A parameterizable methodology for Internet traffic flow profiling

Kimberly C. Claffy, Hans-Werner Braun, George C. Polyzos

Abstract— **We present a parameterizable methodology for profiling Internet traffic flows at a variety of granularities. Our methodology differs from many previous studies that have concentrated on end-point definitions of flows in terms of state derived from observing the explicit opening and closing of TCP connections. Instead, our model defines flows based on traffic satisfying various temporal and spatial locality conditions, as observed at internal points of the network. This approach to flow characterization helps address some central problems in networking based on the Internet model. Among them are route caching, resource reservation at multiple service levels, usage based accounting, and the integration of IP traffic over an ATM fabric. We first define the parameter space and then concentrate on metrics characterizing both individual flows as well as the aggregate flow profile. We consider various granularities of the definition of a flow, such as by destination network, host-pair, or host and port quadruple. We include some measurements based on case studies we undertook, which yield significant insights into some aspects of Internet traffic, including demonstrating (i) the brevity of a significant fraction of IP flows at a variety of traffic aggregation granularities, (ii) that the number of host-pair IP flows is not significantly larger than the number of destination network flows, and (iii) that schemes for caching traffic information could significantly benefit from using application information.**

I. Introduction

The current Internet aggregates flows of traffic among many end systems, users, and applications. Characterizing the nature of these flows will be critical to accommodating their increasing number and diversity. The goal of this paper is to present a parameterizable methodology for profiling Internet traffic flows, and to utilize some results from case studies to demonstrate its usefulness through (1) observations that are generated using this methodology, and (2) explicit examples of how to use the methodology to optimize the carriage of IP traffic over various fabrics, such as traditional IP routers, possibly with address caching optimizations, and IP routers connected to an ATM network.

Our methodology for profiling flows is inspired by the *packet train* model of packet arrivals, which Jain and Routhier offered to describe traffic on a token ring local area network [1]. They defined a *packet train* as a burst of packets arriving from the same source and heading to the same destination. If the spacing between two packets exceeds some inter-train gap, they are said to belong to separate trains. The packet train model reflects the fact that much of network communication involves many packets spaced closely in time between the same two endpoints.

Another suggested definition of a flow derives from a different motivation: the need for service functionality inherently incongruous with the datagram architecture of the Internet. Clark [2] proposes:

> ...a better building block than the datagram for the next generation of architecture. The general characteristic of this building block is that it would identify a sequence of packets traveling from the source to the destination, without assuming any particular type of service... I have used the word "flow" to characterize this building block. It would be necessary for the gateways to have flow state in order to remember the nature of the flows which are passing through them...

Clark refers to this concept of *soft state* as a potential method of achieving the "goals of survivability and flexibility, while at the same time doing a better job of dealing with the issue of resource management and accountability" [2].

The two motivating factors, the packet train phenomenon and the ability to support special service capabilities, are not unrelated. The ability of equip-

This research is supported by a grant of the National Science Foundation (NCR-9119473)

Claffy and Braun are with Applied Network Research, San Diego Supercomputer Center, San Diego, CA 92186-9784, email: kc,hwb@sdsc.edu

George Polyzos is with the Computer Systems Laboratory, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0114, email: polyzos@cs.ucsd.edu

ment in a datagram network such as the Internet to maintain soft state will be directly related to the number and intensity of network flows, the nature of service they require, and their distribution in geographic space, or locality characteristics. Understanding the effect of the packet train phenomena on router behavior, and vice-versa[1] will be essential to optimizing router efficiency. Although many have extended the packet train model of flows to the transport or application layers [5], [6], [7] or focused only on TCP traffic flows [8], [9], we offer a comprehensive methodology of timeout-based flow characterization, primarily at the IP layer, for use in datagram environments. In particular, we investigate the effect of the range of several flow parameters, such as the flow granularity and timeout, in a variety of environments.

Our methodology also differs from previous studies that have concentrated on end-point definitions of TCP flows, such as by the SYN and FIN control mechanism[2] of the TCP protocol [4]. The strength of a TCP SYN/FIN based approach is that the beginning and end of a connection based flow are unambiguous, independent of the location of the observation. However several other factors, described in section II-D, motivate an alternative approach. Most notably, not all traffic uses transport layer protocols that support SYN and FIN functionality. In order to maintain generality across all traffic, we do not associate flows with connections, but rather define flows based on traffic satisfying various temporal and spatial locality conditions, as observed at internal points of the network. That is, we ground the definition of a flow only in the appearance of packets within a given time interval to, from, or between entities, as perceived at a given network measurement point. This approach to the definition and characterization of network flows can address some central Internet problems, including performance requirements of routers at Internet hotspots, route caching, resource reservation at multiple service levels, usage based accounting, and the transport of IP traffic over an ATM fabric. A principal objective is finding an appropriate balance between retaining vs. recreating flow state, with otherwise the former leading to excessive use of resources for stale information, and the latter in thrashing by excessively removing and recreating such flow state information.

[1] Router behavior tends to intensify packet train phenomena [3], [4].

[2] The SYN packet serves to establish virtual connections and to synchronize packet sequence numbers during the opening of a TCP connection. The FIN packet serves to clear the connection.

While this paper principally focuses on methodology, we are also demonstrating its applicability to measurements at different locations in the Internet fabric. Our metrics fall into two categories: metrics of individual flows, and metrics of the aggregate traffic flow.

In section II we formally define flows and discuss several aspects of flow structure that frame our analysis. In section III we focus on metrics and evaluate the parameter space of individual flows, including flow volume, in packets and bytes, and flow duration. In section IV we present aggregate flow metrics, as seen from the network perspective, which include parameters of the flow arrival process. Applying the methodology to our measurements yields significant insights into Internet traffic characteristics, which we review in section V. Details of our instrumentation for deriving the flow information for our case studies is included in Appendix A. There, we describe the environments in which we apply our methodology and the trace-driven analysis procedure that allows us to characterize flow state information from specified transit points within the network.

II. Parameterizable definition of flows

In this section we specify the definition of parameterizable *flows* and then discuss four aspects of a flow that structure flow measurement and subsequent analysis.

We ground our model of a flow on actual traffic activity from one or both of its transmission endpoints as perceived at a given network measurement point. A flow is *active* as long as observed packets that are meeting the flow specification are observed separated in time by less than a specified timeout value, as figure 1 illustrates. The lower half of the figure depicts multiple independent flows, of which many thousands may be active simultaneously at wide area network transit points.

Our definition of the *timeout* is similar to that used in other studies of timeout-based traffic behavior [8], [1], [9], [7], [10], although most studies adopt a single timeout value rather than investigating the effect of varying it across a range as in our study. Jain and Routhier originally selected for their investigation of local network traffic a timeout of 500 milliseconds.

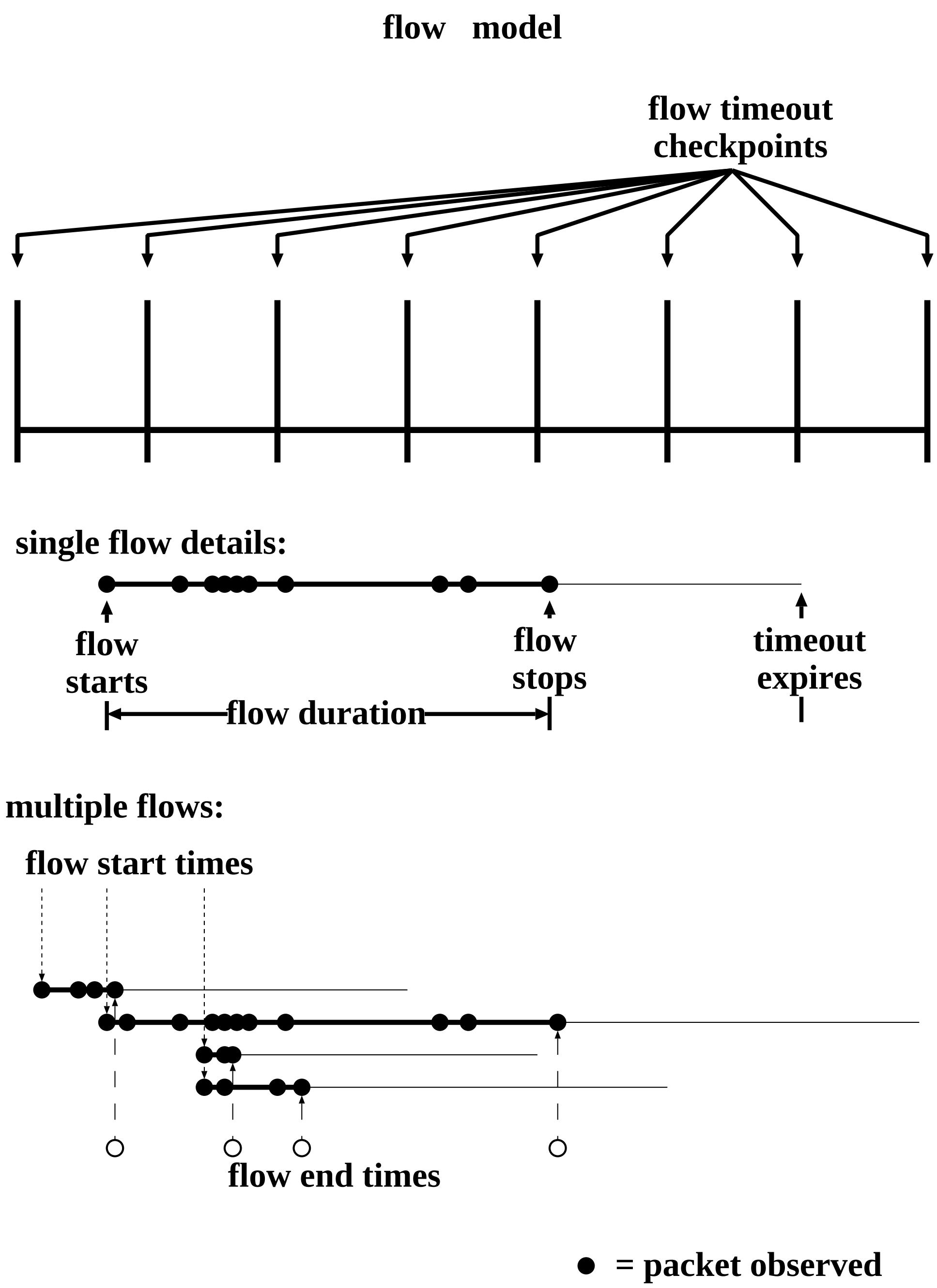

Fig. 1. defining a flow based on timeout during idle periods

For their studies of wide-area traffic at the transport layer, Caceres *et al.* [8] used a 20-minute timeout, motivated by the FTP idle timeout value of 15 minutes, and after comparison to a 5-minute timeout yielded minimal differences. Estrin and Mitzel [9] also compared timeouts of 5 and 15 minutes and found little difference in conversation duration at the two values, but chose to use a timeout of 5 minutes. Acharya and Bhalla [7] used a 15-minute timeout.

In contrast, in our case studies we are seeking to systematically explore a larger range of the time parameter, ranging from 2 seconds to 2048 seconds by powers of two, in order to determine its effect on flow characteristics, e.g., the accompanying tradeoff in router requirements between flow setup and flow maintenance.

This timeout-based flow definition allows flexibility in how one further specifies a flow. In particular, we describe four aspects of a flow that structure a *flow specification*:[3] directionality, one-sided vs. two-sided, endpoint granularity, and functional layer. The remainder of the paper then explores how these aspects interact with the timeout-based definition.

A. Flow directionality

First, one can define a flow as unidirectional or bidirectional. While the connection-oriented TCP traffic generally exhibits bidirectionality, it often exhibits very significant asymmetries in the traffic profile of the two directions. Each TCP flow from A to B also generates a reverse flow from B to A, at the very least for small acknowledgement packets.

In this study we define flows as unidirectional, i.e., bidirectional traffic between A and B would show up as two separate flows: traffic from A to B, and traffic from B to A. While the effect of bidirectionality of flows is generally important to investigate, unidirectional flows are more relevant to the issues that motivate us, among them routing optimization, accounting, and multiplexing IP on top of an ATM substrate [12], [13], [14]. Furthermore the unidirectional flows can be transformed into bidirectional flows during the analysis process, based on actual requirements at that time.

[3] Partridge [11] uses the term *flow specification* to refer to a data structure used by internetwork hosts to specify the nature of service needed, likely involving guarantees about how the internetwork will handle some of the hosts' traffic. His definition is motivated by the future need to request services on behalf of distributed applications such as multimedia conferencing. As we will describe in this section, our definition is different and is grounded in the requirements of the network rather than the end user.

B. One versus two endpoint aggregations of traffic

This second aspect of a flow is related to the first. We distinguish between single and double endpoint flows, that is, flows aggregated at the source or the destination of the traffic, versus flows defined by both the source plus the destination. An example is the difference between all traffic to a given destination network number, versus all traffic from another given network number and also to the same network number, that is, with a common network number pair.

In this study we explore both single and two endpoint flows. The single sided definitions specify the source or destination host or network number, while the two-sided definitions use an IP network number pair, a host pair, or process identifiers, consisting of source and destination host plus source and destination application identifier (i.e., UDP/TCP port number).

C. Flow endpoint granularities

The third aspect of a flow is the endpoint granularity, or the extent of the communicating entities. Potential granularities include aspects such as traffic by application, end user, host, IP network number, Administrative Domain (AD), backbone client service provider, external interface of a backbone node, backbone node, single backbone at large, or multi-backbone environment (e.g., of different agencies). These granularities do not necessarily have an inherent order, as a single user or application might straddle several hosts or even several network numbers. One example flow granularity of interest derives from the fact that IP routers make forwarding decisions based on routing tables which contain next-hop information for a given destination network, a task implicitly grounded in one-sided destination network layer flows at the granularity of IP network number. Eventually, as policy routing issues render the source as well as the destination of a packet relevant to routing decisions, the issue of two-sided flow assessment will also become important. Furthermore, as new routing mechanisms utilize alternative hierarchical definitions related to IP network numbers (e.g., CIDR masks [15]), the desired granularity will have to evolve somewhat.

Network service providers may want to define coarser-grained flows to aggregate network number pairs for which they create virtual circuits crossing their transit network, for example an ATM cloud, with each of the circuits possibly bundling many finer-grained IP flows. Conversely, a more detailed granularity would be necessary for providing special service to a single instance of an application, e.g., a videoconference.

These examples illustrate the importance of flexibility in the parameterization of a flow model. We ground a flow specification in the requirements of the network, and even allow at any point in the network for multiple simultaneous flow specifications. One may want to assume flows: by destination network address for routing; by process pair for accounting; by source address for accounting and policy routing; by destination address or host or network address pair for bundling flows across ATM virtual circuits; or by address plus precedence information for flows at multiple priority levels. For our case studies we selected the granularities of network and host; to highlight certain issues we also use IP address/port quadruples.

D. Protocol layer

Finally, there is the functional, or protocol, layer of the network flow. For example, one could define flows at the application layer. Alternatively one could use the transport connection, e.g., via SYN and FIN packets of the TCP protocol which support explicit connection setup and teardown. In order to maintain generality across all traffic, we consistently do not associate flows specifically with virtual connections, but rather define flows based on packet transmission activity based on specified endpoints at the network layer. Such a flow definition will not have a one-to-one mapping to active TCP connections; under certain conditions a single flow could include multiple active TCP connections, or a TCP connection may be contained in multiple observed flows over time. TCP traffic may furthermore be interleaved with UDP traffic, or a flow may consist entirely of non-TCP traffic.

Several factors motivate our decision to restrict ourselves to an observed state model, all reflective of one circumstance: the Internet is inherently a connectionless datagram environment, and thus connection-oriented information cannot always be assumed available. For example, at transit points within an IP network, equipment is only expected to have access to IP level information, and reliance on higher level information may be inappropriate as a requirement, especially as the network evolves towards new applications that may utilize any available technology.

As a further example, if routes change during the lifetime of a connection, some routers will carry datagrams for flows for which they never observed the transport layer SYN/SYN-ACK packets, and other routers that did see earlier datagrams in a flow will never see the FIN/FIN-ACKs. State information that is dependent on this data will be vestigial in some cases and unavailable in others. Fragmentation will also pose a problem, since all but the first fragment lack the TCP/UDP port information, making it impossible to track fragmented packets to a higher layer flow.[4] Routing and accounting are other functions that will often occur at the IP layer of a distributed datagram network, with no consideration for transport level connections. Therefore management of them must be independent of explicit transport connections.

A recent proposal for flow labels [16] highlights the need for end systems to identify and recognize "a sequence of packets sent from a particular source to a particular (unicast or multicast) destination that require special handling by the intervening routers." A flow label field would obviate the need for the strict connect-data-disconnect phases (SYN/FIN functionality of TCP), since each packet, or every nth packet, or periodic control packets (e.g., RSVP) can establish flow state. Flow labels can also reflect a coarser grain, e.g., aggregating multiple transport connections, or a finer grain, e.g., to support different qualities of service for different packets within a single transport connection, such as with a videoconference where loss of audio is much more detrimental than of video. Since the flow label assumes no router intelligence regarding the endpoint addresses and IP-TCP/UDP address-port quadruples, it allows improved switching time. However, as with other proposed definitions [11] not all traffic will make use of flow IDs; indeed often the network will be able to more efficiently use resources if it can decide itself what constitutes a flow.

[4] An additional constraint is that many wide area environments of today would have to rely on sampling for operational flow assessment, in which case SYN/FIN requirements would lead to "losing" flows, including high volume ones, or requiring a timeout anyway, for the cases of lost FIN packets.

Finally, new technologies for link level traffic forwarding, e.g., based on ATM, may not have access to higher layer information; with Internet related transmission decisions typically having to rely only on IP level information. In particular, until end-to-end ATM is a reality, IP gateways attached to ATM style networks will have to multiplex IP traffic onto the ATM substrate. Mapping higher level (IP) flows to underlying link level virtual circuits [12], [13], [14], [17] will require effective setup, maintenance, timeout strategies, and accounting schemes, most critical at the border gateways. One requirement will be graceful handling of idle connections, regardless of transport layer behavior. Service providers may charge for these idle connections, therefore terminating and restarting of circuits in real-time will have to be efficient [18]. In fact, measurements that we will present in this paper indicate that a very high potential burden for circuit management in an ATM switch will likely come from non-connection-oriented traffic, e.g., non-TCP flows, microscopic in duration and volume. Network layer flow assessment will thus be essential to graceful adaptation between the IP and ATM layers, where such microscopic flows should perhaps be mapped to already existing PVCs, rather than specifically allocating bandwidth and circuits for them.

The four aspects we have described – directionality, one-sided vs. two-sided aggregation, endpoint granularity, and functional layer – structure our selection of which flow specifications we examine.

III. Metrics of individual flows

To illustrate the metric space we varied several parameters in our case studies, with details of the data collection being described in Appendix A:

- flow timeout (we used timeouts from 2 to 2048 seconds);
- traffic aggregation specification (such as destination network (*dn*), destination host (*dh*), source host (*sh*), network pair (*np*), host pair (*hp*));
- the network environment (we used five sites, measured for two separate hours each);
- network usage information above the IP layer, such as transport or application information.

A. Parameter space

In order to isolate the effects of the parameter under investigation, we fix parameters values to a selected default, while we vary the specific one of interest. The values we use for these defaults are: a timeout of 64 seconds, a host pair endpoint granularity, the UC-NSF PM network environment, and all traffic, i.e., not split up by protocol.

A.1 Flow timeout

Figure 2 shows, for a range of flow timeouts, the cumulative distributions of flow byte volume, flow packet volume, and flow duration. The byte values include the protocol headers of IP and above, e.g., TCP, UDP, or ICMP; the minimum size is 28 bytes, for example for ICMP messages with a 20 byte IP header. All three graphs in the figure use the host pair flow granularity, i.e., aggregate all host pair flows regardless of application type, while using the UC-NSF PM data set. The data indicate that the 80th percentile of the flows reflects about 40 packets or less, and about 3.2 kilobytes of data or less. Calculating an average packet size per flow out of this data, one can see that this average is considerably less than the overall average packet size observed in the Internet environment, which is due to an often strong domination of flows by frequent short, name server transactions. For timeout values of 64 seconds or less, 90% of the flows show less than 50 packets, 5.5 kilobytes and 100 seconds of duration. For a 2048 timeout, essentially infinite relative to the 3600-second data duration of our case studies, 27% of the flows consist of a single packet of less than one hundred bytes.

We list in the legend of figure 2 the number of flows in the hour-long data set given each timeout value. Shorter flow timeouts tend to split longer flows into several short ones, so naturally smaller timeouts will yield a much larger number of flows, and a greater proportion of flows of smaller duration. The top line in each graph corresponds to a 4-second timeout value.

A.2 Flow aggregation specification

Figure 3 shows cumulative distributions of flow byte volume, flow packet volume, and flow duration for multiple flow aggregation specifications, or endpoint granularities. The graphs include source host, destination host, destination network, network pair, and host pair aggregations. We use the UC-NSF PM data set with a 64 second flow timeout and aggregate all flows of a given granularity regardless of application

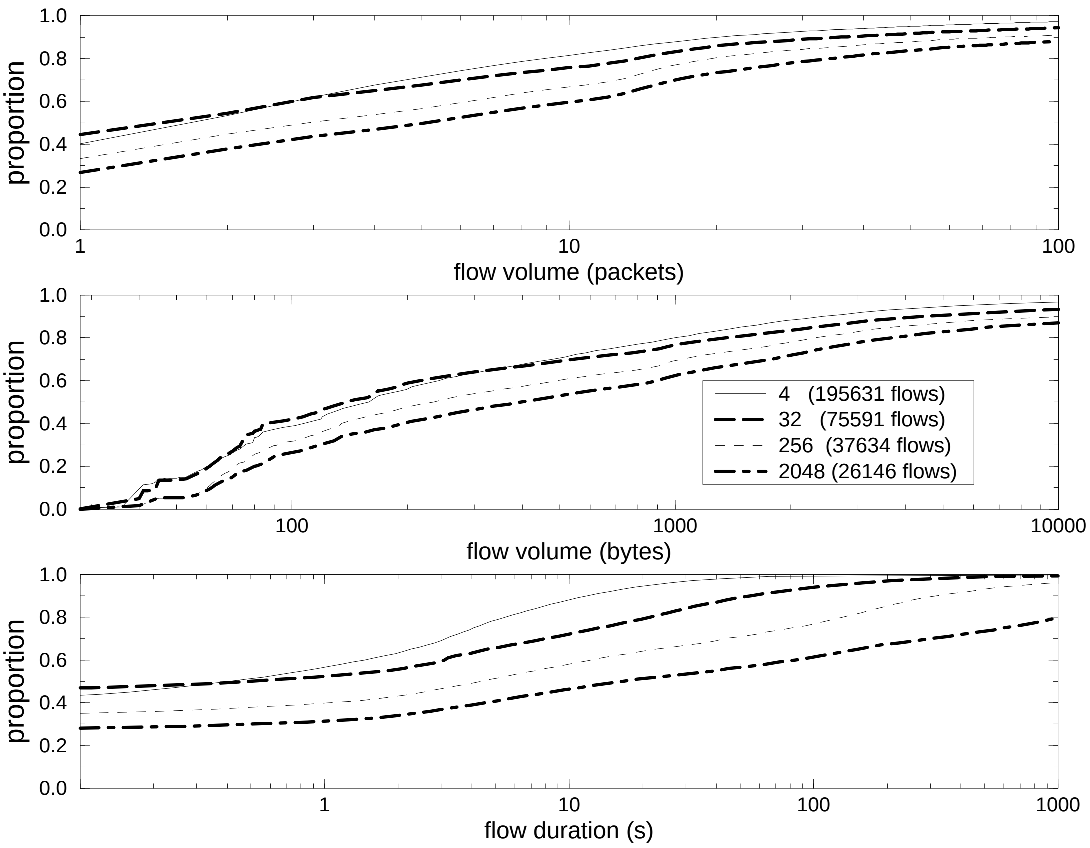

Fig. 2. cumulative distributions of host pair flow packet volumes, byte volumes, and flow durations for a range of flow timeout values: 4, 32, 256, and 2048 seconds (UC-NSF PM)

type. The graphs reflect how multiple host pair flows aggregate into flows of coarser granularities, consistent with environments that simultaneously support flows from many active host pairs which share a common source or destination IP network number, such as backbone network entrance points. This phenomenon yields flow duration and volume distributions which are skewed lower for host pair than for destination networks. Indeed, figure 3 indicates that the fiftieth percentile for host pair byte flow volume is less than 200 bytes, while for destination network numbers it is more than a kilobyte. The fact that this data set includes traffic only in one direction contributes to the disparity, since the ratio of the number of sources to destinations is considerably lower than with the bidirectional collection of the other data sets (see next section). For example, distributions for the bidirectionally collected UCSD campus network data set would exhibit much less disparity among the distributions of flows at different granularities.

The legend of figure 3 shows another interesting aspect which is also applicable to the other wide area backbone entrance point data sets we measured: the number of host pair flows is only two to six times the number of network pair or destination network flows. The number of host pair flows certainly does not scale on the order of the square of the number of network number pair flows, as a uniform matrix of traffic volume among connected sites might imply. In reality, as also illustrated elsewhere [19] the matrix of traffic among sites is sparsely filled. These measurements indicate that maintaining host pair state in the Internet may not impose a prohibitive load on the routers, auspicious for recently proposed soft-state based Internet routing and reservation schemes [20], [21], [22], [23], [24], [25].

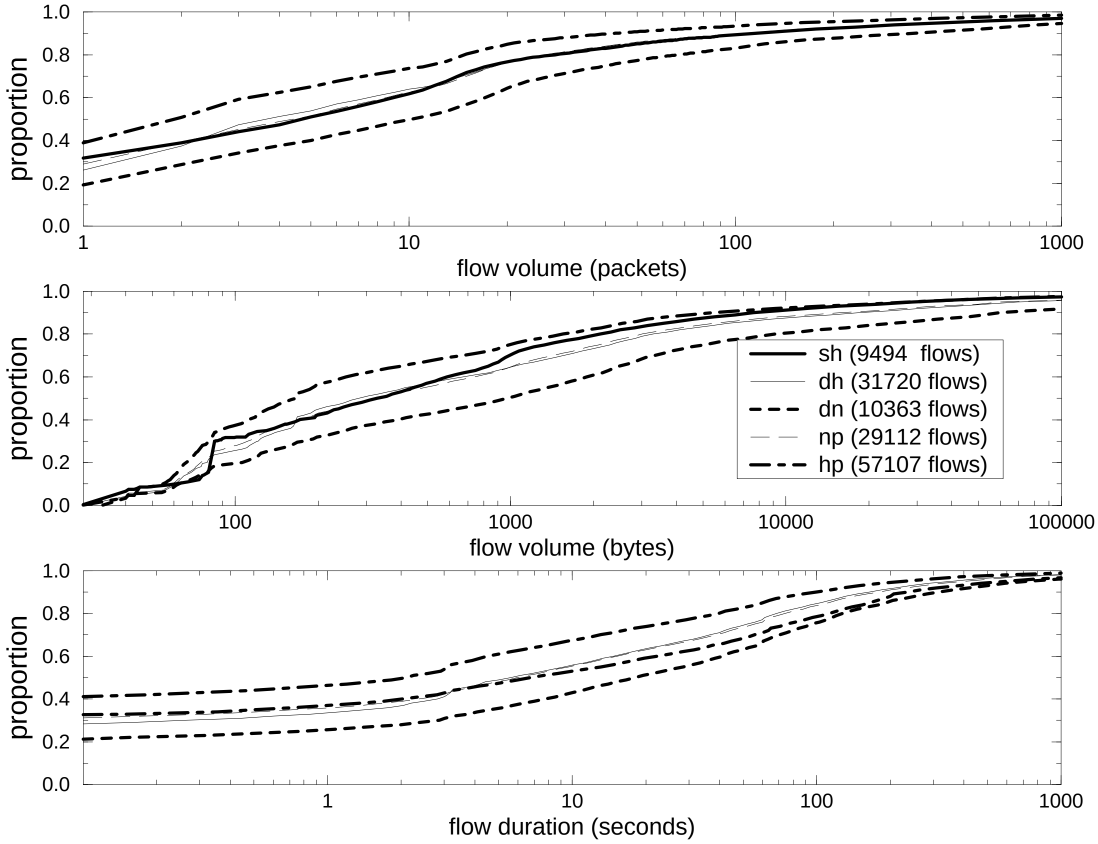

Fig. 3. cumulative distributions of flow packet volumes, byte volumes, and flow durations for five flow aggregation specifications: source host (sh); destination host (dh); destination network (dn); network pair (np); host pair (hp) (UC-NSF PM, 64 second flow timeout)

A.3 Network Environment

Figure 4 shows for five network environments, for which we collected data, the cumulative distributions of flow byte volume, flow packet volume, and flow duration. All three graphs in the figure use the host pair flow specification and a 64 second flow timeout.

These three graphs further confirm the surprising features of figures 2 and 3. For the backbone environments, approximately 40% of the host pair flows consist of a single packet, and less than 100 bytes. For these wide area environments generally between 50% and 60% of flows are less than 200 bytes; between 70% and 80% consist of less than ten packets. This distribution may imply that a flow cache designer should implement a two-phase timeout, e.g., flows that pass a one-second threshold would receive space in a longer term cache. Several studies have investigated similar questions regarding caching [9], [26], [27] and we consider it an important area for further empirical research as the nature of Internet flows changes.

The disparities between the environment are explainable by the differences in network usage. The LAN environments tend to have a greater proportion of higher volume and longer flows, consistent with typical long-term local usage patterns of workstations and terminals [28], as well as distributed file systems and print servers. The SDSC visualization laboratory data trace, representing a LAN of high performance graphics workstations connected to large file severs, is almost entirely (93% of packets; 95% of bytes) UDP/*NFS* traffic. There is also substantial *NFS* traffic during the daytime on the SDSC internal network trace (62% packets; 50% bytes), and some amount of *IPIP* traffic. [5]

[5] *IPIP* (Encapsulated IP) is a generic protocol to allow for tunneling IP, i.e., carrying IP packets within IP packets. An

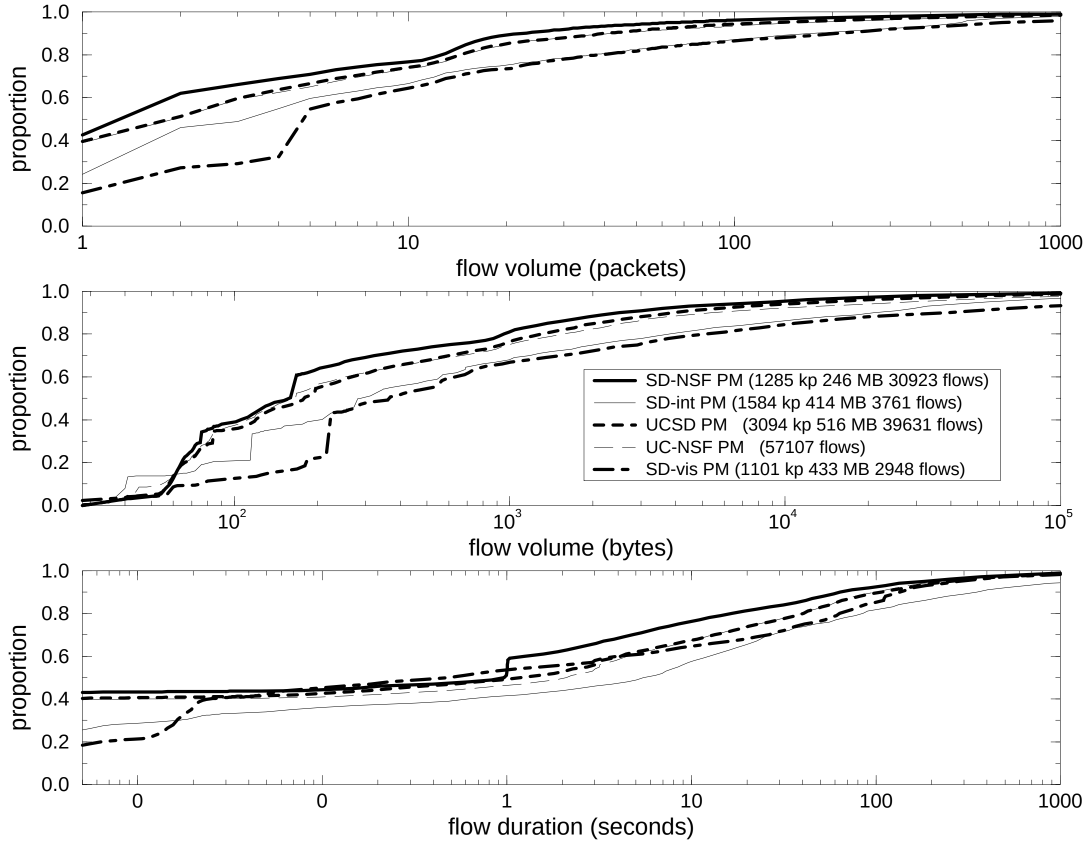

Fig. 4. cumulative distributions of host pair flow packet volumes, byte volumes, and flow durations for five network environments (64 second flow timeout)

In contrast, the SD-NSF and UC-NSF data sets, both at inflow points into the T3 backbone, are composed of mostly TCP with a small amount of UDP traffic, in about a 7:1 ratio during the evening and 10:1 during the day. The UCSD campus backbone data set has a slightly higher proportion of UDP traffic, closer to a 3:1 ratio at night, 6:1 during the day. These differences are consistent with Paxson [30] who find that traffic mix varies a great deal throughout the day.

We do see significant peaks for flows of 5 packets and 224 bytes for the SD-viz PM data set in the top and middle graphs, respectively. To gain further insight into this phenomenon we isolated the 554 host pair flows that consisted of 5 packets in the SD-viz PM data set. The 37 host pairs responsible for these flows were executing a *timeslave* system program that periodically exchanges a 32-byte UDP timestamp packet and four 48-byte ICMP time request/reply packets in order to synchronize time between machines. Aggregating traffic into host pair flows as we have done for this graph combines the UDP and ICMP time requests into a single host pair flow.

In almost every environment in our study and using a 64 second timeout, at least 80% of all host pair flows consist of less than 100 packets, less than 10 kilobytes, and less than 30 seconds in duration. These measurements suggest that most flows are short, which is consistent with the findings of Paxson [30] and Caceres *et al.* [8], even though these studies use different definitions from ours for the notion of a flow, which makes the results not directly comparable.

example application in popular use today is the creation of a virtual network: the Mbone [29]. The Mbone is a multicast service over an infrastructure that does not itself support multicasting. The implementation creates tunnels of multicast paths across the unicast infrastructure, while multicasting or even replicating packets at the tunnel exit point to create the multicast effect.

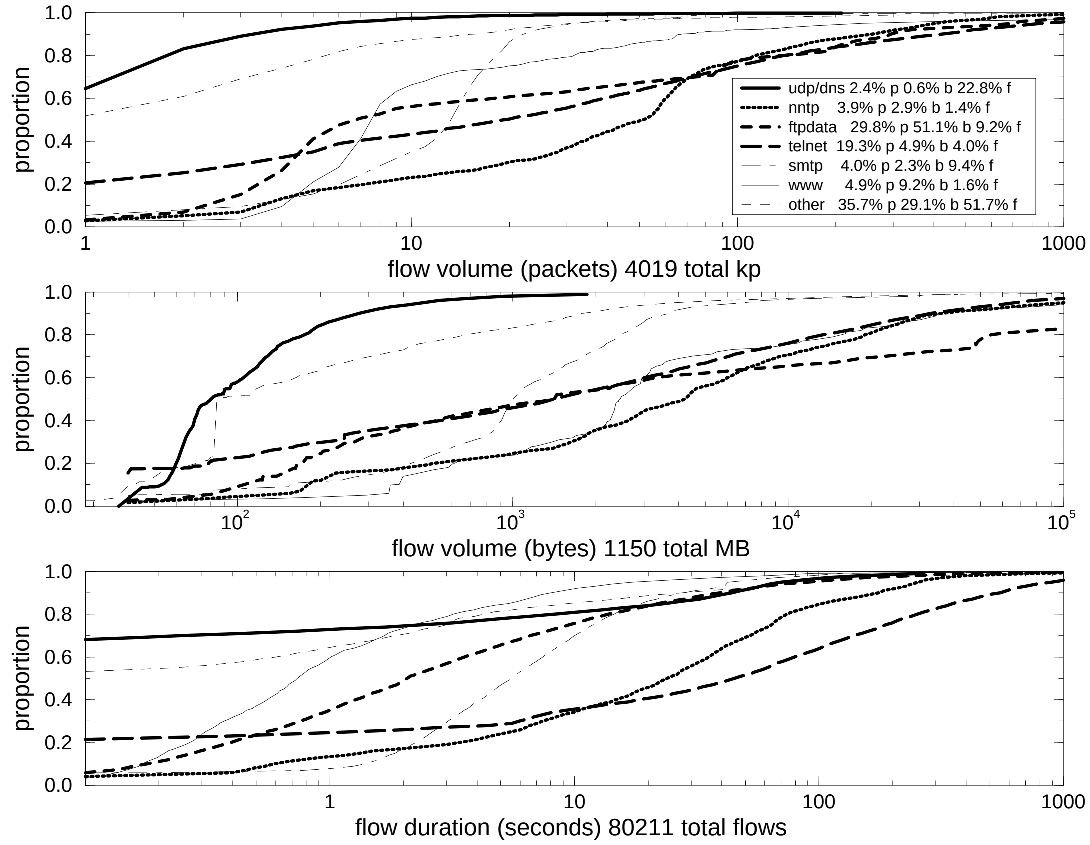

Fig. 5. cumulative distributions of flow packet volumes, byte volumes, and flow durations for six port-based applications (UC NSF PM, 64 second flow timeout)

A.4 Network usage information

The metrics we have shown thus far do not differentiate among various network usages, or applications, which turns out to have a significant effect on the expected size and duration of a flow. An example is the differences of flow characteristics based on transport protocols being used, especially as we often see a strong domination of microscopic DNS flows, which are based on short UDP transactions, and do not require the overhead of the setup and teardown of TCP connections. On the other hand, other often UDP based flows may exist for long periods of time at a high data volume, such as real time audio and video applications. As such a further emphasis in end system application will yield further insight into the application layer.

Figure 5 shows the volume and duration distributions for six common applications on the Internet, specifically UDP/*dns*, *nntp*, *ftpdata*, *telnet*, *smtp*, and *www-http*. We aggregate all other flow types into an "other" category. The legend indicates the percent of the total packets, bytes, and flows that each category contributes. The data indicate that a large majority of the single packet UDP flows are from the *dns* protocol, unsurprising given the short transaction-style nature of the *dns* protocol. With a 64-second timeout, 65% of *dns* flows consist of a single packet.

The longest average flow durations in our samples seem to characterize the *www-http* protocol, which during our measurements in March 1993, appeared significantly only in the UC NSF PM data set. Since that time *www-http* traffic has become quite pervasive across the Internet; the absolute volume as well as proportion of *www-http* traffic on the NSFNET has grown dramatically, from 0.056% in March 1993 to almost 14% of the NSFNET traffic byte volume during the month of November 1994.

The distribution of *ftpdata* flow durations is not significantly different from that of other studies that focus on the transport layer; using timeouts of 64 seconds or larger, approximately 65% of the *ftpdata* flows are less than 10 kilobytes.[6]

B. *Envelope presentations*

The multi-dimensional parameter space of network flows is rich, complicating the task of presenting results. One possibile method of simultaneously displaying the range of two dimensions of flows is using *envelopes*. For example, we can portray the packet-count vs. duration or byte-count vs. duration space of flows while varying one of the parameters that we have explored: flow timeout, environment, flow endpoint granularity, higher layer protocol or application. The top half of figure 6 provides an example, outlining the packet-count vs. duration space of the same Internet applications shown in figure 5. Each diamond delineates the population of flows of that application; the vertices are at the 5th percentile, median, and 95th percentile for both the packet count and the duration axes.

We use the *ftp-data* diamond to illustrate how to read this graph. The x-coordinates of the left and right vertices of the *ftp-data* diamond indicate that, given a 64 second flow timeout, 5% of the *ftp-data* flows are less than 0.1 seconds in duration, and 95% of them are less than 90 seconds. The y-coordinates of these points are equal, at 7 packets, which indicates the median of the distribution of packets per *ftp-data* flow. Similarly, the y-coordinates of the top and bottom vertices of this diamond indicate that 5% of the *ftp-data* flows in this data set were less than two packets, and 95% of them were less than 650 packets. The x-coordinates of these vertices indicates the median duration of an *ftp-data* flow from this hour: 2 seconds. As with previous figures, the legend indicates the percent of the total packets, bytes, and flows that each category contributes. The graph of byte-duration space appears similar; we do not include it here.

The graph highlights visible differences among the applications, especially given the log scaled axes. For example, as an interactive protocol, *telnet* exhibits a large spread in flow duration, with the longest duration flows lasting much longer than those of bulk data protocols such as *ftpdata* or transaction style protocols such as *www-http* and *smtp*. Bulk transfer and transaction protocols also exhibit more predictable duration, consistent with their typical single burst usage. An extreme case is the packet volume distribution of the UDP/*dns* flows, many of which consist of a single packet, with the 95th percentile at six packets. *Dns* zone transfers use the TCP protocol and so are excluded from this statistic.

The top half of figure 6 elaborates on the implications of figure 5 regarding the influence of packet type on the utility of maintaining flow state. Table I provides further insight, ranking the ten protocols with the highest number of flows in the UC-NSF PM data set using a 64 second timeout. The table also lists the number, proportion, and rank of flows, packets, and bytes for each protocol. Although *dns* constituted only 2.4% of the total packets, it constituted 22.8% of the total number of flows, given a 64-second flow timeout.[7] Designers of routing or accounting tables or caches may improve performance by selectively choosing not to store information for packets that are highly likely to represent short flows, e.g., *dns*, *gopher*, *ntp*. Such flows, whose packet-duration profiles are largely in the lower left of the top graph in figure 6, will contribute to cache thrashing because they consume valuable memory that will likely not require future reference. Disregarding higher layer information regarding the nature of the traffic clearly imposes a high opportunity cost.

This data also has implications for multiplexing IP flows onto a wide-area ATM substrate using virtual circuits (VCs). Caceres [17] suggests that the optimal multiplexing policy[8] for bulk transfer conversations whose packet transmission rates are limited by transport protocol window sizes and available bandwidth, e.g., *ftp-data*, *smtp*, *nntp*, would be one virtual circuit per conversation. He further finds that interactive conversations, e.g., *telnet*, should not share vir-

[6] Caceres *et al.* [8] using a 20-minute flow timeout, found that 75-90% of bulk transfer conversations consist of less than 10 kilobytes of data. Our results, although slightly different, are also consistent with their measurements that indicate that over 90% of interactive conversations consist of less than a thousand packets. We note the difficulty of getting a picture of *telnet* flows given unidirectional traffic; Caceres *et al.* [8] and Paxson [30] show that interactive applications can generate an average of twenty times more traffic in one direction than the other. Caceres *et al.* find in their measurements that bulk transfer flows are often bidirectional as well, but Paxson's measurements do not show a strong degree of bidirectionality.

[7] Using a 4-second timeout increased this proportion to 27.6%.

[8] Caceres uses low interactive delay and high bulk transfer throughput under increasing network load as metrics to judge optimality.

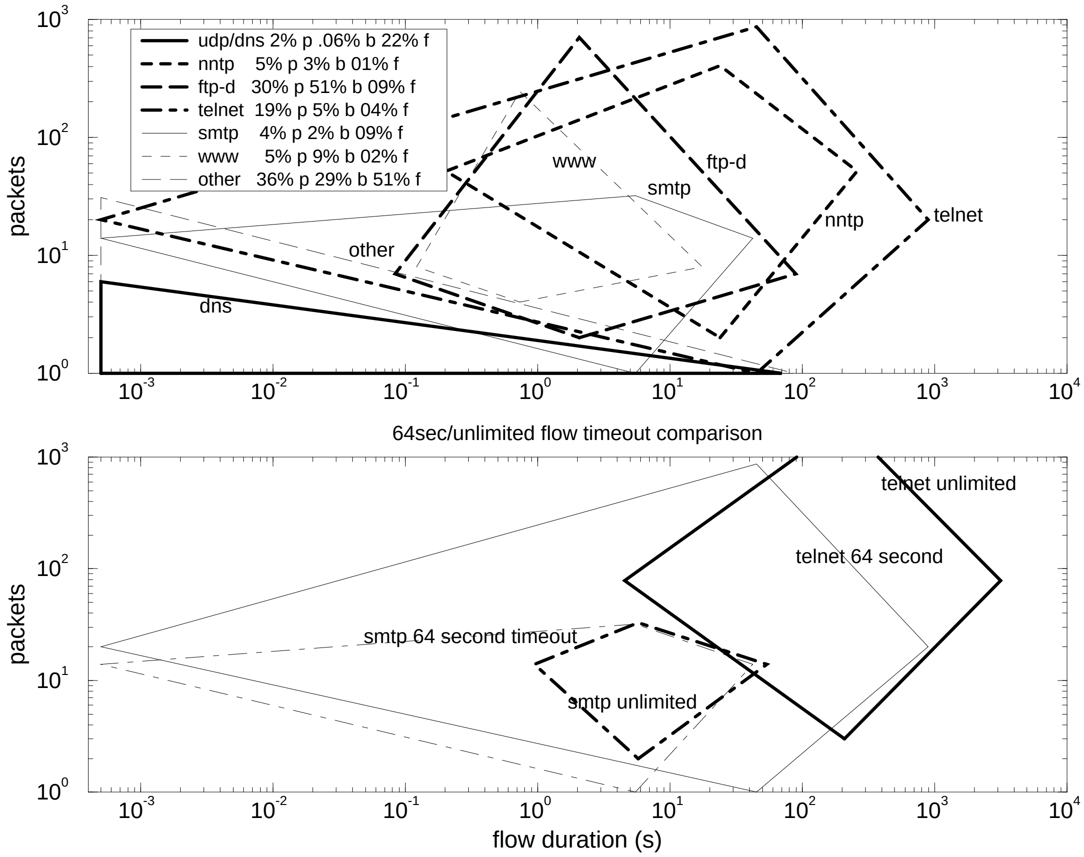

Fig. 6. distributions in packet-duration space of host pair flows by application (UC-NSF, 64 second flow timeout) (a) top: seven common application categories (b) bottom: difference in packet-duration space between 64-second and unlimited timeout values for two applications: *telnet* and *smtp*

TABLE I

PROPORTION OF FLOWS, PACKETS, AND BYTES ATTRIBUTED TO MAJOR PROTOCOLS (UC-NSF PM, 64 SECOND TIMEOUT)

prot	service	number			% total			rank		
	totals:	106848	4019.1	1150.9						
		flows	kp	MB	flows	pkts	bytes	flows	pkts	bytes
UDP	*dns*	24345	97.4	6.50	22.8	2.4	0.6	1	10	14
TCP	*gopher*	11507	146.7	57.71	10.8	3.7	5.0	2	6	5
TCP	*smtp*	10044	161.7	26.64	9.4	4.0	2.3	3	4	8
TCP	*ftp-data*	9807	1197.1	587.53	9.2	29.8	51.1	4	1	1
TCP	*ftp-ctrl*	5307	108.3	7.49	5.0	2.7	0.7	5	9	13
TCP	*telnet*	4222	774.9	56.88	4.0	19.3	4.9	6	2	6
TCP	*finger*	2253	10.6	.89	2.1	0.3	0.1	7	30	33
TCP	*www-http*	1662	196.9	105.55	1.6	4.9	9.2	8	3	2
UDP	*ntp*	1592	3.6	.27	1.5	0.1	0.0	9	50	63
TCP	*nntp*	1532	154.9	33.15	1.4	3.9	2.9	10	5	7
	subtotal	72271	2852.1	882.60	67.6	71.0	76.7			

tual circuits with bulk transfer conversations to avoid high delays, although many interactive conversations can share a virtual circuit among themselves without detrimental effect. Our data suggests that flows that do not fall into either of these categories, e.g., *dns*, *gopher*, *ntp*, and many unknown traffic flows, would likely require the suboptimal router behavior of assigning a virtual circuit for a flow that would not make any further use of it. One alternative policy would be to tear down such VCs immediately without waiting any timeout interval. Another potential technique for reducing the number of circuits needed for such flows would be to bundle them together with others to the same destination over the same virtual circuit. We discuss the maintenance of virtual circuits further in section IV.

The application layer protocol also influences the interaction between other parameters and individual flow metrics. We discuss one example of such interaction using the flow timeout parameter. As figure 5 showed earlier, with a 64-second timeout, 65% of *dns* flows consisted of a single packet. Allowing an infinite timeout does not change the situation significantly: 50% of flows consist of a single packet. The *ftpdata* profile also does not depend significantly on the timeout value. Using a 64-second timeout, the median number of packets in an *ftpdata* flow was 7, and with an unlimited timeout, the median was about the same, 8 packets.

However the timeout value does affect the packet volume and duration of other protocols. The lower half of figure 6 highlights two flow types for which the timeout affects the profile: *telnet* and *smtp*. We show for these two flow types the difference in flow packet volume and duration when using a 64-second versus an infinite timeout. For *telnet* flows using a 64-second timeout the median number of packets per flow was 20; with an unlimited timeout this median jumps to 78 packets, suggesting that telnet flows are often idle for more than 64 seconds. Similarly, for this data set the median and 95th percentile of *smtp* flows do not depend on the timeout, but the 5th percentile at the 64 timeout value is at a single packet, indicating a number of single packet *smtp* flows between the same two hosts separated by more than 64 seconds. These measurements indicate that for at least a few types of network traffic, the timeout value will affect flow assessment.

IV. Aggregate flow metrics

In this section we focus on aggregate flow metrics, e.g., the number of active flows and the appearance of new flows as an arrival process. These metrics are particularly relevant to memory and processor resource requirements for keeping state information based on flows. Our measurements illustrate how a flow methodology approach to traffic analysis can yield implementation guidelines for router designers, and in particular how flow parameters will affect design decisions under new technologies.

Simple flow state maintenance mechanisms, e.g., routing caches, hold entries for destination addresses with high imminent reference probability, often due to recent reference, in a separate, smaller and perhaps faster, memory. Even if one does not use a hardware cache which is typically considerably faster, keeping recently referenced records in a software cache can save time because the switching process does not have to search the entire routing table.

Recent discussions of flow state have involved extensions to the Internet service model and routers that support it. These discussions assume a more general definition of a flow as a single stream of packets from a specified set of one or more sources to a specified set of one or more destinations that are subject to a single path selection constraint and queueing behavior in intermediate nodes. Maintaining the flow state typically will require holding an entry for each active flow which then controls forwarding behavior for packets belonging to that flow.

Flow state will also be an important component of effectively using IP over the virtual circuits of an ATM network. Implementing a datagram service over a virtual circuit-oriented network service such as ATM requires setting up and tearing down virtual circuits, and accounting and charging for them, based on actual traffic activity.

For example, an ATM service provider could offer a fully connected mesh of permanent virtual circuits (PVCs) among all the entry and exit points of its ATM cloud, comparable to a current IP switching backbone. But ATM functionality allows more efficient service and bandwidth allocation by extending a simple PVC network through the use of multiple parallel PVCs, or even switched virtual circuits (SVCs) for specific traffic flows, allowing multiple service qualities in the network. Supporting SVCs in real time as needed requires establishing them upon appearance of traffic not fit for an existing PVC or SVC. For example when a packet from network A to network B arrives at A's interface to the ATM network, and an appropriate circuit between network A and network B does not already exist, the router at the inflow point would establish an SVC for this path, and then tear the circuit down if it becomes idle. This scenario imposes some VC setup overhead compared

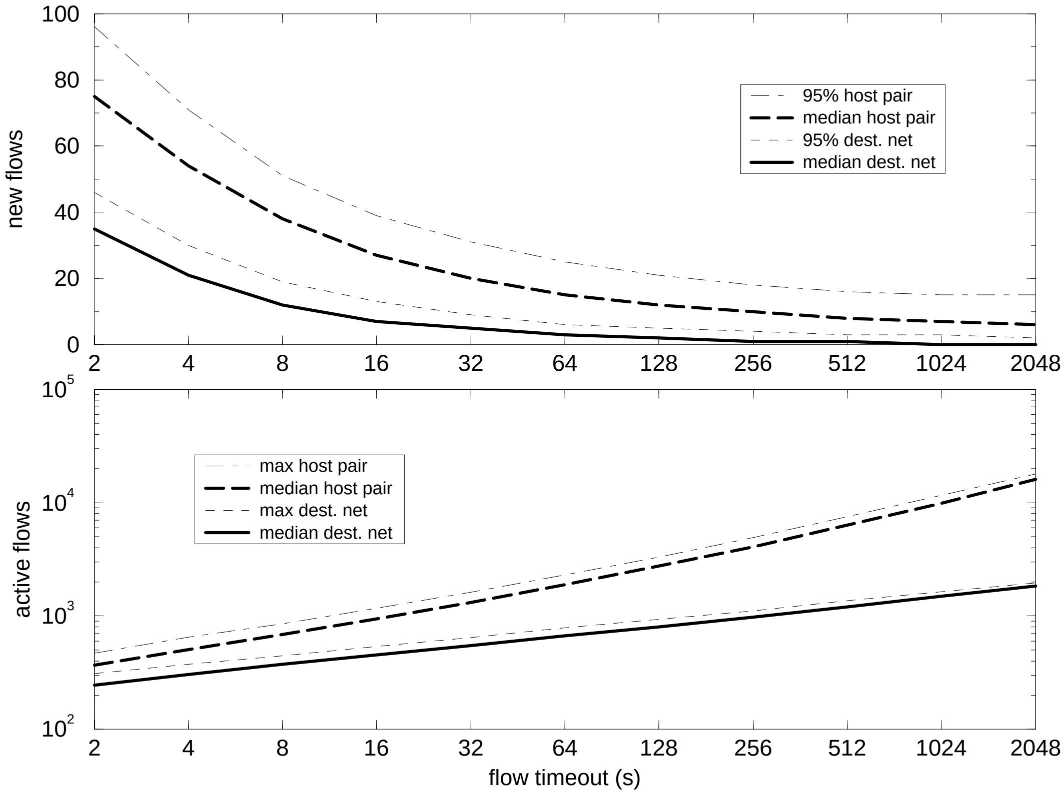

Fig. 7. as a function of timeout value (a) top: median and 95th percentile of new destination network and host pair flows per second (b) bottom: median and maximum of number of active flows per second (UC-NSF PM)

to the fully meshed PVC scenario, but saves resources if not all PVCs are necessary at all times.

In reality the general purpose infrastructure will have to take incremental steps from current IP switched networks toward an ATM cell-switched environment. For example, new federal agency wide area backbones may transition to an ATM environment using multiple fully connected meshes of PVCs, with different priority and bandwidth parameters.[9] Parameters of flows that currently traverse wide area infrastructures are essential to engineering such configurations that can effectively replace existing dedicated point-to-point channels (e.g., T3 links).

[9] One could use the IP precedence field to signal multiple underlying service qualities, so that for example level three traffic could travel on a standard PVC, traffic below level three in precedence goes on a secondary PVC, and traffic above level three would incur dedicated SVC setup when resources are available.

A. Flow timeout

We now examine flow arrival percentile statistics over the course of our 1 hour (= 3600s) duration data sets. We first highlight the impact of the flow timeout parameter, fixing all other parameters, on the number of new and active flows for each environment. For the other parameters we select the destination network flow granularity, the UC-NSF PM data set, and consider all traffic (i.e., all protocols aggregated). We will also use the host pair flow granularity for comparison to destination network flows.

Maintaining state in a router, or virtual circuits in an ATM network, requires memory and computational resources for each flow. One objective of a router maintaining flow state is to optimize the tradeoff between maintaining state for many flows, which requires both memory for the information and search time for accessing the large state table for each packet switched, and maintaining state for few flows by means of a short flow timeout, which requires less memory but greater CPU power and memory management effort to set up and tear down flows more often. If the timeout value is too low, flows may time out even though traffic between the two endpoints has not stopped, leading to potentially large delays and processing costs for reestablishing the flow. The analogy to virtual memory caching is *thrashing*, which will occur if flow demands are larger than available router resources and require constant closing and reopening of flows. In this section we explore how the timeout value influences the probability of thrashing.

Saran and Keshav [18] investigate one timeout strategy: timeout occurs if a packet for a flow has not appeared in the time between the last packet of the flow and the one immediately preceding it, i.e., within the last intra-flow packet interarrival time. Mankin and Ramakrishnan [31] suggest another possible strategy where the timeout value dynamically changes based on the number of currently active flows. They propose three administratively controlled variables: a minimum time; a maximum time and an adaptation factor in seconds per available flow. A flow times out if it has been idle for a time period equal to the minimum plus the adaptation factor times the number of available circuits, limited by the maximum time. The authors suggest that administrative adjustment of these variables can provide considerable flexibility in meeting the needs of a specific gateway, but do not offer any analysis, simulation, or empirical data to test the efficacy of such a scheme.

In figure 7 the upper graph demonstrates a measure of the *flow turnover* rate, while the bottom graph indicates the number of entries required in a state table to hold entries for all active flows. The top graph in figure 7 plots for a single environment (UC-NSF) the median and 95th percentile of the number of new destination network and host pair flows. [10] The bottom graph in figure 7 plots the median and maximum number of active destination network and host pair flows. As expected, the larger the timeout, the greater the number of active flows, but the smaller the turnover rate as measured by the number of new and timed out flows. More specifically, the median number of active destination network flows per second using a 64-second timeout on this data set is 660, and the median number of new flows (requiring set up) is 3. Dropping to a 4 second timeout would have required maintaining a median number of only 300 active destination network flows per second but setting up a median number of 20 flows per second.

The behavior of the host pair flow counts are similar although characterized by higher means, as one would expect. Using a 4-second timeout yielded on average 54 new flows per second and 506 active host pair flows. Using a 64-second flow timeout yielded approximately 2,000 for the median number of active flows, but trades off the greater number of active flows with a reduced setup requirement: only a median of 15 new flows requiring setup per second, and a maximum of 35 new flows per second. The contrast between the values of these metrics for host pair versus destination network flows highlights the difference in flow setup for the two flow granularities. Both cases show clearly the tradeoff between memory requirements for storing more flows at higher timeouts versus processing requirements for frequent flow setup and teardown at lower timeouts. The other flow granularities exhibit similar behavior with different parameters.

Figure 8 provides another perspective on how the flow timeout will affect flow statistics. Note that many flows with a given endpoint granularity, e.g., host pair, occur more than once within the same trace. For example, for the UC-NSF data trace there were only 25,358 unique host pair flows; the different numbers in the legend of figure 2, and in the top line in the top graph of figure 8, reflect the fact that at shorter timeouts many flows will reappear multiple times in this tally.

Given this information we can better understand the implications of figure 8. The top graph shows the total number of host pair flows as a function of flow timeout, and the number of flows that were recreated within a time interval equal to the timeout value. The bottom graph shows the ratio of flows that were recreated within the flow timeout value to the 25,358 unique host pair flows. At a timeout value of two, each of the 25 thousand unique host pair flows is set up and torn down almost ten times *on average* during the hour; a timeout of 16 seconds brings this factor down to 2.9. Dividing by the total number of flows rather than the number of unique flows is also indicative: at a 16 second timeout value, approximately 74% of all flows are ones that have been

[10] We use the 95th percentile rather than the maximum of the distribution of new flows because in the first few seconds the number of flows is still establishing a steady state and so the many new flows for those seconds skew the distribution of the number of new flows per second. Similarly with the number of timed out flows; a few flows lasted beyond the hour interval of the data set and so were all timed out at the end of the hour, skewing the distribution of flow durations.

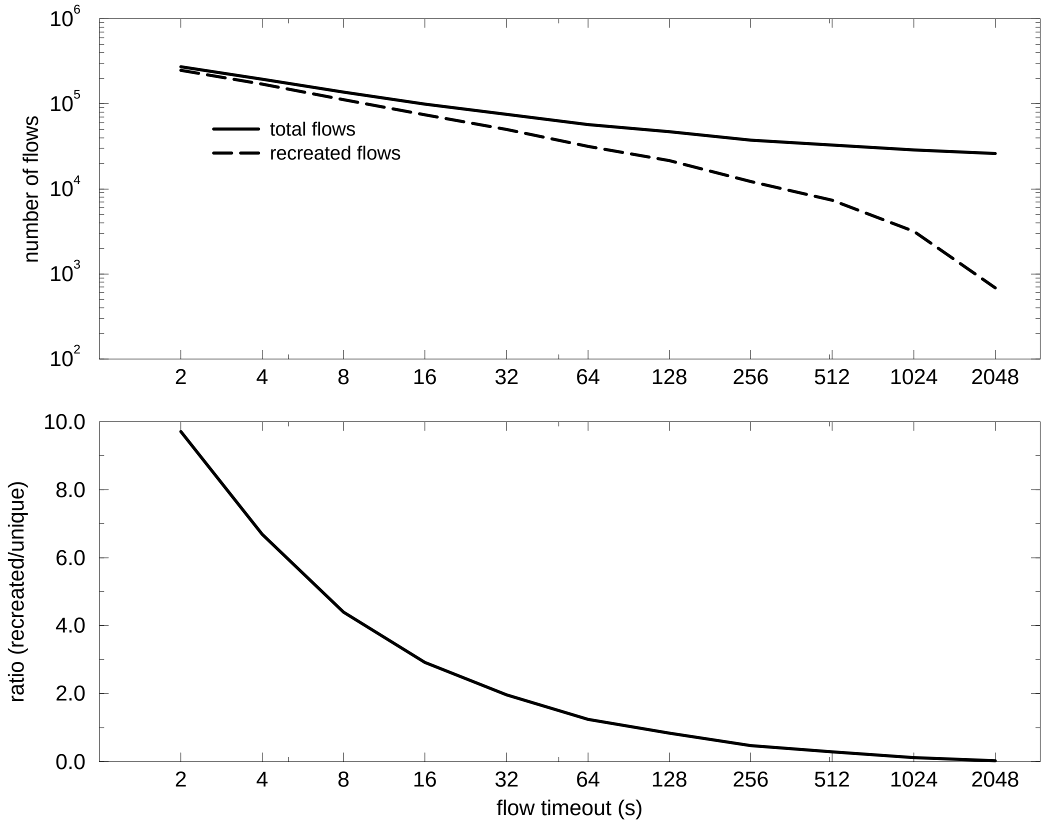

Fig. 8. (a) top: total number of host pair flows as a function of flow timeout, and the number of flows recreated within a time interval equal to the timeout value; (b) bottom: ratio of flows that were recreated within the flow timeout value to the 25,358 unique host pair flows (UC-NSF PM)

recreated in the last 16 seconds. Using a 64-second timeout, half of the flows had been recreated within the last 64 seconds. Both graphs plot the x-axis on a log scale; the top graph also plots the y-axis on a log scale. This graph is consistent with figure 2 where lower timeouts seem to incur undue thrashing, reflected here by ratios of over 2 for flow timeouts under 32 seconds.

A.1 Case study: towards an optimal flow timeout

A router designer would likely want to use empirical knowledge of flow parameters in a target environment in order to determine appropriate timeout values to use for flow setup and tear down. In particular, he needs to quantify how much it will cost to set up and tear down flows versus holding them open, where *cost* may involve several components. The user may perceive cost as a delay in setting up a flow (d), and perhaps a dollar charge for holding the circuit open. Costs for the network or router itself include the number of instructions required to set up a flow (i), memory cost in holding a flow open (m), and possibly a search time or computational burden involved in maintaining flow state (s). Assuming we can normalize these to cost units, we can refer to the cost for establishing a new flow as $N = i + d$ and of holding an active flow as $A = m + s$. Given a active and n new flows per interval the router operating cost is $Aa + Nn$. Using the UC-NSF PM trace flow statistics, figure 9 shows where this value is minimized for several values of the ratio $F = N/A$.

A.2 Case study: flows deserving state

Equipment and protocol designers will also ask a related question: for which flows does state establishment pay off? We present one possible evaluation of when to establish state for a flow given the UC-NSF PM trace flow parameters. In particular we illustrate how the number of packets per flow constrains a router that must create state for a population of flows. Assuming that it takes p instructions to forward a regular packet, r instructions to forward a packet for which special flow information exists, q instructions to install a flow, and a flow comprises n packets on average, then maintaining flow state makes sense when

$$p + q + (n-1) \times r < n \times p$$

$$q < (p-r) \times (n-1)$$

so we can maximize efficiency by not installing unproductive (e.g., single packet) flows, i.e, by ensuring that $q << (p-r)(n-1)$

For example, if $p = 600$ instructions, $r = 300$ instructions, and $n = 10$, then flow installation must require less than $q = 2700$ instructions. A router designer could take advantage of empirical data such as in figure 5, discussed earlier in section III-A.4, and in figure 11, which we discuss later in section IV-C, to determine which flow establishments to avoid, or a preferred PVC/SVC configuration of an ATM network. For example, a PVC substrate could still offer standard service for background traffic, while other traffic flows would travel on dynamically established switched circuits based on actual flow requirements. A high priority video stream could travel on a separate SVC, while an aggregation of low priority video streams and other services whose flow profiles are in the upper left of figure 11 may travel on a PVC network allocated for such background or non-premium traffic.

B. Envelope presentations

Eventually a metric as simple as a flow count may be too limited to measure the impact of flows on the overall workload in a specific environment. To explore the interaction between the number of flows and the total traffic volume, we depict a two-dimensional profile, using the mean number of active flows per second for the hour in one dimension, and the mean per-second traffic volume in packets in the other dimension. Figure 10 plots this two-dimensional metric for each environment, using host pair flows and a 64 second timeout.

As expected, the flow-traffic volume product is highest for the UC-NSF PM environment, which has both high flow counts and high utilization. UCSD PM is the second highest, supporting many host pair flows at a fairly high per second packet rate. Dividing figure 10 into quadrants leaves these two in the upper right with everything else toward the lower left. This use of the flow-traffic volume metric can help assess multiple dimensions of the workload of a specific networking environment. If the value representing a specific environment on this graph moves up the diagonal of this graph, then the compounded characteristics are fairly similar, just more of the same. On the other hand, if the packet volume rate (x-value) stays the same, and the number of flows (y-value) significantly changes, then the traffic composition is changing.

Our measurements indicate that at least in the environments we studied, current IP traffic still consists more of short transaction behavior rather than longer term flows. The short packets and short flows together shed doubt on a strategy of optimizing for long flows that are in fact the minority case. However in the next section we discuss trends that may change this situation and have ominous implications for operators of Internet components.

C. Higher layer protocol

We next explore how the flow type affects aggregate flow metrics. Differences in flow profiles by protocol may lead a designer of an accounting scheme to choose not to charge for small flows at all, either by not maintaining records for certain flow types, or by having a special cache for (free) flows until they exceed a certain packet volume. Small flows will also have a dramatic impact on ATM or other link-level circuit multiplexing policies, if setup or teardown costs more than the entire flow. In this section we extend these results to aggregate flow metrics, using a two-dimensional perspective to show insight into the effect of the protocol on router workload.

To illustrate our point in the next two graphs we use a newer data set from March 1994 at the UC-NSF site in order to get enough IPIP and *www-http* traffic to profile their behavior. Figure 11 uses a 64 second flow timeout in the UC-NSF environment and plots the number of flows of a given packet type versus the mean number of packets per that type of flow. The graph highlights the range from extremely low packet volume flows of *dns* to short duration transactions of *smtp* and *www-http*, to heavier flows carried by file transfers and interactive communications. Note that the otherwise transaction-like *nntp* protocol exhibits a relatively high average packet per flow ratio.

In the same graph we include the traffic profile of a recently popular protocol, IPIP, used for carrying video and audio multicast flows on the Mbone. The lower right quadrant of this graph emphasizes the impact of continuous-stream protocols, where relatively few flows will have packet and byte volumes orders of magnitude higher than conventional applications. Applications in the lower right quadrant of this figure will fundamentally change the nature of workloads in Internet environments.

V. Conclusion

The proliferation of different traffic types on the Internet with fundamentally different workload characteristics, including those not using a transport protocol such as TCP to delineate the beginning and end of a flow, makes it even difficult to define an Internet flow, but also more critical. In this paper we have presented a methodology for profiling traffic flows on the Internet, parameterized in a variety of aspects. We have applied our methodology to a case study based on packet traces from multiple networking environments, and have presented some findings based on those data sets.

The objective is to derive flow metrics relevant to a variety of applications, including route caching, usage-based accounting, and other specialized routing and resource reservation algorithms. Table II summarizes key results of our case study of individual flow profiles.

However, describing individual Internet flows is not sufficient for understanding the aggregate traffic behavior at a systems level. We also explored descriptors of the entire population of flows. Table III summarizes key findings of our study of aggregate flow profiles for the data traces we studied. We presented metrics of the quantity and turnover rate of flows, relevant to route caching strategies in network switching equipment. One important characteristic of our measured data is that the number of host pair flows appears far less than proportional to the square of the number of network number pair flows, as a uniform matrix of traffic volume among active sites might imply. This phenomenon is prominent even in wide area environments that aggregate among a large number of users, and bodes well for applications requiring end host pair state. Traffic locality can compensate somewhat for the brevity of such a large proportion of Internet flows that we found in section III. We note that many Internet environments will face a challenging workload shift with the increasing use of real-time continuous media applications, which tend to exhibit much greater volume-duration products.

The methodology we describe in this paper can form a complementary approach to existing operational statistics collection, if applied to continuous or spot measurements, yielding insights into larger issues of Internet evolution, i.e., how environments of different aggregation can cope with resource contention by an ever-changing composition and volume of flows.

VI. Acknowledgments

We would like to express our appreciation for the cooperation and assistance we received from Randy Butler at NCSA, and Jim Madden and his staff at UCSD who allowed us access to specific data collection points. We also would like to thank Vern Paxson of Lawrence Berkeley Laboratory and Fred Baker of Advanced Computer Communications for helpful comments on earlier drafts of this paper.

References

[1] R. Jain and S. A. Routhier, "Packet trains—measurement and a new model for computer network traffic", *IEEE Journal on Selected Areas in Communications*, vol. 4, no. 6, pp. 986–995, September 1986.

[2] D. D. Clark, "The design philosphy of the Darpa Internet protocols", in *Proceedings of ACM SIGCOMM '88*, August 1988, pp. 16–19.

[3] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic", in *Proceedings of ACM SIGCOMM '91*, September 1991, pp. 133–148.

[4] J. Mogul, "Observing TCP dynamics in real networks", in *Proceedings of ACM SIGCOMM '92*, August 1992, pp. 305–317.

[5] J. Mogul, "Network locality at the scale of processes", in *Proceedings of ACM SIGCOMM '91*, September 1991, pp. 273–285.

[6] M. Acharya, R. Newman-Wolfe, H. Latchman, R. Chow, and B. Bhalla, "Real-time hierarchical traffic characterization of a campus area network", in *Proceedings of the Sixth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 1992, University of Florida.

[7] M. Acharya and B. Bhalla, "A flow model for computer network traffic using real-time measurements", in *Second International Conference on Telecommunications Systems, Modeling and Analysis*, March 24-27 1994.

[8] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel, "Characteristics of wide-area TCP/IP conversations", in *Proceedings of ACM SIGCOMM '91*, September 1991, pp. 101–112.

TABLE II

KEY RESULTS OF INDIVIDUAL FLOW PROFILING CASE STUDY (SEE APPENDIX A FOR DETAILS OF MEASUREMENTS AND ENVIRONMENTS)

1. Host pair flow timeout values between 16 and 128 seconds seem to be an appropriate tradeoff between router processing and memory, i.e., incurring thrashing in a cache vs. requiring too many flow state entries that are often never again used after the first packet in the flow caused the creation of the entry in the router.
2. For timeout values of 64 seconds (or less), 90% of the host pair flows are less than 50 packets, 5.5 kilobytes and 100 seconds. For virtually infinite timeouts, 27% of the host pair flows consist of a single packet of less than one hundred bytes.
3. The large proportion of flows are very short in duration: using a 64-second timeout for the busiest backbone data set (UC-NSF PM), almost 60% of the host pair flows are less than one second (40% consisting of a single packet).
4. Flow volume and duration are correlated to the higher level protocol. Given a 64 second timeout, only 10% of TCP flows consist of a single packet, but over 60% of UDP flows consist of a single packet, and 52% of ICMP flows consist of a single packet.
5. TCP/UDP ports also provide an indication of the expected duration and volume of a flow: e.g., the large majority of the single packet UDP flows are from the *dns* protocol (unsurprising given the nature of the *dns* protocol). Using a 64-second flow timeout, although *dns* constituted only 2% of the total packets, it constituted 27% of the total number of host/port-pair flows.
6. When examining the shorter flows as a function of protocol, it is evident that many of the short flows (approximately one third of all flows) are *dns*, *gopher*, *ntp*, or *finger*.

TABLE III

KEY RESULTS OF AGGREGATE FLOW PROFILING CASE STUDY (SEE APPENDIX A FOR DETAILS OF MEASUREMENTS AND ENVIRONMENTS)

1. For the data sets we studied, the number of host pair flows is only two to six times the number of network pair or destination network flows.
2. The number of flows a router must be able to maintain, create and delete in a state table depends on the timeout value and environment, but for a busy entrance point into the NSFNET backbone, the median number of active host pair flows per second using a 64-second timeout is approximately 2000, with a median of 15 new or deleted flows per second, and a maximum of 35 new flows per second.
3. The tradeoff between timing out flows too early and allowing them to continue using memory resources unproductively is significant. For our busy backbone data set, at a timeout value of two, each of the 25 thousand unique host pair flows is set up and torn down almost ten times *on average* during the hour; a timeout of 16 seconds brings this redundancy factor down to 2.9.
4. Our measurements indicate that, at least in the environments we studied, current IP traffic still consists more of short transaction type traffic rather than longer term flows. The short packets and short flows together shed doubt on a strategy of optimizing for long flows that are in fact the minority case. However we note that many new applications may change this characteristic of Internet environments, as they introduce traffic flows with different behavior, particularly real-time continuous media flows, which tend to exhibit greater duration and flow volume.
5. The data sets we examined exhibited significant flow locality, confirming results of several previous studies. For the SDSC internal FDDI LAN, there was an 80% chance that the address of a packet was the same as one of the last two packets. For the UCSD campus, 19 references of address history was enough to account for 80% of the packets. The backbone inflow points require more history to account for the same percent of traffic, but it is still noteworthy that given hundreds of thousands of possible destination network addresses, 60% of the traffic across the San Diego and Illinois inflow points was to destinations that were referenced within 30 and 90 previous address references, respectively. These measurements indicate that even environments of wide scope are conducive to flow state maintenance mechanisms.

[9] D. Estrin and D. Mitzel, "An assessment of state and lookup overhead in routers", in *Proceedings of IEEE Infocom 92*, May 1992, pp. 2332–42.

[10] J.-S. Ahn, Peter B. Danzig, D. Estrin, and B. Timmerman, "Hybrid technique for simulating high bandwidth delay computer networks", in *Proceedings of ACM SIGMETRICS '93*, May 1993.

[11] C. Partridge, "A proposed flow specification", Internet Request for Comments Series RFC 1363, September 1992.

[12] M. Laubach, "Classical IP and ARP over ATM", Internet Request for Comments Series RFC 1577, January 1994.

[13] Juha Heinanen, "Multiprotocol encapsulation over ATM adaptation layer 5", Internet Request for Comments Series RFC 1483, July 1993.

[14] D. Grossman, M. Perez, F. Liaw, E. Hoffman, and A. Mankin, "ATM signaling support for IP over ATM", Internet-Draft, April 1994.

[15] P. Ford, Yakov Rekhter, and H.-W. Braun, "Improving the Routing and Addressing of the Internet protocol", *IEEE Network*, vol. 7, no. 3, pp. 10–15, May 1993.

[16] Steve Deering, "Simple internet protocol plus (SIPP) specification", Available as ds.internic.net:internet-drafts/draft-ietf-sipp-spec-00.txt, February 1994.

[17] R. Caceres, "Multiplexing data traffic over wide-area cell networks", Matsushita Information Technology Laboratory Technical Report, March 1993.

[18] H. Saran and S. Keshav, "An empirical evaluation of virtual circuit holding times in IP-over-ATM networks", in *Proceedings of IEEE Infocom 94*, June 1994.

[19] K. Claffy, G. C. Polyzos, and H.-W. Braun, "Traffic char-

acteristics of the T1 NSFNET backbone", in *Proceedings of IEEE Infocom 93*, March 1993, pp. 885–892.
[20] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm", in *Proceedings of ACM SIGCOMM '89*, September 1989, pp. 1–12.
[21] Scott Shenker, David C. Clark, and Lixia Zhang, "A scheduling service model and a scheduling architecture for an integrated services packet network", Internet Draft, March 1994.
[22] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism", in *Proceedings of ACM SIGCOMM '92*, August 1992, pp. 14–26.
[23] Sally J. Floyd, "Link-sharing and resource management models for packet networks", unpublished memorandum, September 1993.
[24] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new resource reservation protocol", *IEEE Network*, vol. 7, no. 5, pp. 8–18, September 1993.
[25] D. J. Mitzel, D. Estrin, S. Shenker, and L. Zhang, "An architectural comparison of ST-II and RSVP", in *Proceedings of IEEE Infocom 94*, March 1994.
[26] R. Jain, "Characteristics of destination address locality in computer networks: a comparison of caching schemes", *Computer networks and ISDN systems*, vol. 18, no. 4, pp. 243–54, May 1990.
[27] David C. Feldmeier, "Improving gateway performance with a routing table cache", in *Proceedings of IEEE Infocom 88*, March 1988, pp. 298–307.
[28] Riccardo Gusella, "A measurement study of diskless workstation traffic on an Ethernet", *IEEE Transactions on Communications*, vol. 38, no. 9, pp. 1557–68, September 1990.
[29] M. R. Macedonia and D. P. Brutzman, "Mbone provides audio and video across the Internet", *IEEE Computer*, vol. 27, no. 4, pp. 30–36, April 1994.
[30] V. Paxson, "Empirically-Derived Analytic Models of Wide Area TCP Connections", to appear in *IEEE/ACM Transactions on Networking, vol. 2 no. 2*, August 1994.
[31] A. Mankin and K. Ramakrishnan, "Gateway congestion control survey", Internet Request for Comments Series RFC 1254, August 1991.
[32] K. Claffy, H.-W. Braun, and G. C. Polyzos, "Tracking long-term growth of the NSFNET backbone", *Communications of the ACM*, vol. 37, no. 8, pp. 34–45, August 1994.
[33] K. Claffy, G. C. Polyzos, and H.-W. Braun, "Internet traffic flow profiling", UCSD TR-CS93-328, SDSC GA-A21526, November 1993.

Appendix

I. Data collection of the case study

Figure 12 provides an abstract illustration of a subset of U.S. Internet interconnectivity. The actual implementation forms a much more complex framework. The right half of figure 12 depicts (and Table IV lists) the five sites at which we collected traffic data, covering a range of Internet traffic aggregation points. We collected packet traces in early 1993 from two T3 NSFNET backbone sites for traffic going from those nodes into the backbone. In particular we selected the FDDI interfaces into the NSFNET backbone node at San Diego and Urbana-Champaign.

These two data trace locations generally allow us to investigate and recommend improvements in operational statistics collection for the NSFNET backbone. However, to test traffic flow methodologies across a broader range of environments, we analyzed packet traces from three more locations: an internal FDDI LAN of one of these national supercomputing facilities, the San Diego Supercomputer Center (SDSC); an FDDI campus backbone of the University of California, San Diego (UCSD), and a smaller departmental subnet, the SDSC visualization laboratory Ethernet, which connects several workstations served by common file servers. For each site we used a dedicated SGI Indigo R4000 workstation to capture two one-hour traces: one during workday hours and the other during the night. For the first two traces, the backbone environments, we only collected data going in one direction, utilizing the FDDI MAC level address of the NSFNET node to filter the traffic into the backbone. The unidirectional collection allows us to assess the impact of the inflow into the NSFNET backbone, and to mirror the results of the NNStat/ARTS traffic characterization described in [32], which operationally collects statistics on inbound NSFNET traffic at each backbone node. For the remaining three data collection locations we did not apply filters but rather collected all IP traffic on the LANs.

Table IV lists the sites at which we collected data, including the starting times and size of each one-hour trace. Table V provides basic population parameters of our data sets, including per-second packet and byte volumes.[11] For the graphs and tables in this paper we will refer to the data sets with the following five acronyms: SD-NSF, UC-NSF, SDSC, UCSD, and SD-viz. An AM or PM suffix identifies whether the data set was a nighttime (approximately 02:00-03:00 AM) or a workday (approximately 14:00-15:00 PM) hour of collection. Our preliminary comparisons of all ten data sets revealed that the AM sets were quite similar to the PM sets with respect to the parameters of interest, e.g., flow durations and volume, although characterized by lower means, e.g., in flow counts and traffic volume. We therefore focus on the PM data sets for the majority of the graphs in this paper.

[11] For more statistics on the composition of these traces, see [33].

TABLE IV

COLLECTION SITES FOR FLOW PROFILING INVESTIGATION

code	site	layer	date	AM start	kpkt	PM start	kpkt
SD-NSF	San Diego FDDI into NSFNET	backbone	23 march 93	02:00	637	14:00	1285
UC-NSF	Urbana-Champaign FDDI to NSFNET	backbone	29/25 march 93	02:00	1274	14:11	4019
SDSC	SDSC FDDI	supercomp. ctr.	29/28 june 93	03:30	316	14:00	1584
UCSD	UCSD academic FDDI	university	11 march 93	02:00	840	14:00	3094
SD-viz	SDSC visualization lab Ethernet	dept. subnet	8/7 march 93	02:00	81	14:00	1101

TABLE V

POPULATION PARAMETERS FOR ONE-HOUR DATA SETS MEASURED BY ONE-SECOND INTERVALS

	total	min	mean	95%	max	SD	total	min	mean	95%	max	SD
data set	kpkt	pck/sec					MB	kbytes/sec				
SD-NSF AM	637	44	178	272	1416	90	134	4	37	72	209	20
SD-NSF PM	1285	79	358	475	924	70	246	16	68	123	225	29
SDSC AM	316	30	88	176	708	46	67	3	17	46	307	17
SDSC PM	1584	55	441	724	1141	160	414	6	115	241	974	83
UCSD AM	840	102	234	361	1083	78	103	7	29	61	229	22
UCSD PM	3094	424	861	1212	1765	185	516	44	143	298	540	73
UC-NSF AM	1274	24	355	661	1032	165	436	2	122	320	653	94
UC-NSF PM	4019	659	1118	1549	2355	228	1151	92	320	548	913	122
SD-Viz AM	81	0	29	155	397	55	51	0	18	108	423	52
SD-Viz PM	1101	1	314	695	1177	213	433	0	124	449	886	148

We then used the collected traces to drive a simulation of soft-flow-state maintenance. To dynamically assess flows, we simulate a state-maintenance machine with an entry for each active flow, as defined above. The simulation proceeds as follows. Each time we see a new flow we create a new timestamped entry. We retain an entry as long as traffic exists for its associated flow. A flow "garbage collection" procedure executed every second deletes all flows which no longer qualify as active according to the timeout value, and for those deleted flows reports the flow volume in packets and bytes, and the flow duration. The flow byte volume includes IP and transport protocol headers. In addition, each second we report the total number of active, new, and deleted flows for that second. Although we only time-out flows synchronously at discrete observation points, every second, as figure 1 depicts, we do record the actual duration based on the difference between the timestamp of the last packet seen in the flow and that of the first packet that incurred creation of the flow.

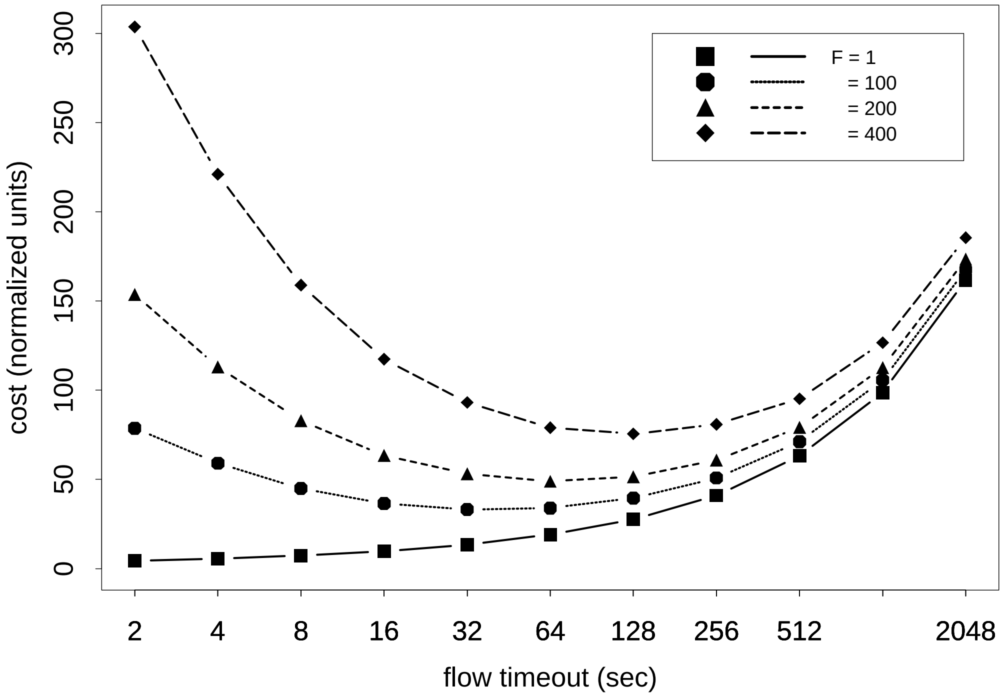

Fig. 9. operating cost versus timeout value for various values of the ratio of flow setup cost to maintenance cost (F) using UC-NSF PM traffic data

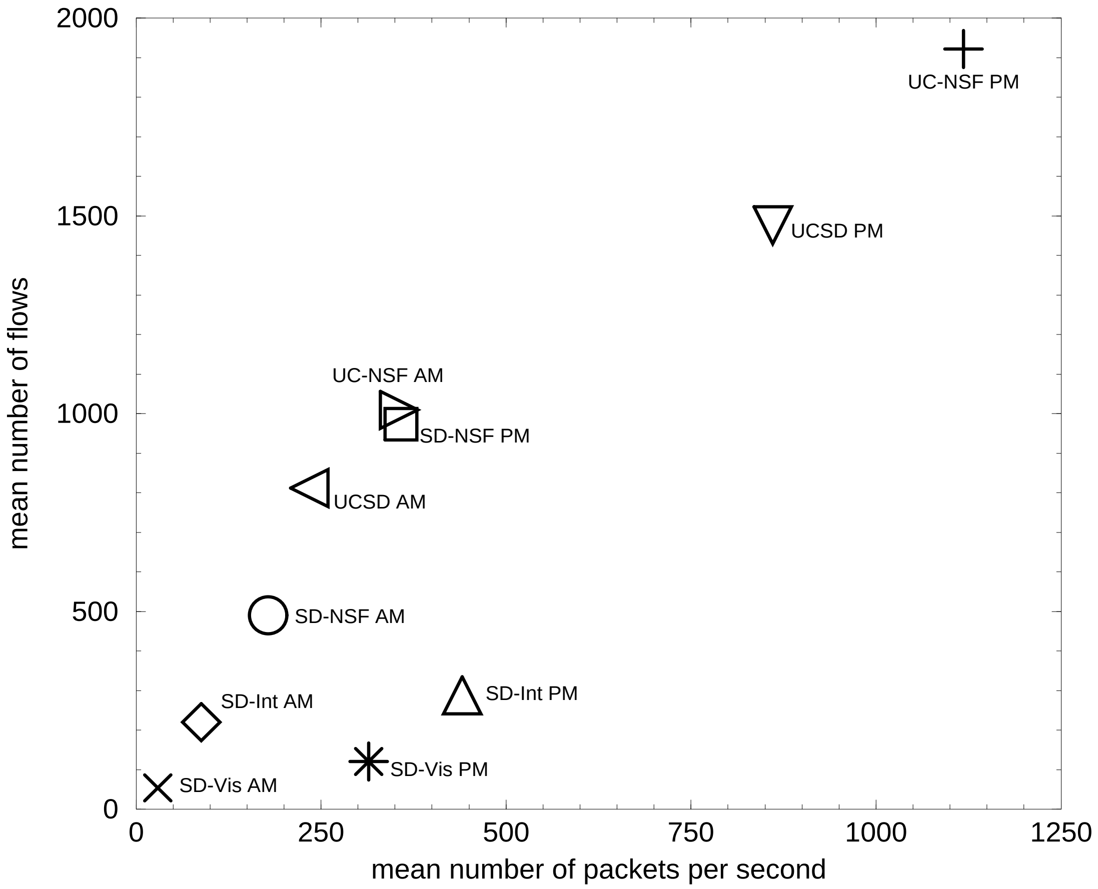

Fig. 10. mean number of host pair flows versus mean per-second packet rates for each environment (64 second flow timeout)

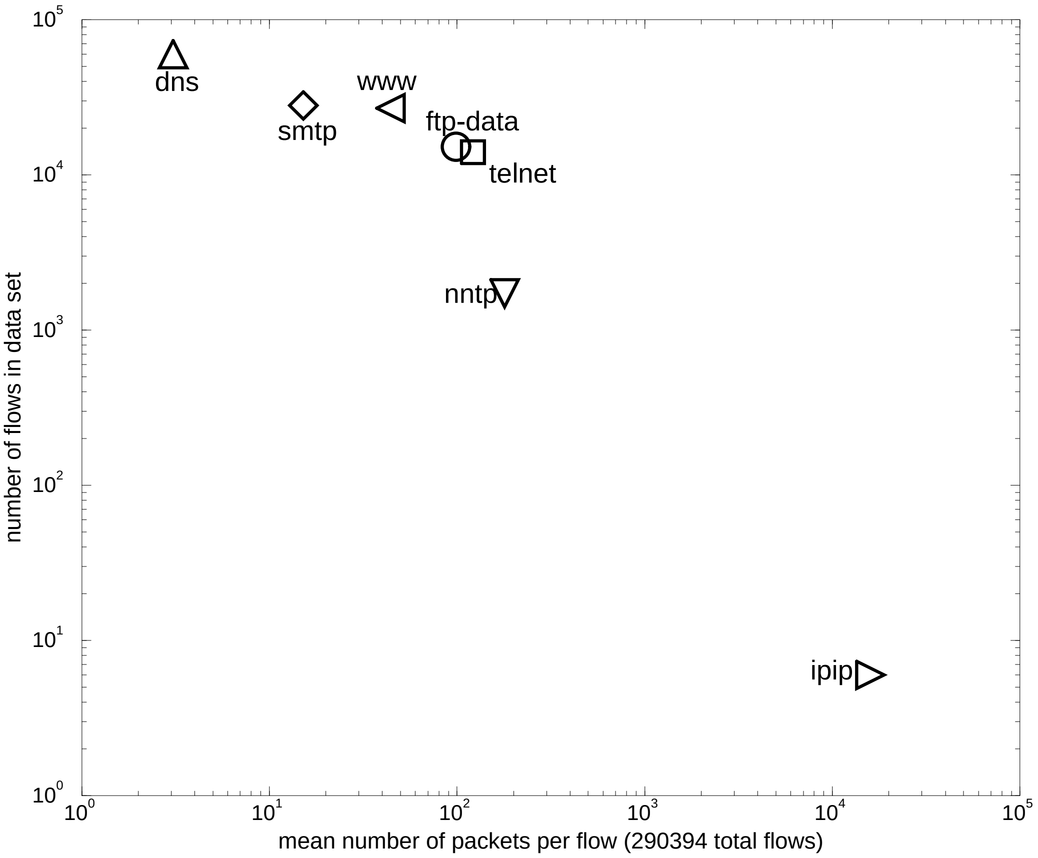

Fig. 11. number of flows per protocol versus packet volume per flow including two newer protocols: IPIP (which includes Mbone) and *www-http* traffic (UC-NSF 1994, 64 second flow timeout)

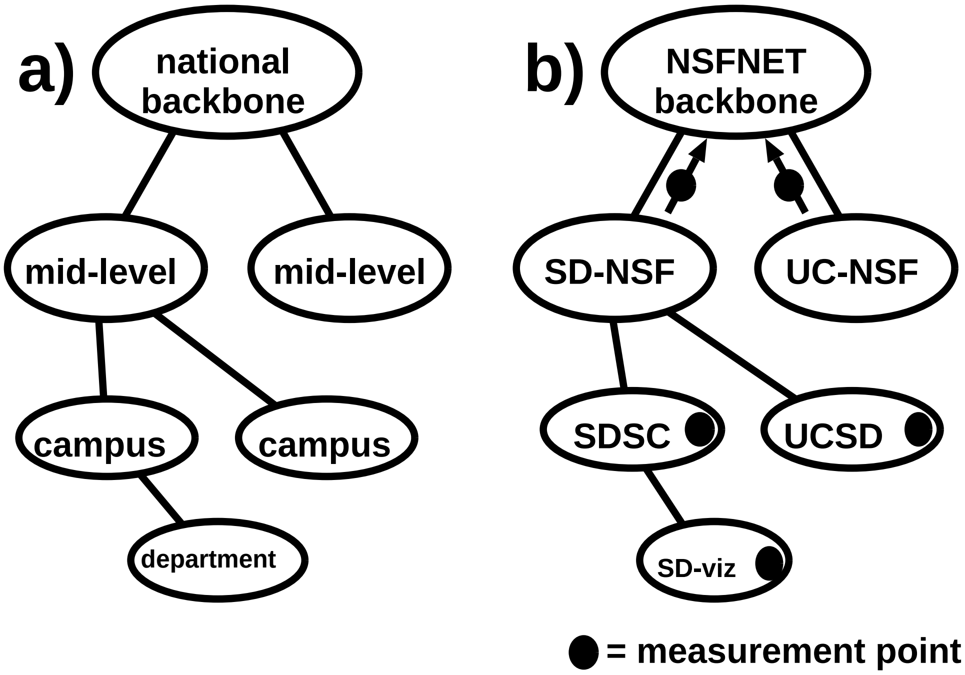

Fig. 12. (a) abstract hierarchical model of U.S. Internet interconnectivity; (b) Internet locations we selected for characterization (SD-NSF: San Diego NSFNET node, traffic going into the backbone; UC-NSF: Urbana-Champaign NSFNET node, traffic going into the backbone; UCSD: UC, San Diego campus backbone; SDSC: San Diego Supercomputer Center, internal FDDI LAN; SD-viz: San Diego Supercomputer Center, visualization laboratory (small subnet of SDSC)