Characteristics of Fragmented IP Traffic on Internet Links

Colleen Shannon, David Moore, k claffy

Abstract— **Fragmented IP traffic is a unique component of the overall mix of traffic on the Internet. Many assertions about the nature and extent of fragmented traffic are anecdotal rather than empirical. In this paper we examine the causes and attributes of measured fragment traffic and contrast those results with commonly cited beliefs. In particular, the effects of NFS, streaming media, networked video games, and tunneled traffic are quantified, and we estimate the prevalence of packet fragmentation due to improperly configured machines.**

To understand the prevalence, causes, and effects of fragmented IP traffic, we have collected and analyzed seven multi-day traces from three sources. These sources include a university commodity access link, a highly aggregated commercial exchange point, and a local NAP. Although there is no practical method of ascertaining whether any data provide a representative sample of all Internet traffic, we do include data sources that cover several different types of WANs with traffic from commercial entities, educational and research institutions, and large government facilities.

Keywords—**fragmentation, fragment, CoralReef, TCP/IP**

I. Introduction

The Internet protocol (IP) was designed to facilitate communication between heterogenous networks and thus serves as a least-common-denominator protocol with which computers differing in architectures, operating systems, and applications, connected by varying routes, paths, and protocols, can exchange information. IP must have the capability to handle differences in maximum sizes of transmitted packets on dissimilar networks. While it is trivial to move packets from a network with a smaller MTU (maximum transmission unit) to a network with a larger MTU, the reverse is challenging. To overcome this obstacle the IPv4 protocol performs fragmentation: a router breaks the datagram up into smaller individual pieces called fragments. Each fragment has its own IP header, which is a replica of the original datagram header. Thus each fragment has the same identification, protocol, source IP address, and destination IP address as the original IP packet. To distinguish fragments and allow correct reassembly, the offset field of each fragment contains the distance, measured in 8-byte units, between the beginning of the original datagram and the beginning of that particular fragment. The first fragment by definition has its offset set to 0, the second fragment has as its offset value the payload size of the first fragment, and so on. All of the fragments except the last have the 'more fragments' bit set so that the end host waits to receive all of the fragments before reassembling them into the original IP datagram. The size of each fragment usually corresponds to the size of the MTU of the subsequent link minus the length of the header that is added to each fragment. After disassembly of the original datagram, fragments are sent out into the network and are routed independently towards their destination. By providing an automatic network mechanism for handling disparate MTU sizes, IP allows end hosts to exchange traffic with no explicit knowledge about the path between them.

In their 1987 paper "Fragmentation Considered Harmful," Kent and Mogul [1] established that packet fragmentation is a suboptimal method of handling packets as they traverse a network due to the increased consumption of bandwidth, packet switching, and CPU resources. While some of their assertions apply to hardware and software that are now deprecated, their overall argument remains valid for several reasons. Modern routers have sufficient buffering capabilities to receive back-to-back packets and current computers generally have sufficient buffer space to reassemble very large packets. As a result, fragmentation is no longer an insurmountable problem for end hosts. However, the adverse effects of fragmentation on network performance and infrastructure continue to negatively impact wide area transport. First, an intermediate router must perform the fragmentation. This CPU-intensive operation impairs the ability of the fragmenting router to efficiently process non-fast path traffic. The additional fragmented packets increase the load on all routers and networks be-

All authors are with CAIDA, San Diego Supercomputer Center, University of California, San Diego. E-mail: {cshannon,dmoore,kc}@caida.org.

Support for this work is provided by DARPA NGI Contract N66001-98-2-8922, NSF grant NCR-9711092, and Caida members.

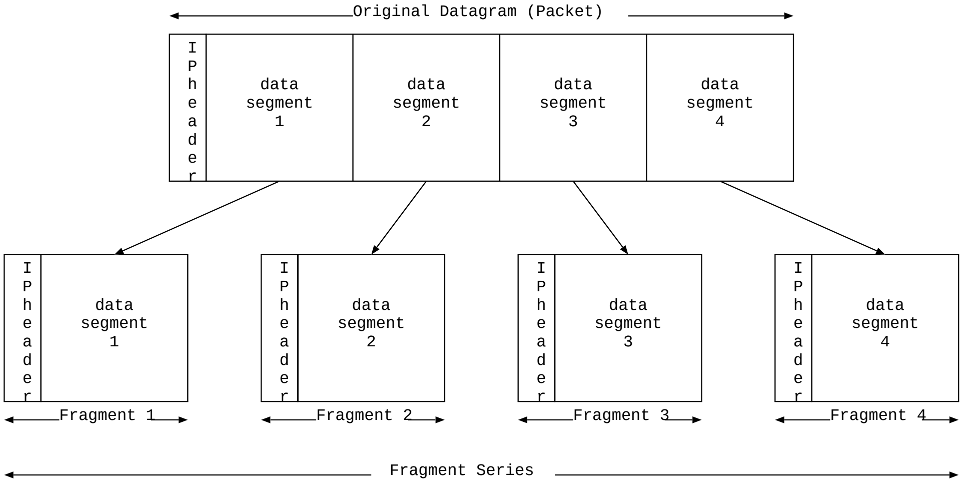


Fig. 1. Composition of a fragment series.

tween the initial router and the end host. Finally, once the fragments reach their destination they must be reassembled by the end host in another CPU intensive operation. The loss of any fragment results in the destination host dropping the entire packet. This in turn forces the source host to repeat transmission of a datagram that will be fragmented once again. Researchers have shown that in certain specific, controlled circumstances fragmentation can improve performance [2]; however, those observations do not apply to backbone links. Despite widespread advances in the intervening thirteen years, IP packet fragmentation is still "considered harmful".

Since the work of Kent and Mogul, many untested hypotheses about the causes and effects of fragmented IP traffic have come to be treated as fact. Foremost is the assertion that fragmented traffic does not exist. Others in the networking community accept the existence of fragmented traffic on LANs, but believe its scope does not extend to backbone links. Further common beliefs include that only UDP traffic is fragmented, that NFS is the source of all fragmented packet traffic, that fragmented IP traffic on the whole is decreasing, and that certain misconfigurations are causing an increase in fragmented traffic. These beliefs as a group are not tenable, since several are mutually exclusive (e.g. the overall volume of fragmented traffic cannot be simultaneously increasing and decreasing). While one recent publication suggests that IP packet fragmentation is increasing [3], all other fragment folklore has no basis in current network measurement.

Yet, IP packet fragmentation continues to play a small but vital role in facilitating communication between hosts on the Internet. The proliferation of protocols that send packets with different MTUs necessitates a system flexible enough to accommodate these variations. IP packet fragmentation increases the robustness and efficacy of IP as a universal protocol. In this paper, we examine the character and effects of fragmented IP traffic as monitored on highly aggregated Internet links.

The paper is organized as follows. Section II defines terminology we use to describe fragmented traffic. Sources of data and our methodologies for analysis are presented in Section III. In Section IV we present our results characterizing fragmented traffic. Finally, Section V summarizes the current effects of fragmented traffic on the monitored links.

II. Terminology

This section introduces the terminology used in our discussion of IP packet fragmentation. Several of these terms are illustrated in Figure 1.

As described in RFC 1191 [4], the *Path MTU* is the smallest MTU of all of the links on a path from a source host to a destination host. In the context of this paper, values observed for a Path MTU reflect the smallest MTU of all links between the source and the passive monitor.

We define an *original datagram* as an IP datagram that will be fragmented because its size exceeds the MTU of the next link on its path to its destination. *Packet fragment*, or simply *fragment*, refers to a packet containing a portion of the payload of an original datagram. While for the purposes of this paper, the terms packet and datagram are synonymous, we will use *original datagram* and *packet fragment* in the interest of clarity. A *fragment series*, or simply *series*, is the ordered list (as monitored on the network) of fragments whose source is a single original datagram.

The *size* of the series will be used to refer to the total number of bytes in the series, while the *length* of the series describes the number of fragments in the series.

The *first fragment* is the packet containing the original IP header and the first data segment of the payload of the original datagram. The *last fragment* is the packet containing the last portion of the payload of the original datagram. Because fragments can be transmitted in any order and because packets can be reordered as they pass through a network, the first observed and last observed fragments do not necessarily contain the first and last segments of the payload of the original datagram (respectively), and are thus not necessarily the first or last fragment of the series.

The first fragment is frequently equal in size to the largest fragment in each series. The *largest fragment size* is greater than or equal to the size of the other fragments in the series. If all of the fragments in a series are the same size, the largest and smallest fragments of the series will be identical.

Similarly, the last fragment is not always the smallest fragment in a series. So the *smallest fragment size* is less than or equal to the other fragment sizes in a series.

Because the IP protocol permits networks to drop, duplicate or reorder packets, the individual fragment packets for a single original datagram may not arrive at the destination in transmission order. We define a series as *complete* when the fragmented packets monitored provide sufficient coverage of the original data segment to allow reconstruction of the transmitted datagram (i.e. reordering or duplication may have occurred, but no fragments are missing). Conversely an *incomplete* series (Figure 2) does not have sufficient information to reconstruct the original datagram; some part of the payload never reached our monitor.

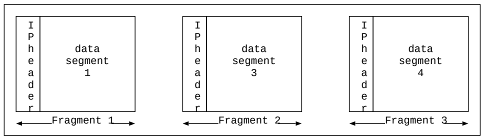


Fig. 2. Example incomplete series.

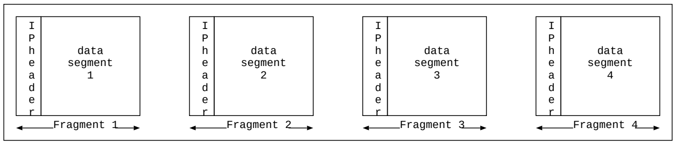


Fig. 3. Example in-order series.

A series is *in-order* (Figure 3) if the fragments are observed arriving sequentially; we never monitor a fragment with an offset lower than its predecessors. Conversely, a series is considered in *reverse-order* (Figure 4) if its fragments have offsets that never increase. A computer producing in-order series transmits data segment 1 through data segment N, while a computer producing reverse-order series transmits data segment N through data segment 1. However, we cannot necessarily correlate the order in which we received the packet fragments and the order in which they were transmitted by the fragmenting router, since the fragments can be reordered by the network. Only one fragment needs to be delivered out of order for us to observe a reverse order two-fragment series. For longer series, it is less probable that an exact reversal of the fragment order occurs in the the network than it is that the ordering is due to reverse order transmission.

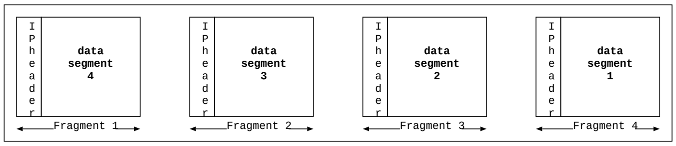


Fig. 4. Example reverse-order series.

A series contains a *duplicate* (Figure 5) if at least two of its fragments cover the exact same portion of the original payload.

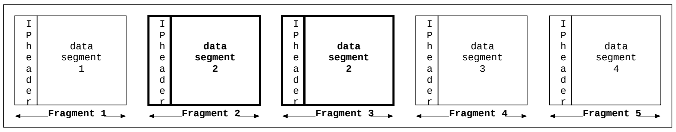


Fig. 5. Example duplicate series.

An *overlapping* series (Figure 6) has at least two fragment packets that contain overlapping portions of the original payload when the two fragments are not duplicates. Conversely, a *non-overlapping* series has no overlapping fragments. Note that the 'teardrop' denial of service attack [5][6] sends large fragments that are overlapping except for a single byte, thereby exhausting buffer resources in certain fragment reassembly implementations.

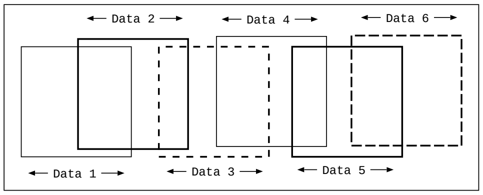


Fig. 6. Example overlapping series.

We define a *correct* series (Figure 7) as a series that is complete, with no overlapping or duplicated fragments. Any order of fragment arrival is acceptable in a correct series.

Dataset	Length		Characteristics		
	Start Time (UTC)	Duration (hours)	Packets (kpkts)	Bytes (MB)	Src Hosts[1]
CERF-IN	Fri Mar 9 02:01	252.00	2,797,266	1,439,570	2,745,493
CERF-OUT	Fri Mar 9 02:01	252.00	3,394,283	1,559,170	37,242
SDNAP	Fri Mar 9 01:36	259.58	1,073,321	646,677	328,094
MAEWEST-1	Fri Mar 9 01:35	75.00	5,307,429	2,203,614	1,277,423
MAEWEST-2	Tue Mar 13 02:12	132.00	8,991,449	3,963,302	1,691,880
AIX-1	Fri Mar 9 01:38	58.00	8,781,881	3,281,324	2,684,104
AIX-2	Mon Mar 12 04:35	49.00	8,070,586	3,743,040	2,478,624

TABLE I

DATASETS USED IN STUDY – MARCH, 2001

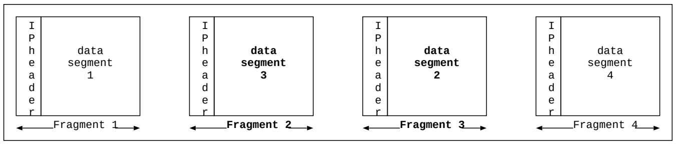


Fig. 7. Example correct series. Note that this series is *not in-order.*

III. METHODOLOGY

Measurement Sites

Data sets for this study were collected from three different locations, summarized in Table I. The first data source for this study was a link at MAE-west. We used an Apptel Point card to collect traffic exchanged by customers that peer at MAE-west. No intra-customer traffic is observed at this location. Traffic across SDNAP, a regional exchange point located in San Diego, California, was the second data source for this paper. We used `libpcap` [7] to monitor this Gigabit Ethernet traffic. Using a FORE ATM OC3 card, we monitored the commodity access link that connects the University of California, San Diego campus (including such entities as the San Diego Supercomputer Center and the Scripps Institute of Oceanography) to CERFnet. At our final location, traffic was collected from a link between Ames Internet Exchange (AIX) and MAE-west, using a WAND DAG card [8].

The numbers of unique source hosts for each data set as shown in Table I were limited to hosts which sent at least 3 packets over the lifetime of the trace. This filtering was applied to provide a more accurate count of the actual number of hosts transmitting across the link, since the MAE-west data sets contained at least one random source Denial of Service attack.

Traffic Monitoring

A specialized tool, `crl_frag_capture`, collected the data for this study due to the high volume of traffic at some of the measurement sites. `crl_frag_capture` relies on the CoralReef [9] software suite for header capture, interval handling and data aggregation. `crl_frag_capture` examines only packet headers; we attempted no analysis of the payload portion of the traffic. We organized the data we collected into hour long time intervals for post-processing. We collected four types of data:

frags.pcap — a full header trace in libpcap [7] format containing only fragmented traffic packets (either offset > 0 or 'more fragments' set).

src_ip.t2 — an aggregated table of non-fragmented traffic containing the number of packets and bytes seen per source IP address.

proto_ports_folded.t2 — an aggregated table of non-fragmented traffic with the number of packets and bytes seen per 3-tuple of IP protocol, source port, and destination port. Since a significant amount of monitored traffic travels between a well known port and an ephemeral port, additional aggregation was done for commonly occurring ports. A list of 19 ports[2] was chosen from preliminary studies of non-fragmented traffic on these links. For each packet with a source or destination matching one of these common ports, the ephemeral port is set to 0, causing all traffic for each of these 19 services to fall into a single bucket. Additionally, all ports above 32767 were bucketed as 32768, since the ports in this range are typically dynamically allocated and we found in preliminary studies that no well known ports above 32767 had a significant volume of traffic.

length.t2 — a table of non-fragmented traffic aggregated

[1] Unique IP source addresses which sent at least 3 packets over the trace lifetime.

[2] Specific ports in aggregation application order: 80, 53, 25, 443, 27015, 110, 113, 37, 20, 119, 5000, 6112, 6667, 6688, 6699, 6970, 8888, 9000, 27005.

by the number of packets and bytes seen with each IP packet size.

The collection of full header traces for non-fragmented traffic was not feasible due to the high volume of traffic on the monitored links. Furthermore, partitioning of the data into independent tables for source IP address, protocol/ports, and packet length obscures the original relationships between these fields.

Fragment Processing

For an in-depth analysis of IP packet fragmentation, constituent fragments from each original datagram were assembled into a fragment series. Fragments were separated into discrete series using the identification, protocol, source IP address and destination IP address fields, since those fields uniquely define fragments of an original datagram. We used a timeout of 600 seconds for each series to provide sufficient time for all fragments to be monitored even with significant network delays.

The payload of the original packet was not reconstructed, since the offset and size of each fragment are sufficient to infer the basic properties of fragmented traffic.

Application Mapping

To discern which applications and services produce the most fragmented traffic, we map the protocol, source port and destination port fields of each IP packet header to a named application by choosing the first matching rule from an ordered collection of protocol/port patterns. For this study, we used CAIDA's passive monitor report generator application list.[3] The list contained 81 entries, which includes common well known ports from the IANA port assignment list [10], as well as emerging multimedia and video game applications (such as RealAudio, Quake, Napster). For example, traffic to and from ports 80 and 8080 are classified as WWW traffic, while connections to port 21 are classified as FTP data. Because passive FTP utilized dynamically allocated ports, we cannot distinguish it from other traffic to ephemeral ports.

IV. Results

A. Overall trends in Fragmented Traffic

Table II shows the percentage of fragmented and non-fragmented traffic found in each data set. We observed hosts sending both fragmented traffic and non-fragmented traffic, so the host percentages may total more than 100%. Although the overall volume of fragmented traffic was small, it was also highly variable. Figure 9 shows the variance in the number of fragmented packets, number of fragmented bytes, and number of hosts sending fragmented traffic.

The non-fragmented traffic measured by both the AIX and MAE-west monitors demonstrated diurnal cycles. The traffic at SDNAP does not share the strongly cyclical nature of traffic at the other two locations, although it does show a daily decrease in traffic late at night (Pacific Standard Time). Figure 8 shows time series plots of the non-fragmented traffic. Note that Figures 8(e) and 8(f) do not exclude random source Denial of Service attacks. These attacks produce spikes in the number of hosts generating traffic with no periodic temporal patterns [11].

B. Classification of Fragmented Traffic

Fragment series can be categorized by the order in which the monitor received their constituent packets. Table III shows the breakdown of all series based on the following attributes (as defined in Section II): correct, complete, in-order, reverse-order, overlapping, or duplicate. Of all series, 98.1% are complete, meaning they contain sufficient information to reconstruct the original datagram. Correct series (Figure 7) account for 89.6% of all series. Of complete series, 81.5% are in-order (Figure 3) and 9.2% are reverse-order (Figure 4); the remaining 9.7% are either overlapping (Figure 6) or duplicate (Figure 5) series; both are attributes which impede exact determination of ordering. Of all complete series, 1.1% have overlapping fragments and 8.6% contain duplicates.

1.62% of all monitored series were correct series that were neither in-order nor reverse-order; they were likely reordered in transit. In May 2000, Paxson et. el. [12] observed that approximately 0.3% of all packets arrive out of order. Thus we hypothesize that fragmented traffic is reordered by the network at a greater rate than non-fragmented traffic. However, we have no way to quantify the overall frequency of out of order non-fragmented packets in our data sets so we cannot prove that this is the case.

Reverse-order series are not problematic; in fact, they can actually be beneficial since a host receiving a reverse-order series can use the fragment length and offset fields to immediately allocate correctly sized buffers, rather than growing or chaining buffers as subsequent fragments arrive.

C. Characteristics of Fragment Traffic

To clearly portray the characteristics of fragmented traffic, we use images generated from data collected at the Ames Internet Exchange because they demonstrate the ba-

[3] The mapping code and application/port list used in this study, as well as the current CAIDA list can be obtained from the authors or by emailing `coral-info@caida.org`.

Trace	Fragmented			Non-Fragmented		
	Pkts(%)	Bytes(%)	Hosts[1](%)	Pkts(%)	Bytes(%)	Hosts[1](%)
CERF-IN	0.675	1.556	0.042	99.325	98.444	99.989
CERF-OUT	0.742	1.283	0.177	99.258	98.717	100.000
SDNAP	0.069	0.090	0.023	99.931	99.910	99.998
MAEWEST-1	0.534	1.459	0.174	99.466	98.541	99.994
MAEWEST-2	0.578	1.573	0.183	99.422	98.427	99.996
AIX-1	0.269	0.835	0.172	99.731	99.165	99.973
AIX-2	0.250	0.590	0.162	99.750	99.410	99.974

TABLE II
PREVALENCE OF FRAGMENTED AND NON-FRAGMENTED IP TRAFFIC

Category						Occurrence(%)
Correct	Complete	In-Order	Reverse	Overlap	Duplicate	# Series
YES	YES	YES	-	-	-	79.922
YES	YES	-	YES	-	-	8.051
-	YES	-	-	-	YES	7.493
YES	YES	-	-	-	-	1.620
-	YES	-	-	YES	YES	1.093
-	-	YES	YES	-	-	1.016
-	-	YES	-	-	-	0.595
-	-	-	YES	-	-	0.111
-	-	YES	YES	-	YES	0.044
-	-	-	-	-	-	0.030

TABLE III
TOP SERIES KINDS FROM ALL SERIES - ACROSS ALL DATASETS

sic properties of fragmented IP traffic as observed on all links studied. We analyzed the size (in bytes) of monitored fragment series, the number of fragments in each series, the sizes of the largest and smallest fragments in each series, and the effects of fragments larger than 1500 bytes.

Bytes per Fragment Series (Figure 10):

The size of the payload carried by each fragment series is highly variable. It has a random component similar to distributions of packet size in general, with a band around the range 1520-1636 of bytes per fragment series. Tunneled traffic is a major cause of series of these sizes. The source host sends these original datagrams at 1500 bytes – the MTU of Ethernet (and many other link types) – and then have between 1 and 4 additional IP (or other) headers prepended. This banding effect and the prevalence of original datagram sizes around 1500 bytes can be seen in Figure 11, an enlargement of the 0-3000 byte range of Figure 10. The most frequently occurring series size across all of the data sets was 1572 bytes. We observe a background, relatively uniform distribution of packet sizes that stretches across series size graphs. In this case, series between 597 and around 4,000 bytes total occurred with a uniform frequency of approximately one hundred series per size. A background level of approximately 10 series spanned the range from 4,000 bytes to 10,000 bytes.

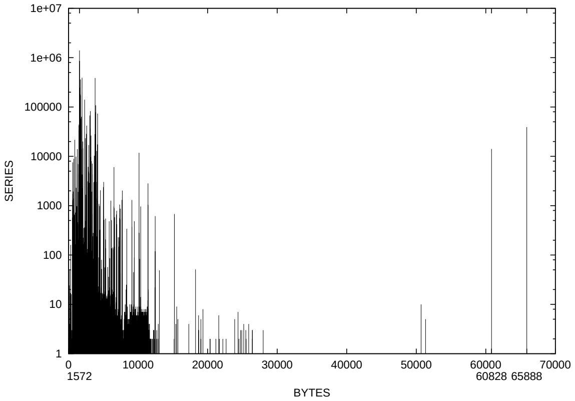
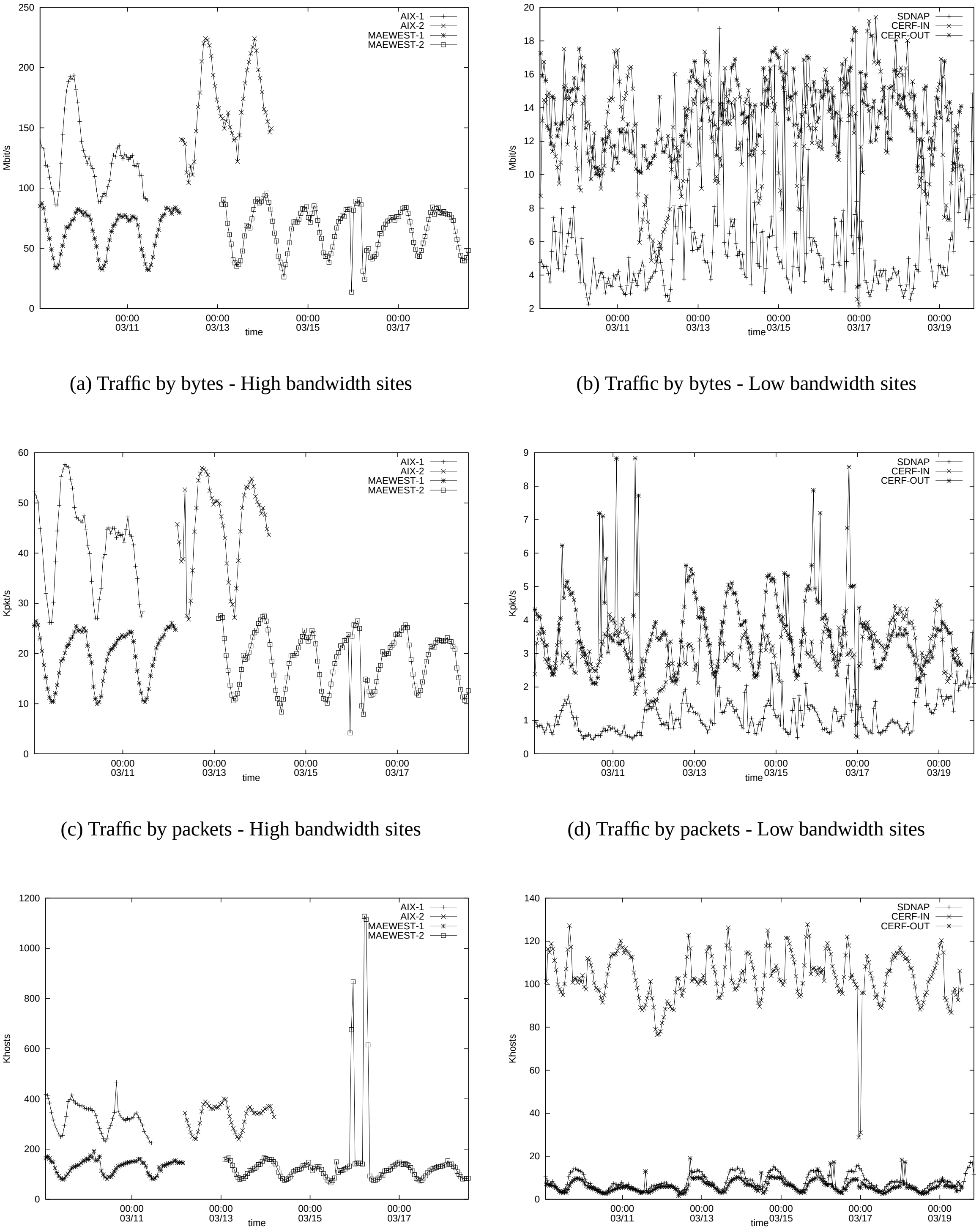


Fig. 10. Number of bytes transmitted per correct series for trace AIX-2. Note this includes the bytes in all of the IP headers for each fragment.



(a) Traffic by bytes - High bandwidth sites

(b) Traffic by bytes - Low bandwidth sites

(c) Traffic by packets - High bandwidth sites

(d) Traffic by packets - Low bandwidth sites

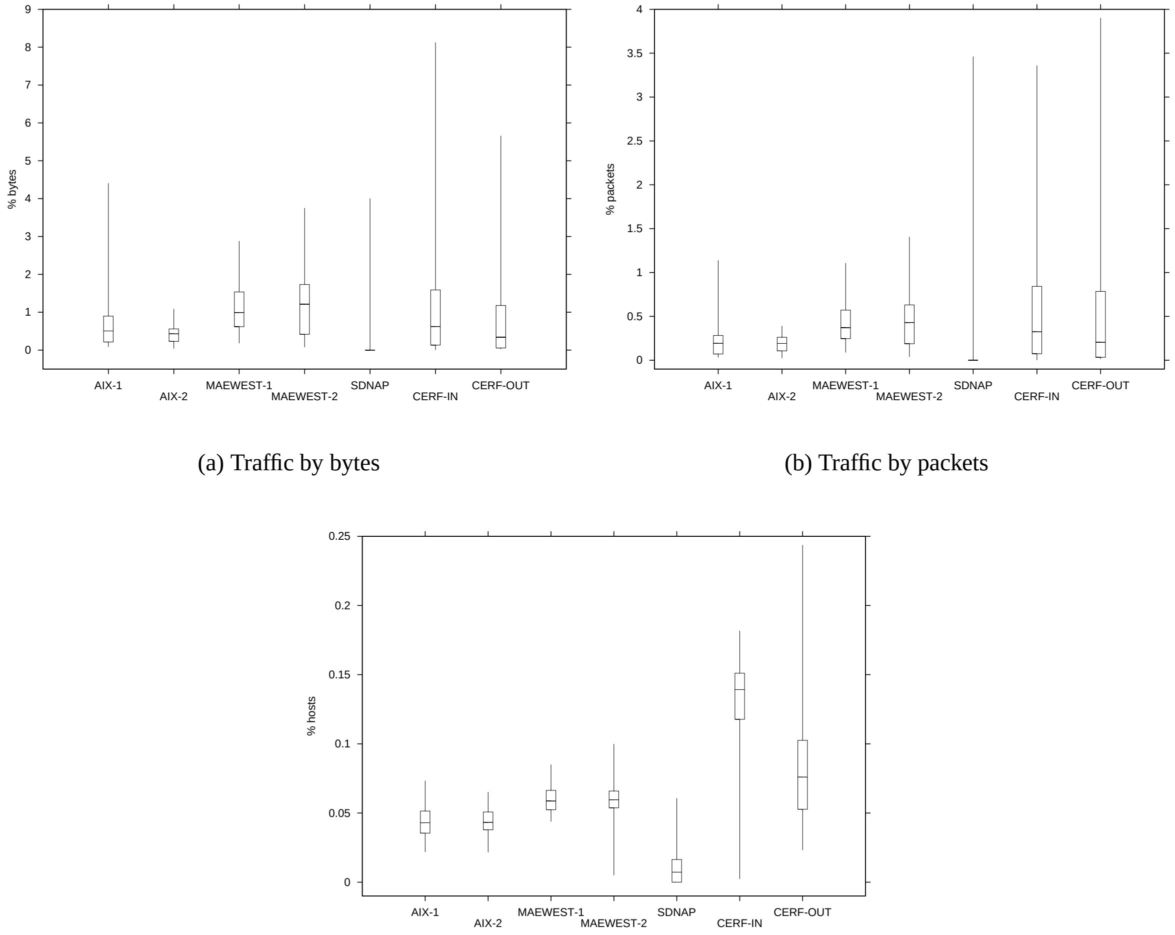
(e) Unique source hosts - High bandwidth sites

(f) Unique source hosts - Low bandwidth sites

Fig. 8. Average hourly bandwith (a,b), packets (c,d), and unique hosts (e,f) for non-fragmented traffic.

Figure 12 shows the overall packet size distribution for this data set, including both fragmented and non-fragmented traffic. All packet sizes above 30 bytes occur at a frequency in excess of 100,000 packets. The most frequently occurring packet size was 40 bytes with 2.69 billion packets, followed by 1500 bytes at 1.49 billion packets and 576 bytes with 514 million packets.

Figure 11 shows evidence of fragmentation caused by



(a) Traffic by bytes

(b) Traffic by packets

(c) Unique hosts

Fig. 9. Percentage of fragmented traffic by (a) bandwidth, (b) packets, or (c) unique hosts for 1 hour intervals for each trace. The candlestick lines show minimum and maximum percentage of traffic seen in an hour, the bottom and top of the box show the 25th and 75th percentiles, and the line inside the box shows the median value.

MTU misconfiguration. We monitored a total of 93 series less than 256 bytes. Indeed, the smallest series, at 92 bytes, had only 52 bytes of payload. The overhead for this series, 40 bytes, is nearly as large the size of the payload. An additional 252 series are considered ‘poorly configured’ because they have series lengths less than 576 bytes. While in a few instances (i.e. routers handling predominantly voice over IP traffic) a low MTU is an optimal configuration, MTUs lower than 576 bytes are generally evidence of mistaken or misguided configuration. Some end hosts that use modem connections with SLIP set low MTUs for their dial-up link; however an MTU of 576 is sufficient to preserve interaction even during large file transfers.

One phenomenon we often observe in series size histograms is a large original datagram occurring at a disproportionally large frequency for its size. These spikes appear to be a transient property of the traffic on each link; they vary in datagram size and magnitude of occurrence over time on the same link, and also vary across wide-area network locations. This data set contains two easily identifiable manifestations of this phenomenon: 14,087 fragment series of 60828 bytes and 39,090 series of 65888 bytes. Because these large datagrams occur on all links monitored, we will make note of the effects of these occurrences throughout the following sections.

These fragment series have the following compositions:

The 60828 byte fragment series consist of 40 fragments of 1500 bytes followed by 1 fragment of 828 bytes, with original datagram length of 60028 bytes. In this case 800 bytes of overhead (60828 - 60028) were caused by the 40 additional IP headers needed to transmit the series.

The 65888 byte fragment series consist of 43 fragments of 1500 bytes followed by 1 fragment of 1388 bytes, with original datagram length of 65028 bytes. In this case 860

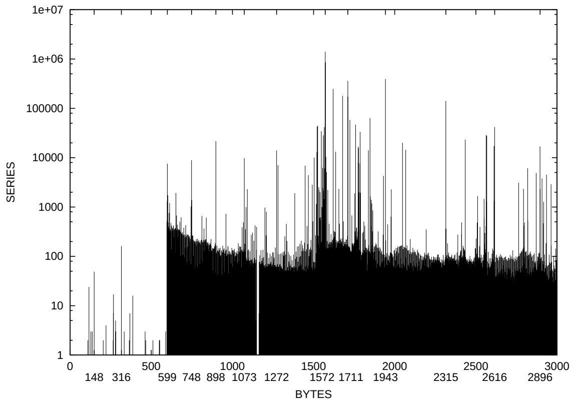


Fig. 11. Enlargement of 0-3000 byte range of the number of bytes transmitted per correct series for trace AIX-2. Note this includes the bytes in all of the IP headers for each fragment.

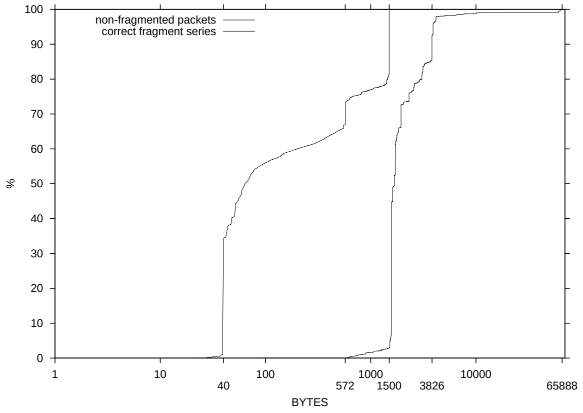


Fig. 12. Number of fragmented and non-fragmented bytes in each IP packet for trace AIX-2.

bytes of overhead (65888 - 65028) were the result of the 43 additional IP headers needed to transmit the series.

Fragments per Fragment Series (Figure 13):

Fragment series are typically two fragments in length. A high number of two-fragment series are consistent with a high volume of tunneled fragmented traffic, since this series length accounts for original datagrams that range from just exceeding the MTU of the next link to forty bytes (for the next packet header) less than double the MTU of the next link:

$$MTU < datagram \leq (2 * MTU) - 40$$

This spike in two-fragment series in Figure 13 is generally followed by decreasing numbers of packets with increasing length of the series. We often observe a pairing of even and odd lengths that results in a step-like decrease in the frequency of occurrence of long fragment series. This behavior can be seen in the pairs (4,5), (6,7), (10,11), (14,15), (21,22), (23,24), and (25,26).

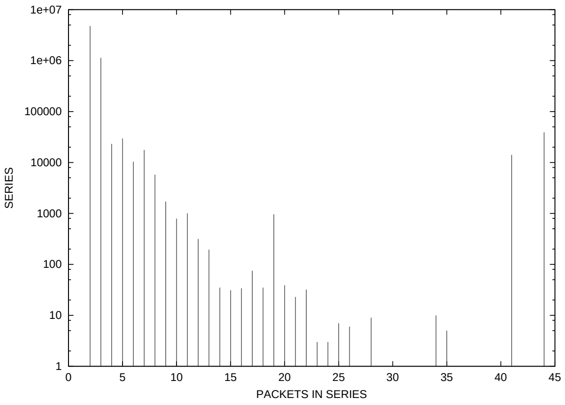


Fig. 13. Number of fragment packets for correct series for trace AIX-2.

We observed an unusually large number of forty-one and forty-four fragment series at AIX because of the unusual frequency of packets of lengths 60028 and 65028 bytes, respectively. These packets were broken up into 1500-byte fragments with one oddly sized leftover fragment.

Largest Fragment Size Distribution (Figure 14):

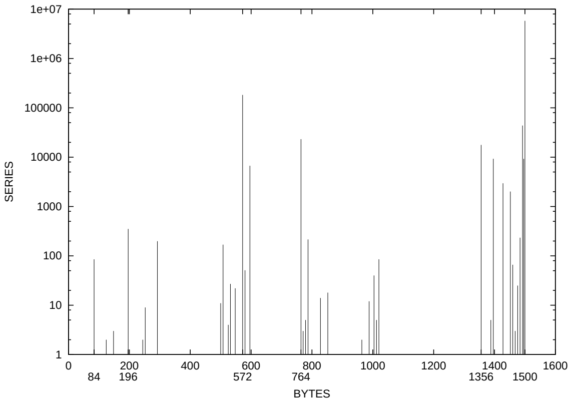


Fig. 14. Largest fragment size for correct series for trace AIX-2.

The size of the largest fragment found in a fragment series is indicative of the MTU of the link requiring fragmentation. Typically the first fragment in a fragment series is equal to this maximum size, but this is not universally true. We identified in the AIX and MAE-west data a total of 237,263 two-fragment series in which the smallest fragment was sent first, with the largest trailing. While only 8.1% of the total correct fragment series were transmitted in reverse order, we cannot make the assumption that the first fragment of each series is always the largest.

The same misconfigurations that were apparent in the bytes per fragment series graph are visible here: it is unlikely that a packet would need to be fragmented to a size less than 576 bytes as it travels towards an exchange point.

However, there are no observable artifacts of the 60028 or 65028 original datagram phenomena in this graph. All of those anomalies result in 1500 byte largest fragments, and since 1500 is by far the most common largest fragment size, the anomalies are not visible in the largest fragment size distribution.

Fragment	Occurrence(%)
Size (bytes)	# Series
1500	85.314
1484	11.112
572	1.186
1492	1.086
1496	0.455
1356	0.183
1396	0.120
124	0.115
764	0.097
1452	0.068

TABLE IV
TOP TEN LARGEST FRAGMENTS FROM CORRECT SERIES - ACROSS ALL DATASETS

Many of the largest fragments occur at sizes easily predicted from the MTUs of common link types. Table IV shows the largest fragment size per series seen across all data sets. 1500 bytes is by far the most common largest fragment size; it is the maximum packet size for Ethernet networks. Ethernet networks using LLC/SNAP, in accordance with RFC 1042 [13] produce 1492 byte IP packets. DEC Gigaswitch traffic results in packets of length 1484 bytes. 572 bytes is a widely used PPP MTU and also results from usage of the default 576 byte transmission size. The largest size packet that a host is required to accept is 576 bytes by RFC 791 [14] and RFC 879 [15], therefore when Path MTU discovery fails or is not implemented, packets are sent at a size less than or equal to 576 bytes. Note for IPv6, the minimum MTU of any link must be 1280 bytes [16].

The default packet packet size of 576 bytes results in fragments of 572 bytes because the length of the payload of each fragment packet except the last must be divisible by eight. This size requirement is based on the design of the IP packet header that specifies that the offset field holds the position of each fragment within the original datagram in eight-byte units[14]. The size of the entire fragment is the sum of the length of the IP header and the payload. Since IP options rarely occur, the IP headers of these fragments are 20 bytes in length. [3] Therefore, the entire packet size for non-last fragments is $20 + N * 8$ for some N. The largest valid fragment packet size less than or equal to the default transmission size of 576 bytes is 572. Such a packet would consist of 20 bytes of IP header and 69 eight-byte units of fragment payload.

Many large fragment sizes evince configuration errors. This is evidence for the utility of Path MTU Discovery, since there is no "safe" transmission size at which a host can send packets to prevent fragmentation without an unacceptable increase in per-packet overhead.

Smallest Fragment Size Distribution (Figure 15):

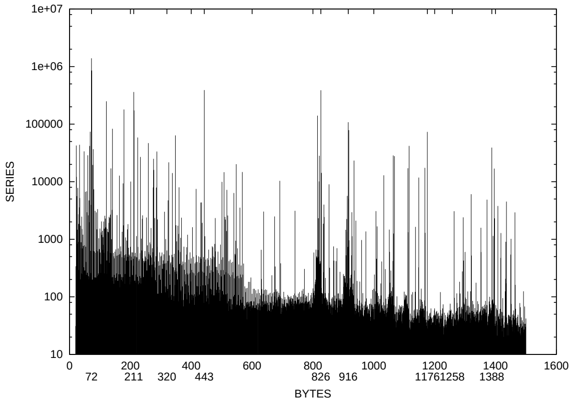


Fig. 15. Smallest fragment size for correct series for trace AIX-2.

The comparatively small variation in the MTUs of common links, in contrast to the wide variations in original datagram sizes, produces a background frequency of approximately one hundred series across a wide variety of smallest fragment sizes. This is the result of the inherent diversity in the sizes of the original datagrams. The background level decreases across the range of packet sizes because the frequency of occurrence of packet sizes decreases with a rate of $(sizeofpacket)^2/2$. In each fragment series, all of the fragments except the last are of uniform size: the MTU of the subsequent link. Thus the diversity in the sizes of the original datagrams manifests itself only in the size of the smallest fragment. The most commonly occurring smallest fragment size is 72, which corresponds to the most common fragment series size at 1572 bytes. After a 1500 byte largest fragment has been composed, 72 bytes is the leftover value. Each spike in the graph corresponds to a frequently occurring combination of original datagram length and MTU of the fragmenting router.

Our 60028 and 65028 byte fragment series anomalies are not visible in this graph. The 60028 byte original datagrams result in smallest fragments that are 320 bytes in size. Likewise, 1258 bytes corresponds to the smallest fragments from the 65028 byte original datagrams. However, since there are only 41 and 45 examples of each unusual series, these occurrences are diluted by the background occurrence of smallest fragment sizes.

The Effects of Fragments Larger than 1500 Bytes

As we have seen in the previous graphs, the most frequently occurring original datagram sizes shape the characteristics of their resulting fragments. Fragment traffic at MAE-west is unusual in that a common largest fragment size for this link is 4348 bytes, rather than the usual sizes smaller than 1500 bytes.

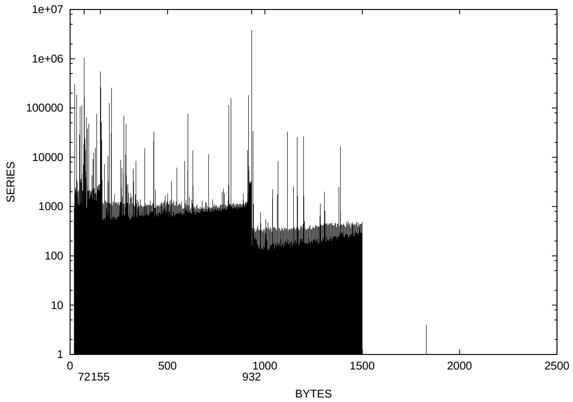


Fig. 16. Smallest fragment size for correct series for trace MAEWEST-1.

Smallest fragments larger than 1500 bytes accompany the largest fragments that are greater than the usual size, as shown in Figure 16. They occur less frequently, however, due to their "leftover" nature. Large packets resulting from a higher MTU do not necessarily have last fragments of increased size. Fragments larger than 1500 bytes also retain the possibility of being fragmented again before they reach their destination.

D. Fragmented Traffic Protocols and Applications

This section examines which services, protocols and applications contribute to fragmented traffic. Values presented are for all data sets combined.

Services Causing Fragmentation

While many hypothesize that NFS causes all of the fragmented traffic on LANs and backbone networks, in our data, fragmented tunneled traffic is the dominant cause of IP packet fragmentation. For example, on the link between the UCSD campus and CERFnet, IPIP tunneled traffic is the largest cause of fragmented traffic by several orders of magnitude. The fragmented tunneled traffic consists of IP packets sized at the MTU of their local network (generally 1500 bytes) which were then tunneled, causing the addition of at least one additional 20 byte IP header. The resulting 1520 byte datagram exceeds the MTU of the subsequent link, and is fragmented into a 1500 byte first fragment and a 40 byte second fragment. This fragmentation may be preventable – a machine that is known to send traffic through an IPIP tunnel could set the MTU of the interface associated with the tunnel to 1480 bytes, rather than 1500. This would reduce the switching load resulting from the tunneled traffic by 98.7% – the machine would generate an extra packet for only every seventy-fifth packet sent, rather than requiring a second packet for every original datagram sent from the machine.

Protocol	Fragmented		
Name	Pkts(%)	Bytes(%)	Series(%)
UDP	68.256	72.033	74.981
IPENCAP	13.857	9.687	11.645
ESP (IPSEC)	3.234	2.273	4.881
ICMP	10.049	12.494	2.546
TCP	1.734	1.588	2.261
GRE	1.162	0.793	1.906
IPIP	0.972	0.669	1.084
AH (IPSEC)	0.399	0.285	0.664
IGMP	0.334	0.177	0.030
AX.25	0.001	0.000	0.001

TABLE VI
PROTOCOL BREAKDOWN FOR FRAGMENTED TRAFFIC. SERIES COLUMN IS FOR CORRECT SERIES ONLY.

Tunneled traffic is not a local phenomenon. The combination of IPENCAP, IPIP, GRE, and UDP-L2TP accounts for 15% of all fragmented traffic – by far the largest single cause of fragmentation. None of these protocols support any form of Path MTU discovery. NFS accounts for only 0.1% of wide-area fragmented traffic. The most frequently fragmented protocol is IGMP – some 78% of IGMP packets are fragments. However, since IGMP accounts for only 0.001% of all traffic, this fact is of purely academic import.

As shown in Table VI, UDP accounts for more fragmented traffic than any other protocol – 68.3% of fragmented traffic, followed by IPENCAP at 13.9%, ICMP at 10.0%, and ESP at 3.2%. Fragmented ICMP traffic consists primarily (98.1%) of echo requests and replies, although a small but significant number of timestamp requests were also monitored. Path MTU Discovery successfully limits the amount of TCP traffic that is fragmented; however, its effects are not quite as ubiquitous as some might claim. More than three million packets over the course of a week, 0.009% of the total TCP traffic, consisted of fragmented packets. Fragmented TCP traffic exists on highly aggregated links.

Protocol		Fragmented		Non-Fragmented	
Name	Number	Pkts(%)	Bytes(%)	Pkts(%)	Bytes(%)
UDP	17	0.300	0.800	12.197	3.713
IPENCAP	4	0.061	0.108	0.123	0.039
ESP (IPSEC)	50	0.014	0.025	0.278	0.277
ICMP	1	0.044	0.139	1.860	0.447
TCP	6	0.008	0.018	84.815	94.213
GRE	47	0.005	0.009	0.176	0.130
IPIP	94	0.004	0.007	0.033	0.021
AH (IPSEC)	51	0.002	0.003	0.053	0.042
IGMP	2	0.001	0.002	0.000	0.000
AX.25	93	0.000	0.000	0.004	0.001

TABLE V

Protocol Breakdown for Fragmented and Non-Fragmented IP Traffic. Percentages are of total traffic.

Protocol	Non-Fragmented	
Name	Pkts(%)	Bytes(%)
UDP	12.251	3.754
IPENCAP	0.123	0.040
ESP (IPSEC)	0.280	0.280
ICMP	1.868	0.452
TCP	85.189	95.271
GRE	0.177	0.132
IPIP	0.033	0.021
AH (IPSEC)	0.053	0.043
IGMP	0.000	0.000
AX.25	0.005	0.001

TABLE VII

Protocol breakdown for non-fragmented traffic.

TCP Application	Occurrence(%)
	# Series
SMTP	53.827
FTP_DATA	32.288
WWW	5.096
NAPSTER_DATA	4.979
Unclassified TCP	2.993
GNUTELLA	0.635
X11	0.070
BGP	0.020
SSH	0.018
KERBEROS	0.008

TABLE VIII

Top TCP applications from correct series - Across All Datasets

TCP Applications (Table VIII)

53.9% of fragmented TCP traffic is composed of SMTP packets. FTP data and WWW follow, with 32.3% and 5.1%, respectively. Napster accounts for 5.0% of all fragmented TCP traffic, and Gnutella produces 0.6%, for a total of 5.6% of fragmented TCP traffic from these two peer-to-peer file-sharing applications. However, we only identify Napster and Gnutella traffic on the most commonly used ports. Because Gnutella servers, and to a lesser extent, Napster servers often use alternate ports (typically to circumvent blocks intended to impede use of these applications), we underestimate, perhaps significantly, the prevalence of both fragmented and non-fragmented peer-to-peer file-sharing application use. The top five most commonly fragmented TCP applications appear in the top six TCP applications overall. WWW traffic is the most common TCP transmission (54.4%), followed by Napster (8.35%)and NNTP (5.90%). FTP data is fourth with 2.94% of all TCP traffic, followed by SMTP at 2.43% and Gnutella at 2.07%. SMTP is actually the most frequently fragmented TCP application, followed by FTP data, Napster, Gnutella, and WWW.

UDP Applications (Table IX)

L2TP accounted for 28.9% of the fragmented UDP traffic from identifiable applications. RealAudio followed close behind with 22.9%. Microsoft's Windows Media Player weighed in with 3.6%, for a total of 26.2% streaming media. The Squid caching protocol (SQUID_ICP) composes 11.7% of the identifiable UDP applications, followed by video-conferencing software CUSEEME at 11.0%. 10.4% of identifiable UDP application traffic composed NFS packets. Quake accounted for 5.4% of iden-

UDP Application	Occurrence(%)
	# Series
Unclassified UDP	98.512
L2TP	0.412
REALAUDIO_UDP	0.326
SQUID_ICP	0.166
CUSEEME	0.157
NFS	0.149
QUAKE	0.077
MS_MEDIA	0.048
HALFLIFE	0.048
DISCARD	0.043

TABLE IX
TOP UDP APPLICATIONS FROM CORRECT SERIES - ACROSS ALL DATASETS

ICMP Application	Occurrence(%)
	# Series
ICMPECHOREQUEST	61.280
ICMPECHOREPLY	36.905
ICMP 13/0	1.767
ICMPNOPORT	0.039
ICMP 11/1	0.004
ICMPNOHOST	0.003
ICMP 3/4	0.002
ICMP 69/0	0.000
ICMPTTL	0.000

TABLE X
TOP ICMP APPLICATIONS FROM CORRECT SERIES - ACROSS ALL DATASETS

Application	Occurrence(%)
	# Series
Unclassified UDP	73.866
ICMPECHOREQUEST	1.560
SMTP	1.217
ICMPECHOREPLY	0.940
FTP_DATA	0.730
L2TP	0.309
REALAUDIO_UDP	0.245
SQUID_ICP	0.125
CUSEEME	0.118
WWW	0.115
NAPSTER_DATA	0.113
NFS	0.111
Unclassified TCP	0.068
QUAKE	0.058
ICMP 13/0	0.045
MS_MEDIA	0.036
HALFLIFE	0.036
DISCARD	0.032
NETBIOS	0.018
DAYTIME	0.015
GNUTELLA	0.014

TABLE XI
TOP APPLICATIONS FROM CORRECT SERIES - ACROSS ALL DATASETS

tifiable UDP traffic. Halflife followed with 3.6%, for a total of 8.8% of identifiable UDP application traffic from video games. Unfortunately, we were unable to classify the majority (73.9%) of UDP traffic. As we have ruled out many possible sources of this traffic, including multicast, we conjecture that dynamic H.323 video-conferencing applications account for a significant portion of the unknown UDP applications.

The top eight applications that generate fragmented UDP traffic appear in the top 15 UDP applications overall. DNS accounted for 26.8% of all (fragmented and non-fragmented) UDP traffic. Halflife followed, with 14.4% of UDP traffic. RealAudio traffic caused 5.7% of all UDP traffic, with Quake accounting for 4.3% and Netbios producing 3.8%. L2TP generated the 8th largest volume of UDP traffic, 0.28%, with SQUID_ICP close behind with 1.1%. CUSEEME was 12th overall, at 0.084%. NFS was the 15th most common cause of all UDP traffic on highly aggregated links, with 0.063% of all packets. NFS is the most frequently fragmented UDP application, followed by CUSEEME, L2TP, Windows Media Player, and SQUID_ICP. Although NFS was had the highest rate of fragmentation of any UDP application, L2TP accounted for more than five times the volume of fragmented traffic. Thus, fragmented tunneled traffic has a far greater impact on wide-area networks than does NFS.

IPv6 and Packet Fragmentation

The next version of the IP protocol, IPv6, eliminates the IP packet fragmentation mechanism in routers [16]. IPv6 also requires a checksum in the UDP header of all UDP packets. The UDP checksum field does appear in the IPv4 UDP header, but its use is optional. One proposed mechanism for bridging IPv4 and IPv6 networks is that UDP packets lacking checksums will have checksums computed and applied before they are transmitted onto IPv6 networks. This process of checksum computation is difficult for fragmented traffic since all of the fragments of the original datagram must be reassembled before a checksum can be computed. If all of the fragments do not share the same egress point from the IPv4 network, checksum computation is impossible. However, we are aware of no available data on the prevalence of IPv4 UDP fragments without UDP checksums. In our data, we observe that only 0.42% of all UDP fragments lacked a UDP checksum. However, 25.5% of all hosts sending fragmented traffic sent UDP packets without checksums. 82.3% of all hosts that sent UDP packets without a checksum also sent UDP packets *with* checksums. These results is consistent with application-specific checksum incorporation, rather than host-specific behavior, which complicates a user-transparent IPv4 to IPv6 transition.

V. Conclusion

Many assertions about the nature and extent of fragmented traffic are based in folklore, rather than measurement and analysis. Common beliefs includes: fragmented traffic is decreasing in prevalence or nonexistent, fragmented traffic exists only on LANs (due to NFS) and not on backbone links, misconfiguration causes most fragmentation, and only UDP traffic is fragmented.

While the majority of fragmented traffic is UDP (68% by packets and 72% by bytes), ICMP, IPSEC, TCP, and tunneled traffic are fragmented as well. Tunneled traffic is the single largest cause of fragmented traffic, and accounts for at least 16% of packets and 11% of bytes.

NFS accounts for only 0.1% of fragment series seen. We were unable to classify the applications associated with most UDP traffic because of the use of ephemeral ports and dynamically exchanged ports. The classifiable UDP traffic was comprised primarily of tunneled, streaming media and game traffic.

Fragmented traffic does occur regularly at highly aggregated exchange points as well as on access links. Fragmented traffic is detrimental to wide-area network performance. Fragmented traffic causes increased load on routers, through both the division of the original packet and the increased number of packets handled by all subsequent routers. The traffic also causes increased load on links due to the overhead of an extra IP header for each fragment. Additionally, because all of the fragments are necessary to reassemble the original packet, the probability of successfully delivering a fragmented packet exponentially decreases as a function of the number of fragments, in contrast to the normal packet loss rate. This partial packet loss may further increase link and router loading as higher layers retransmit packets.

With the advent of IPv6, all packets that are currently fragmented will be dropped by routers, with a “Packet Too Big” ICMP message returned to the source host [17]. The proposed mechanism for transition between IPv4 and IPv6 networks requires checksums for all fragmented UDP traffic, yet 26% lacks a UDP checksum. Understanding the actual prevalence and causes of fragmented traffic is critical to the success of currently proposed protocols and security efforts.

VI. Acknowledgments

We would like to thank Ryan Koga for writing the program used to collect data for this study, and Ken Keys for suggestions and help using CoralReef. This paper would not have been possible without the assistance of Steve Feldman and Bobby Cates with the MAE-west and AIX passive monitors. We are especially grateful for the assistance of Daniel Plummer, David Meyer, Bill Fenner, Craig Partridge, Ken Keys and the reviewers for useful suggestions and in the editing of this paper. As always, the support of the CAIDA staff was invaluable.

References

[1] C. A. Kent and J. C. Mogul, “Fragmentation considered harmful,” *WRL Technical Report 87/3*, Dec. 1987.

[2] G. P. Chandranmenon and G. Varghese, “Reconsidering fragmentation and reassembly,” in *PODC: 17th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 1998.

[3] Sean McCreary and k claffy, “Trends in wide area IP traffic patterns: A view from Ames Internet Exchange,” in *ITC Specialist Seminar on IP Traffic Modeling, Measurement and Management*, Sept. 2000.

[4] J. C. Mogul and S. E. Deering, “RFC 1191: Path MTU discovery,” Nov. 1990.

[5] “CERT Advisory CA-1997-28 IP Denial-of-Service Attacks,” `http://www.cert.org/advisories/CA-1997-28.html`.

[6] Jason Anderson, “An Analysis of Fragmentation Attacks,” Mar. 2001, `http://www.sans.org/infosecFAQ/threats/frag_attacks.htm`.

[7] S. McCanne, C. Leres, and V. Jacobson, *libpcap*, Lawrence Berkeley Laboratory, Berkeley, CA, available via anonymous ftp to ftp.ee.lbl.gov.

[8] Waikato Applied Network Dynamics group, “The DAG project,” `http://dag.cs.waikato.ac.nz/`.

[9] Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and k claffy, “The architecture of CoralReef: an Internet traffic monitoring software suite,” in *PAM2001 — A workshop on Passive and Active Measurements*. CAIDA, Apr. 2001, RIPE NCC, http://www.caida.org/tools/measurement/coralreef/.

[10] “IANA port assignments,” `ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers`.

[11] David Moore, Geoffrey M. Voelker, and Stefan Savage, “Inferring Internet Denial-of-Service Activity,” *Usenix Security Symposium*, 2001.

[12] Yin Zhang, Vern Paxson, and Scott Shenker, “The stationarity of internet path properties: Routing, loss, and throughput,” *ACIRI Techinical Report*, May 2000.

[13] J. Postel and J. K. Reynolds, “RFC 1042: Standard for the transmission of IP datagrams over IEEE 802 networks,” Feb. 1988.

[14] J. Postel, “RFC 791: Internet Protocol,” Sept. 1981.

[15] J. Postel, "RFC 879: The TCP Maximum Segment Size and Related Topics," Nov. 1983.

[16] S. Deering and R. Hinden, "RFC 2460: Internet Protocol, Version 6 (IPv6) specification," Dec. 1998.

[17] A. Conta and S. Deering, "RFC 2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," Dec. 1998.