OC3MON: flexible, affordable, high performance statistics collection

Joel Apisdorf
k claffy (NLANR)
Kevin Thompson
and
Rick Wilder
MCI/vBNS

(***current version of software at ftp://nlanr.net/Software/Oc3mon***)

part I: design and implementation

introduction

The Internet is rapidly growing in number of users, traffic volume, and topological complexity. At the same time it is increasingly driven by economic competition. These developments render it more difficult, and yet more critical, to characterize network usage and workload trends, and point to the need for a high performance monitoring system that can provide workload data to Internet users and administrators. To ensure the practicality of using the monitor at variety of locations, implementation on low cost, commodity hardware is a necessity.

In its role as the network service provider for NSF's vBNS (very high speed Backbone Network Service) project, MCI has undertaken the development of an OC3 based monitor to meet these needs. We will describe and demonstrate our current prototype. The goal of the project is to specifically accommodate three incompatible trends:

- current widely used statistics gathering tools, largely FDDI and Ethernet based, are running out of gas, so scaling to higher speeds is difficult
- ATM trunks at OC3c are increasingly used for high volume backbone trunks and interconnects
- detailed flow based analysis is important to understanding usage patterns and growth trends, but such analysis is not possible with the data that can be obtained directly from today's routers and switches

Specific design goals that led to the current prototype are

- a flexible data collection and analysis implementation that can be modified as we codify and refine our understanding of the desired statistics
- low cost, in order to facilitate widespread deployment

The project schedule calls for deploying the monitor in third quarter 1996 in the vBNS. As soon as we demonstrate its stability, we will make the software freely available to others for use elsewhere. Both the flow analysis code and monitor architecture will be public domain.

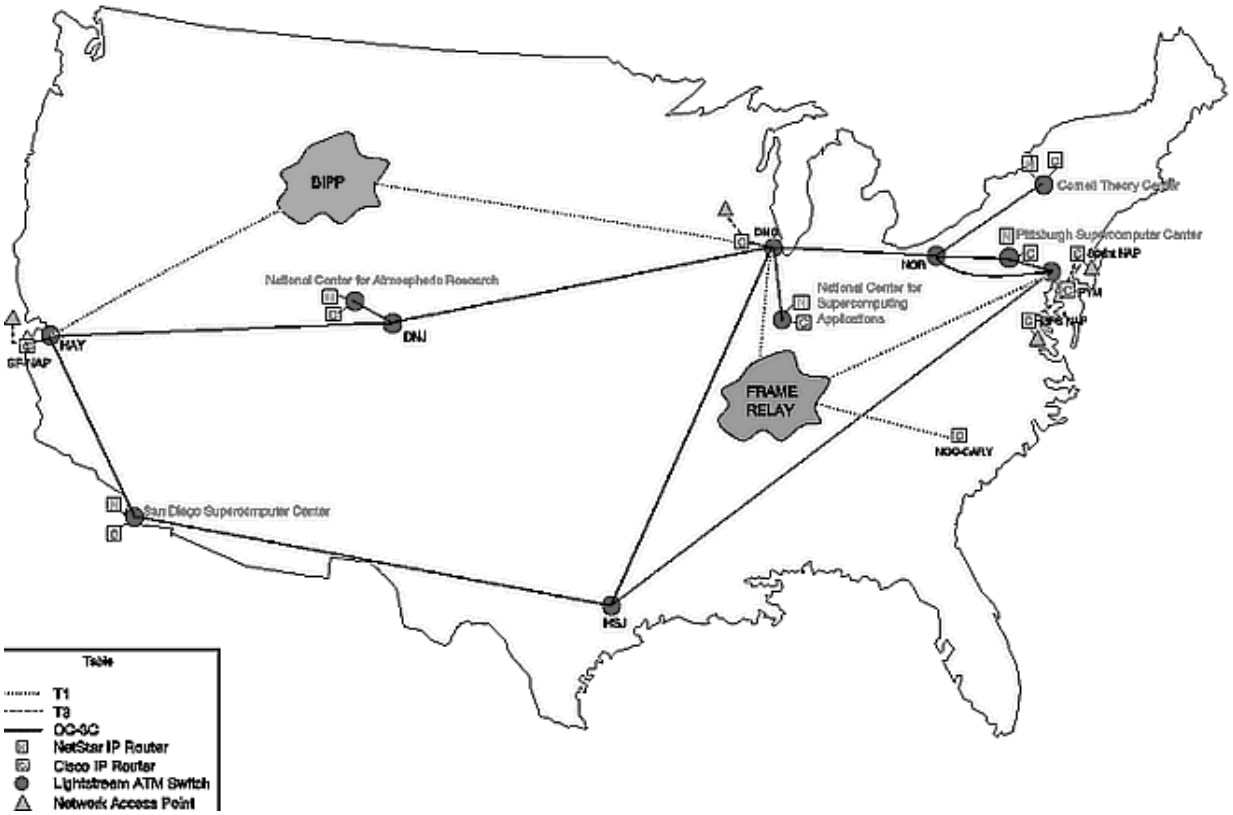
description of the OC3 monitor

hardware

OC3MON is an IBM personal computer clone with 128 MB of main memory, a 166 MHZ Intel Pentium processor, an Ethernet interface, two ATM interface cards, and a 33 MHz 32-bit-wide PCI bus. Our first implementation used ATM interface cards built around Texas Instrument's SAR (segmentation and reassembly) chips due to early availability and low cost. The current version of OC3MON uses a Fore Systems ATM network interface card (NIC) for the PCI bus. The Intel i960 processor on this card allows us to optimize OC3MON operation with custom firmware. We made arrangements with Fore to obtain the necessary source code and freely distribute the custom firmware executables along with the source code developed for the OC3MON system processor.

We attach the OC3MON ATM NICs to an OC3 fiber pair carrying IP traffic, connecting the receive port of each ATM card to the monitor port of an optical splitter, which carries 5% or 10% (depending on the type of splitter used) of the light from each fiber to the receive port of one NIC. as of January 1997, the dual splitter cost is about $800; the NICs run about $1200. Attached to an OC3 trunk terminated on a switching device (e.g., ATM switch or router), one of the OC3MON NICs sees all traffic received by the switching device and the other NIC sees all traffic transmitted by the switching device. In the vBNS, we will attach an OC3MON to each connection from the wide area ATM backbone to the primary nodes at the supercomputer centers. (See figure 1).

figure 1: National Science Foundation very high speed Backbone Network Service (vBNS) topology. The backbone connects the 5 supercomputer centers at Cornell, Pittsburgh, Urbana-Champaign, Boulder, and San Diego at OC3, with T3 connectivity to the 4 original NAPs in DC, Pennsauken, Chicago, and San Francisco. MCI is currently upgrading the backbone to OC12 in 1997.



software: why we didn't choose unix

The DOS-based software running on the host PC consists of device drivers and a TCP/IP stack combined into a single executable; higher level software performs the real-time flow analysis. Several design constraints motivated our decision to use DOS-based functionality rather than a UNIX kernel. First, the Texas Instruments cards in the original OC3MON design required polling the interface card for cells at 1/128 the cell rate in order to obtain accurate timestamp granularity at full OC3 rate, since the card itself did not timestamp the cells. Monitoring a full duplex link requires two cards in the machine, which meant that we had to reprogram the timer interrupt to occur every 128/(2*353207.5) = 1/5518 second. Because Unix has a higher interrupt latency than DOS, we were better off with DOS at that point.

Our latest design uses Fore cards that can attach timestamps to the cells on their own; the host no longer needs to poll the card at all. We need to interrupt only at most every 1/40 second (i.e., if both links received 40 byte packets simultaneously), so low latency is no longer a constraint. However, we would not have gotten a prototype working without the control that DOS provided.

Second, we needed the ability to monopolize the entire machine, which is easier with DOS than Unix. OC3MON needs to provide the hardware with large blocks of contiguous physical memory, so we did not want the operating system to have to maintain knowledge about the memory and possibly fragment, resulting in lower efficiency on card-to-host buffer transfers. We did not want the kernel to suddenly decide it needed to capture the PCI bus to swap a page to disk, nor did we want the analysis software to fall behind because the kernel scheduled another process. We also wanted more control than Unix provides over when TCP/IP could have a time slice. These concerns will become even more important

for OC12MON, which faces four times the data rate, or even if we dedicate a separate machine to each direction of traffic faces twice the data rate.

The disadvantage of DOS is its blocking I/O routines, whereas Unix would provide non-blocking I/O. Ports of OC3MON to Linux and NetBSD are now underway by vBNS researchers at NCSA and SDSC, respectively.

software: background on ATM

The host software directs each ATM NIC to perform AAL5 (ATM Adaptation Layer) reassembly on a specified range of virtual circuit and virtual path identifiers (VCI/VPI) (defaults is 7 bits of VPI and 13 bits of VCI, zeroing unused bits of both). Note that Cisco routers and Fore switches also support AAL3/4, but MCI does not use it on either the vBNS or their commodity infrastructure because it consumes an additional 4 bytes from each cell (above the 5 already used for the ATM header) to support submultiplexed channels within a given VP/VC. Since the LLC/SNAP 8-byte per-frame header, which RFC1577 says a router should insert by default in front of all packets, already includes a 2-byte ethertype field that allows, if needed, multiplexing of different protocols (IP, IBM SNA, Novell IPX, etc) on the same VC, including AAL3/4 support in the design would not have been beneficial. Note that the addition of any bytes to a simple (no data attached) TCP ACK causes the 8-byte ATM AAL5 trailer to require another cell, doubling the bandwidth used by such packets. Also note that the AAL5 trailer already has room for a user-defined 16-bit field, and the router does not care about protocol information until after receiving a complete packet, so supporting protocol multiplexing did not require the per-packet, variable-length, upfront overhead of an LLC/SNAP header.

AAL5 makes use of a user-defined single-bit field in the ATM header to indicate whether a cell is the last in a frame. AAL5 also assumes that cells for a given frame will not be interspersed with cells for another frame within the same VP/VC pair -- no multiplexing occurs inside a VC. Combined with a single bit of state per VP/VC pair maintained by the receiver, which indicates that the cell is in the middle of a frame for that VP/VC pair, there is enough information to reassemble the frame.

The receiving card normally also needs a pointer to the location in host memory (or card memory if the card were to buffer received frames before DMAing them to the host, which ours does not) where it has put previous cell payloads for incomplete frames, so that it can store future cells contiguously, or at least maintain a linked list. Once a SAR (segmentation and reassembly) engine design involves this leap from one bit to the size of a pointer, most go even further and use several more words for management purposes. VC table entries on the order of 16 to 32 bytes are not uncommon. Thus most ATM NICs are limited to on the order of 1024 VC/VP combinations active at a time.

Since OC3MON has no need for data beyond the first cell, and since it already maintains per-flow state on the host, we chose to limit the per-VC state on the card to the bare minimum: one bit (2 bits when we implement up-to-3-cell capture for OC12MON). Although the Fore cards have 256KB of memory, some of it is used for the i960 code (about 32K), the OS, reassembly engine data structures, and the stack. Since the VP/VC lookup needs an exact power of two, the largest we could get was 128KB. (Single-bit state for 2^20 VP/VC combos = 2^17 bytes = 131072 bytes = 128KB).

The cards for OC12MON will have 4MB of memory, all of which will be available for state information, allowing us to keep single-bit state for all 2^24 = 16777215 possible ATM UNI (user to network interface) VPI/VCI combinations. Using 2 bits of state per VPI/VCI combination (for example

if we wanted 3 cells per packet sent to the host) leaves room for only have as many VPI/VCI combinations. Note that when we copy multiple cells of the same packet to the host, the card will not place them near each other so the host must do further reassembly using the ATM headers.

Examining twenty bits of VCI/VPI information allows OC3MON to monitor over one million VCs simultaneously. The host controls exactly how many bits of the VCI this 20-bit index will include; the rest derive from the VPI. The host also specifies at startup what to expect for the remaining bits of the VPI/VCI, i.e., those not used for indexing into the card's state table. The card can then complain about, or at least drop, non-conforming cells.

Many SAR engines choose to completely ignore the VPI and any bits of the VCI not used for indexing. When presented with the arbitrary VPI/VCI combinations we expect to see on a general purpose monitor, inevitable aliasing will cause collisions in reassembly state among VPI/VCI pairs. OC3MON avoids this situation by: (1) using a large number of VPI and VCI bits for its table lookup, leading to more successful reassemblies in the presence of arbitrary channel usage; and (2) comparing the bits it does not use for indexing with the expected values as described above, which keeps unsuccessful reassemblies from corrupting successful ones.

Since we want OC3MON to be able to see traffic on (almost) any VPI/VCI without prior knowledge of which circuits are active, and because the fast SRAM (static random access memory) used on such ATM cards for state tables is expensive and not amenable to modification by the consumer, this design turned out to be extremely advantageous.

software: description

The AAL reassembly logic is customized to capture and make available to the host only the first cell of each frame. The 48 bytes of payload from these cells typically contain the LLC/SNAP header (8 bytes), IP and TCP header (typically 20 bytes each). Copying the 5-byte ATM header also allows us the flexibility of doing ATM based analysis in the future. The SAR engine discards the rest of each AAL5 protocol data unit (PDU, equivalent to a frame or IP packet), limiting the amount of data transferred from the NICs over the PCI bus to the host. Although as yet unimplemented, one could increase the amount collected to accommodate IP options or larger packet headers as specified for IP version 6. Currently, however, the cards only pass the first cell of each packet, so when IP layer options push part of the TCP header into the second cell, these latter portions will not be seen by the host. Although suboptimal, we decided the savings in PCI (peripheral component interconnect) bus cycles, host memory, and CPU usage justified this decision.

Each NIC (network interface card) has two 1MB buffers in host memory to hold IP header data. These cards are bus masters, able to DMA (direct memory access) header data from each AAL5 PDU into the host memory buffers with its own PCI bus transfer. This capability eliminates the need for host CPU intervention except when a buffer fills, at which point the NIC generates an interrupt to the host, signaling it to process that buffer up to memory while the NIC fills the other buffer with more header data. This design allows the host to have a long interrupt latency without risking loss of monitored data. The NICs add timestamps to the header data as they prepare to transfer it to host memory. Clock granularity is 40 nanoseconds, about 1/70 of the OC3 cell transmission time.

The resulting trace is conducive to various kinds of analysis. One could just collect a raw timestamped packet level trace in host memory, and then dump the trace to disk. This technique is useful for

capturing a detailed view of traffic over a relatively brief interval for extensive future study. However, because we currently use the DOS-supplied disk I/O routines, which are blocking, we cannot write to disk simultaneously with performing flow analysis. Therefore one can only collect a trace as big as the size of host memory, which in our case would be 128 million bytes, and then must stop OC3MON header collection to let OC3MON transfer the memory buffer to disk. In the future we hope to develop separate I/O routines that directly use the hardware, allowing us to keep up with OC3 line rate, assuming some minimum packet size.

Because the amount of data captured in a packet level trace and the time needed for our disk I/O inhibits continuous operational header capture, the default mode of OC3MON operation is to maintain IP flow statistics that do not require the storage of each header. In this mode of operation, concurrently with the interrupt driven header capture, software runs on the host CPU to convert the packet headers to flows, which are analyzed and stored at regular intervals for remote querying via a web interface. The query engine we use is similar to that found in the NLANR FIX West workload query interface (**http://www.nlanr.net/NA/**). We will describe the methodology for deriving flow information in the second half of the paper, followed by some example snapshot statistics taken with OC3MON.

internal oc3mon data transfer rate

We tested OC3MON on a concatenated OC3, or OC3c, link fully occupied with single cell packets (as would occur in the admittedly unlikely event of continuous TCP ACKs with no data and LLC/SNAP disabled on the routers), which yields 353207.5 packets per second (or in the single-cell packet case, the same number of cells) across each half-duplex link. Each header, including timestamp, ATM, LLC/SNAP, IP and TCP headers consumes 60 bytes, so the internal bus bandwidth required would be 353207.5 * 2 * 60 * 8 = 339 Mbits. The 32-bit, 33MHz bus in the PC is slated at 1.056 gigabits, so we do not expect bus bandwidth to be the bottleneck until we need to support OC12. There are already extensions to the PCI standard to double the bus width and speed, so when we need to support the worst OC12 workload (i.e., single-celled traffic), the bus technology will likely be available. (Digital has already demonstrated the 64-bit part.)

sampling

We do not currently support sampling in the capture or flow analysis software. Although sampling is one option to avoid losing gaps of data during traffic burst, it changes the statistics of a timeout-based flow analysis in a most unclear way. For simply collecting packet headers, or to venture into the murky statistics zone that flows of sampled packet streams would involve, we could modify the card to support sampling in the future. Testing OC3MON with single-cell packets on OC3 indicated that we do not lose packets, so supporting sampling is not high priority for us at this time.

security

Secure access is a problem for any machine, especially monitoring-capable ones. On both the vBNS and the commodity MCI Internet backbone, the monitoring machines live in locked machine rooms or secure terminal facilities. One can also require the monitor to only accept packets from certain IP addresses, or configure nearby routers to block packets from unknown addresses from reaching it.

We obtain the flow data summary from OC3MON via a passwd-protected remote query to a port on the machine. This level of security is equivalent to that provided by most SNMP implementations. The

query process triggers OC3MON to clear the current flow summary, but OC3MON retains the active flows in memory.

An ANNEX terminal server supports console one-time password access to OC3MON.

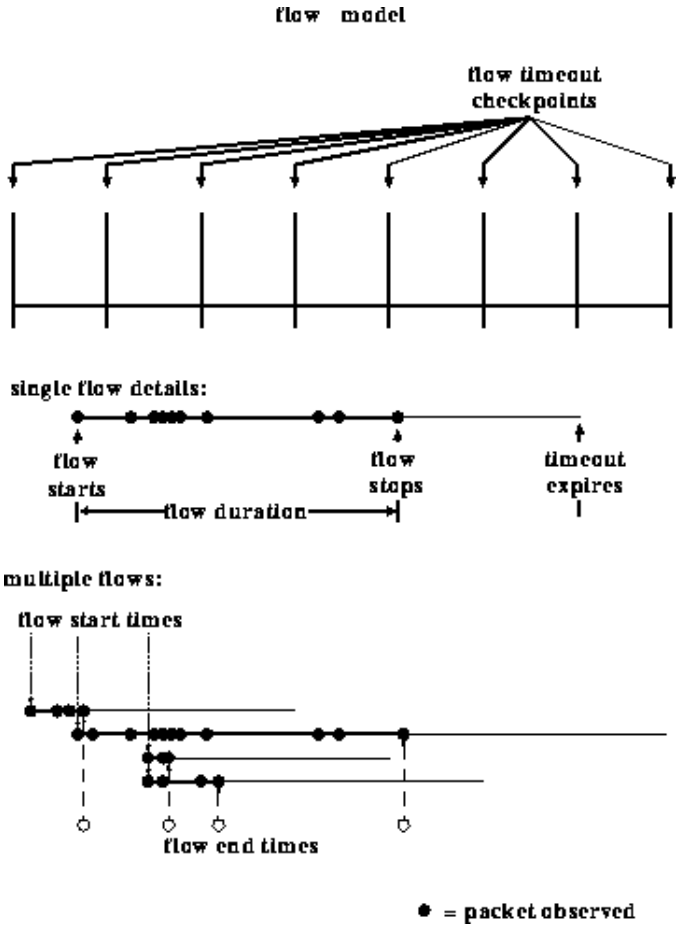
part II: methodology and results

flow profiling methodology

In deriving flow profile information from packets, we need to establish a definition of what constitutes a flow. Since the appropriate constraints to put on what one labels a *flow* depend on the analysis objective, our methodology specifies a set of parameters that are configurable based on the analysis requirements.

We specifically do not restrict ourselves to the TCP connection definition, i.e., SYN/FIN-based, of a flow. Instead, we define a flow based on traffic satisfying specified temporal and spatial locality conditions, as observed at an internal point of the network, e.g., where OC3MON sits. That is, a flow represents 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 meet the flow specification arrive separated in time by less than a specified timeout value, as figure 2 illustrates. The lower half of the figure depicts multiple independent flows, of which hundreds of thousands may be active simultaneously at WAN transit points.

figure 2: defining a flow based on timeout during idle periods



This approach to flow characterization allows one to assess statistics relevant to issues such as route caching, resource reservation at multiple service levels, usage based accounting, and the integration of IP traffic over an ATM fabric. Our definition of the timeout is similar to that used in other studies of timeout-based traffic behavior [4,6,8]. Jain and Routhier originally selected for their investigation of local network traffic a timeout of 500 milliseconds. Wide area traffic studies of the transport layer have typically used longer timeouts, between 15 and 20 minutes [8,10]. Caceres *et al.* 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 [10] 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 [6] used a 15 minute timeout.

We explored a range of timeouts in Claffy *et al.* [3], and found that 64 seconds was a reasonable compromise between the size of the flow table and the amount of work setting up and tearing down flows between the same points. The timeout parameter is configurable in OC3MON, we have used the default of 64 seconds for the measurements in this paper. Initial tests with timeouts as large as 10 minutes did not significantly increase the number of flows, but we have not yet tested it under heavier data streams.

This timeout-based flow definition allows flexibility in how one further specifies a flow. There are other aspects that structure a flow specification: directionality, one sided vs. two sided, endpoint granularity, and functional layer.

- **flow directionality**: one can define a flow as unidirectional or bidirectional. While connection-oriented TCP traffic is bidirectional, the profiles of the two directions can be quite asymmetric. Each TCP flow from A to B also generates a reverse flow from B to A, at the least for small acknowledgement packets. 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.

- **one versus two endpoint aggregations of traffic**: This second aspect of a flow is related to the first. One can 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 and to a specific pair of network numbers. Although single endpoint flows can be configured, OC3MON uses two endpoint flows by default, specifically at the host pair granularity.

- **flow endpoint granularities**: The third aspect of a flow is the endpoint granularity, or the extent of the communicating entities. Possible granularities include traffic by application, end user, host, IP network number, Autonomous System (AS), external interface of a backbone node, backbone node, backbone, or multibackbone environment (e.g., of different agencies or countries). 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 that 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. When 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 is also important. Furthermore, as new routing mechanisms utilize alternative hierarchical definitions related to IP network numbers (e.g., CIDR masks), the desired granularity will likely shift.

Network administrators may want to define flows at a coarser granularity, such as aggregating network number pairs for which they create virtual circuits across their transit network. For example, an ATM cloud may bundle many finer grained IP flows within each ATM circuit. Conversely, a finer 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, and the need to 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.

- **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 transport connection information, e.g., SYN and FIN packets of the TCP protocol which support explicit connection setup and teardown. Because we want 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. We provide further details in an earlier study [3].

configurability

These four aspects -- directionality, one-sided vs. two-sided aggregation, endpoint granularity, and functional layer -- provide a framework for specifying a flow profile structure. We designed the flows analysis software in OC3MON as flexibly as possible. One can specify a specific flow timeout (in seconds), an endpoint granularity (network, host, or host/port), and one-sided or two-sided flows(source, destination, or pair). One can also restrict OC3MON to analyzing flows for a specific transport protocol, port number, or host address.

For our flow profiling we use host pair plus source and destination application identifier (i.e., UDP/TCP port number), if they exist. That is, for the measurements in this paper, OC3MON considers a flow unique based on its protocol, source IP address, destination IP address, source port, and destination port, and a 64 second timeout. A packet is considered to belong to the same flow if no more than 64 seconds have passed since the last packet with the same flow attributes. When flows time out, they are passed up to the statistics routines that update accumulators for remote querying via the Ethernet interface at regular intervals. The results of these queries, still in raw flow format, are then stored on a web server that supports a menu-driven interface. The menus, illustrated in figures 4, 5, and 6, allow users to customize graphs of the data according to their interest.

address-space analysis

The current release of OC3MON supports the granularity of classless network using a CIDR-aware IP-to-AS-path mapping derived from a periodically updated core internet backbone routing table. This includes flow conversion to the autonomous system (AS) granularity, which allows for assessment of traffic flow at a more convenient macroscopic level. Table 1 shows an example AS report. Note the actual data shown represents one hour of vBNS traffic collection, and is not necessarily representative of long-term traffic patterns on the vBNS.

Table 1: AS Report from OC3MON

Flow Report

Top AS Pairs sorted by bytes

Site Location: ncar

Date: 01/30/97, Time: 16:05:00
AS Table

#	AS src name	AS src number	AS dest name	AS dest number	#Flows	#Packets	#
1	XXXX	xxx	XXXX-XXXX	xxxxx	942	335260	1.
2	XXXX	xx	XXXXXXXXX	xxxxx	117	180185	7.
3	XXXX	xxx	XXXX	xxx	14888	141317	7.
4	XXXXXXX	xxx	XXXXXXXXXXXXXXXX	xxx	60	128541	7.
5	XXXXXXXXXX	xxxx	XXXXXXXXX	xxxxx	69	139690	6.
6	XXXXXXXXXXXXXXX	xxx	XXXXXXXXX	xxxxx	115	95079	4.
7	XXXXXXXXXX	XXXX	XXXXXXXXX	xxxxx	68	47265	3.
8	XXXXXXX	xxx	XXXXXXXXX	xxxxx	30	55648	3.
9	XXXXXXX	xxxx	XXXXXXXXX	xxxxx	73	64028	2.
10	XXXXXXX	xxx	XXXXXX	xxx	51	15293	7.

On the microscopic level, it is useful to identify the heavy traffic sources and destinations. OC3MON reports on "heavy hitters": IP addresses that are the most frequently seen according to flows, packets, and bytes.

Table 2: Heavy-Hitter Report from OC3MON

Flow Report

Top Heavy Hitters sorted by bytes

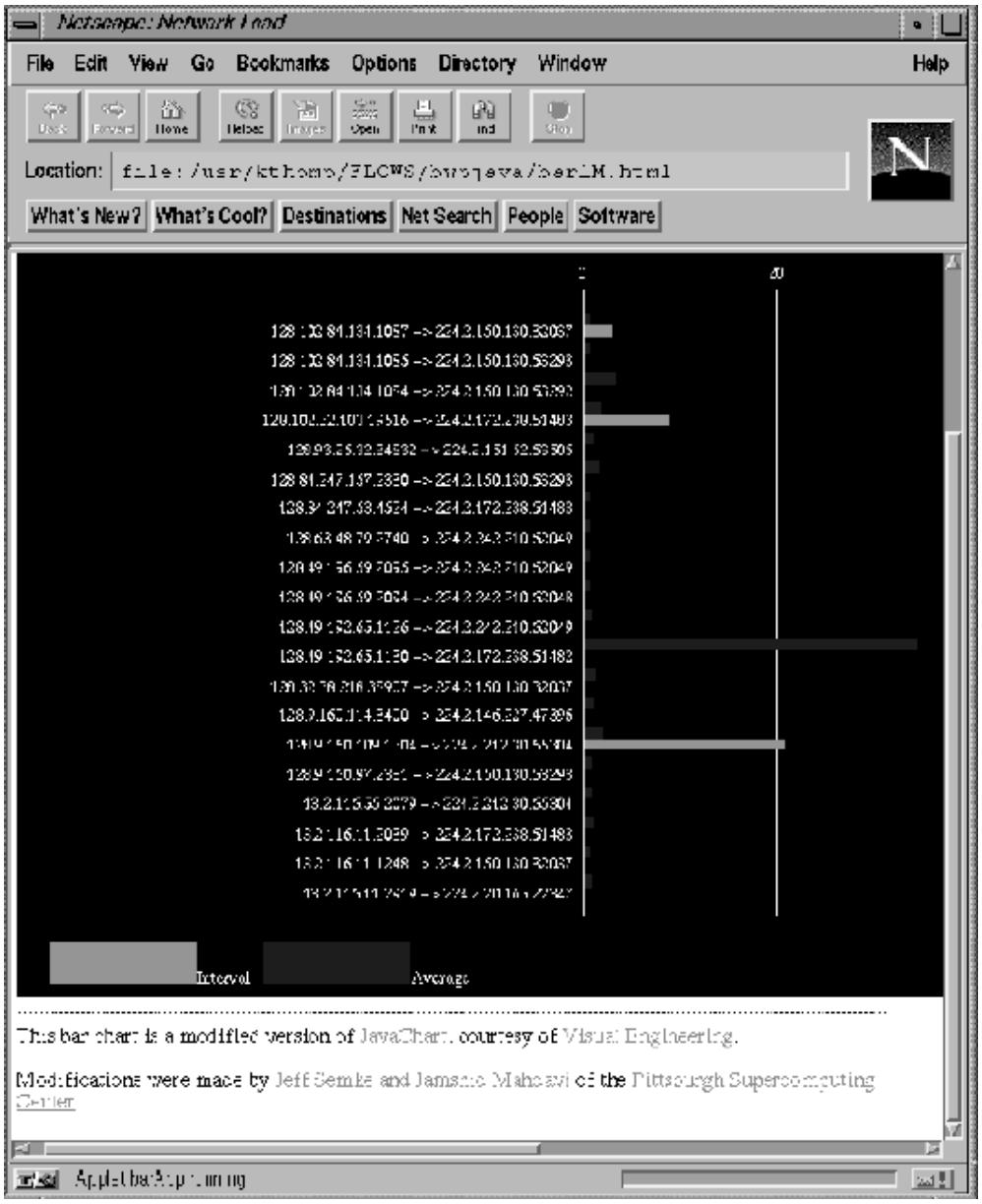
Site Location: ncar

Date: 01/30/97, Time: 14:35:01 a 5-minute sample
Heavy Hitters

#	Source IP address	Hostname	proto	port	#Flows	#Packets	#Bytes	Cum Percenta of Tota Trunk Traffi
1	yyy.yyy.yyy.yyy	xxxxxxxxxxxx	TCP	http	82	908	4.08e+05	1.06
2	yyy.yyy.yyy.y	xxxxxxxxxxxxxxxxxxxxxx	TCP	http	43	421	1.59e+05	1.47
3	yyy.yyy.yyy.yy	xxxxxxxxxxxxxxxxxx	TCP	http	14	227	1.17e+05	1.77
4	yyy.yyy.yyy.yy	xxxxxxxxxxxxxxxxxxxxxxx	TCP	http	83	303	1.14e+05	2.07
5	yyy.yyy.yyy.yy	xxxxxxxxxxxxxxxxxx	TCP	http	10	145	1.13e+05	2.36
6	yyy.yyy.yyy.yy	xxxxxxxxxxxxxxxxxxxxxx	TCP	http	42	299	1.01e+05	2.63
7	yyy.yyy.yyy.yy	xxxxxxxxxxxxxxxxxxxxxx	TCP	http	11	192	8.62e+04	2.85
8	yyy.yyy.yyy.y	xxxxxxxxxxxxxxxxxxxxxx	TCP	http	15	202	7.74e+04	3.05
9	yyy.yyy.yyy.y	xxxxxxxxxxxxxxxx	TCP	http	4	155	7.16e+04	3.24
10	yyy.yyy.yyy.yy	xxxxxxxxxxxxxxxxxx	TCP	http	5	73	6.42e+04	3.40

OC3MON also reports active flows in conjunction with a remote real-time graphic display. In addition to reporting based on expired flows, OC3MON can periodically multicast summary information, including source/destination IP addresses, of active flows. These multicast updates allow for monitoring the data at multiple locations. The vBNS has this feed activated on its 5 OC3MONs throughout the network. Each monitor is given a multicast address on the command line on which to transmit, and a tunnel endpoint used as the destination for encapsulating the IP multicast. The vBNS Cisco router at each node acts as the tunnel endpoint, forwarding multicast data to vBNS-connected destinations. The receiveing process is a web-based java applet which displays overall average and instantaneous bit rates of IP address pairs. Figure 3 shows a screen capture of the program displaying the active feed from the vBNS OC3MON at PSC.

figure 3: OC3MON Active Flow Display



example statistics

To illustrate the kind of graphs and tables one can retrieve, we provide sample graphs of OC3MON measurements on an OC3 trunk of MCI's IP backbone and MCI's vBNS during the period between 25 January and 30 January 1997. Space restrictions prevent us from showing every graph type in this paper; we only provide a small set of possible plots to illustrate the utility of the tool.

basic counters: packets, bytes, flows

Figures 4, 5, and 6 show the main menu and two submenu form interfaces for the query engine, respectively.

figure 4: The main menu lets you choose either a snapshot of a specific monitoring interval (configured for 5 minutes here), or select one of the submenus.

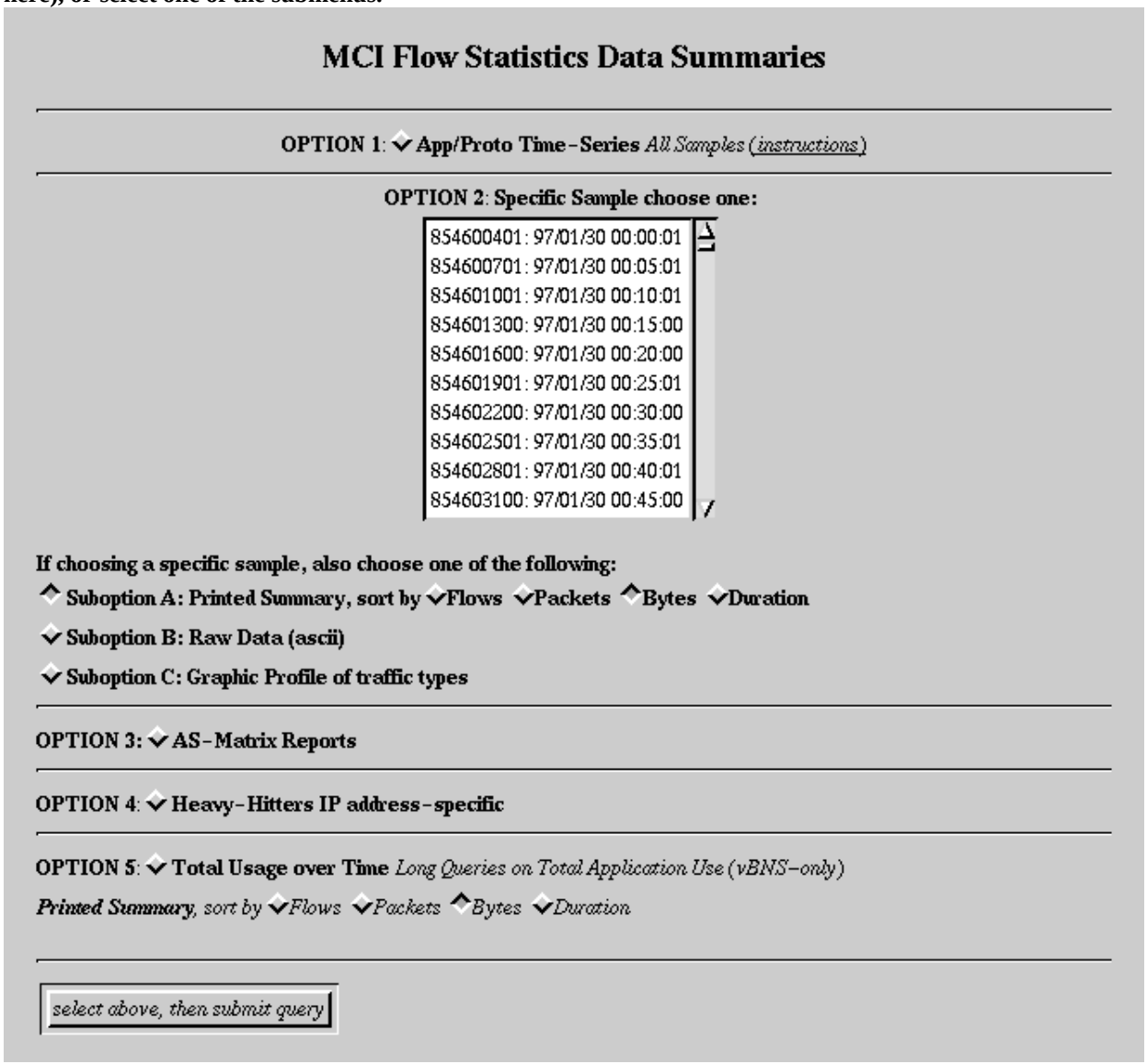


figure 5: time-series menu for OC3MON query engine (invoked upon clicking option 1 *time-series* in main menu)

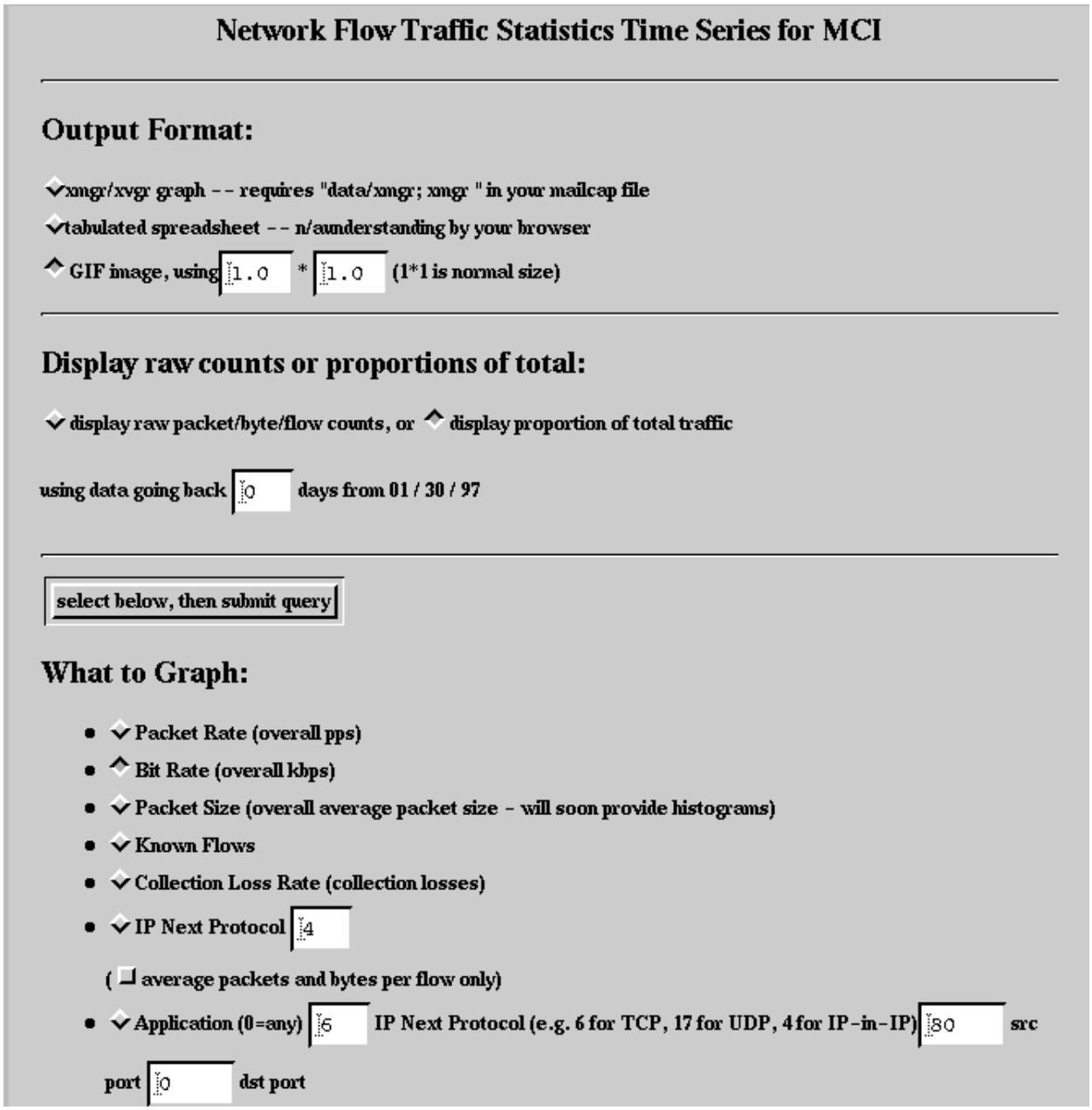
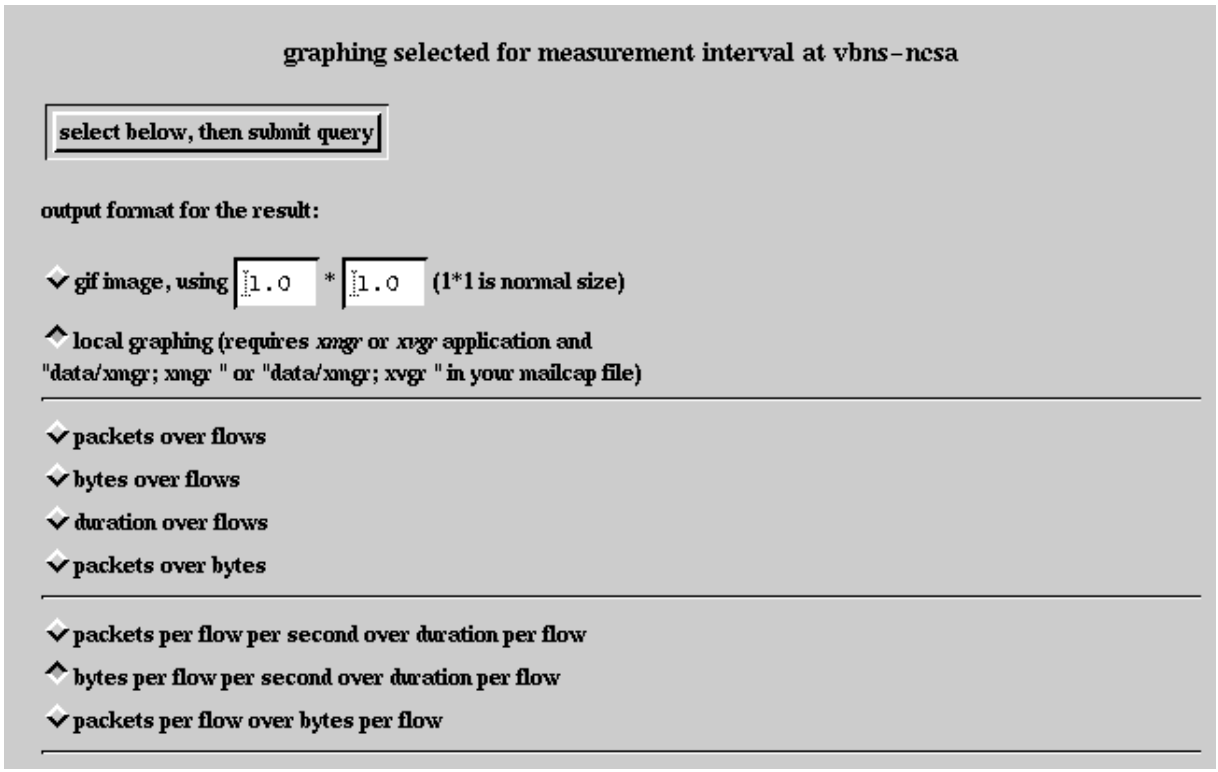
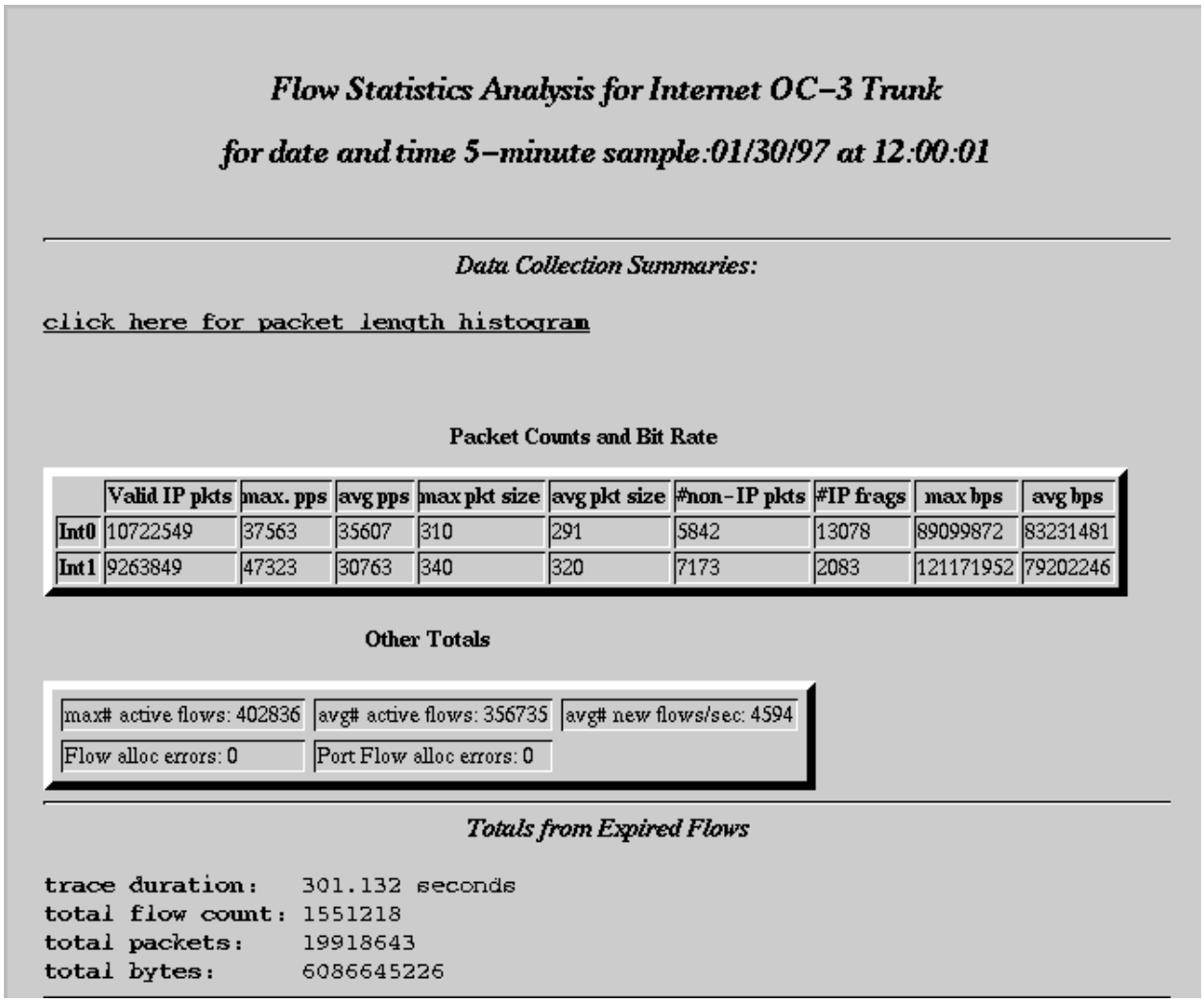


figure 6: 2D profiles menu for OC3MON query engine (invoked upon clicking suboption C *Graphic profile* option in main menu)



The main menu lets you choose either a snapshot of a specific monitoring interval (configured for 5 minutes here), or select one of the submenus. The snapshot returns a page similar to that shown in figures 7 and 8.

figure 7: flow assessment snapshot of traffic during single interval of OC3MON collection - Page 1 (invoked upon clicking suboption A, *printed summary* in the main menu)



Flow Statistics Analysis for Internet OC-3 Trunk

for date and time 5-minute sample:01/30/97 at 12:00:01

Data Collection Summaries:

click here for packet length histogram

Packet Counts and Bit Rate

	Valid IP pkts	max. pps	avg pps	max pkt size	avg pkt size	#non-IP pkts	#IP frags	max bps	avg bps
Int0	10722549	37563	35607	310	291	5842	13078	89099872	83231481
Int1	9263849	47323	30763	340	320	7173	2083	121171952	79202246

Other Totals

max# active flows: 402836	avg# active flows: 356735	avg# new flows/sec: 4594
Flow alloc errors: 0	Port Flow alloc errors: 0	

Totals from Expired Flows

```
trace duration:   301.132 seconds
total flow count: 1551218
total packets:    19918643
total bytes:      6086645226
```

figure 8: flow assessment snapshot of traffic during single interval of OC3MON collection - Page 2 (invoked upon clicking suboption A, *printed summary* in the main menu)

Itemization by IP protocols

Proto#	Protocol	Flows	Packets	Bytes	Duration	flows%	pkts%	bytes%	dur%
1	ICMP	18303	103149	10285824	565039	1.20	0.50	0.20	2.50
2	IGMP	44	6041	2611081	6300	0.00	0.00	0.00	0.00
4	IP	40	94521	29592623	3954	0.00	0.50	0.50	0.00
6	TCP	1190310	17836747	5814263294	17641946	76.70	89.50	95.50	78.90
17	UDP	342450	1876733	229625512	4125010	22.10	9.40	3.80	18.50
41	IPv6	8	479	121620	1862	0.00	0.00	0.00	0.00
47	GRE	1	13	738	273	0.00	0.00	0.00	0.00
57	SKIP	5	322	67688	454	0.00	0.00	0.00	0.00
80	ISO-IP	1	5	270	240	0.00	0.00	0.00	0.00
83	VINES	1	13	614	296	0.00	0.00	0.00	0.00
93	AX.25	47	348	37575	1876	0.00	0.00	0.00	0.00
94	IPIP	6	255	36953	303	0.00	0.00	0.00	0.00
255	Reserved	2	17	1434	75	0.00	0.00	0.00	0.00

#simultaneous flows at a given time is estimated at: 66183.481384

Application Details

Sorted by bytes

Protocol	src port	dest port	Flows	Packets	Bytes	Duration	flows%	pkts%	bytes%	dur%
TCP	http	0	495030	6589476	4010854477	6082525	31.90	33.10	65.90	27.20
TCP	0	http	543224	6717744	462438250	6514203	35.00	33.70	7.60	29.10
TCP	0	smtp	36947	683154	255154870	663189	2.40	3.40	4.20	3.00
TCP	ftp-data	0	1871	272506	188725120	107693	0.10	1.40	3.10	0.50
TCP	0	nntp	1313	276622	115969817	135043	0.10	1.40	1.90	0.60
UDP	domain	domain	228406	608700	69478806	2734533	14.70	3.10	1.10	12.20
TCP	nntp	0	1153	223887	67332972	111175	0.10	1.10	1.10	0.50
UDP	7648	7648	140	73543	35068973	16652	0.00	0.40	0.60	0.10
TCP	0	ftp-data	2010	224338	30193135	112631	0.10	1.10	0.50	0.50

Figures 9, 10, 11, and 12 show packets, bytes, flows, and average per-second packet size for the 6 day period. Rates for packets and bytes are measured for each direction on the OC-3 trunk. During this measurement interval, the average number of packets per second cycled between ten and 45 thousand, depending on direction. (MCI has also installed parallel OC3s where traffic demands require them, in addition to OC12s, into its IP infrastructure.)

The average number of flows per second goes from around 150,000 at night to over 450,000 in the case of Tuesday, 28 January. Note the average packet size goes in the opposite direction -- the per-second average packet size gets larger at night, presumably due to less interactive traffic and likely occurrence of automatic backups.

figure 9: average packets per second over 5 minute intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97

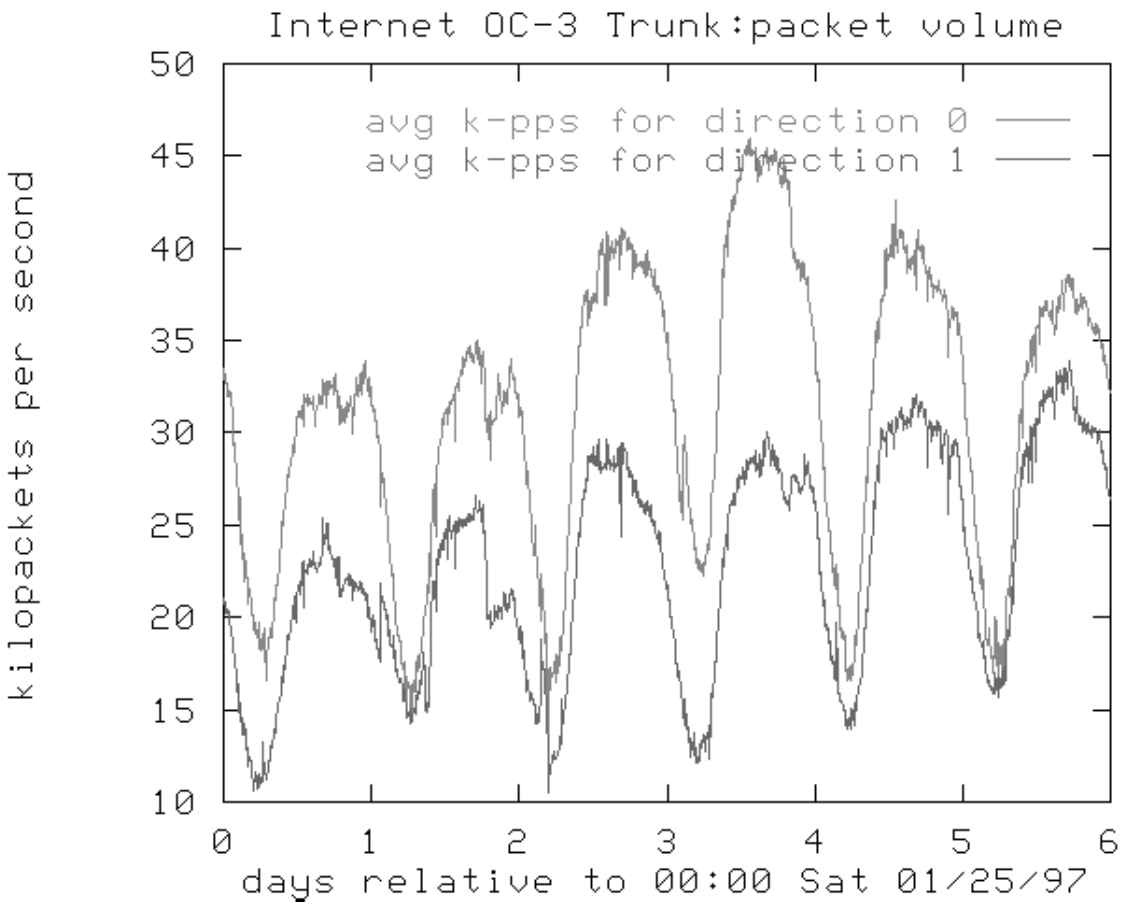


figure 10: average bits per second over 5 minute intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97

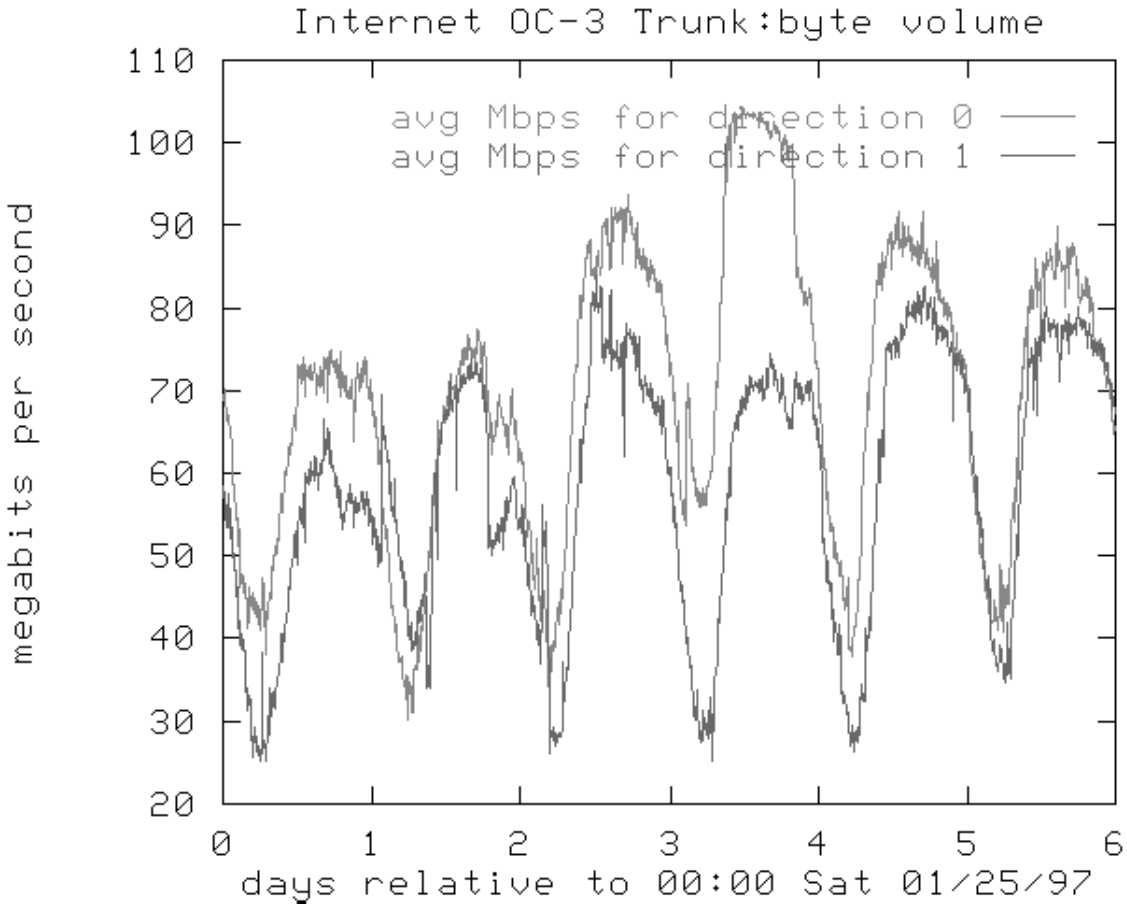


figure 11: average and maximum flows per second over 5 minute intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97

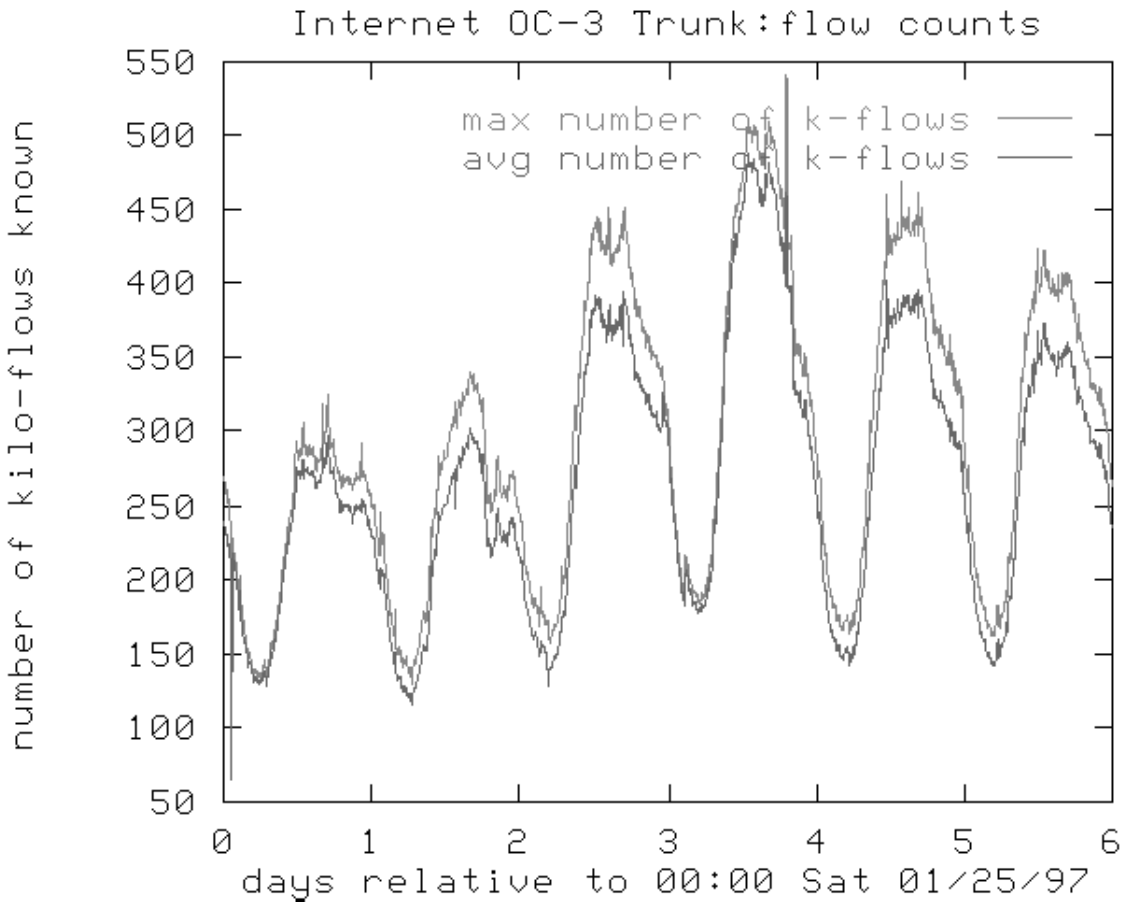
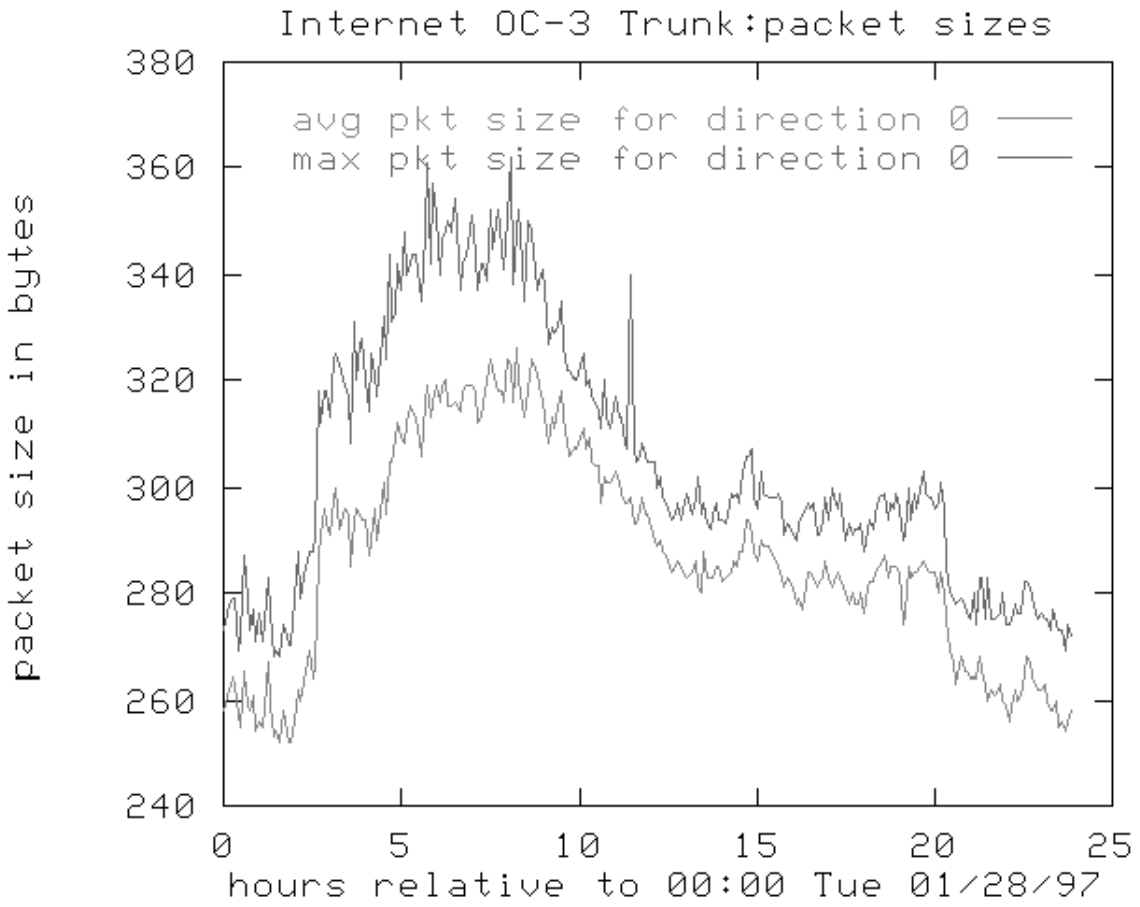


figure 12: average and maximum packet size over 5 minute intervals on MCI IP OC3 backbone trunk tue 28 jan



application specific: web, dns, mbone

OC3MON also supports analysis by TCP/UDP application type. Figure 13 illustrates the proportion of traffic from web servers using the well-known *http* port 80 (web servers can also use other ports, whose traffic will not be reflected in the graph) measured in packets, bytes, and flows. Note that web traffic consumes approximately the same proportion of flows as it does packets, but a somewhat larger proportion of bytes, indicating the use of larger packet sizes relative to other Internet traffic.

figure 13: proportion of web server-to-client traffic, i.e., from port 80 to any port, measured in packets, bytes, and flows over 5 minute sample intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97

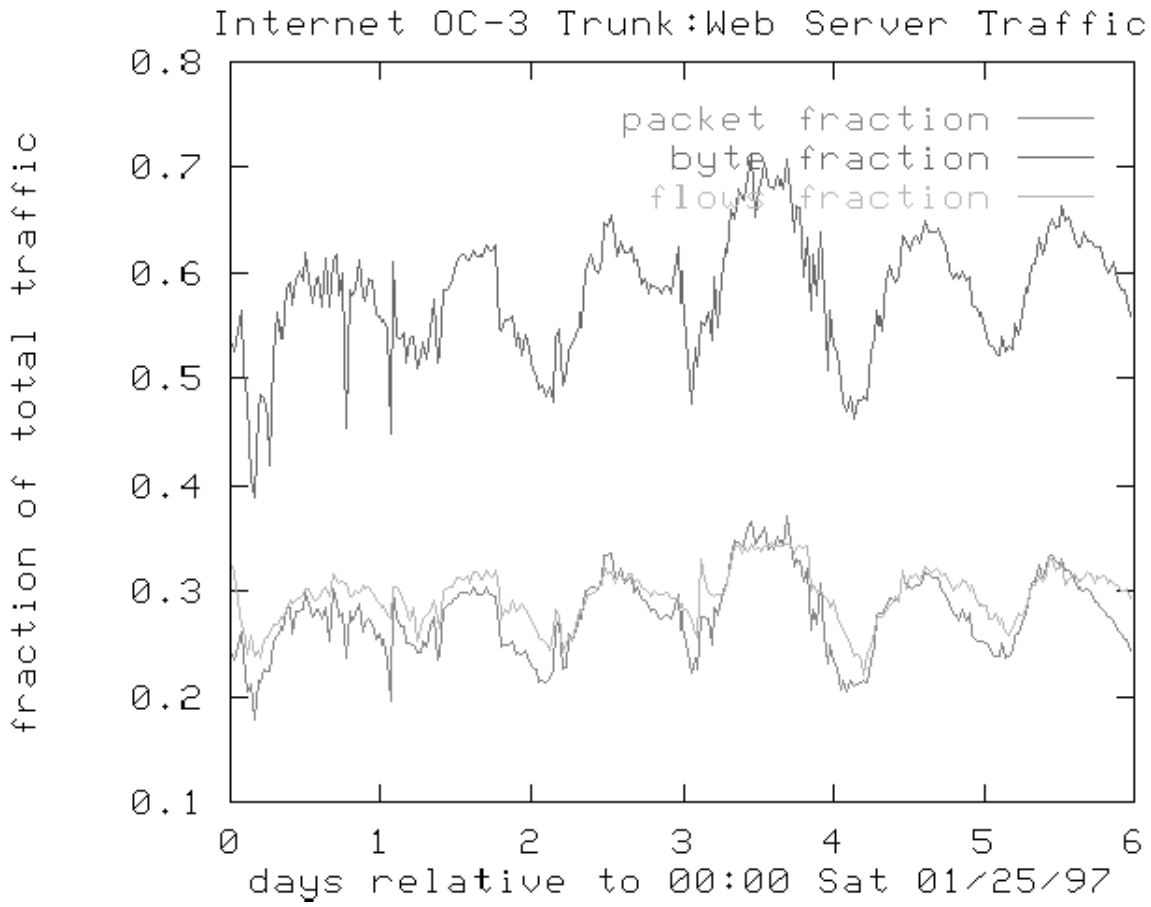
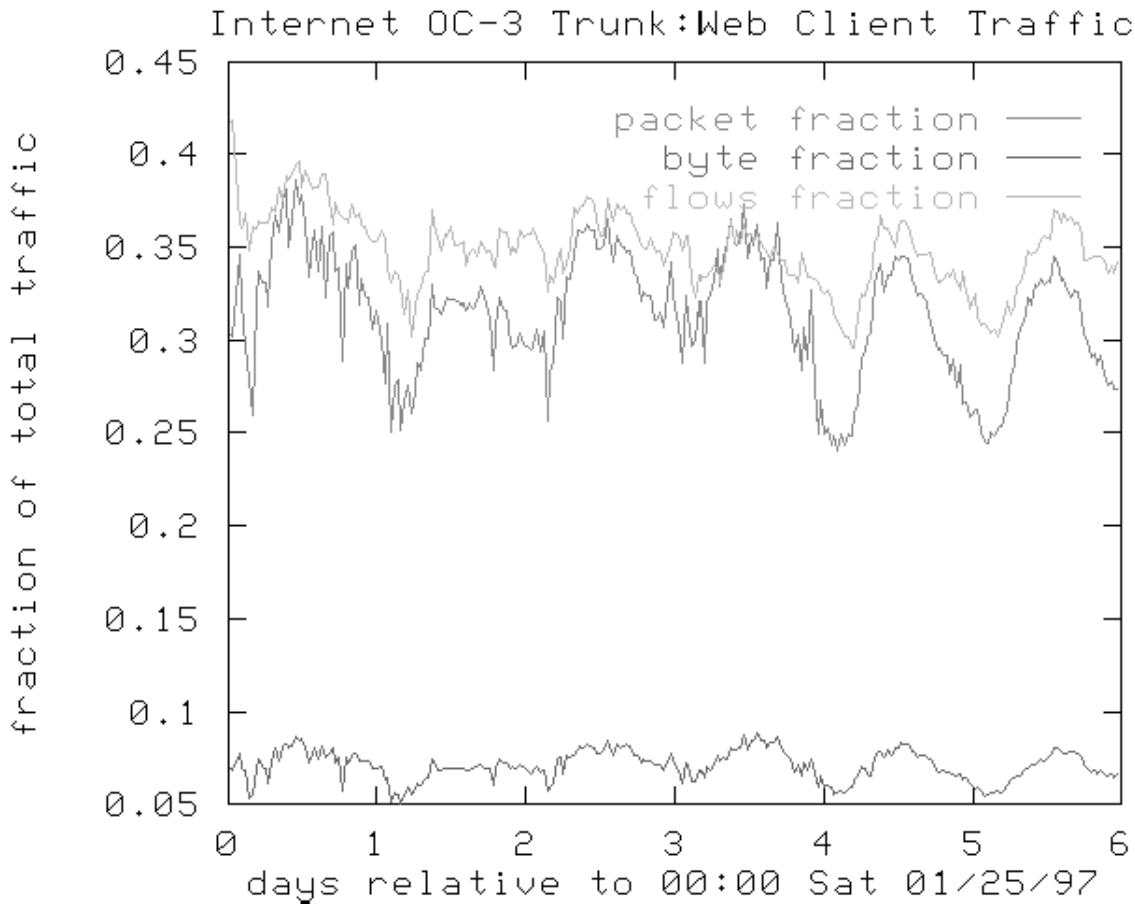


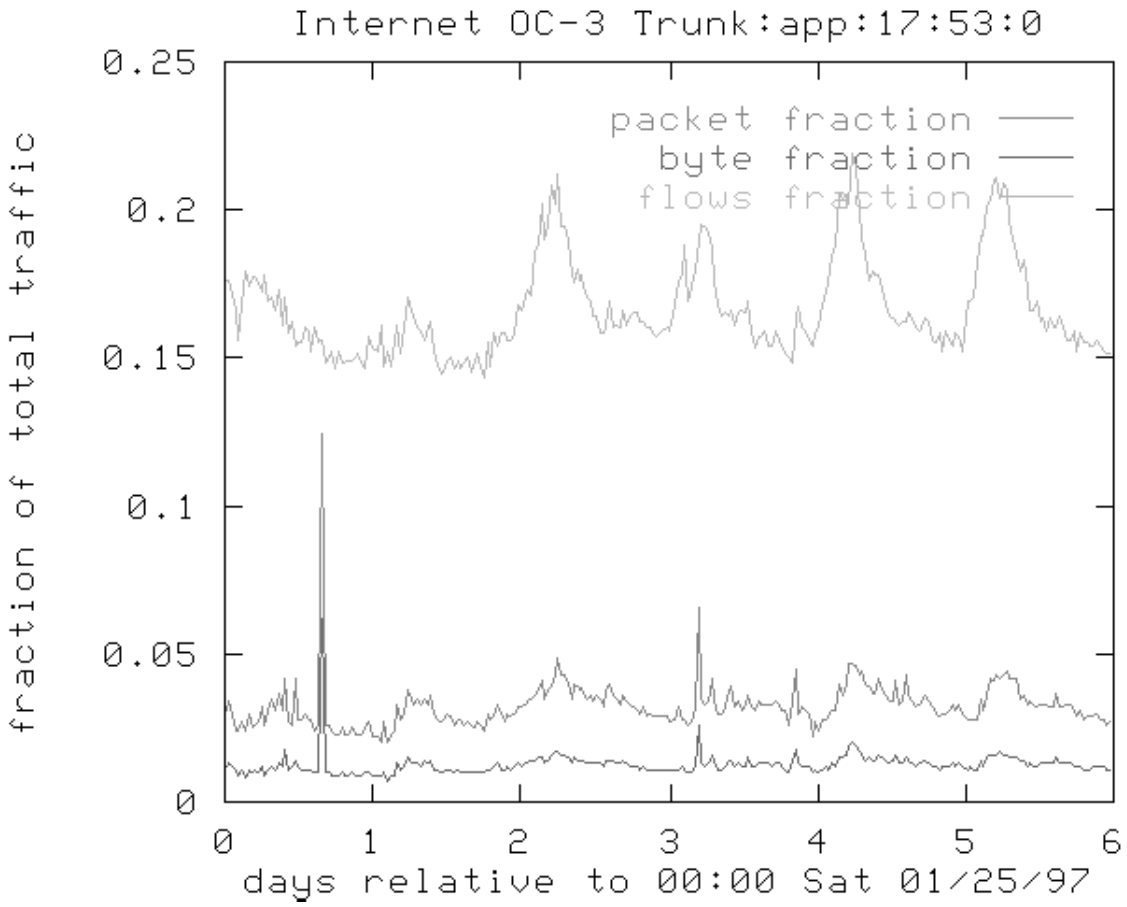
Figure 14 plots flows in the opposite direction, from clients to web servers; these flows have much lower byte proportions, being mostly query and acknowledgement traffic, slightly lower packet traffic, but similar flow counts.

figure 14: proportion of web client-to-server traffic, i.e., to port 80 from any port, measured in packets, bytes, and flows over 5 minute sample intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97



Domain name system (*dns*) traffic is also characterized by short query/response packets and thus, as shown in figure 15, comprises a huge proportion of (single packet, 40-80 byte) flows, but less than 3% of the byte traffic.

figure 15: proportion of *dns* traffic measured in packets, bytes, and flows over five minute sample intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97



We can also look at the traffic by transport layer protocol; figure 16 is the proportion of UDP packets, bytes, and flows (which includes all of the *dns* traffic plotted in figure 12).

figure 16: proportion of *udp* traffic measured in packets, bytes, and flows over 5 minute sample intervals on MCI IP OC3 backbone trunk sat 25 jan - thur 30 jan 97

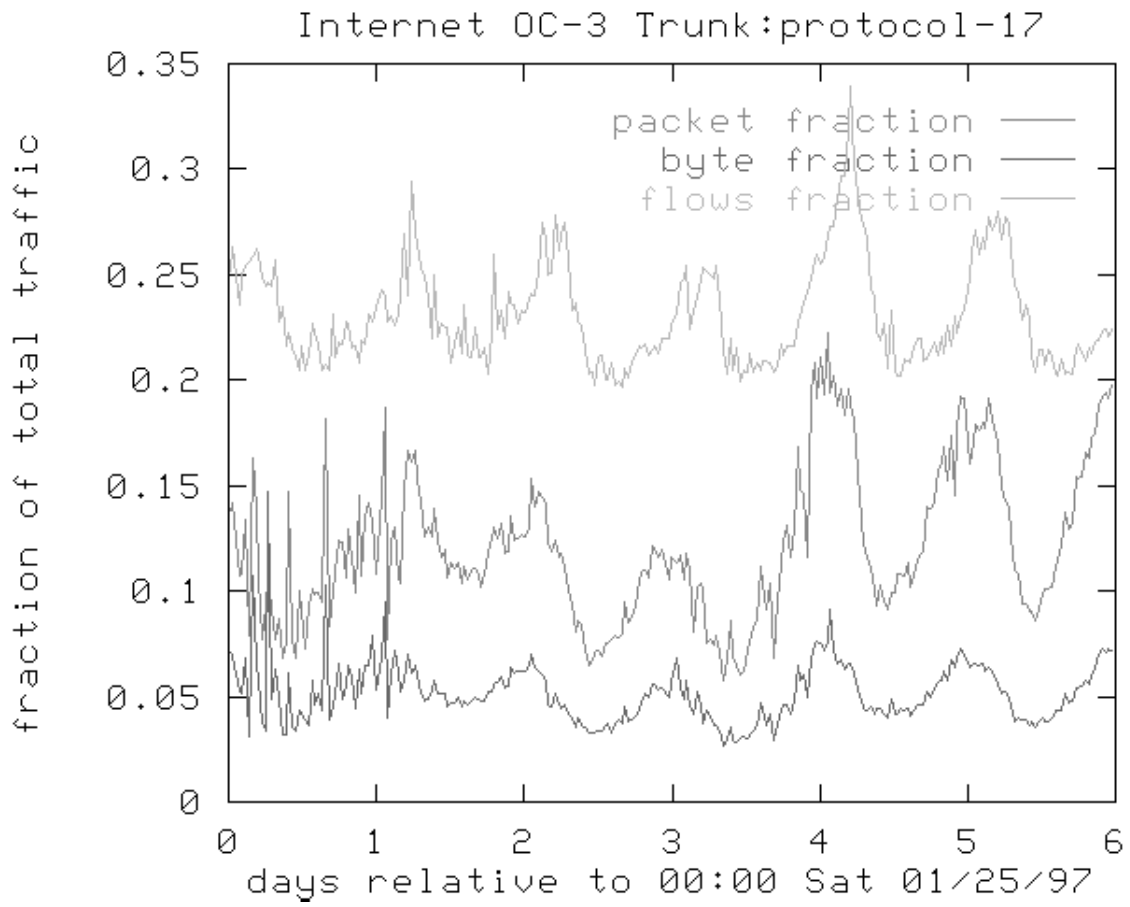
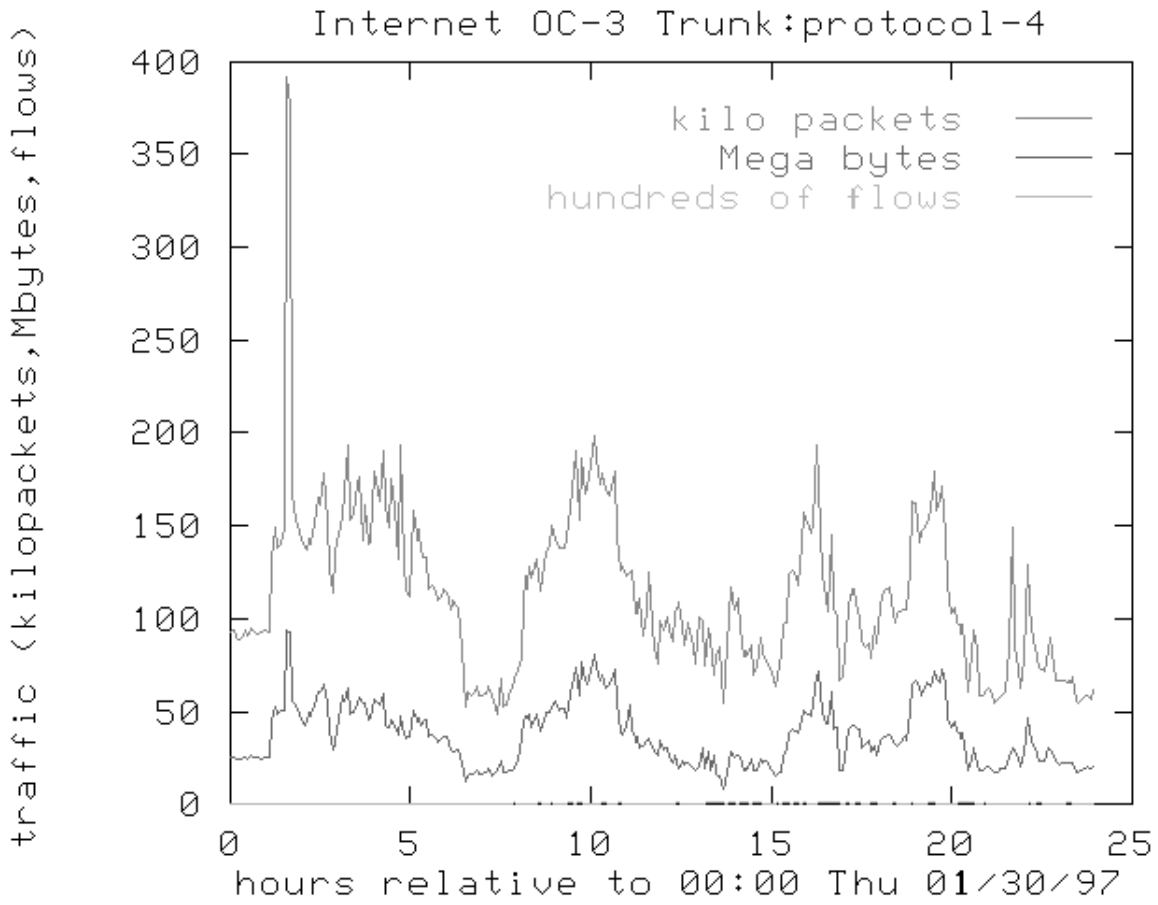


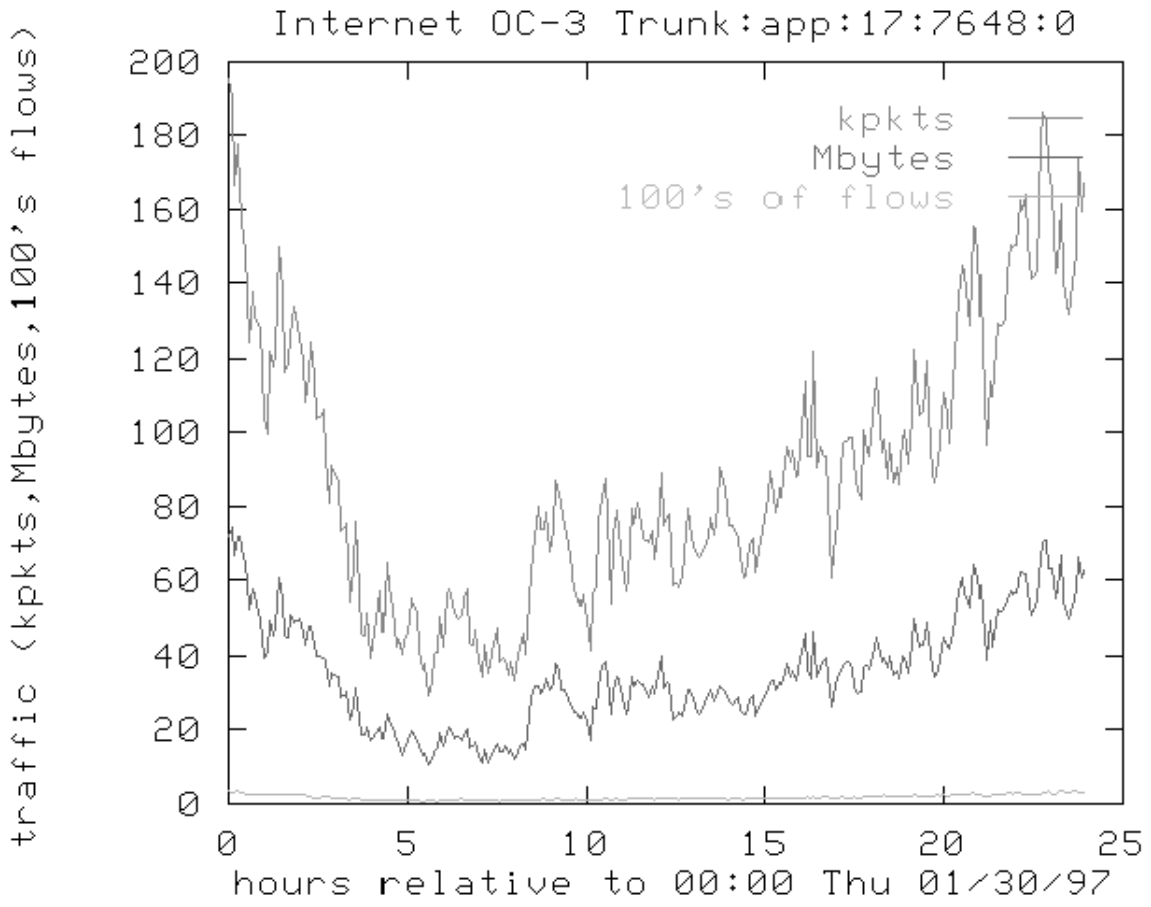
Figure 17 shows the absolute counts of IPIP traffic, again measured in packets, bytes, and flows. In this case a 24-hour period is shown. IPIP (IP protocol 4) traffic includes Mbone tunnel traffic, where very few flows each typically consume a substantial proportion of packets and bytes.

figure 17: counts of *IPIP* (IP protocol 4) traffic measured in packets, bytes, and flows on MCI IP OC3 backbone trunk thur 30 jan 97



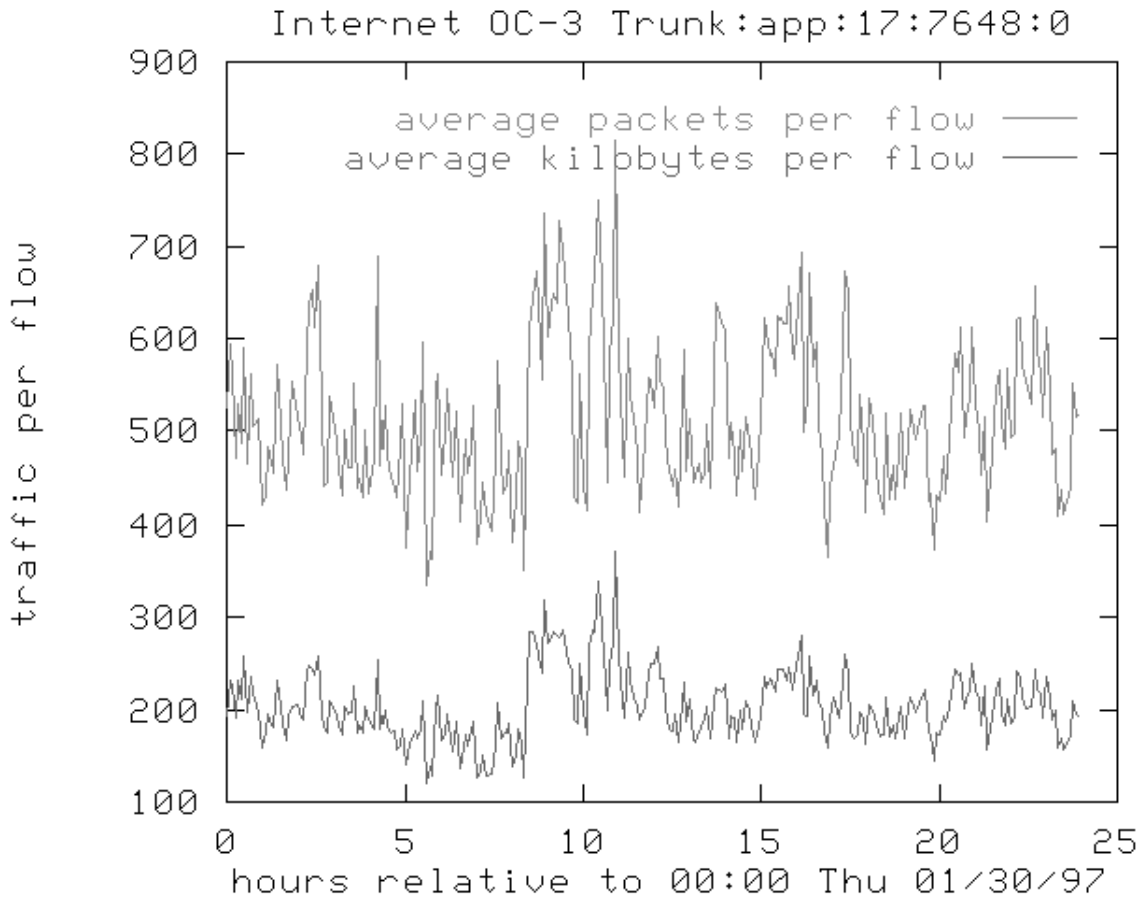
Although each mbone flow seems to consume an inordinate amount of resources, note that in the expected case, the mbone flows represent tunneled multicast traffic, and thus potentially serve a larger number of customers than just the single flow depicts. In contrast, the *cuseeme* audio/video teleconferencing application, plotted in figure 18 with a profile similar to the Mbone flow profile, is not multicast, and so poses a definite threat to Internet service providers trying to grow, or even maintain, a (largely flat-priced) customer base.

figure 18: counts of *cuseeme* traffic measured in packets, bytes, and flows on MCI IP OC3 backbone trunk thur 30 jan 97



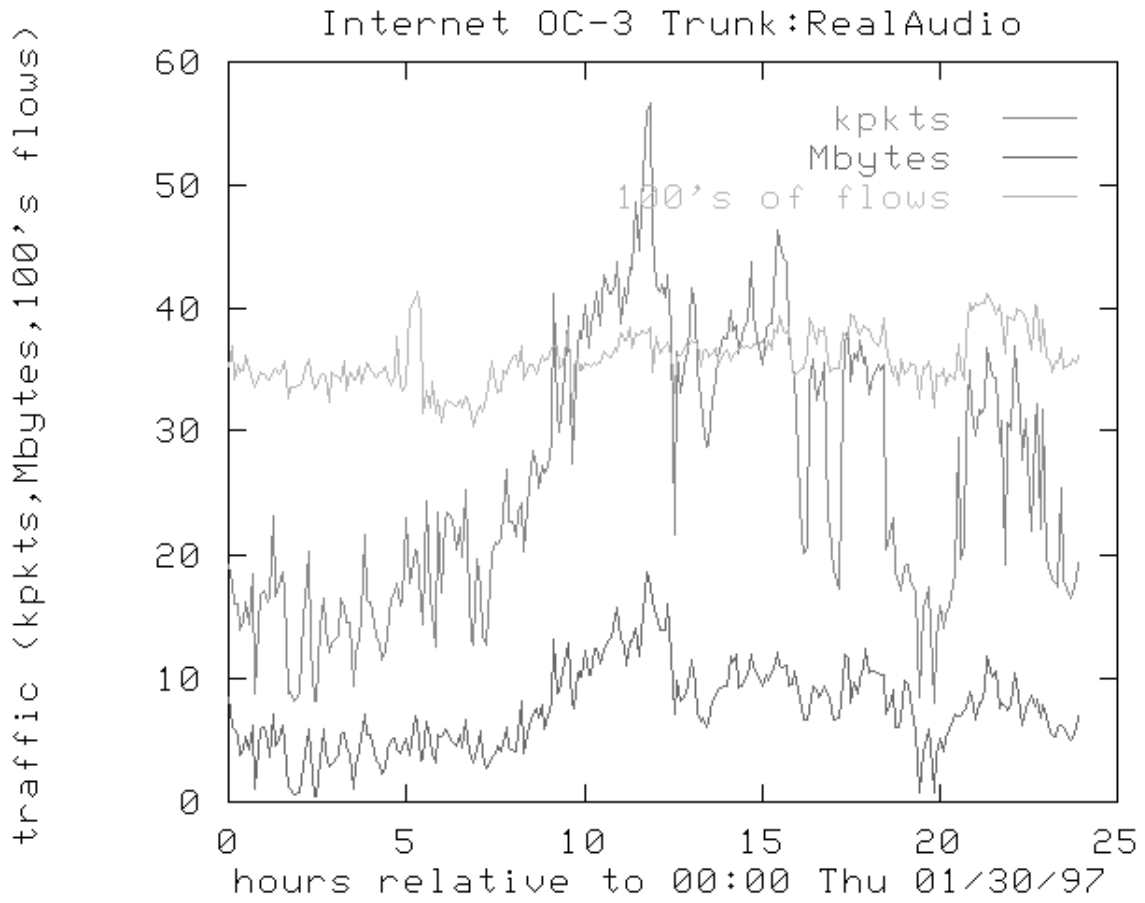
We might also want to know an average of how many packets and bytes are in a flow of a given type; figure 19 shows this metric for *cuseeme* traffic.

figure 19: average packet size of *cuseeme* traffic on MCI IP OC3 backbone trunk thur 30 jan 97



The use of ports as an application classifier limits us to applications that use a single port. *Realaudio* is an emerging application that uses more than one port: TCP port 7070, and UDP ports 6970 through 7170. Because we were particularly interested in the growth of this application, we modified the post-processing analysis script to support a query for this set of ports. (Note that this will be an upper bound, since other applications may also use these ports. In particular, *afs* uses port 7000.

figure 20: counts of *realaudio* traffic (which uses a set of ports: TCP port 7070, and UDP ports 6970 through 7170) measured in packets, bytes, and flows on MCI IP OC3 backbone trunk thur 30 jan 97



future work

OC3MON's design is conducive to several extensions. Enhancements to the analysis of packet trace and flows data and the web interface to the data are limited only by the imaginations of software developers. Our most immediate concerns are procuring OC12c interface cards for OC12MON, and as soon as possible being able to process both IP/ATM/SONET and IP/PPP/SONET encapsulations at OC12 rates with the same reasonably priced hardware.

We are also investigating moving more of the functionality of OCnMON, such as flows extraction from the packet header trace onto the interface card, in order to offload the host processor. This optimization would be increasingly useful at OC12c and OC48c speeds, where buses and host CPUs run out of steam. We believe that Field Programmable Gate Arrays (FPGA's) can provide this migration with a high degree of parallelism, without sacrificing the iterative design process and flexibility of software. We are also considering writing routines to access enhanced integrated drive electronics (EIDE) controller and the DMA engine on the Intel PCI ISA accelerator (PIIX3) directly to obtain much better asynchronous disk I/O.

Other extensions we examine with interest include:

- extending the current real time graphic display of traffic behavior on a per application, per subnet, or per trunk basis

- re-creation of traffic patterns from a previously monitored packet trace
- WAN simulation by adding delay, jitter, and errors to a traffic stream

We hope to pursue collaborations with those interested in extending OC3MON's utility.

conclusion

We have described the design, implementation, and use of a high performance yet affordable Internet monitoring tool. We have also described and shown examples from the web-based interface to the associated library of post-processing analysis utilities for characterizing network usage and workload trends. By using low cost, commodity hardware, we have ensured the practicality of using the monitor at a wide range of locations. Our network flow analysis tools have proven useful to us in understanding, verifying, debugging or spotting traffic behavior anomalies in the locations that we have deployed it.

availability

The web-based query engine for vBNS OC3MON data is at http://www.vbns.net/stats/flows/html/level0/. An electronic html version of this paper and pointers to the OC3MON software is at **http://www.nlanr.net/NA/Oc3mon/**. The software itself is available via ftp from **ftp://nlanr.net/Software/Oc3mon**. The original prototype for the web query engine, written by Hans-Werner Braun for the FIX-West FDDI, is still housed at **http://www.nlanr.net/NA/**.

appendix: hardware specifications

1. **adapter card** : Fore E-series PCA-200EPC PCI, about $US1000/card
2. **compilers** :
    1. Borland C++ 4.5
    2. Borland PowerPack for DOS
    3. Borland Turbo Assembler
3. **minimum PC configuration**:
    1. motherboard with Intel 82439HX PCI bridge chip (e.g., SuperMicro's P5STE)
    2. at least 2 full-length PCI slots
    3. 2 EIDE drives (C: can be small, D: should be big to hold traces)
    4. at least 128 MB memory
    5. at least 166 MHz Pentium CPU
    6. PCI VGA video card or better
    7. any multisync monitor (we only use 80x50 text mode)
    8. at least 1 free serial port (usually 2 on motherboard)
    9. 3COM 3C509B combo ISA ethernet card
    10. DOS (not win 95) 6.2 or 6.22

    it is being tested with:
4. Pentium Pro 200MHz instead of Pentium
5. Kingston KNE-40BT PCI ethernet 10 Mbps adaptors

it may work, but has not been tested, with:

1. multiple CPUs on motherboard
2. other ethernet cards, as long you install the DOS packet driver

- **optical splitters**: (available from ADC Telecommunications or AMP)

1. for single mode fiber: 90/10 split ratio 1260 to 1360 nm and 1480 to 1580 nm dual bandpasses, grade A (0.8 / 11.2 dB insertion loss)
   AMP part# 7-107804-8: 1x2 singlemode splitter with 3 mm cables, SC connectors with SPC polish, -50 dB backreflection (note that AMP splitters are outside an enclosure, with male connectors; need t 2 to monitor both directions on duplex link)
   ADC standard part# VSM-K7U7U281B0000, dual splitters in a single metal box so need one box per full duplex link, your choice of connectors
2. for multi mode fiber: 50/50 split ratio, 62.5/125 um core/cladding diameters, 4.0 dB max insertion loss, 0.5 dB max uniformity
   AMP part# 2-107842-3
   ADC standard part# VSM-K9B9B202B0000
   (about $800 per dual splitter)

ADC also makes racks to mount 8-12 splitter modules (if you tap multiple links at same site)

acknowledgments

We are grateful to Hans-Werner Braun for prototyping the original flow statistics software and making it freely available.

This material is based on work sponsored by the National Science Foundation, grants NCR-9415666 and NCR-9321047. The very high speed Backbone Network Service (vBNS) project is managed and coordinated by MCI Communications Corporation under sponsorship of the National Science Foundation. The Government has certain rights to this material. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] k claffy and H W Braun, *Post-NSFNET statistics collection*, Proceedings Inet 95

[2] k claffy and H W Braun *Web traffic characterization: an assessment of the impact of caching documents from the NCSA's web server* World Wide Web conference, 1994, Chicago, IL

[3] k claffy, Hans-Werner Braun, George C. Polyzos *A parameterizable methodology for Internet traffic flow profiling* IEEE JSAC

[4] 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.

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

[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 Modeling 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.

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

[10] D. Estrin and D. Mitzel, *An Assessment of state and lookup overhead in routers*, Proc. Infocom ’92,