1500 byte transaction demonstrating the significant advantages of Jumboframes in small TCP transactions.

IV. DISCUSSION

The experiments in Section 3 found that Jumboframes outperform 1500 byte frames in both large and small file transfer scenarios. The two reasons that explain Jumboframe performance in these environments are: superior resistance to packet loss and faster TCP Growth

A. *Resistance to Packet Loss*

Packet loss may be caused by congestion or link problems, but all dropped packets are interpreted as congestion by TCP. These packet losses cause multiplicative decreases in TCP's window size. Equation 2 was proposed by Mathis in their seminal paper "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm" [21]. Note that equation 2 uses MSS rather than MTU, but the MSS is simply the MTU minus the IP and TCP headers. This equation can be used to describe how packet loss affects throughputs on a WAN link. This equation also states that: if packet loss is limiting the connection, the throughput can be doubled by doubling the packet size [2]. This is one of the frequently missed arguments concerning packet size.

$$Throughput <= 0.7 \times \frac{MSS}{RTT\sqrt{PLoss}} \qquad (2)$$

B. *Faster TCP growth*

Jumboframes also accelerate the growth of a TCP window. When a TCP sender starts transmitting, the ICW (Initial Congestion Window) used is based on the on the guidelines specified in RFC 5681 [22]. According to this RFC [22], the

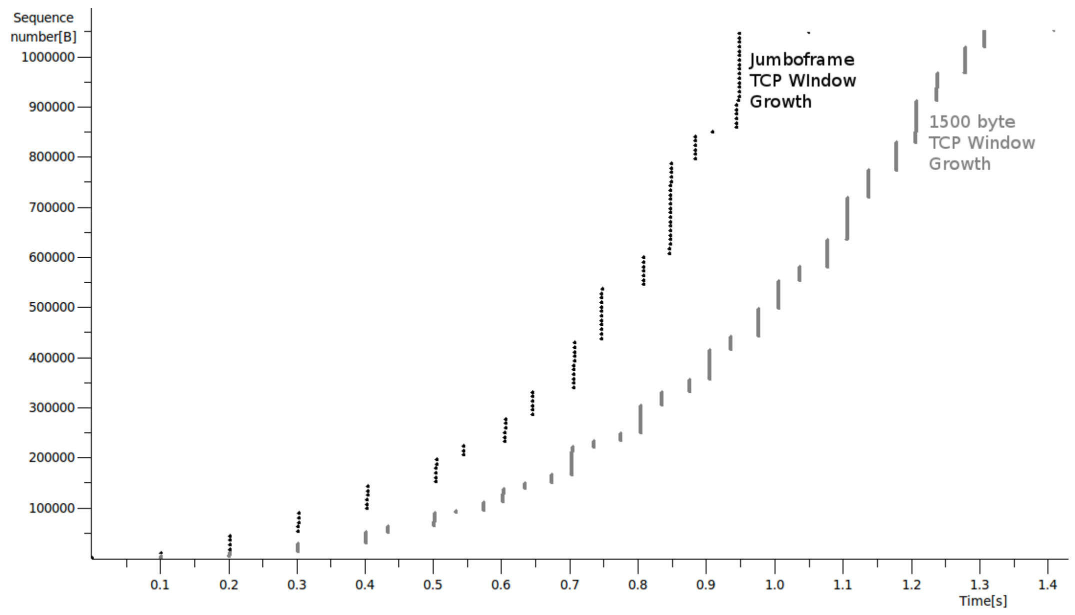

Fig. 5. The growth rate of 1500 byte and 9000 byte Transfers

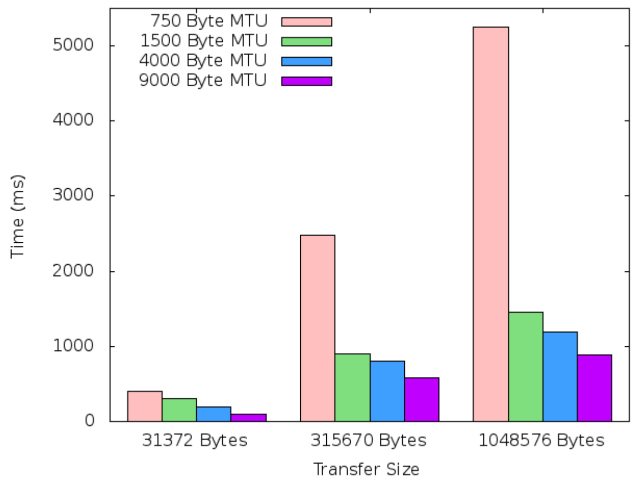

Fig. 6. Completion time of 31372, 315670 and 1048576 byte transfers with a 100ms RTT

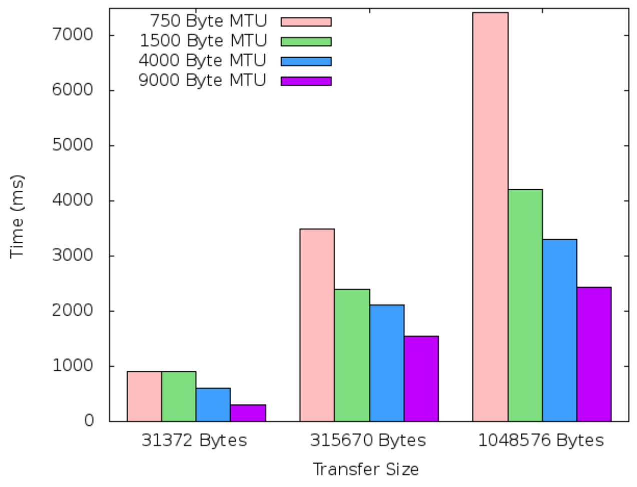

Fig. 7. Completion time of 31372, 315670 and 1048576 byte transfers with a 300ms RTT

ICW for a 9000 byte MTU connection will be set at the size of two segments (18,000 bytes). The ICW for a 1500 byte MTU connection be three segments (4,500 bytes).

After sending these initial segments, TCP will be in a phase known as slow start. During this phase the TCP sender increments the congestion window by one segment size for every new TCP acknowledgement [22]. This phase will continue until a packet is lost or the congestion window exceeds the slow start threshold. At this stage, the congestion window will be reduced and the TCP sender will enter the congestion avoidance phase. Congestion avoidance is a period of linear TCP growth. During the congestion avoidance phase, the congestion window is incremented by one full sized segment per RTT (Round-Trip Time) [22]. These rules indicate that growth during congestion avoidance will be six times faster for 9000 byte frames. To summarise, jumboframe connections will start with a larger congestion window. Additionally, as jumboframe segments are approximately 6 times larger than a standard segment, the window grows faster.

To visualize the accelerated window increase of jumboframe transfers, downloads of a 1MB file transfer over a 100ms link were captured in Wireshark. Fig 5 shows the “Stevens” graph of the 1500 byte and 9000 byte transfer of a 1MB file. “Stevens” graphs show the progression of TCP sequence numbers against time. Fig 5 demonstrates the increased acceleration of jumboframes during the exponential slow start phase.

There are many different schemes [20], [19] being proposed to speed up the initial stages of TCP. The use of 9000 byte jumboframes lessen the need ICW modifications currently being proposed [23].

V. Jumboframe Support

IP supports large packet sizes. IPv4 supports packet sizes up to 65,535 bytes. IPv6 supports packets up to 4,294,967,295 bytes. Evidently, the restriction is the Physical/Data-link layer technologies, such as Ethernet. Jumboframe support has existed in some Gigabit and 10-Gigabit Ethernet switches for some time. Many of the Intel networking cards support 9000+ byte frames. Jumboframe support is also inexpensive. Home user switches, such as the Buffalo WZR-HP-G300NH Gigabit switch, support Jumboframes.

Jumboframe transfers require support for that packet size end-to-end across all the link layer technologies. The adoption problems are similar to those of IPv6. Until support is available end-to-end, there is little benefit to adopt this technology. Despite healthy debate [11], IEEE Ethernet standardization groups have not mandated that 802.3 compliant equipment support frames sizes significantly larger than 1500 bytes. It is simply undesirable for the vendors of 802.3 equipment to have it mandated in modern equipment [11]. If Jumboframes were mandated in Ethernet, it would break some compatibility with legacy Ethernet equipment. For many, compatibility is more important than performance.

VI. Conclusion

This paper described the traditional arguments made for and against the use of Jumboframes. Experiments demonstrated the performance of Jumboframes in large file transfers. Optimal performance was obtained using a tuned TCP stack and Jumboframes, however, even under default conditions, Jumboframes still provided significant performance benefits.

A sub-study on web transactions with four popular websites found that large numbers of small TCP transactions were opened. Using the size of these transactions as a guide, a second round of performance tests were performed. For small transactions, jumboframe connections completed significantly faster.

The reasons for the superior performance of Jumboframes were discussed and demonstrated. A paper by Mathis [21] shows that 9000 byte jumboframe connections are more resistant to packet loss. By investigating RFC 5681 [22], it is also clear that Jumboframes begin a TCP transaction with a larger ICW. Furthermore, TCP's rules [22] also accelerate the growth of TCP windows in both slow start and congestion avoidance phases. Wireshark 'Stevens' graphs were used to visualize these window growth differences.

The CPU and overhead problems of 1500 byte frames and data rate increases will worsen in the future. This problem is well known. The originality of this work is the experimental exploration and investigation into jumboframe throughputs for Internet transfers. It is hoped that this paper encourages further exploration, discussion and experimentation with Jumboframes.

References

[1] P. Sayer, "Market Overview: US Ethernet Services," Forrester Research, 2010.

[2] P. Dykstra, "Gigabit Ethernet Jumbo Frames, And why you should care," http://sd.wareonearth.com/p̃hil/jumbo.html, 1999.

[3] A. Foong, T. Huff, H. Hum, J. Patwardhan, and G. Regnier, "TCP Performance Re-visited," in *In IEEE International Symposium on Performance of Systems and Software*, 2003, pp. 70–79.

[4] W. Rutherford, L. Jorgenson, M. Siegert, P. V. Epp, and L. Liu, "16000-64000 B pMTU experiments with simulation: The case for super jumbo frames at Supercomputing '05," *Optical Switching and Networking*, vol. 4, no. 2, pp. 121 – 130, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B7GX5-4MD46C3-2/2/d52a67bbbed98b275d31fda645473ef5

[5] N. Garcia, M. Freire, and P. Monteiro, "The Ethernet Frame Payload Size and Its Effect on IPv4 and IPv6 Traffic," in *Information Networking, 2008. ICOIN 2008. International Conference on*, jan. 2008, pp. 1 –5.

[6] S. Makineni and R. Iyer, "Architectural Characterization of TCP/IP Packet Processing on the Pentium; M Microprocessor," in *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, ser. HPCA '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 152–. [Online]. Available: http://dx.doi.org/10.1109/HPCA.2004.10024

[7] L. Foundation, "TOE," www.linuxfoundation.org/collaborate/workgroups/networking/toe, 2009.

[8] R. Hughes-Jones, P. Clarke, and S. Dallison, "Performance of 1 and 10 Gigabit Ethernet cards with server quality motherboards," *Future Gener. Comput. Syst.*, vol. 21, pp. 469–488, April 2005. [Online]. Available: http://dx.doi.org/10.1016/j.future.2004.10.002

[9] R. Stevens, *TCP/IP illustrated (vol. 1): the protocols*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[10] Chelsio, "Ethernet Jumbo Frames, The Good, the Bad and the Ugly," www.chelsio.com/assetlibrary/solutions/Chelsio_Jumbo_Enet_Frames.pdf, 2011.

[11] M. Mathis, "Arguments about mtu," http://staff.psc.edu/mathis/MTU/arguments.html, 2011.

[12] R. Jain, "Error characteristics of fiber distributed data interface (FDDI) ," *Communications, IEEE Transactions on*, vol. 38, no. 8, pp. 1244 –1252, aug 1990.

[13] C. Kent and J. Mogul, "Fragmentation Considered Harmful," in *In ACM SIGCOMM*, 1987, pp. 390–401.

[14] M. Mathis and J. Heffner, "Packetization Layer Path MTU Discovery," RFC 4821, 2007.

[15] M. Ravot, Y. Xia, D. Nae, X. Su, H. Newman, and J. Bunn, "A Practical Approach to TCP High Speed WAN Data Transfers," in *Proceedings of PATHNets 2004*. San Jose, CA, USA: IEEE, 2004.

[16] M. Mathis, R. Reddy, and J. Mahdavi, "Enabling High Performance Data Transfers System Specific Notes for System Administrators," http://www.psc.edu/networking/projects/tcptune/, 2011.

[17] M. Hassan and R. Jain, *High Performance TCP/IP Networking: Concepts, Issues, and Solutions*. Prentice-Hall, 2003.

[18] W. Feng, J. Hurwitz, H. Newman, S. Ravot, R. Cottrell, O. Martin, F. Coccetti, C. Jin, X. Wei, and S. Low, "Optimizing 10-Gigabit Ethernet for Networks of Workstations, Clusters, and Grids: A Case Study," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, ser. SC '03. New York, NY, USA: ACM, 2003, pp. 50–. [Online]. Available: http://doi.acm.org/10.1145/1048935.1050200

[19] S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan, "Tcp fast open," in *Proceedings of the 7th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2011.

[20] Michael and Scharf, "Comparison of end-to-end and network-supported fast startup congestion control schemes," *Computer Networks*, vol. 55, no. 8, pp. 1921 – 1940, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128611000491

[21] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 67–82, July 1997. [Online]. Available: http://doi.acm.org/10.1145/263932.264023

[22] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681, 2009.

[23] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," IETF Draft, 2011.