Beyond Folklore: Observations on Fragmented Traffic

Colleen Shannon, David Moore, k claffy

Abstract— **Fragmented IP traffic is a poorly understood 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, in particular, the effects of NFS, streaming media, networked video games, tunneled traffic, and 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 four sources. These sources include a university commodity access link, two highly aggregated commercial exchange points, and a local NAP. Although there is no practical method of ascertaining whether any data provide a representative sample of all Internet traffic, we include data sources that cover several different types of WANs with traffic from commercial entities, educational and research institutions, and large government facilities.

The dominant causes of fragmentation are streaming media and tunneled traffic. Although rumored to be the main impetus for IP packet fragmentation, NFS is not among the top ten causes.

Keywords—**fragmentation, fragment, measurement, traffic measurement, TCP/IP**

I. Introduction

The Internet protocol (IP) was designed to facilitate communication between heterogenous networks. It serves as a least-common-denominator protocol that allows computers differing in architectures, operating systems, and applications, connected by varying routes, paths, and protocols, to exchange information. IP must be able 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 destination 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 intranetwork 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. Although current technology mitigates some of the described problems with consumption of bandwidth, packet switching, and CPU resources, the overall argument that fragmentation is detrimental remains valid. 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. The router must process the fragment with its main CPU, rather than utilizing the specialized hardware in a line card, commonly called the "fast path". This CPU-intensive operation may impair 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 between the initial router and the end host. Finally, once the fragments reach their destination they must be reassembled by the end host. The loss of any fragment causes the destination host to drop the entire packet. This in turn

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.

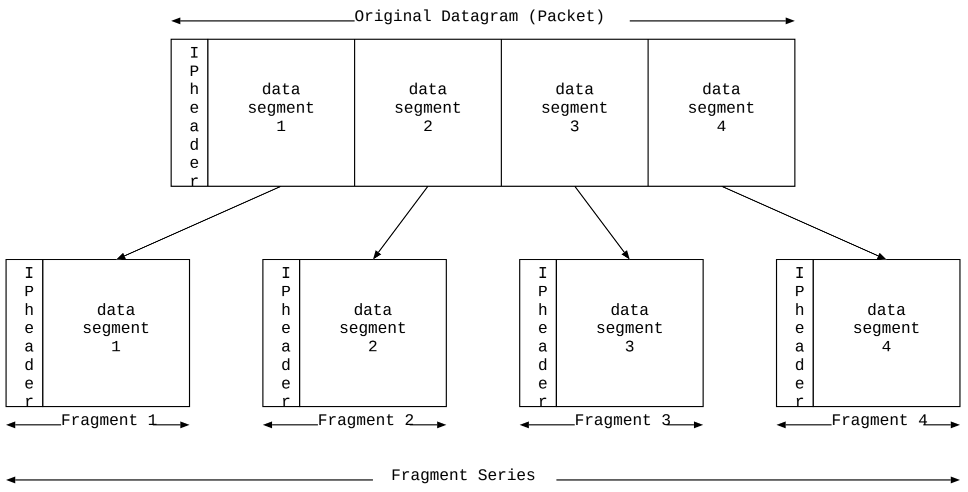


Fig. 1. Composition of a fragment series.

forces the source host to repeat transmission of a datagram that will likely 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 no longer exists. 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.

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 derived from 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. A *data segment* is a portion of the original packet payload that becomes the payload of a single fragment.

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. 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. RFC 791 [5] does not specify how fragments must be sized, other than that the payload of all of the non-last fragments must be a multiple of 8 bytes in length.

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.

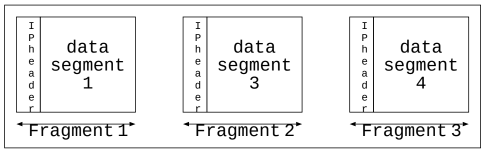


Fig. 2. Example incomplete series.

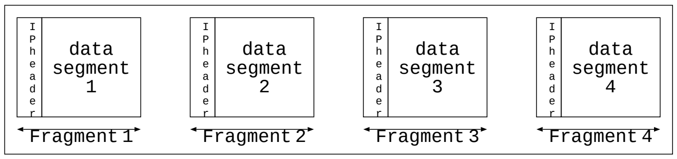


Fig. 3. Example in-order series.

A series is *in-order* (Figure 3) if the fragments are observed arriving sequentially; i.e., 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. Only one fragment needs to be delivered out of order for us to observe a reverse-order two-fragment series. 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. 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.

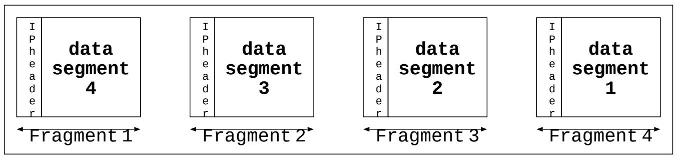


Fig. 4. Example reverse-order series.

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

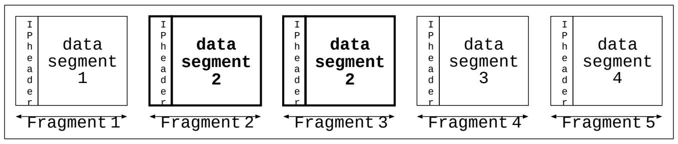


Fig. 5. Example duplicate series.

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 [6][7] sends large fragments that are overlapping except for a single byte, thereby exhausting buffer resources in certain fragment reassembly implementations.

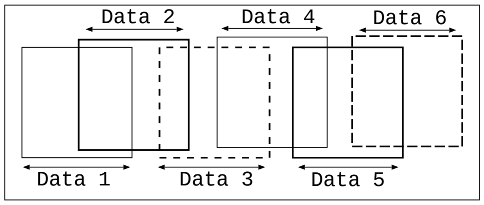


Fig. 6. Example overlapping series.

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

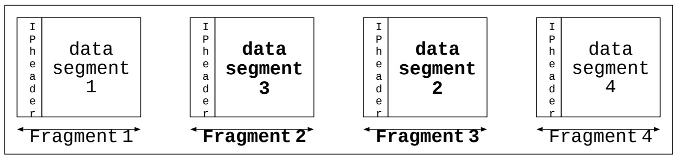


Fig. 7. Example correct series. Note that this series is *not in-order.*

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 OC12 card [8] 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` [9] and an off-the-shelf 100Mbit ethernet card to monitor this traffic. Using a FORE ATM OC3 card [10], 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 OC3 card [11].

The numbers of unique source hosts for each data set (shown in Table I) were limited to hosts that 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.

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

Traffic Monitoring

Due to the high volume of traffic at some of the measurement sites, a specialized tool, `crl_frag_capture`, collected the data for this study. `crl_frag_capture` relies on the CoralReef [12] software suite for header capture, interval handling and data aggregation. We gleaned only packet headers; we attempted no analysis of the payload portion of each packet. We organized the data we collected into hour-long intervals for post-processing. We collected four types of data:

frags.pcap — a full header trace in `libpcap` [9] format containing only fragmented traffic packets (either offset $>$ 0 or 'more fragments' bit 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 to keep the tuple table size from exceeding memory space on the machine. 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 only 19 entries in our tuple table. This method maintains the port that matches the traffic to a specific application, while discarding the dynamically generated, meaningless port. Additionally, all ports above 32767 were bucketed as 32768, since the ports in this range

[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.

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 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. 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.

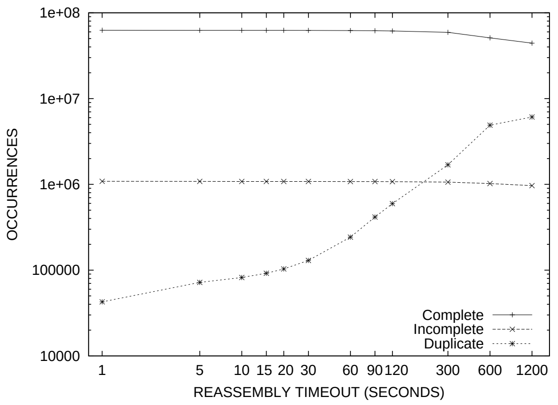


Fig. 8. The effects of various reassembly timeouts on complete, incomplete, and duplicate fragment series.

The grouping of fragments into series is sensitive to the chosen reassembly timeout. On one hand, we wish to provide sufficient time for all fragments in each series to be monitored even with significant network delays. However, we also need to account for the possibility of a wraparound of the IP ID field. While we would not ordinarily expect the IP ID field to wrap in a short period of time, there are a few cases in which we do monitor duplicate IP ID fields from a single host in a short period of time. For example, we observed tunnel ingress points that generated sufficient traffic to wrap their IP ID fields in only a few minutes. If the timeout is too lengthy, the likelihood of incorrectly assigning fragments from disparate original datagrams into the same fragments series increases. As shown in Figure 8, the number of duplicate fragments in each series (which can be an indicator of IP ID wraparound) increases across tested fragment series timeouts. Conversely, the number of incomplete series decreases with increasing timeout magnitude. Because there exists no point at which erroneous duplicates are minimized while complete series are maximized, we have chosen to use a timeout of 15 seconds, as it is the maximum advisable delay before reassembly recommended in RFC 791 [5].

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 92 application to port mappings, including common well known ports from the IANA port assignment list [13], as well as emerging multimedia, file sharing, and video game applications (such as RealAudio, Quake, Napster, eDonkey2000, and FastTrack (KaZaA)). 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 using well-known ports. As described in Section IV-D, we were able to use control traffic to identify almost all applications using ephemeral ports.

IV. Results

A. *Overall trends in Fragmented Traffic*

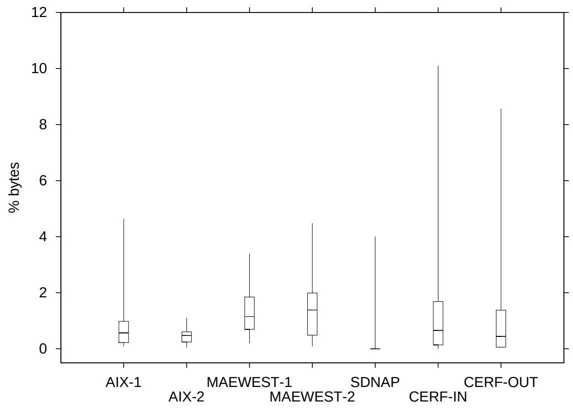
Table II shows the percentage of fragmented and non-fragmented traffic found in each data set. We observed hosts sending both fragmented traffic and non-fragmented traffic, so the host percentages may total more than 100%. Although the overall volume of fragmented traffic was small, it was also highly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

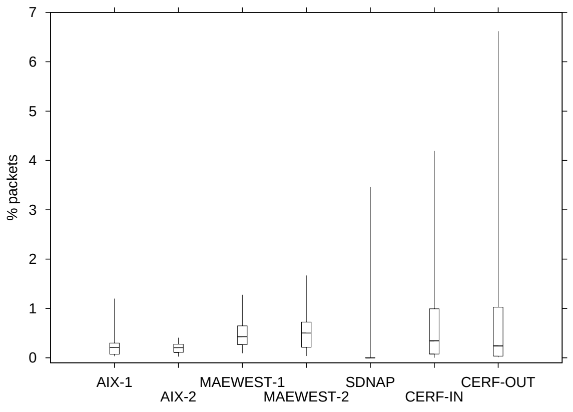
[3]The mapping code and application/port list used in this study (from CoralReef 3.5.2), as well as the current CAIDA list, can be obtained from the authors or by emailing `coral-info@caida.org`.

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

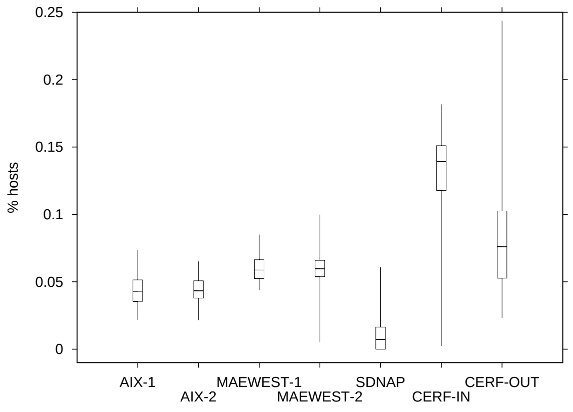
TABLE II
PREVALENCE OF FRAGMENTED AND NON-FRAGMENTED IP TRAFFIC



(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.

ture of traffic at the other two locations, although it does show a daily decrease in traffic late at night (Pacific Standard Time). Figure 10 shows time series plots of the non-fragmented traffic. Note that Figures 10(e) and 10(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 [14].

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, and duplicate. Of all series, 98.3% are complete, meaning they contain sufficient information to reconstruct the original datagram. Correct series (Figure 7) account for 98.2% of all series. Of complete series, 90.4% are in-order (Figure 3) and 7.9% are reverse-order (Figure 4). 0.1% of all complete series are either overlapping (Figure 6) or duplicate (Figure 5) series; both are attributes that impede exact determination of ordering. Of all complete series, 0.002% have overlapping fragments and 0.15% contain duplicates.

Of all monitored series, 1.6% are correct series that were neither in-order nor reverse-order; they were likely reordered in transit. In May 2000, Paxson et al [15] observed that approximately 0.3% of all packets arrive out of order. Thus it appears that fragmented traffic has a greater probability of being reordered by the network 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 test this hypothesis.

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 of the first received packet 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 graphs generated from data collected at the Ames Internet Exchange because they demonstrate the basic 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.

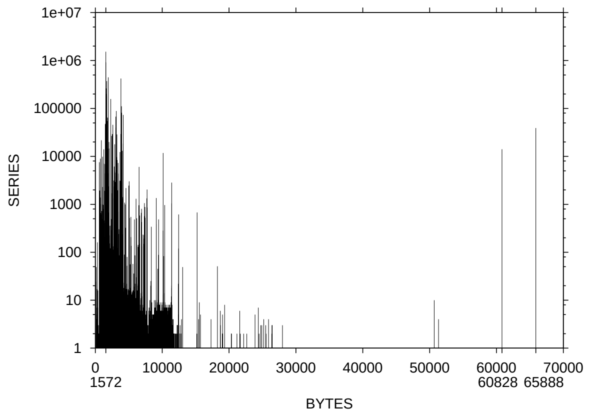


Fig. 11. Number of bytes transmitted per correct series for trace AIX-2. Note this includes the bytes in all of the IP headers for each fragment.

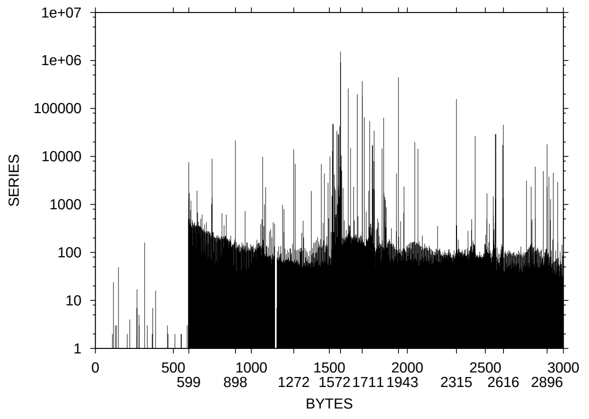
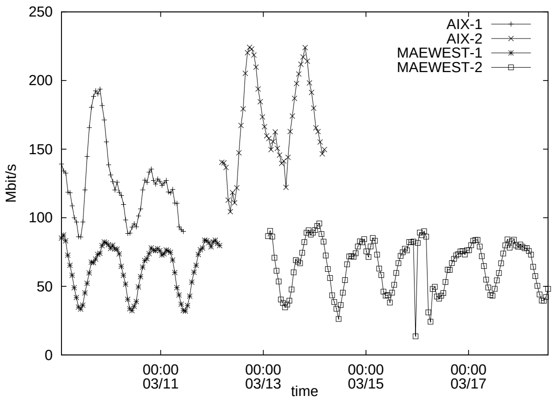


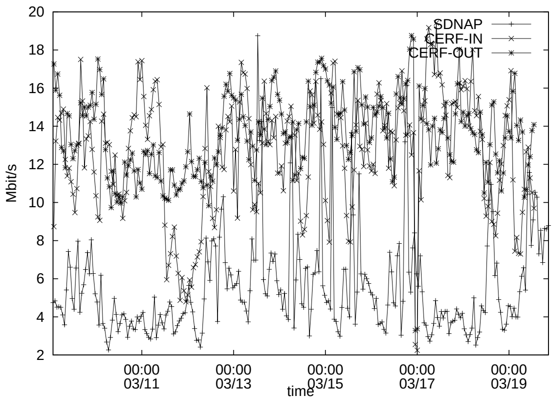
Fig. 12. Enlargement of 0-3000 byte range of the number of bytes transmitted per correct series for trace AIX-2. Note this includes the bytes in all of the IP headers for each fragment.

Bytes per Fragment Series (Figure 11)

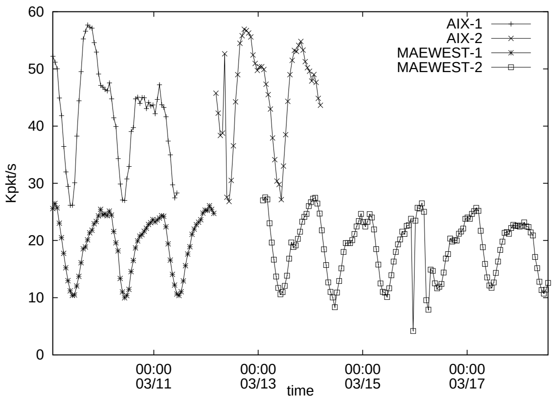
The size of the payload carried by each fragment series is highly variable. It has a random component similar to distributions of packet size in general, with a band between 1520 and 1636 bytes per fragment series. Tunneled traffic is a major cause of fragment series in this size range. The source host sends these original datagrams at 1500 bytes – the MTU of Ethernet (and many other link types) – and then they have between 1 and 4 additional IP (or other) headers prepended at the tunnel ingress point. This banding effect and the prevalence of original datagram sizes around 1500 bytes can be seen in Figure 12, an enlargement of the 0-3000 byte range of Figure 11. 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 with total sizes between 597 and around 4,000 bytes occurred with a uniform frequency of



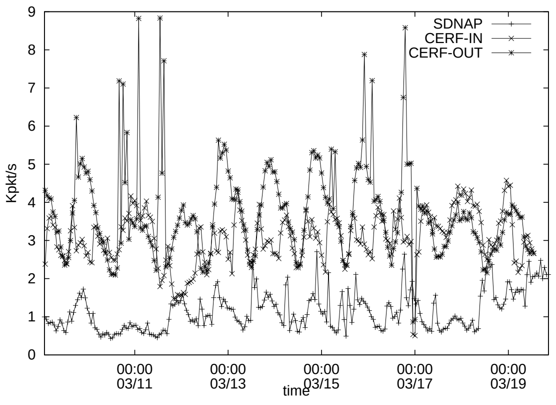
(a) Traffic by bytes - High bandwidth sites



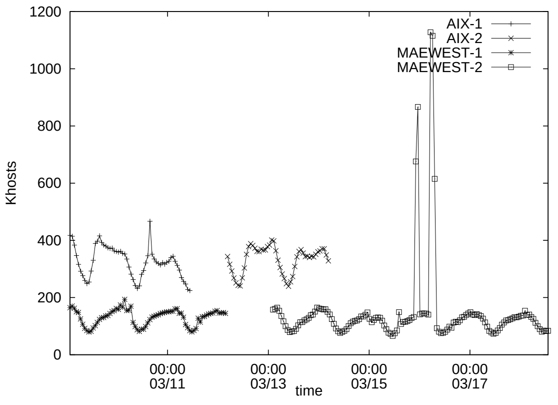
(b) Traffic by bytes - Low bandwidth sites



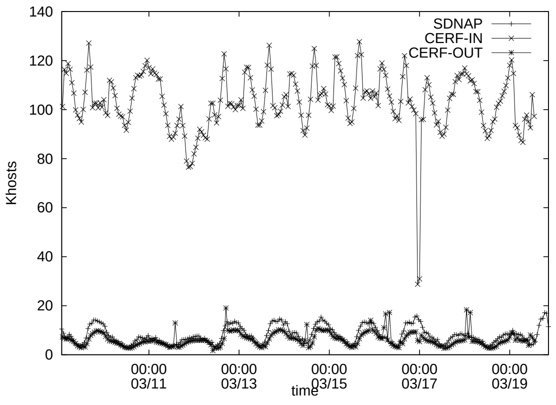
(c) Traffic by packets - High bandwidth sites



(d) Traffic by packets - Low bandwidth sites



(e) Unique source hosts - High bandwidth sites



(f) Unique source hosts - Low bandwidth sites

Fig. 10. Average hourly bandwidth (a,b), packets (c,d), and unique hosts (e,f) for non-fragmented traffic.

Category						Occurrence	
Correct	Complete	In-Order	Reverse	Overlap	Duplicate	# Series	% Series
YES	YES	YES	-	-	-	56,411,943	89.
YES	YES	-	YES	-	-	4,936,210	7.8
YES	YES	-	-	-	-	1,028,327	1.6
-	-	YES	YES	-	-	653,167	1.0
-	-	YES	-	-	-	339,362	0.53
-	YES	-	-	-	YES	89,288	0.14
-	-	-	YES	-	-	69,627	0.11
-	-	-	-	-	-	17,626	0.028
-	-	YES	YES	-	YES	990	0.0016
-	YES	-	-	YES	YES	974	0.0015

TABLE III
TOP SERIES KINDS FROM ALL SERIES

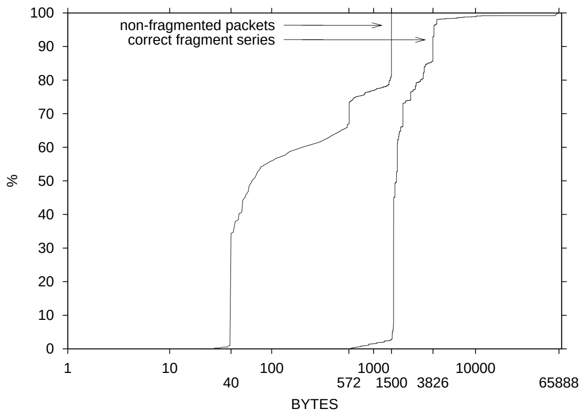


Fig. 13. Cumulative distribution of fragmented and non-fragmented bytes in each IP packet for trace AIX-2. For fragmented traffic this includes the bytes in all of the IP headers for each series.

approximately one hundred series per size. A background level of approximately 10 series spanned the range from 4,000 bytes to 10,000 bytes.

Figure 13 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 12 shows evidence of fragmentation caused by MTU misconfiguration. We monitored a total of 93 series less than 256 bytes. While two of these series appeared to be deliberate optimizations for slow links, the majority appear to be errors. 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 (e.g., 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 make modem connections via SLIP set low MTUs for their dial-up link; however, an MTU of 576 is often sufficient to preserve interaction even during large file transfers. The additional overhead incurred should be considered whenever a smaller MTU is chosen: 7% of 576 byte TCP packets is used by headers necessary for delivery of the packet, while 16% of 256 byte packets and 31% of 128 byte packets constitute overhead. To deliver the payload of a 576 byte TCP packet, three 256 byte packets (generating 80 extra bytes) or seven 128 byte packets (generating 240 extra bytes) are necessary.

One phenomenon we often observe in series size histograms is a large original datagram occurring at a frequency disproportionate to 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 39,114 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. Both of these sets of large series are ICMP echo requests and thus may have been “ping of death” attacks[1] [16].

[1]Machines running older versions of many Operating Systems can be crashed by sending them a ping packet larger than 65,535 bytes. Because few protocols allow packets that large, the “ping of death” packet

These fragment series have the following compositions:

Each 60828 byte fragment series consists 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.

Each 65888 byte fragment series consists of 43 fragments of 1500 bytes followed by 1 fragment of 1388 bytes, with original datagram length of 65028 bytes. In this case 860 bytes of overhead (65888 - 65028) were the result of the 43 additional IP headers needed to transmit the series.

Fragments per Fragment Series (Figure 14)

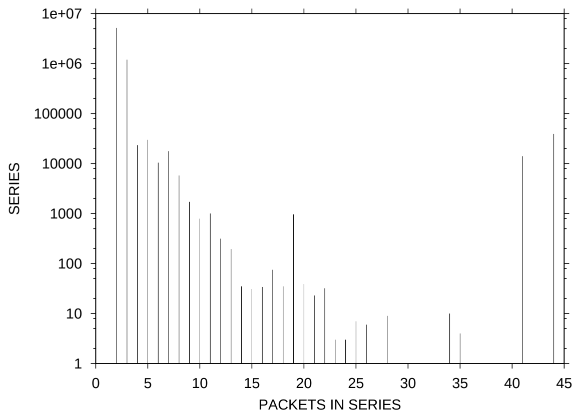


Fig. 14. Number of fragment packets for correct series for trace AIX-2.

Fragment series are typically two fragments in length. A high number of two-fragment series is 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 14 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).

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.

typically arrives as a fragment series, is reassembled by the target machine into the original 65,535 byte packet, and then crashes the target machine

Largest Fragment Size Distribution (Figure 15)

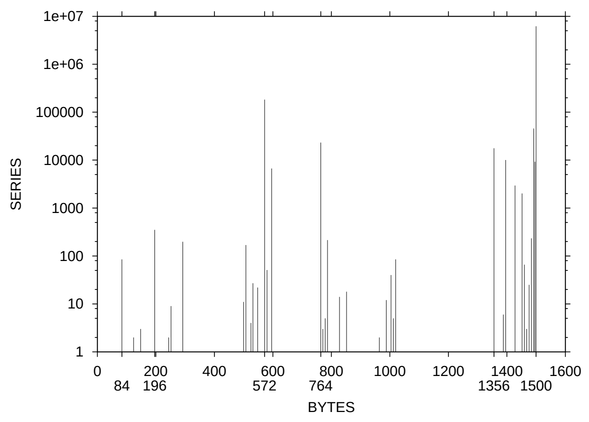


Fig. 15. Largest fragment size for correct series for trace AIX-2.

The size of the largest fragment found in a fragment series is indicative of the MTU of the link provoking fragmentation. Typically the first fragment in a fragment series has 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 received first, with the largest trailing. While only 7.8% 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.

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 [17] 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. According to RFC 791 [5] and RFC 879 [18], the largest size packet that a host is required to accept is 576 bytes.

Protocol		Fragmented		Non-Fragmented	
Name	Number	Pkts(%)	Bytes(%)	Pkts(%)	Bytes(%)
UDP	17	0.30	0.80	12.	3.7
IPENCAP	4	0.061	0.11	0.12	0.039
ESP (IPSEC)	50	0.014	0.025	0.28	0.28
ICMP	1	0.044	0.14	1.9	0.45
TCP	6	0.0076	0.018	85.	94.
GRE	47	0.0051	0.0088	0.18	0.13
IPIP	94	0.0043	0.0074	0.033	0.021
AH (IPSEC)	51	0.0018	0.0032	0.053	0.042
IGMP	2	0.0015	0.0020	0.0004	<0.0001
AX.25	93	<0.0001	<0.0001	0.0045	0.0014

TABLE V

PROTOCOL BREAKDOWN FOR FRAGMENTED AND NON-FRAGMENTED IP TRAFFIC. PERCENTAGES ARE OF TOTAL TRAFFIC.

Fragment Size (bytes)	Series(%)
1500	79.
1484	18.
1492	1.0
572	0.96
1496	0.37
1356	0.15
1396	0.10
124	0.093
764	0.081
1452	0.055

TABLE IV

TOP TEN LARGEST FRAGMENTS FROM CORRECT SERIES

Therefore when Path MTU discovery fails or is not implemented, packets are sent at a size less than or equal to 576 bytes. Note that for IPv6, the minimum MTU of any link must be 1280 bytes [19].

The default 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[5]. 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 bytes. 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.

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. The MAE-west location monitored for this study is an ATM link, and the MTU for IP over ATM is 9180 bytes [20][21]. Fragments larger than 1500 bytes are more likely to be fragmented again before they reach their destination than smaller fragments.

D. Fragmented Traffic Protocols and Applications

This section examines the services, protocols and applications that contribute to fragmented traffic.

Services Causing Fragmentation

While many hypothesize that NFS causes all of the fragmented traffic on LANs and backbone networks, in our data streaming media and tunneled traffic are the dominant cause of IP packet fragmentation.

Streaming media accounts for 53% of all fragmented traffic. A single application in this group, Microsoft Media Player, is responsible for the 52% of all fragmented traffic. The use of a protocol that utilizes Path MTU Discovery could significantly reduce the impact of this application on packet fragmentation.

The other major culprit, fragmented tunneled traffic, consists of IP packets sized at the MTU of their local network which were then tunneled, causing the addition of at least one additional IP header. For example, a 1500 byte packet with a 20 byte IP header added as it is tunneled via IPENCAP results in a 1520 byte datagram that exceeds the MTU of the subsequent link, and is fragmented into a 1500 byte first fragment and a 40 byte second fragment. This fragmentation is entirely preventable – an end host that is known to send traffic through an IP 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.

Path MTU discovery should allow the end host to discover an MTU that minimizes fragmentation of its tunneled traffic. However, 98.98% of the fragmented IPENCAP traffic monitored on the link between UCSD and CERFnet in the course of this study consisted of an IP packet with its Don't Fragment bit set, encapsulated by an IP header with no Don't Fragment bit. The end host believes that it is correctly performing Path MTU Discovery, and is oblivious to the fact that packets it sends through the tunnel are fragmented.

To prevent unnecessary fragmentation, implementations of tunneling protocols need to perform Path MTU discovery between the tunnel ingress point and the tunnel egress point and correctly forward ICMP "datagram too big" messages back to source hosts [22]. As the use of tunneling protocols becomes increasingly widespread, increased fragmentation caused by improperly implemented tunneling software may cause performance problems.

Tunneled traffic is not a local area network phenomenon. The combination of IPENCAP, IPIP, GRE, and UDP-L2TP accounts for 22% of all fragment series – the second largest single cause of fragmentation. None of these protocols currently implement any form of Path MTU discovery. NFS accounts for only 0.1% of wide-area network fragment series. The most frequently fragmented protocol is IGMP – some 78% of IGMP packets are fragments. However, since IGMP accounts for only 0.0004% of all measured packets, this fact is of purely academic import.

As shown in Table VI, UDP accounts for more fragmented packets than any other protocol – 68.3% of fragmented traffic. 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 does exist on highly aggregated links.

Protocol	Fragmented		
Name	Pkts(%)	Bytes(%)	Series(%)
UDP	68.	72.	70.
IPENCAP	14.	9.7	19.
ESP (IPSEC)	3.2	2.3	4.4
ICMP	10.	12.	2.1
TCP	1.7	1.6	2.0
GRE	1.2	0.79	1.6
IPIP	0.97	0.67	1.0
AH (IPSEC)	0.40	0.29	0.54
IGMP	0.33	0.18	0.024
AX.25	0.0009	0.0001	0.0011

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

Protocol	Non-Fragmented	
Name	Pkts(%)	Bytes(%)
UDP	12.	3.8
IPENCAP	0.12	0.040
ESP (IPSEC)	0.28	0.28
ICMP	1.9	0.45
TCP	85.	95.
GRE	0.18	0.13
IPIP	0.033	0.021
AH (IPSEC)	0.053	0.043
IGMP	0.0004	<0.0001
AX.25	0.0045	0.0015

TABLE VII
PROTOCOL BREAKDOWN FOR NON-FRAGMENTED TRAFFIC.

TCP Applications (Table VIII)

50.8% of fragmented TCP series are composed of SMTP packets. Two peer-to-peer file-sharing applications, Napster and Gnutella, account for a total of 5.3% of fragmented TCP traffic, despite being a larger portion of non-fragmented traffic. 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 circum-

TCP Application	Series(%)
SMTP	51.
FTP_DATA	36.
HTTP	4.8
NAPSTER_DATA	4.7
Unclassified TCP	2.8
GNUTELLA	0.60
X11	0.066
BGP	0.019
SSH	0.017
KERBEROS	0.0079

TABLE VIII
TOP TCP APPLICATIONS FROM CORRECT FRAGMENT SERIES

vent 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.

ICMP Application	Series(%)
Echo Request	61.
Echo Reply	37.
Timestamp Request	1.8
Port Unreachable	0.039
TTL Expired – Reassembly	0.0041
Host Unreachable	0.0028
Frag Needed But "DF" Set	0.0020
Type 69, Code 0	0.0003
TTL Expired – Transit	0.0001

TABLE IX
TOP ICMP APPLICATIONS FROM CORRECT FRAGMENT SERIES

ICMP Applications (Table IX)

The majority (98%) of ICMP traffic observed was associated with echo request and reply. 13% of ICMP echo requests were series of 65888 bytes (previously discussed in Section IV-C). The 1.8% of fragmented ICMP traffic which was a timestamp request appears to have been either a misconfiguration or denial of service attack. An ICMP timestamp request packet is normally 20 bytes plus the IP header and thus should never be larger than 80 bytes. The observed fragmented ICMP timestamp traffic predominantly used payloads of 5013 bytes with less than 1% using 4013 and 7513 bytes. In some instances multiple source hosts were sending the the same type of 5013 byte ICMP timestamp messages to the same destination, suggesting a denial of service attack.

UDP Applications (Tables X and XI)

UDP Application	Series(#)	Series(%)
MS_MEDIA	6,461,564	82.
UNKNOWN	1,370,495	17.
ASHERONS	12,132	0.15
TALK	12,057	0.15
ISAKMP	7,543	0.095
BIFF	5,871	0.074
DAYTIME	4,067	0.051
POINTCAST	4,038	0.051
ICU_II	3,030	0.038
NFS	3,027	0.038

TABLE X
THE TOP TEN UDP APPLICATIONS USING EPHEMERAL PORTS.

Ninety-eight percent of all fragmented UDP traffic occurs on dynamically allocated ports. To identify the applications associated with traffic on these ephemeral ports, we must match these ambiguous fragments with control traffic on known ports. However, because our initial study did not include a packet-level trace of unfragmented traffic, we collected 16 hours (including business hours) of TCP, UDP, and ICMP flow data from our MAEWEST, UCSD, and SDNAP taps on February 26, 2002. The distribution of applications in this follow-up study was similar to that of the traces used in the earlier study. Specifically, Unclassified UDP traffic accounted for 93.11% of all observed UDP traffic, a reduction of 5% from the previous study. The utilization of a few applications, including the games Quake, Doom, and Halflife, Microsoft's Media Player, and AOL increased, otherwise the overall distribution of applications remained the same.

At each location, we built a tuple table based upon the source IP address, destination IP address, protocol, and ports of each measured flow. We identified UDP flows with fragmented packets and extracted all other flows involving the same source and destination IP that occurred within 120 seconds of the fragmented flow. We explored the effect of the timeout window size on the number of matched and unmatched fragment series. We chose 120 seconds as the window size for this study because it minimizes unmatched fragment series while minimizing multiple matches for each fragment series.

When we generated multiple matches for a single unknown UDP series despite tuning the window size, we attempted to determine which of the matches was the most likely control stream for the fragments. We removed duplicate matches and eliminated a few common applications (traceroute, http, netbios) that are never control streams for UDP traffic on ephemeral ports. If multiple matches remained, we chose the match that occurred most commonly on its own. For example, if we narrowed the choices to Microsoft's RealMedia Player (81%) and Squid (<0.1%), we'd assign the unidentified fragments to Microsoft's RealMedia Player. Varying our method of resolving multiple matches has no significant effect on the final results.

As seen in Table XI, the vast majority of fragmented UDP traffic appears to be caused by Microsoft's Windows Media Player. Other culprits include the games Quake, Doom, and Asheron's Call, and L2TP, a tunneling protocol.

We also categorized applications identified via well-known port numbers into thirteen groups[4] based on similarities among application functions. The top ten application groups contributing to fragmented traffic, as measured in February 2002, are shown in Table XI. While Streaming applications are the primary contributors of fragmented traffic, the effects of tunneled traffic are under-represented in this table because several tunneling protocols (IPSEC, GRE) are not identifiable via port numbers.

IPv6 and Packet Fragmentation

The next version of the IP protocol, IPv6, eliminates the IP packet fragmentation mechanism in routers [19]. 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. This result is consistent with application-specific checksum incorporation, rather than host-specific behavior, which complicates a user-transparent IPv4 to IPv6 transition.

[4]Application Groups: Conferencing, Encryption, File Systems, File Transfer, Games, Login, Mail/News, Network Infrastructure, Other, P2P, Streaming, Tunneling, and WWW

V. Conclusion

Many assertions about the nature and extent of fragmented traffic are based in folklore, rather than measurement and analysis. Common beliefs include: 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 commonly fragmented as well. Microsoft's Media Player is the single largest source of fragment series, accounting for 52% seen in this study. Tunneled traffic is a major cause of fragmented traffic, and accounts for at least 16% of fragmented series.

NFS accounts for only 0.1% of fragment series observed. We were unable to classify the applications associated with a small percentage of 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. Although fragmented traffic is a small percentage of traffic overall, its prevalence is highly variable; fragmented traffic accounted for 8% of all packets during hour-long periods on some links. The bursty nature of fragmented traffic makes it difficult to measure accurately with short samples of network traffic.

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 must retransmit packets.

With the advent of IPv6, all packets that are currently fragmented in the network will be dropped by routers, with a "Packet Too Big" ICMP message returned to the source host [23]. The proposed mechanism for transition

Application	Series(%)
MS_MEDIA	78.
UNKNOWN	16.
QUAKE	2.4
L2TP	1.7
DOOM	0.86
REALAUDIO_UDP	0.19
ASHERONS	0.14
TALK	0.14
AOL	0.11
ISAKMP	0.090

Application Group	Series(%)
Streaming	78.
UNKNOWN	16.
Games	3.4
Tunneling	1.9
Other	0.28
Network_Infrastructure	0.25
File_Systems	0.14
Encryption	0.12
Conferencing	0.10
WWW	0.076

TABLE XI
THE TOP TEN TCP AND UDP APPLICATIONS AND APPLICATION GROUPS OVERALL (INCLUDING APPLICATIONS ON EPHEMERAL PORTS IDENTIFIED VIA CONTROL STREAMS).

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 the first part of this study, and Ken Keys for suggestions and help using CoralReef. This paper would not have been possible without the assistance of Steve Feldman, Bobby Cates, and Joe Pucik with the MAE-west and AIX passive monitors. We are especially grateful for the assistance of Daniel Plummer, David Meyer, Bill Fenner, Evi Nemeth, Craig Partridge, Ken Keys, Geoff Voelker, and the SIGCOMM Internet Measurement Workshop reviewers and participants, and the ACM Transactions on Networking reviewers for useful suggestions for 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] J. Postel, “RFC 791: Internet Protocol,” Sept. 1981.
[6] “CERT Advisory CA-1997-28 IP Denial-of-Service Attacks,” `http://www.cert.org/advisories/CA-1997-28.html`.
[7] Jason Anderson, “An Analysis of Fragmentation Attacks,” Mar. 2001, `http://www.sans.org/infosecFAQ/threats/frag_attacks.htm`.
[8] Applied Telecom (apptel), “Product overview: Pci optical interface network transceiver,” `http://www.apptel.com/pdf/ov_point.pdf`.
[9] S. McCanne, C. Leres, and V. Jacobson, *libpcap*, Lawrence Berkeley Laboratory, Berkeley, CA, available via anonymous ftp to ftp.ee.lbl.gov.
[10] Marconi Corporation, “Marconi - ForeRunner 200e nics,” `http://www.marconi.com/html/solutions/forerunner200enics.htm`.
[11] Waikato Applied Network Dynamics group, “The DAG project,” `http://dag.cs.waikato.ac.nz/`.
[12] 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/.
[13] “IANA port assignments,” `ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers`.
[14] David Moore, Geoffrey M. Voelker, and Stefan Savage, “Inferring Internet Denial-of-Service Activity,” *Usenix Security Symposium*, 2001.
[15] Yin Zhang, Vern Paxson, and Scott Shenker, “The stationarity of internet path properties: Routing, loss, and throughput,” *ACIRI Techinical Report*, May 2000.
[16] Malachi Kenney, “Ping of death,” `http://www.insecure.org/sploits/ping-o-death.html`.
[17] J. Postel and J. K. Reynolds, “RFC 1042: Standard for the transmission of IP datagrams over IEEE 802 networks,” Feb. 1988.
[18] J. Postel, “RFC 879: The TCP Maximum Segment Size and Related Topics,” Nov. 1983.
[19] S. Deering and R. Hinden, “RFC 2460: Internet Protocol, Version 6 (IPv6) specification,” Dec. 1998.
[20] R. Atkinson, “RFC 1626: Default IP MTU for use over ATM AAL5,” May 1994.
[21] M. Laubach and J. Halpern, “RFC 2225: Classical IP and ARP over ATM,” Apr. 1998.
[22] C. Perkins, “RFC 2003: IP Encapsulation within IP,” Oct. 1996.
[23] A. Conta and S. Deering, “RFC 2463: Internet Control Mes-

sage Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," Dec. 1998.