JumboGen: Dynamic Jumbo Frame Generation for Network Performance Scalability

David Salyers, Yingxin Jiang, Aaron Striegel, Christian Poellabauer
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN. USA 46556
{dsalyers,yjiang3,striegel,cpoellab}@nd.edu

ABSTRACT

Network line speeds have increased at a significant rate. Unfortunately, network performance has not been able to keep pace with increases in line speed. This is due to the majority of packets being less than or equal to 100 bytes in addition to network routers not being able to scale well with the increased number of packets. In this paper we present our solution, JumboGen, an approach that will allow for a higher utilization of larger packet sizes on a domain-wise basis. Through simulations and experimentation, we show that the dynamic creation of jumbo packets decreases the number of packets processed by core routers and does not have an adverse impact on link utilization or fairness. The final result of JumboGen is a reduction in the number of packets seen by core routers which directly improves network scalability.

Categories and Subject Descriptors

C.2.6 [**Internetworking**]: Routers

General Terms

Design, Performance

Keywords

JumboGen, Dynamic Jumbo Frames, Router Scalability

1. INTRODUCTION

Network (LAN and WAN) speeds are continuously increasing and are such that computer networks can now easily transfer billions of bits of information every second. However, while line speeds are increasing, the effective MTU size has remained stagnant at 1500 bytes due to the dominance of Ethernet at the edge of the network. The stagnant MTU, in addition to the fact that the majority of packets transferred in the Internet are small [1], has lead to a situation where the number of packets needed to be processed by routers scales roughly linearly with line speed. Furthermore, conservative estimates have the Internet traffic roughly doubling every year [2], with the expectation that this trend will continue. Unfortunately, it has also been shown that router capacity only doubles every eighteen months [3].

For example, consider a 10 Mb/s network that can handle N packets every second. If the network is upgraded to 10 Gb/s the routers would be expected to handle $1000N$ packets. As there is a fixed amount of overhead processing per packet, it is a challenge for CPUs to scale with increasing network speeds [4]. The traditional solution of simply adding more hardware to solve the problem is not sufficient, as current router designs typically suffer from at least one of two different problems; unpredictable performance[1] and/or poor scalability.

	Total Frame Size (Bytes)				
	128	**256**	**512**	**1024**	**1500**
BW (Mb/s)	771	842	890	940	947
Packet Loss (%)	25	19	14	9	7

Table 1: Network Efficiency vs. Packet Size

Although, new router technologies, such as found in [3], which use an optical switch fabric, have been introduced that improve router scalability. Unfortunately, such routers have significant hardware costs in addition to the possibility of introducing out-of-order packets. Moreover, these new router technologies do not attempt to address the primary cause of the scalability problem, which is, the majority of packets transferred across the Internet are small.

To better illustrate the problem of small packets, consider a simple experiment with results shown in Table 1. UDP CBR data was sent from multiple systems on-campus over a 1 Gb/s link to multiple systems at a downtown data center with varying total packet sizes. To avoid host limitations, each individual pair of hosts (campus, downtown) was validated to be capable of loss-free communications for the entire range of packet sizes. Total bandwidth consumption was kept constant with results averaged over multiple runs. Despite the routers managing relatively simple routing tables, significant packet loss occurs as the number of packets increases even though the total bandwidth (including all layer headers and the inter-frame gap) stays the same[2]. Finally, results were validated over multiple runs in order to eliminate transient congestion.

[1]Performance refers to the ability of the router to sustain a consistent transfer rate regardless of packet size distribution and/or destination distribution.
[2]For example the 128 byte frame consists of 12 bytes for the interframe gap (IFG), 8 byte MAC preamble, 14 byte MAC header, 20 byte IP header, 8 byte UDP header, 62 byte data payload, and 4 byte FCS.

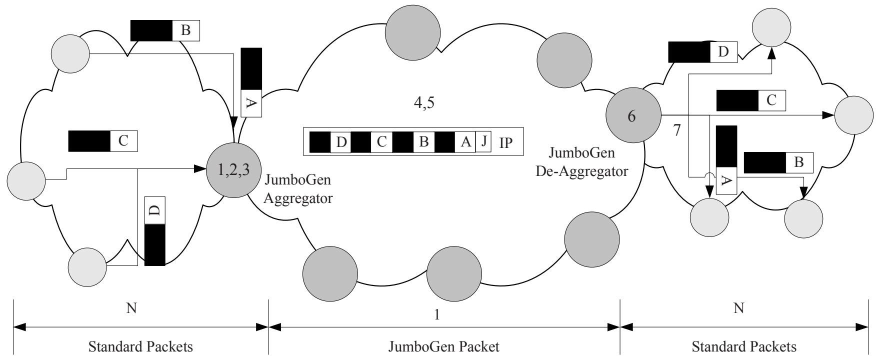


Figure 1: JumboGen Operation Overview

Given that network speeds will be increasing for the foreseeable future, a reduction in the volume of packet processing is critical. However, the legacy nature of the current Internet dictates that attempts to impose end-to-end solutions for larger packet sizes will be plagued by fragmentation, thus removing their benefit. As a result the weakest link (Ethernet) will likely continue to determine the effective MTU of the network. Moreover, the MTU is only part of the problem as small packets still dominate the majority of network traffic.

Thus, in this paper, we propose the JumboGen approach in order to reduce core router load. In short, JumboGen dynamically encapsulates smaller packets at the ingress to a domain into a single larger JumboGen packet (JumboGram). At the egress point, the original packets are decapsulated, shaped, and transmitted toward their final destinations. The end result of JumboGen is a significant reduction in the number of packets processed by the *core* nodes.

The reason JumboGen focuses on the core of the network, is because the core of the network will process many more packets than the routers at the edge, thus the core routers are likely to create a bottleneck for the network, while edge routers are likely to have spare computational ability. Thus, JumboGen aims to distribute the computational load of the network away from the core by adding additional load to the edge routers.

The remainder of this paper is as follows: Section 2 provides a detailed description of JumboGen. Section 3 discusses other issues related to JumboGen, including its relation to other networking technologies. Next, simulations are presented in Section 4 and results from experimental studies on university network traffic are presented in Section 5. Background and other related research is provided in Section 6 and finally, in Section 7, we present several conclusions regarding JumboGen and discuss future work.

2. JUMBOGEN OVERVIEW

JumboGen improves core router scalability by encapsulating packets with the same next Autonomous Systems (AS) and egress point, into larger packets for transmission across the domain. Critically, the design of JumboGen functions on a *domain-wise* scale, not end-to-end, with external entities (other domains and end hosts) unaware that any conversion took place. The overall JumboGen process is shown in Fig. 1 with the basic flow of events described below:

1. Packets arrive at an ingress node to the domain.
2. At the ingress node, the packets are sorted into queues based upon their egress point of the network in their path, which is obtained from the BGP [5] routing table. A *Jumbo Frame Encapsulation Timer* (JFET) is started for the queue.
3. Packets that are being sent through the same egress point are combined into the same JumboGram, subject to MTU.
4. Once the JFET for the queue has expired, the JumboGram is released towards the next AS.
5. The JumboGram is routed through the core of the network. Routing is provided by using the standard routing mechanism of the network[3].
6. The JumboGram arrives at the egress node and the original packets are separated out.
7. The original packets are forwarded onto their final destinations.

There are two main benefits of using JumboGen. The first benefit is that JumboGen lowers the number of packets that the core routers are responsible for processing, thus allowing the network to better scale as line speeds increase. The second beneficial aspect of JumboGen is that data is more efficiently transferred by reducing the number of physical layer headers used (due to a lower number of packets).

The JumboGen approach consists of three main components:

- *Fast Packet Encapsulation:* The JumboGram is structured to allow for efficient encapsulation, inspection, and decapsulation. Packet overhead is minimal and is offset by the reduction in physical layer headers.

[3]Information from the header of the first encapsulated packet (MPLS or IP) is simply copied to the header of the JumboGram.

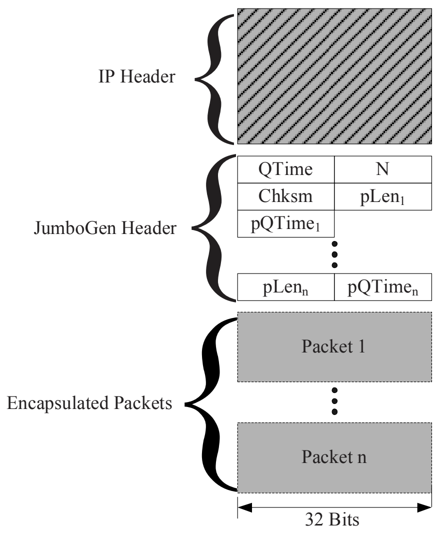


Figure 2: JumboGram Structure

- *Active Queue Management Support:* Active Queue Management support for JumboGen allows for partial dropping of a JumboGram, in addition to modifications to ensure fairness in a mixed jumbo/non-jumbo environment. Specifically, RED queue modifications are presented and discussed in detail.
- *Egress Shaping:* Egress shaping on JumboGrams occurs at the egress to the ISP in order to provide more reliable performance and limit traffic bursts.

2.1 Fast Packet Encapsulation

The structure of the JumboGram is shown in Fig. 2 containing the following fields:

- *QTime* - 16 bits; indicates the amount of time the packets were encapsulated over, which will be bounded by $JFET$.
- N - 16 bits; the number of encapsulated packets.
- *Chksm* - 16 bits; the checksum (1's complement addition) for the JumboGram header[4].
- *pktLen* - 16 bits; indicates each individual packet's length.
- *pQTime* - 16 bits, time offset of when the individual packet is queued. Allows packets to be shaped according to their arrival time.

The destination address of the JumboGram is the same as the first packet stored in the JumboGram. For an MPLS network, the destination address is the MPLS address of the first packet stored in the group. This ensures proper routing for all packets as all encapsulated packets contained in the JumboGram will arrive at the correct next AS in their path.

[4]An optional checksum could be added at the end of each encapsulated packet to improve error detection.

The design of the JumboGram allows the original packets to be decapsulated with minimal effort while also keeping the overhead of the JumboGram to a minimum. As shown, the overhead of the JumboGram is $6 + 4N$ bytes. However, the overhead is offset by the reduction in physical layer headers. The net cost (or benefit) of JumboGen can be stated as:

$$cost = H_{IP} + H_{JG} + (N * 4) - (H_p * (N - 1))$$

The cost of JumboGen in the above equation comes from the size of the IP header (H_{IP}), the JumboGen header (H_{JG}), and the number of encapsulated packets (N). The reduction in bandwidth comes from the reduction in physical headers[5] (H_p). For example, if the network is an Ethernet network and two packets were encapsulated into a JumboGram, then $H_{IP} = 20$, $H_{JG} = 6$, $N = 2$, $H_p = 38$, and the total cost would be -4 bytes. In other words, 4 bytes of bandwidth would be conserved.

The structure of the JumboGrams also allows active queue management schemes (AQM) to partially drop a JumboGram without requiring the restructuring of the JumboGram. When a partial drop occurs, the dropped packets are removed from the end of the JumboGram. Importantly, the N field is not modified, which ensures that the partial drop operation is as efficient as possible. The length information for the dropped packets is set to zero, and not removed, which allows the JumboGram to be forwarded without a rebuild of the JumboGram. While it may be more desirable to be able to drop packets from any location within the JumboGram, this would likely lead to performance issues in a router as the packets would have to be completely restructured. By removing only from the end of the packet, the transmission of the partially dropped packet will only require the modification of a few fields, with no packet reconstruction necessary.

2.2 Efficient Packet Queues

In JumboGen, a naïve approach to encapsulating packets is used. A naïve approach implies packets from the same flow[6] may be combined into the same JumboGram. When multiple packets from the same flow are combined into the same packet, the dropping of an entire JumboGram will unfairly penalize an individual flow. Fortunately, this is negated by two factors; first, with AQM modifications a partial packet drop can be performed for JumboGrams, ensuring that not all packets from the same flow will be lost. Second, in the experiments (Section 5) it is shown that over 70% of the packets encapsulated in a JumboGram are from unique flows.

Finally, if desired, a bitmask can be used to ensure that a JumboGram does not contain more than one packet from the same flow. In this case, there will be multiple queues for each possible egress point. When a new packet is to be placed in a queue, the bitmask is checked to see if the queue already contains a packet from that flow. If it does, then the other queues for the egress point will be checked (oldest to

[5]Includes interframe gap (IFG), Preamble, MAC Header, and FCS.

[6]Flow is defined as packets from the same source to the same destination.

newest) with the new packet being placed in the first queue which does not have a packet from this flow. This technique may change ordering of packets from different flows, but will not upset the per-flow ordering.

2.3 Active Queue Management Interactions

Active Queue Management (AQM) techniques are an important type of technology which aims to improve the utilization of the network. While JumboGen can be combined with any AQM technique, modifications to RED [6] queues are presented. For JumboGen to perform optimally, modification to the RED queue algorithm is required. If RED is not modified, JumboGrams will be treated as any other packet in the system, which is not desirable as the loss of a single JumboGram can lead to the loss of many packets from many flows. Thus, an approach that allows for the partial dropping of packets is used.

In [6], two different methods that RED queues can use to determine the queue utilization are presented, the first is by number of packets in the queue and the second is to determine queue utilization by number of bytes in the queue. In this paper we only discuss the RED modification required when the number of packets is used to determine queue utilization, as the number of packets, and thus routers, will likely be the bottleneck in current high-speed networks.

JumboGen-Aware RED Queue: In order to determine if a packet should be dropped at a router, the following formula is used:

$$dropPercent = (1 - \alpha) * \frac{Red_w}{N} + \alpha * Red_w$$

where α is the percentage of packets that are JumboGrams, Red_w is the probability that an individual packet will be dropped (assuming JumboGen is not used), and N is the number of packets encapsulated in the JumboGram. This formula allows for the probability of dropping a JumboGram to increase or decrease depending on the proportion of JumboGrams in the network. If the percentage of JumboGrams is small, there is little chance for the packet to be dropped. When the amount of JumboGrams in the network is small, it is undesirable to drop the JumboGrams due to the fact that every JumboGram dropped will have a significantly larger impact on the network than if a regular-sized packet is dropped. Conversely, if the percentage of JumboGrams is large, the probability of dropping a JumboGram will approach that of normal RED.

A more effective method for reducing the impact to the flows contained in a JumboGram is to allow for partial drop of the packet. Thus, when a JumboGram arrives at a router and is selected to be dropped, not all of the encapsulated packets will be lost, but only a subset. The fact that only a subset of encapsulated packets will be dropped will minimize the impact of dropping a JumboGram will have on a flow that has multiple packets encapsulated in the JumboGram. The formula to determine the appropriate number of packets to drop is given by:

$$Drop_{num} \Leftarrow (\frac{avgQ - min_{th}}{max_{th} - min_{th}} * (maxD - minD) + minD) * N$$

In the above formula, min_{th} and max_{th} come from the RED settings as explained in [6] and $avgQ$ is the average queue size. Additionally, two new variables are introduced, $minD$ and $maxD$, which define the lower and upper bounds on the percentage of packets to be dropped from the end of the JumboGram. An additional protection against harming flows that have a heavy presence in the JumboGram is the fact that the N field is not modified upon a partial drop. Thus, if a JumboGram is selected to have packets dropped at multiple routers in the core, the effect is not cumulative. Additional packets will only be dropped from the JumboGram, if downstream routers select a number of packets to be removed that is greater than previous values.

For example, suppose a JumboGram, with 12 encapsulated packets, traverses through three routers, A, B, and C. At router A, four packets are selected to be dropped, leaving eight packets unharmed. Upon arrival at router B three packets are to be dropped. However, since the JumboGram has already lost four packets, no additional packets will be removed. Finally, upon arrival at router C five packets are to be dropped. Thus, only one additional packet is dropped, for a total of five dropped packets, and seven successfully transmitted ones.

2.4 Egress Shaping

When the JumboGram reaches its destination, the packets need to be decapsulated and released to the next node on their path to the destination. If all the packets are released as soon as they are removed from the JumboGram, this can lead to dropped packets at the client due to the receive buffer overflowing (ACK Compression problem [7]). Hence packets are shaped at the egress according to the differences in their arrival time ($pQTime$). In other words, if two packets arrive at the ingress node 4 ms apart, they will be released from the egress node 4 ms apart.

3. JUMBOGEN INTERACTIONS

Before any new network technology can be deployed it is important to understand any adverse effects that may occur due to JumboGen conversion. Thus, we discus the main adverse affect of JumboGen, which is added packet delay in both the single and multiple conversion scenario. Additionally, the interactions of JumboGen with other networking technologies are discussed in order to provide insight as to how JumboGen can work with these technologies to provide an increased benefit.

3.1 Increased Packet Delay

Put simply, JumboGen trades a small window of delay for the ability to encapsulate into jumbo-sized packets. While this delay is non-zero, the delay is tolerable for several reasons. First, packets are strictly limited by $JFET$ for the queue time. Assuming that packets queued into a JumboGram at a uniform rate for the full $JFET$ time, the average ingress queue time for the packets in the JumboGram will be $\frac{JFET}{2}$. Egress shaping is also used, and contributes an additional average added delay of $\frac{JFET}{2}$. No packet will have a cumulative delay greater than $JFET$. Also, if the JumboGram reaches its full capacity before the $JFET$ time expires, it is released to the network early and is not held for the full $JFET$, which will lower the average delay experienced by packets.

Additionally, $JFET$ is kept small (1 ms for 1 Gb/s link) and can be decreased for higher network speeds. The additional latency introduced by $JFET$ is minor when compared with the latency tolerance users have for even the most demanding on-line applications. For example, first person shooters, the most demanding of applications in terms of latency, has a latency tolerance up to 180 ms [8]. Real time strategy games, however, have latency tolerances on the order of several seconds, similar to web browsing [9]. Lastly, VoIP, another delay sensitive application, can have up to 250 ms of one-way delay before it becomes noticeable to the user, with a recommendation for the total one-way latency to not be more than 150 ms (ITU G.114).

3.2 Multiple JumboGen Conversions

A packet will likely cross many ISP domains as it travels towards its final destination. Thus, it is important to quantify the effect of multiple JumboGen conversions (assuming multiple ISPs have JumboGen deployed) on the overall end-to-end delay of a packet. In [10], the authors present a five level hierarchy of classification for ISP domains in the Internet. The lowest domain is the customer domain, and is not likely to have JumboGen deployed. Thus, that leaves four levels of the hierarchy where JumboGen could be deployed. This implies that in the most severe case a packet would go through eight JumboGen conversions. One for each level as the packet passes through them in the following order: Small Regional ISP ⇒ Outer Core ⇒ Transit Core ⇒ Inner Core ⇒ Inner Core ⇒ Transit Core ⇒ Outer Core ⇒ Small Regional ISP. This would lead to a total added delay to the packet of $8 * JFET$. Assuming a 1ms $JFET$ this would lead to a total of 8 ms added delay, which is substantially smaller than the latency tolerance of even the most demanding applications. Multiple JumboGen conversion is mitigated in two ways. First, most packets will not likely travel through that many distinct domains, thus, it is unlikely that a packet will ever face all eight conversions (most traffic will cross a maximum of three to five ASs, and thus face three to five conversions). Second, ASs can form peering agreements in order to accept each others JumboGen packets, thus, eliminating the need for multiple JumboGen conversions.

3.3 Other Networking Technologies

To assess how JumboGen will interact in currently envisioned technology, we comment on three specific technologies: Differentiated Services, Expedited Congestion Notification, and Optical Networking. Additional comments made regarding related technologies are left to Section 6.

Differentiated Service (DiffServ): Differentiated Services is a quality of service (QoS) architecture proposed to address scalability concerns by aggregating flows into classes of service [11]. Packets in DiffServ are marked with a DiffServ Code Point (DSCP) that selects a Per Hop Behavior (PHB) [12] to be used in the core. With regards to JumboGen, the primary cause for concern would be the combination of packets from multiple classes into the same JumboGram. While opportunities would exist through ordering when only loss differentiation exists (Assured Forwarding service, AFx0 vs. AFx1 vs. AFx2) [12], the default behavior for JumboGen would be to only allow packets from the same class to be encapsulated in the same packet. The net result will be a reduction of encapsulation opportunities for JumboGen that could potentially be addressed by an increase in $JFET$.

Expedited Congestion Notification (ECN): A second technology that intersects with JumboGen is the usage of Expedited Congestion Notification (ECN) [13]. While the usage of DiffServ has the potential to reduce the effectiveness of JumboGen, the inclusion of ECN with JumboGen is trivial. First, all JumboGrams are marked as ECN capable. Second, as the packets are decapsulated at the egress, the ECN information can simply be copied from the outer JumboGram to the outgoing packet if ECN was requested by the packet. Moreover, the ECN support could be applied even if the core of the network does not provide it. If partial packet drops are enabled, the presence of missing encapsulated packets (dropped in the core), implies that congestion is present in the network to the egress.

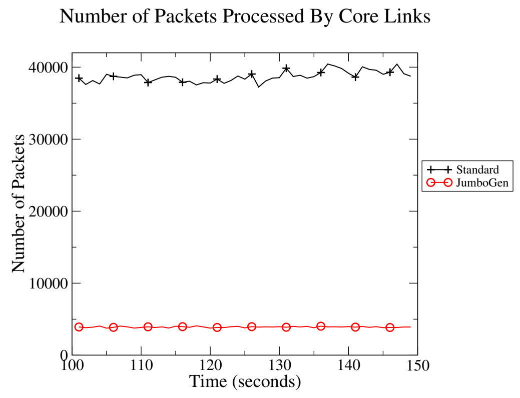
Optical Networking: With the proliferation of optical networking, a natural extension is to consider the impact of JumboGen in such a scenario. The overall reduction of total packets is quite compelling for optical networking as the cost of optical buffers is extremely high. Furthermore, if JumboGen is coupled with even faster line speeds and larger MTUs, JumboGen can offer significant cost savings in terms of the required number of network buffers.

4. SIMULATION RESULTS

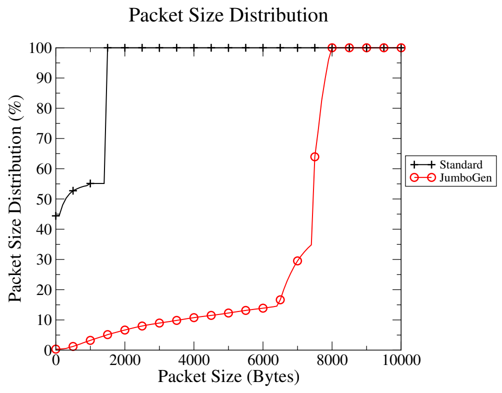
Simulation studies were conducted using the *ns-2* simulator. The goal of our studies was to examine the impact of JumboGen on scalability and quality of service (link utilization, TCP fairness, goodput, UDP end-to-end delay, and delay jitter). For our simulation studies, a network topology which consisted of a small network with five external nodes connected to each edge node and a link bandwidth of 250 Mb/s was used. The remaining information concerning the simulation setup is presented below:

- Traffic was generated between end nodes in order to ensure high link utilization in the network. A typical core link had hundreds of traffic flows active at a given time.
- The default MTU for JumboGen frames is 8 KB.
- The default JFET is 1 ms.
- The network flows consisted of short and long term flows. All short term flows were TCP and long term flows consisted of 20% CBR, 20% VBR, and 60% TCP. For TCP flows, the average size of the flows was 100 KB for short term traffic and 5 MB for long term traffic.
- Network traffic was supplemented with long term UDP traffic with an average lifetime of 20 seconds.

With the basic setup as listed above; four groups of simulations were performed. The first group evaluates the effectiveness of the different RED queue modifications. The second group of simulations demonstrates the effectiveness of egress shaping. The third group of simulations explores the



(a)



(b)

Figure 3: Effect of JumboGen on a) Number of packets seen by core links b) Packet size Distribution

	Link Util (%)	Std. Dev. of Link Util.	Total Goodput (MB)	Fairness	End-to-End Delay (ms)	Mean Delay Jitter (ms)
Standard	90.40	1.66	64.5	0.995	*20.1*	0.147
JumboGen - FD	89.93	1.75	61.4	*0.997*	21.8	*0.049*
JumboGen - PD	*94.32*	*1.39*	*81.4*	0.988	25.0	0.055

Table 2: Performance for various RED queue implementations

effect of MTU size on JumboGen. Finally, the fourth group of simulations demonstrates the effect that a minimum per-packet processing overhead has on the performance of the network. The following metrics were used to evaluate the performance of JumboGen:

- *Packet Distribution:* Number of packets processed by a core link in the network and the distribution of the packet sizes.
- *Link Utilization*
- *Service Fairness:* Four TCP flows were monitored over the life of the simulation in order to determine if fair service was received via Jain's Fairness Index [14].
- *Total Goodput:* Average total goodput for monitored TCP flows.
- *End-to-end Delay:* A CBR flow was monitored over the life time of the simulation in order to determine the end-to-end delay and jitter received by the UDP traffic.
- *Instantaneous Packet Delay Variation (IPDV):* Delay jitter or IPDV [15] was measured between the source and a destination of the monitored CBR flow.

The RED settings were the same for all three experiments with the exception of the α parameter. For the JumboGen variations, α was modified to account for the increase in the coarseness of the sliding average due to the reduction in the number of packets.

- *Standard* - The performance of the network when JumboGen is not used. $\alpha = 0.005$.
- *JumboGen - Full Drop (FD)* - In this simulation when a JumboGram is selected to be dropped, all packets are lost. $\alpha = 0.025$.
- *JumboGen - Partial Drop (PD)* - Full deployment of JumboGen at ingress and egress. Core routers are also JumboGen aware, allowing for partial packet drop. Queue length is determined by number of encapsulated packets. $\alpha = 0.025$.

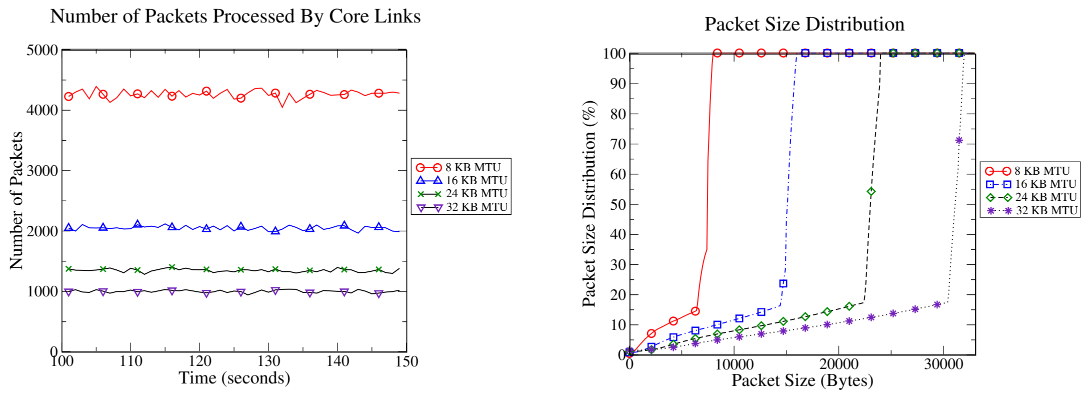
4.1 RED Queue Modification Performance

In Fig. 3(a), the number of packets transmitted over one of the core links is shown. As can be seen in the figure, the number of packets processed by the core links is roughly an 8x reduction over the non-JumboGen case (38,000 packets per second versus 4,500 packets per second). In Fig. 3(b), the packet size distribution for the same core link is shown. It can be seen that without JumboGen, the majority of packets are either 1500 (for the various TCP transfers) or 64 bytes (TCP ACKs) and the remainder are various UDP packets. For JumboGen, the majority of packets are located near the MTU (8 KB), thus reducing the number of packets transferred in the network as demonstrated in Fig. 3(a).

In Table 2, the results for various performance metrics are shown. In terms of end-to-end delay, the baseline case performs better than the JumboGen cases. This is to be expected as JumboGen has to delay each packet slightly in order to determine if it can be combined with other packets. Note that the added delay for the JumboGen full drop case is roughly double the $JFET$ time. This is because the packets are delayed for $JFET$ time at the ingress router, and then the packets are released over $JFET$ (due to shaping) time

	Link Util (%)	Std. Dev. of Link Util.	Total Goodput (MB)	Fairness	End-to-End Delay (ms)	Mean Delay Jitter (ms)
Standard	90.40	1.66	64.5	0.995	*20.1*	0.147
JumboGen - No ES	94.25	*1.31*	76.7	*0.992*	22.5	0.266
JumboGen - ES	*94.32*	1.39	*81.4*	0.988	25.0	*0.055*

Table 3: Effectiveness of egress shaping



(a) (b)

Figure 4: Effect of different MTU sizes on a) Number of packets seen by core links b) Packet size distribution

at the egress router. The reason that the average queue time is a full $JFET$ at the ingress router is due to the fact that the JumboGram is filled quickly, well before $JFET$ expires. Although the JumboGram can not hold any more encapsulated packets the simulation still holds the packet for the full $JFET$ time, leading to the situation where the packets queued are all held for a near $JFET$ time. If a threshold value is introduced to allow for early release of the JumboGram, this problem is negated, and will lower the average ingress queue time. The partial drop case has an even higher end-to-end delay, which is explained by the $JFET$ time and the fact that each packet experiences a greater queuing delay in the core due to a higher link utilization (which leads to higher goodput).

JumboGen with full packet drop, drops the complete JumboGram which introduces a drop-tail like behavior. The drop-tail behavior occurs despite the usage of RED queues, due to the fact that many flows will have packets dropped at the same time. This results in poorer link utilization. The net effect is an increase in packet delay (due to the JumboGen queuing delay) over the standard case even though the link utilization is lower. On the other hand, JumboGen with partial packet drop can react to congestion at a finer grain, and thus does not exhibit a drop-tail behavior. This allows JumboGen with partial drop to have a high link utilization and goodput. Finally, the fairness ratio for the different implementations is shown in Table 2 as well. Since each implementation provides extremely high fairness, it is reasonable to infer the JumboGen/RED implementation does not impair the fairness received by TCP flows.

4.2 Effectiveness of Egress Shaping

In Table 3, the effectiveness of egress shaping is shown (partial drop is enabled). Compared to the standard case and JumboGen without egress shaping case, JumboGen with egress shaping provides the highest goodput and the lowest jitter. When a JumboGram arrives at its destination, which employs egress shaping, the original packets are released according to their encapsulation time rather than released all at once. Thus, for TCP flows, egress shaping solves the ACK Compression problem [7] and improves the mean of total goodput without sacrificing the fairness.

For UDP flows, egress shaping decreases the mean of end-to-end delay jitter as there is no jitter for any two consecutive UDP packets if they are encapsulated into the same JumboGram. However, the improvement in delay jitter comes at the cost of an increase in the mean of end-to-end delay since the original packet has to wait at the egress node before it can be decapsulated from the JumboGram and thus will not be released immediately at the JumboGram's destination.

4.3 Varying MTU Setting

The final set of simulation results presented in this paper is the performance of JumboGen as the MTU size is adjusted. For all simulations, egress shaping was used and partial packet drops were allowed. The following scenarios were simulated:

- 8 KB MTU with 5 ms JFET
- 16 KB MTU with 5 ms JFET
- 24 KB MTU with 5 ms JFET
- 32 KB MTU with 5 ms JFET

MTU Size (KB)	**Link Util. (%)**	**Std. Dev. of Link Util.**	**Total Goodput (MB)**	**Fairness**	**End-to-End Delay (ms)**	**Mean Delay Jitter (ms)**
8	93.5	1.68	71.6	0.989	26.4	0.044
16	93.3	1.70	73.1	0.992	28.9	0.035
24	92.8	1.88	68.2	0.991	29.8	0.031
32	92.7	1.78	67.7	0.991	30.8	0.028

Table 4: Effectiveness of MTU

Min. Processing Time (μs)	**Link Util. (%)**	**Std. Dev. of Link Util.**	**Total Goodput (MB)**	**Fairness**	**End-to-End Delay (ms)**	**Mean Delay Jitter (ms)**
0.0	90.40	1.66	64.5	*0.995*	20.1	0.147
3.9	86.60	1.514	53.3	*0.995*	*19.6*	0.128
5.8	84.65	1.396	48.6	0.998	20.1	0.129
7.8	81.87	*1.008*	43.5	*0.995*	20.2	0.116
JumboGen	*94.32*	1.39	*81.4*	0.988	25.0	*0.055*

Table 5: Results from hard limit on number of packets processed per second.

In Fig. 4(a), the number of packets processed by the core link node is shown. As is expected, as the MTU is increased, the number of packets seen by the core nodes decreases. In Fig. 4(b), the packet size distribution is shown. As can be seen in the figure, as MTU increases, the size of the majority of packets increases as well. However, with larger MTUs the majority is a shrinking majority.

In Table 4, the results for the various MTU settings are shown. The most important result that can be seen is that increasing the MTU to greater values does not always provide for better performance. However, further investigation reveals that with the increase in the MTU the α value is no longer optimal. Thus, only by adjusting α can a greater MTU size be of benefit.

4.4 Varying Minimum Per-Packet Overhead

As demonstrated by Table 1, as the frame size is decreased the maximum throughput of the channel decreases. The maximum throughput decreases because there is a minimum amount of time needed to route each packet, irrespective of packet size. Thus, when only smaller packets are present in the network, the per-packet routing overhead will limit the number of packets the router can process per second. In order to illustrate this issue, the *ns-2* simulator is modified such that each packet will take a minimum amount of time to process, regardless of packet size. The results for the simulation are shown in Table 5. While the real routers can process packets at significantly higher rates than what is shown in Table 5, these values were chosen in order to demonstrate on our simulation set-up the effect that this problem has. Overall, as the per-packet minimum processing time increases, the performance of the network goes down. However, when JumboGen is used the minimum processing has little effect due to the fact that the number of packets the router needs to process has been significantly reduced.

5. EXPERIMENTS

In order to test the effectiveness of JumboGen, traces on live traffic were captured from the university Internet traffic using a 100Mb/s tap. Only outbound traffic was captured over a period of five hours. For each flow, the egress point for the university's ISP was determined. In Table 6, the top five most common paths are shown. Commonality is determined by how many flows will follow the same path to the egress point. As can be seen, over 60% of the flows utilizes the same path through the university's ISP. For over 13% of the flows, no traceroute information could be obtained, thus any packet which is a member of these flows was only combined with other packets of the same flow. This implies that the actual performance of a real deployment of JumboGen will be better than indicated by the experiment, as the egress and ingress for each packet can be determined. The remainder of flows (0.9%) were evenly distributed over a number of less common paths.

Path #	**% of Total Addresses**
1	66.3
2	15.4
3	3.0
4	0.5
5	0.1
No Info	13.8

Table 6: Top 5 Most Common Paths

Typically, JumboGen is envisioned to be deployed on networks that use a much greater line speed. However, these experiments do provide benefit in that they show the potential for benefit from JumboGen when $JFET$ is scaled appropriately to line speed. For the experiments, a $JFET$ value of 5 ms and 10 ms was used. A faster link, such as 1 Gb/s, should have a $JFET$ of 0.5 ms to 1 ms. Additionally, the MTU size was set to 8 KB for all experiments.

In Fig. 5, the average flow population for a JumboGram is shown. It is shown that the majority of the time there will only exist one packet per flow represented in the JumboGram. Two packets per flow only occurs 16% of the time, and anything over five packets per flow occurs less than 1%

	Avg. # Pkts. (K/s)	Std. Dev. (K/s)	Avg. Added Delay (ms)	Avg BW Saved (Mb/s)
No JumboGen	564.8	52.1	N/A	0
JFET 5ms	132.3	10.5	2.83	2.1
JFET 10ms	116.2	9.1	5.29	2.2

Table 7: Results from experimental study of JumboGen.

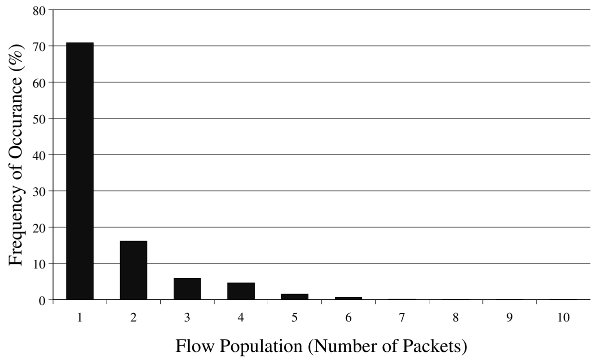


Figure 5: Histogram of Flow Population per JumboGram

of the time. This supports our intuition that a more complex queuing mechanism is not needed for JumboGen in order to ensure fairness.

The results from the experiment are presented in Table 7. As is shown in Table 7 the average number of packets processed per second for JumboGen is significantly less than the number of packets processed per second when JumboGen is not deployed. Additionally, the standard deviation shows that the number of packets processed per second is a much more stable value when JumboGen is deployed, unlike the standard case which varies significantly.

Also shown in Table 7, is the average queuing delay added due to JumboGen. As can be seen the average added delay added by JumboGen is roughly $\frac{JFET}{2}$ at the ingress, which is expected as only the first packet in a JumboGen queue will be delayed the $JFET$ time. Finally, it is shown that the use of JumboGen results in a savings of bandwidth of 2.1 Mb/s to 2.2 Mb/s[7] depending on the setting of the $JFET$ timer.

Overall, while the limited number of egress nodes in our University's ISP is not indicative of the number of egress nodes of a larger ISP, these results do show that JumboGen has promise. This is true because larger ISPs, while having more egress points, it will have a substantially larger amount of traffic from which to form JumboGrams.

6. BACKGROUND AND RELATED WORK

The closest works to JumboGen are the works on packet aggregation [16], Optical Packet Switching (OPS), and Optical Burst Switching (OBS) [17] as well as their numerous derivatives [18, 19, 20]. With regards to previous work on packet aggregation, JumboGen differentiates itself in terms of both its compatibility with RED, the specification of how/when to packets beyond only ACK packets, and the significant performance improvements to jitter and goodput offered by its egress shaping mechanisms.

In relation to OBS and its derivatives, JumboGen is distinct in that it re-uses existing packet switching mechanisms while avoiding the need for complicated fixes to improve performance. For instance, both [18] and [20] as well as others noted the need for larger queuing times (10% to 20% of RTT) and the distinct impact of the randomized loss nature of OBS on well-deployed TCP stacks. In contrast, JumboGen does not require changing the end host to improve performance and its impact in terms of aggregation time is significantly less as validated by our simulations and real network traffic analysis.Hence, the uniqueness of JumboGen versus the above aggregation-based schemes comes from its offering of true transparency in terms of perceived application performance, i.e. keeping the black box nature of the network, while significantly improving router scalability.

Another related work is Gathercast [21], which also uses queuing in order to improve the efficiency of the network. However, they differ in that Gathercast focuses mainly on aggregating TCP ACKs with the same destination into the same packet. Finally, Gathercast, also needs client/server support in order to obtain the most benefit from Gathercast.

Other works that focus on router performance are [3, 22, 23]. These works introduce new routing techniques in an effort to ensure high-speed performance. However, these works fail to address the core problem of increasing line speeds when the average packet size does not scale accordingly. JumboGen addresses this core issue by allowing packet sizes to scale closely with increases in line speed. Hence, the number of packets needed to be processed by the core routers remains consistent as line speed increases.

Finally, as most traffic transferred over the Internet is TCP-based, it is important to be aware of the implications high-speed networks and packet aggregation has on TCP performance. TCP performance in high speed networks has been studied at great length [24, 25, 26]. Additionally, [4] gives an indication of how high speed networks can benefit from large MTU sizes.

7. SUMMARY AND FUTURE WORK

In this paper, we have proposed a dynamic packet encapsulation scheme that allows for the utilization of higher speed networks while lessening the load placed on routers as the number of packets that traverse the network increases. This

[7]The savings come from the reduced number of physical headers on the 100 Mb/s connection.

allows for better utilization of the transfer medium and a more scalable transport. Simulations were conducted to demonstrate that JumboGen can be used without adversely affecting the performance of the network in terms of link utilization and fairness to traffic flows. JumboGen also provides a benefit to the total goodput of the network and reduces the number of packets processed by the core links. These results were also strengthened by experiments on traces obtained from the university network showing the potential benefit of JumboGen on even a small ISP.

Our future work with JumboGen lies in obtaining network traces from the Internet2 backbone, in order to perform a better analysis of the impact of JumboGen on a larger network. Additionally, we are investigating methods on how the peering agreements between various ISPs may work in order prevent multiple encapsulation/decapsulation penalties across domains that support JumboGen. Finally, we are working with an Intel IXP 2350 in order to more fully understand the performance of JumboGen in a real router.

Acknowledgements

This research was supported in part by the National Science Foundation through the grant CNS03-47392 and an Intel IXP Grant (2005).

8. REFERENCES

[1] R. Gass, "Sprint corp. packet size distribution," in *http://ipmon.sprint.com*, 2004.

[2] A. M. Odlyzko, "Comments on the Larry Roberts and Caspian networks study of internet traffic growth," *The Cook Report on the Internet*, pp. 12–15, Dec. 2001.

[3] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet routers using optics," in *SIGCOMM 2003*. New York, NY, USA: ACM Press, 2003, pp. 189–200.

[4] J. Kay and J. Pasquale, "Profiling and reducing processing overheads in TCP/IP," *IEEE/ACM Transactions on Networking*, vol. 4, no. 6, pp. 817–828, Dec. 1996.

[5] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," *IETF RFC 1771*, Mar. 1995.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking.*, vol. 1, no. 4, pp. 397–413, 1993.

[7] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz, "TCP behavior of a busy internet server: Analysis and improvements," in *Proc. of IEEE INFOCOM (1998)*, 1998, pp. 252–262.

[8] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer Quake 3," in *11th IEEE International Conference on Networks*, Sept. 2003, pp. 137–141.

[9] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in Warcraft III," in *NetGames '03*. New York, NY, USA: ACM Press, 2003, pp. 3–14.

[10] L. Subramanian, S. Agarwai, J. Rexford, and R. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. of IEEE INFOCOM 2002*, June 2002, pp. 618–627.

[11] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services field (DS Field) in the IPv4 and IPv6 headers," *IETF RFC 2474*, Dec. 1998.

[12] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," *IETF RFC 2597*, June 1999.

[13] K. Ramakrishnan and S. Floyd, "A proposal to add Explicit Congestion Notification (ECN) to IP," *IETF RFC 2481*, Jan. 1999.

[14] R. Jain, M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared systems," in *DEC Technical Report DEC-TR-301*, 1984.

[15] C. Demichelis and E. Petrov, "Instantaneous packet delay variation," in *Proc. of IEEE Workshop on QoS Support for Real-Time Internet Applications*, June 1999.

[16] H. Tounsi, L. Toutain, and F. Kamoun, "Small packets aggregation in an IP domain," in *IEEE ISCC 2001*, Hammamet, Tunisia, July 2001, pp. 708–713.

[17] C. Qiao and M. Yoo, "Optical burst switching (OBS) - a new paradigm for an optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69–84, 1999.

[18] A. Detti and M. Listanti, "Impact of segments aggregation on TCP reno flows in optical burst switching networks," in *IEEE Infocom 2002*, New York, New York, June 2002, pp. 1803–1812.

[19] J. He and S.-H. G. Chan, "TCP and UDP performance for internet over optical packet-switched networks," *Computer Networks*, vol. 45, no. 4, pp. 505–521, 2004.

[20] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," in *Proc. of IEEE INFOCOM 2004*, Hong Kong, China, Mar. 2004, pp. 774–784.

[21] B. Badrinath and P. Sudame, "Gathercast: The design and implementation of a programmable aggregation mechanism for the Internet," in *Proc. of IEEE ICCCN*, Las Vegas, NV USA, Oct. 2000, pp. 206–213.

[22] K. Y. Yun, "A terabit multiservice switch," *IEEE Micro*, vol. 21, no. 1, pp. 58–70, 2001.

[23] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," in *Proc. of SIGCOMM 2002*. New York, NY, USA: ACM Press, 2002, pp. 251–264.

[24] S. Floyd, "RFC 3649 highspeed TCP for large congestion windows," *IETF RFC 3649*, Dec. 2003.

[25] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[26] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proc. of IEEE INFOCOM 2004*, Mar. 2004.