The following paper was originally published
in the Proceedings of the
Tenth USENIX System Administration Conference (LISA X)
Chicago, IL, USA, Sept. 29 - Oct. 4, 1996

OC3MON: Flexible, Affordable, High Performance Statistics Collection

Joel Apisdorf, k claffy (NLANR), Kevin Thompson, & Rick Wilder
MCI/vBN

OC3MON: Flexible, Affordable, High Performance Statistics Collection

Joel Apisdorf, k claffy (NLANR), Kevin Thompson, & Rick Wilder – MCI/vBNS

ABSTRACT

The Internet is rapidly growing in number of users, traffic levels, 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.

Part I: Design and Implementation

Introduction

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% of the light from each fiber to the receive port of one NIC. (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).

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

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.

constraints motivated our decision to use DOS-based functionality rather than a UNIX kernel. First, the TI cards in the original OC3MON design required polling 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 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 (e.g., 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 wanted more control over when TCP/IP could have a time slice than Unix provides.

The disadvantage is that DOS only has blocking I/O routines, whereas Unix would provide non-blocking I/O. But in experimenting with our own workstations, we did not see evidence that Unix single disk I/O was significantly faster than that which we were going to implement. We do hope to eventually port OC3MON to a Unix platform, most likely Linux on a PC.

Software: Background on ATM

The 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). 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 four bytes from each cell (above the five already used for the ATM header) to support submultiplexed channels within a given VP/VC. Since the LLC/SNAP 8-byte per-frame header that the routers insert already includes a 2-byte ethertype field that allows, if needed, multiplexing of

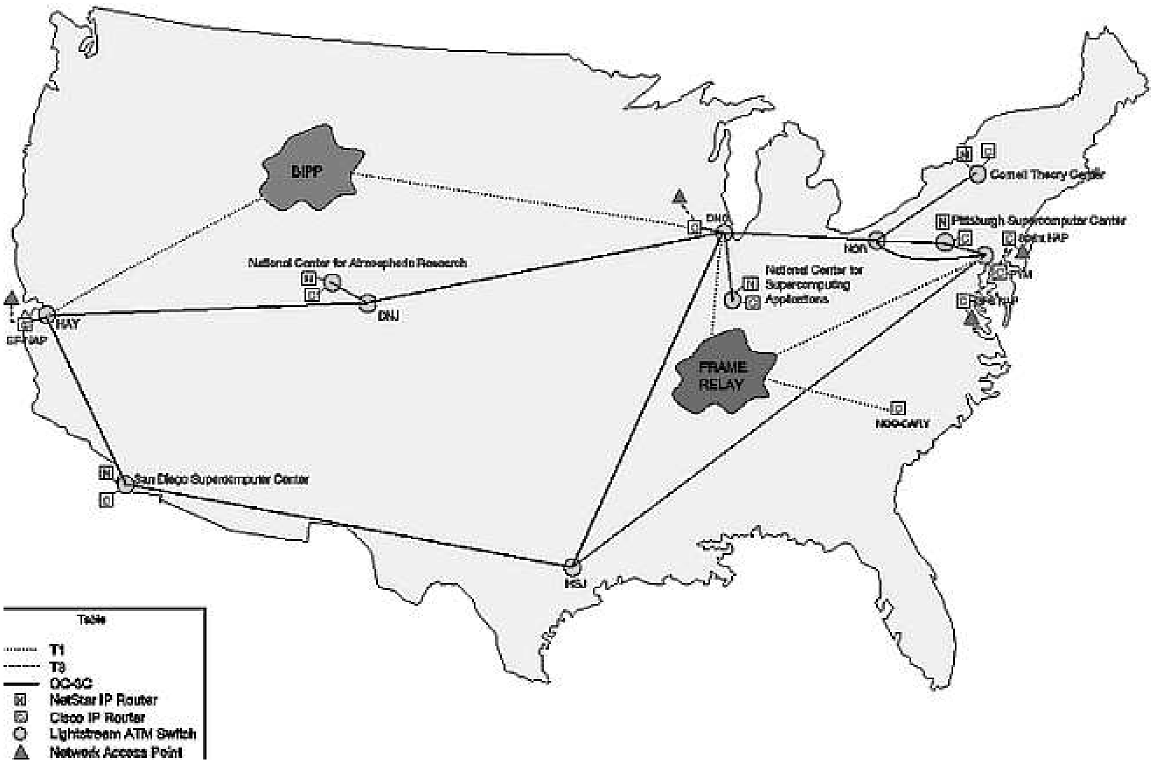

Figure 1: National Science Foundation very high speed Backbone Network Service (vBNS) topology. The backbone connects the five supercomputer centers at Cornell, Pittsburgh, Urbana-Champaign, Boulder, and San Diego at OC3, with T3 connectivity to the four original NAPs in DC, Pennsauken, Chicago, and San Francisco. MCI plans to upgrade the backbone to OC12 in late 1996.

different protocols (IP, IBM SNA, Novell IPX, etc) on the same VC, including AAL3/4 support in the design would not have been beneficial. (In fact the other six bytes of the LLC/SNAP header, for which we have no use, take up so much space in the first cell that even for simple TCP ACKs they squeeze out the 8-byte AAL5 trailer, which then requires its own cell.)

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. 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 it 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). This limit allows us to use 20 bits (19 bits for OC12MON) for VPI and VCI information, yielding a 128KB table size. 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 2MB of memory, all of which will be available 2^{24} = 16777215 UNI (user to network interface) possible VP/VC combinations if we use one bit (or half that if we use two bits) of state per VP/VC. 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 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. In fact, the I/O is not even fast enough to sustain disk transfer of a packet trace without the flow analysis process running. Therefore one can only collect a trace as big as the size of host memory, which in our case would be 114MB (119.5 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, bypassing the slower DOS routines, and allowing us to keep up with continuous collection and storage of full packet headers at OC3 line rate.

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 (see http://www.nlanr.net/NA/) workload query interface. We will describe the methodology for deriving flow information in second half of the paper, followed by example snapshot statistics taken with OC3MON.

Internal OC3MON Data Transfer Rate

We tested OC3MON on an 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 $353207.5 \times 2 \times 60 \times 8 = 339$MB. 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 case OC12 workload (i.e., single-celled traffic), the bus technology will likely be available. (Digital has already demonstrated the 64-bit part.)

Typical production Internet environments exhibit average packet sizes closer to 250 bytes (about five cells), and rarely full utilization in both directions of a link. If we estimate 66% utilization in one direction and full utilization in the other, we get a more realistic: $353207.5 \times 1.6666 / 5 = 117731$ headers per second, or 56.5 MB per second across the internal bus.

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 those capable of monitoring traffic. 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 known IP addresses, or configure the 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 many thousands may be active simultaneously at WAN transit points.

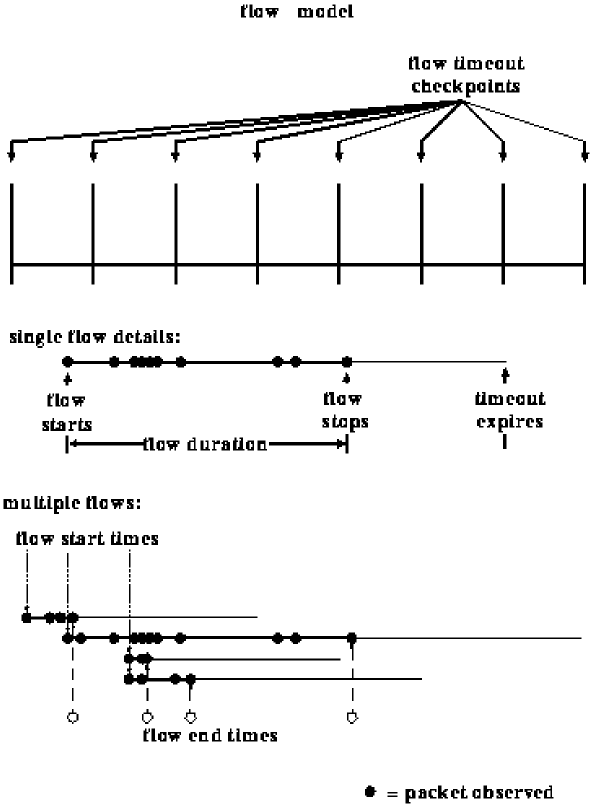

Figure 2: Defining a flow based on timeout during idle periods

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 five minute timeout yielded minimal differences. Estrin and Mitzel [10] also compared timeouts of five and 15 minutes and found little difference in conversation duration at the two values, but chose to use a timeout of five 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 are often 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 application instance, 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

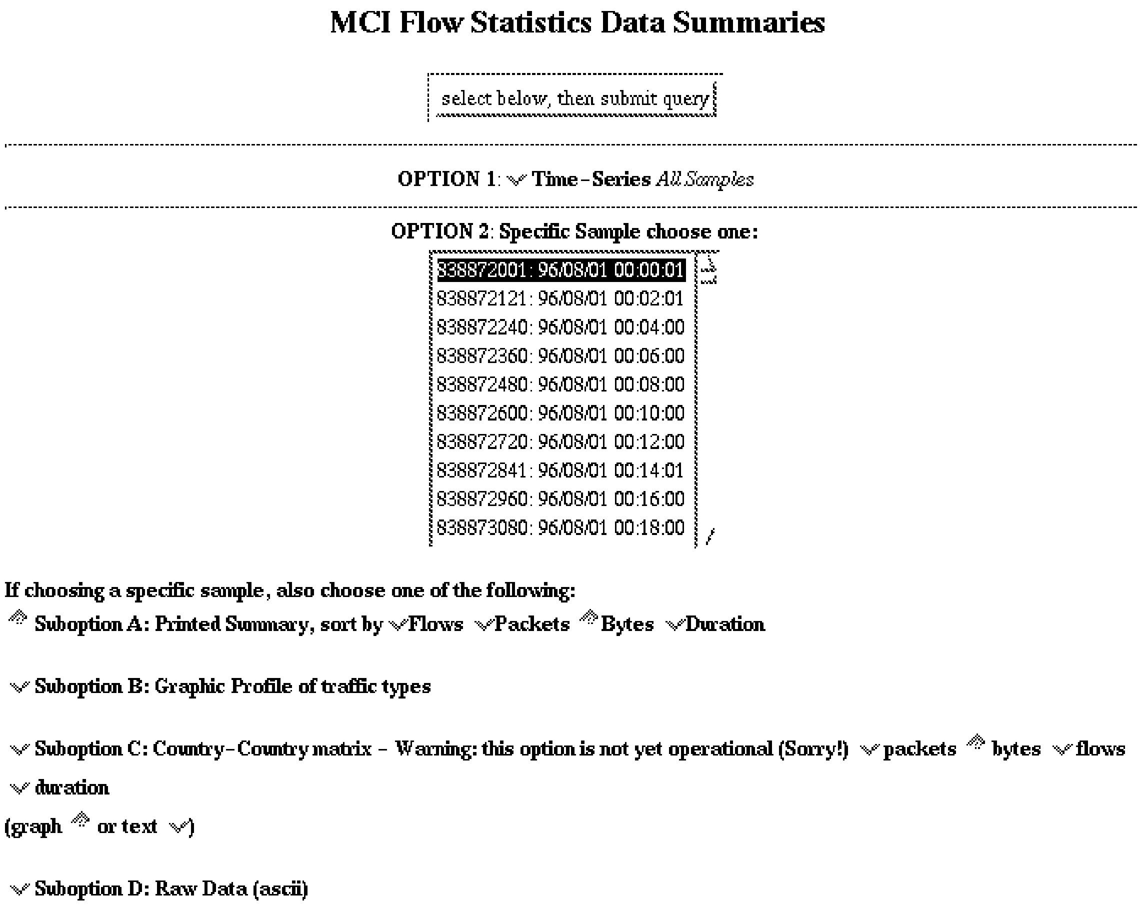

Figure 3: Main menu for OC3MON query engine

layer – provide a framework for specifying a flow profile structure. We designed the OC3MON flows analysis software to be flexible. One can specify a specific flow timeout (in seconds), an endpoint granularity (network, host, or host/port), and one- or two-sided flows (source, destination, or pair). We currently only support the classful IP network granularity, i.e., the most significant two bits of the address choose a netmask of 24, 16, or 8 bits. This is no longer appropriate in today's infrastructure, which instead uses CIDR (classless inter domain routing) variable length netmasks. Besides the network number, each routing update includes length of the netmask, which is unrelated to the contents of any bits of the IP address. OC3MON's next release will support the granularity of classless network by a CIDR-aware IP-to-AS-path mapping derived from a periodically updated dump of an actual routing table. This support will also include flow conversion to the autonomous system (AS) granularity, which will enable assessment of traffic flow at convenient macroscopic level.

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 3, 4, and 5, allow users to customize graphs of the data according to their interest.

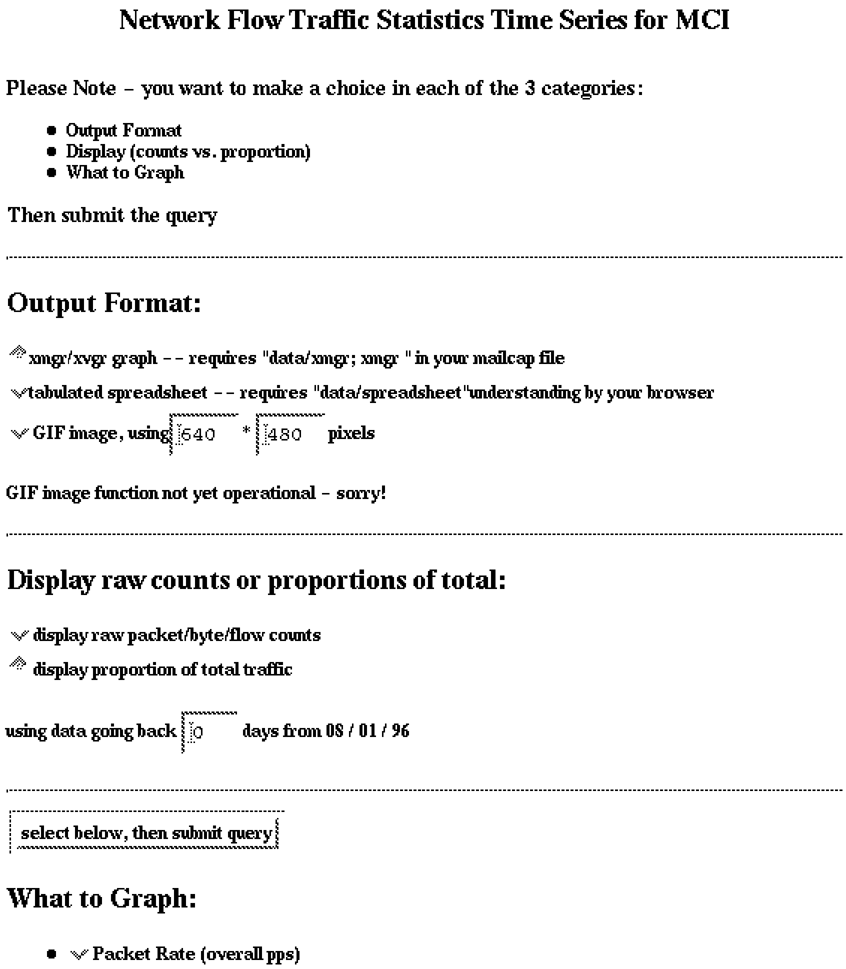

Figure 4a: Time-series menu for OC3MON query engine (invoked upon clicking option 1 *time-series* in main menu)

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 during the period between 29 July and 1 August 1996. We also provide international data from similar (earlier prototype) software running at the FIX West interagency interconnection point at NASA-Ames, in Moffett Field, California. 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 3, 4, and 5 show the main menu and two submenu form interfaces for the query engine, respectively.

The main menu lets you choose either a snapshot of a specific monitoring interval (configured for two minutes here), or select one of the submenus. The snapshot returns a table similar to that shown in Table 1.

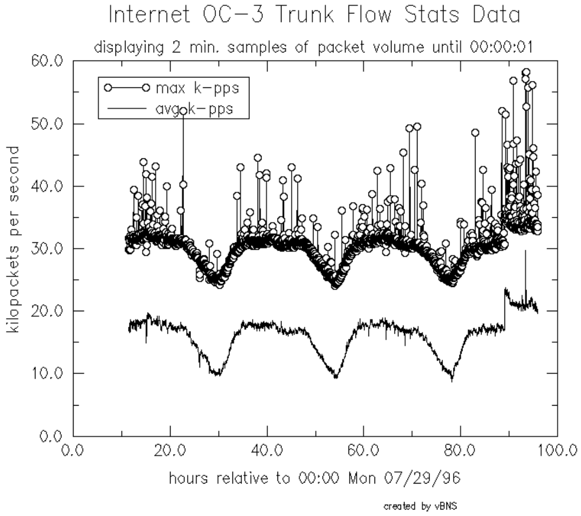

Figure 6: Average and maximum packets per second over two minute intervals on MCI IP OC3 backbone trunk Mon 29 Jul-Thur 1 Aug 96

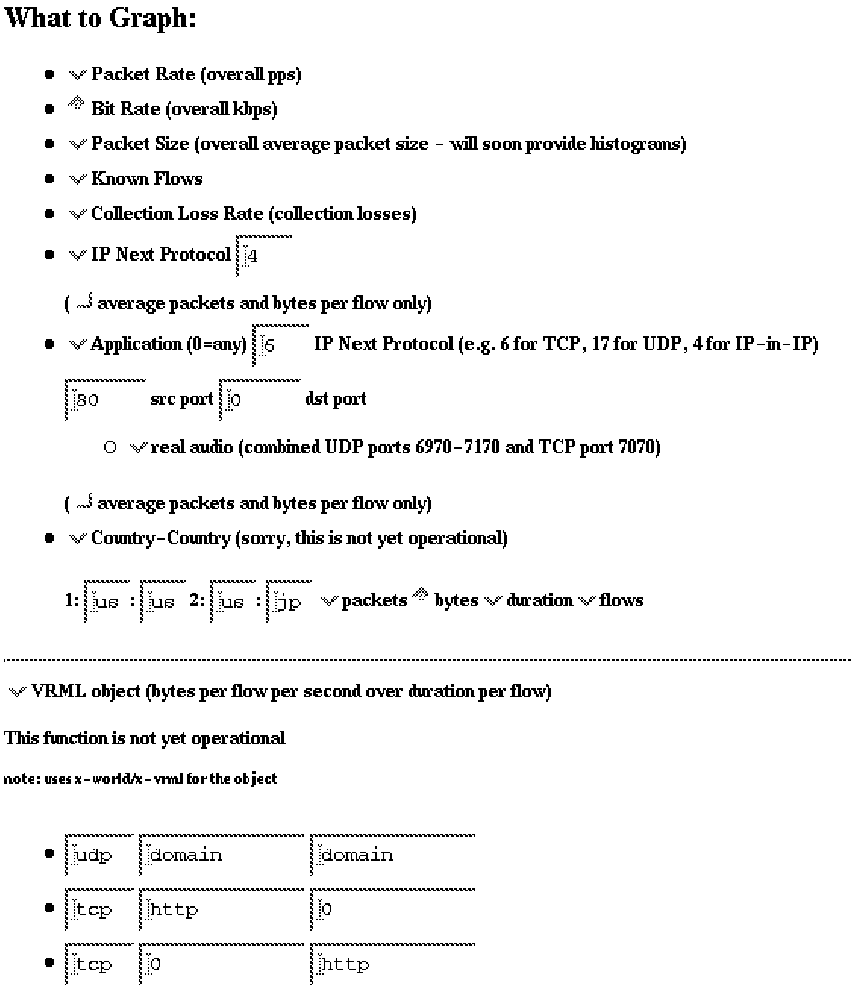

Figure 4b: Rest of Figure 4a time-series menu

Figures 6, 7, 8, and 9 show packets, bytes, flows, and average per-second packet size for the four day period. During this measurement interval, the average number of packets per second cycled between ten and twenty thousand, with per second peaks as high as 58 thousand packets. The link utilization is around 50%, which is a moderately loaded link in the current backbone. (MCI has also installed parallel OC3s where

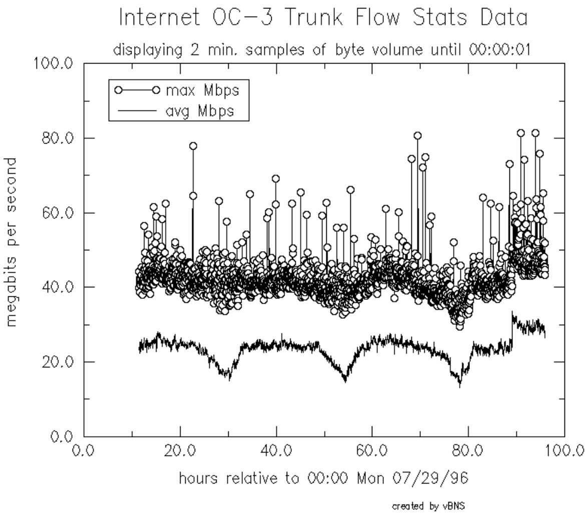

Figure 7: Average and maximum bits per second over two minute intervals on MCI IP OC3 backbone trunk Mon 29 Jul-Thur 1 Aug 96

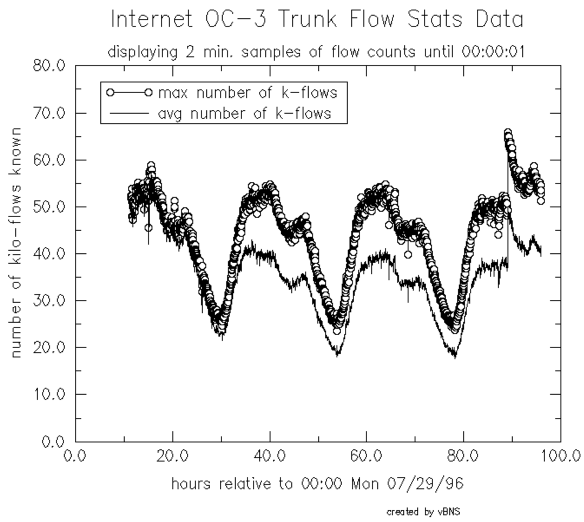

Figure 8: Average and maximum number of flows per second over two minute intervals on MCI IP OC3 backbone trunk Mon 29 Jul-Thur 1 Aug 96

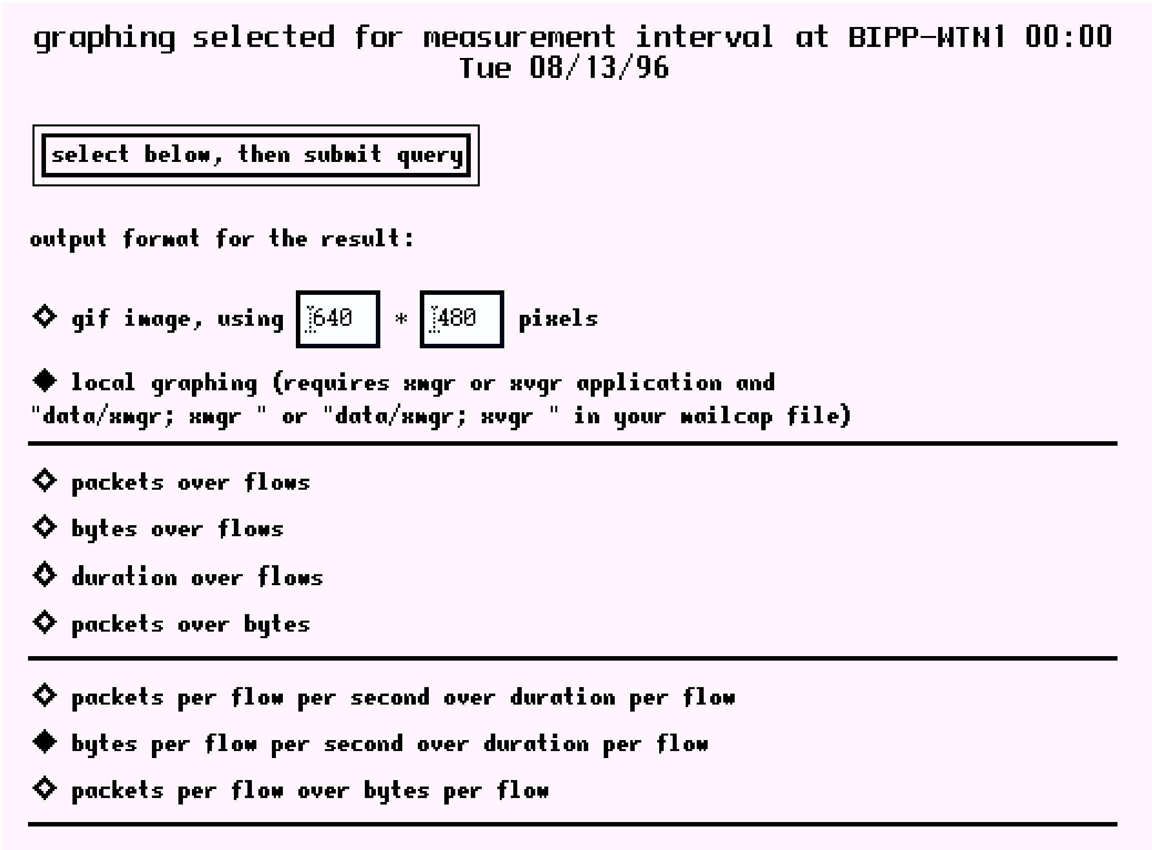

Figure 5: 2D profiles menu for OC3MON query engine (invoked upon clicking suboption B *Graphic profile* option in main menu)

```
         Flow Statistics Analysis for Internet OC3 Trunk
           for date and time sample:08/14/96 at 01:20:01
                        Data Collection Summaries:
packet load  (max/avg per second): 29461/15505
bit volume   (max/avg per second): 41331776/8663743
packet sizes (max/avg per second): 244/194
flow summary:
   maximum number of active flows: 44925
   average number of active flows: 38718
   average number of newly created flows per second: 535
error count/second (max/avg for three counters): 0/0 0/0 0/0
error percentage per second (max/avg):  0.00000/0.00000
                                     Totals
trace duration:   299.502 seconds
total flow count: 160353
total packets:    3116705
total bytes:      861221626
                       Itemization by IP protocols
protocol       Flows packets    bytes     duration
             f-fract p-fract   b-fract     d-fract
============ ======= ======== ========== ========= ================
  1:    ICMP   4755    36024    2753051     168691 absolute
              0.030    0.012      0.003      0.000 fraction
                 76        7        578         35 ave pkt size,pkts,byts,dur
  2:    IGMP      3     1219     165730        886 absolute
              0.000    0.000      0.000      0.000 fraction
                135      406      55243        295 ave pkt size,pkts,byts,dur
  4:      IP     14     5192     913461        926 absolute
              0.000    0.002      0.001      0.000 fraction
                175      370      65247         66 ave pkt size,pkts,byts,dur
  6:     TCP 117449  2778274  808689615 -862387242 absolute
              0.732    0.891      0.939      0.500 fraction
                291       23       6885      -7342 ave pkt size,pkts,byts,dur
 17:     UDP  38085   295191   48604047 -864052262 absolute
              0.238    0.095      0.056      0.500 fraction
                164        7       1276     -22687 ave pkt size,pkts,byts,dur
 47:     GRE     10      500      51942       2727 absolute
              0.000    0.000      0.000      0.000 fraction
                103       50       5194        272 ave pkt size,pkts,byts,dur
 83:   VINES      5       70       4692        554 absolute
              0.000    0.000      0.000      0.000 fraction
                 67       14        938        110 ave pkt size,pkts,byts,dur
 93:   AX.25     24      175      24386        776 absolute
              0.000    0.000      0.000      0.000 fraction
                139        7       1016         32 ave pkt size,pkts,byts,dur
148: unknown      7       59      14632        613 absolute
              0.000    0.000      0.000      0.000 fraction
                248        8       2090         87 ave pkt size,pkts,byts,dur
241: unknown      1        1         70          0 absolute
              0.000    0.000      0.000      0.000 fraction
                 70        1         70          0 ave pkt size,pkts,byts,dur
```

Table 1a: Flow assessment snapshot of traffic during single interval of OC3MON collection (invoked upon clicking suboption A, *printed summary* in the main menu)

Application Details

Sorted by bytes

prot	source port	dst port	Flows f-fract	packets p-fract	bytes b-fract	duration d-fract	
TCP	80	0	45096	782256	362260508	717418	absolute
			0.281	0.251	0.421	0.000	fraction
				17	8033	15	per flow average
TCP	0	119	265	284514	178873180	33343	absolute
			0.002	0.091	0.208	0.000	fraction
				1073	674993	125	per flow average
TCP	0	80	50095	713105	40006387	613333	absolute
			0.312	0.229	0.046	0.000	fraction
				14	798	12	per flow average
TCP	20	0	516	54058	38851597	46553	absolute
			0.003	0.017	0.045	0.000	fraction
				104	75293	90	per flow average
TCP	0	25	3618	68213	22884422	-864616448	absolute
			0.023	0.022	0.027	0.500	fraction
				18	6325	203944101	per flow average
TCP	119	0	593	215040	19874375	39717	absolute
			0.004	0.069	0.023	0.000	fraction
				362	33514	66	per flow average
UDP	7648	7648	90	34543	14214500	11001	absolute
			0.001	0.011	0.017	0.000	fraction
				383	157938	122	per flow average
UDP	53	domain	31350	102557	13217739	-864419840	absolute
			0.196	0.033	0.015	0.500	fraction
				3	421	23536521	per flow average
TCP	5190	0	129	11878	6269981	20769	absolute
			0.001	0.004	0.007	0.000	fraction
				92	48604	161	per flow average
TCP	23	0	304	43325	5714030	28780	absolute
			0.002	0.014	0.007	0.000	fraction
				142	18796	94	per flow average
TCP	1091	0	1	3050	4531532	125	absolute
			0.000	0.001	0.005	0.000	fraction
				3050	4531532	125	per flow average
TCP	6667	0	685	16751	3384800	64961	absolute
			0.004	0.005	0.004	0.000	fraction
				24	4941	94	per flow average
TCP	25	0	2820	54382	3005756	54107	absolute
			0.018	0.017	0.003	0.000	fraction
				19	1065	19	per flow average
TCP	0	20	1052	65855	2850237	35584	absolute
			0.007	0.021	0.003	0.000	fraction
				62	2709	33	per flow average

Table 1b: Application details

traffic demands require them, and is currently installing the first OC12s.)

The average number of flows per second goes from around 20,000 at night to over 60,000 during the day. 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.

Application Specific: Web, DNS, Mbone

OC3MON also supports analysis by TCP/UDP application type. Figure 10 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.

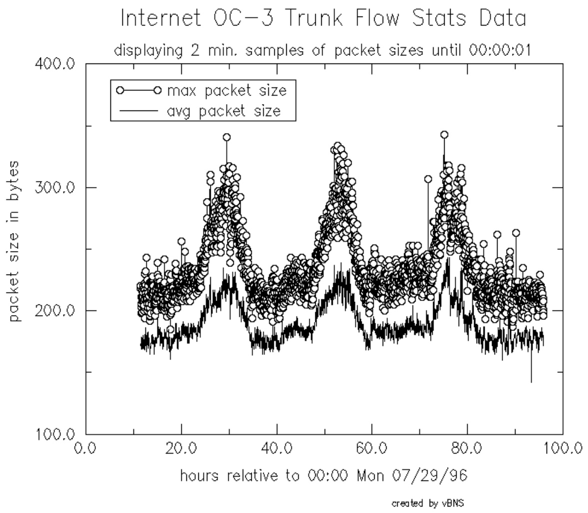

Figure 9: Average and maximum packet size over two minute intervals on MCI IP OC3 backbone trunk Mon 29 Jul-Thur 1 Aug 96

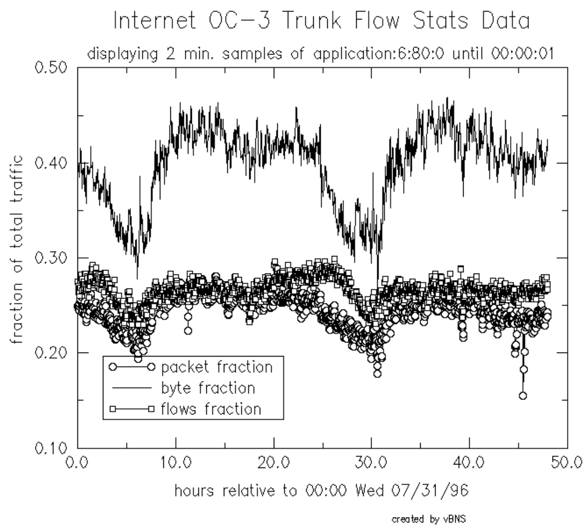

Figure 10: Proportion of web server-to-client traffic, i.e., from port 80 to any port, measured in packets, bytes, and flows over two minute sample intervals on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

Figure 11 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.

Domain name system (*dns*) traffic is also characterized by short query/response packets and thus, as shown in Figure 12, comprises a huge proportion of (single packet, 40-80 byte) flows, but less than 8% of the byte traffic.

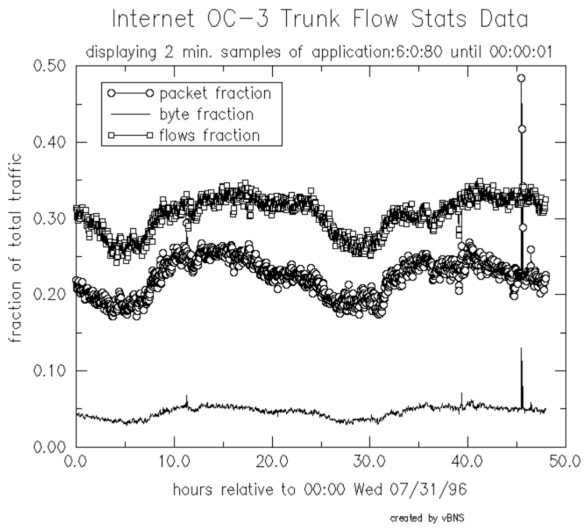

Figure 11: Proportion of web client-to-server traffic, i.e., to port 80 from any port, measured in packets, bytes, and flows over two minute sample intervals on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

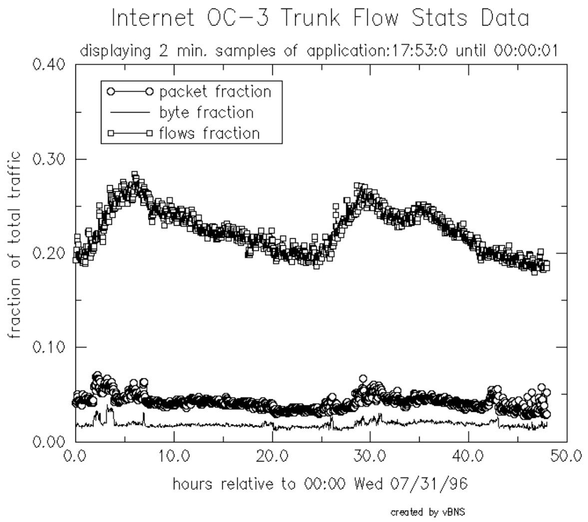

Figure 12: Proportion of *dns* traffic measured in packets, bytes, and flows over two minute sample intervals on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

We can also look at the traffic by transport layer protocol; Figure 13 is the proportion of UDP packets, bytes, and flows (which includes all of the *dns* traffic plotted in Figure 12).

Figure 14 shows the absolute counts of IPIP traffic, again measured in packets, bytes, and flows. IPIP (IP protocol 4) traffic includes Mbone tunnel traffic, where very few flows each typically consume a substantial proportion of packets and bytes.

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 15 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 (still largely flat-priced) customer base.

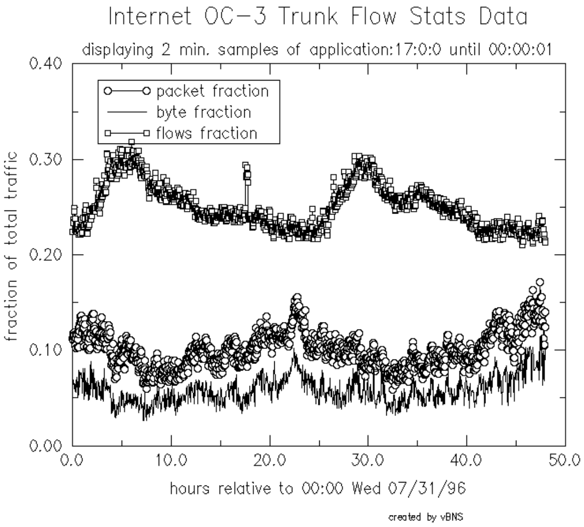

Figure 13: Proportion of *udp* traffic measured in packets, bytes, and flows over two-minute sample intervals on MCI IP OC3 backbone trunk Wed 31 Jul – Thur 1 Aug 96

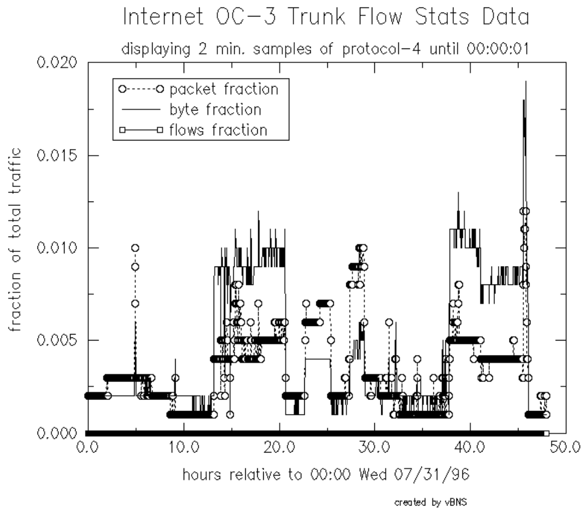

Figure 14: Proportion of *IPIP* (IP protocol 4) traffic measured in packets, bytes, and flows on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

We might also want to know an average of how many packets and bytes are in a flow of a given type; Figure 16 shows this metric for *cuseeme* traffic.

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 this will be an upper bound, since other applications may use those ports as well, e.g., AFS uses port 7000.)

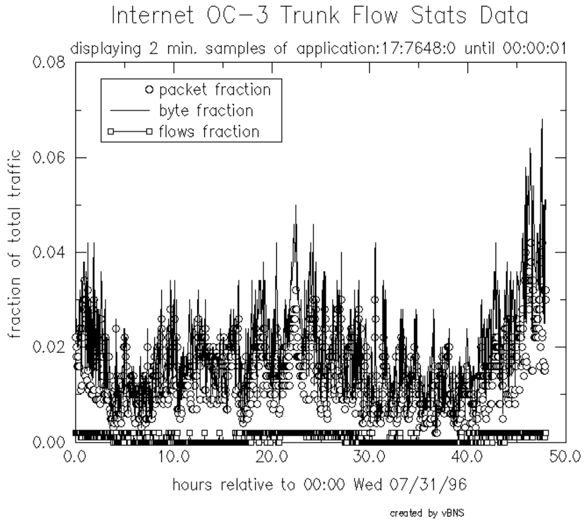

Figure 15: Proportion of *cuseeme* traffic measured in packets, bytes, and flows on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

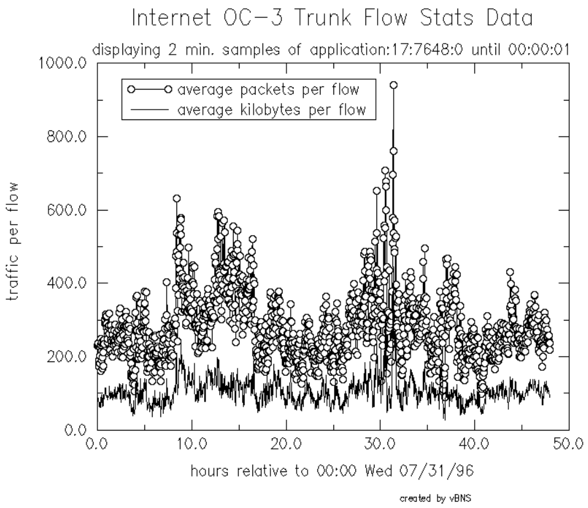

Figure 16: Average packet size of *cuseeme* traffic on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

Profiles of Two Dimensions

The 2D profile menu shown earlier (Figure 5) allows one to classify application flows by two parameters at a time, e.g., their byte-duration or packet-duration products. This will allow figures such as that in Figure 18, which is an example from the FIX West location that has already implemented this feature.

Flow Across Geographic or Administrative Boundaries

OC3MON also is amenable to post-processing to derive geographic or administrative flow information. For example, we can look at the trade balance among countries. For the graphs in this section we use data collected from the FIX West interagency interconnection point at NASA-Ames, in Moffett Field, California. FIX West is a FDDI LAN medium that serves as a

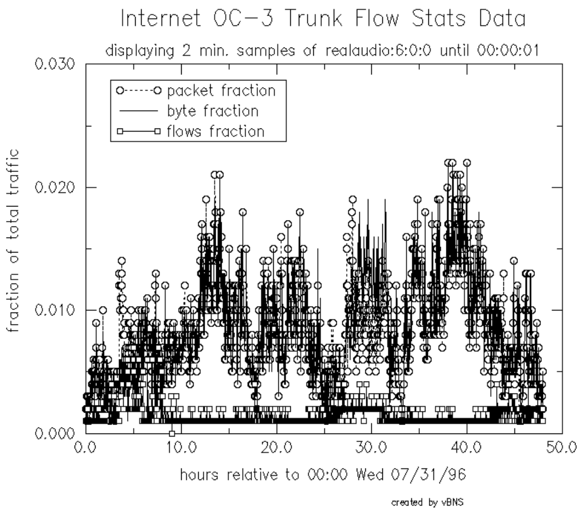

Figure 17: Proportion of *realaudio* traffic (which uses a set of ports: TCP port 7070, and UDP ports 6970 through 7170) measured in packets, bytes, and flows on MCI IP OC3 backbone trunk Wed 31 Jul-Thur 1 Aug 96

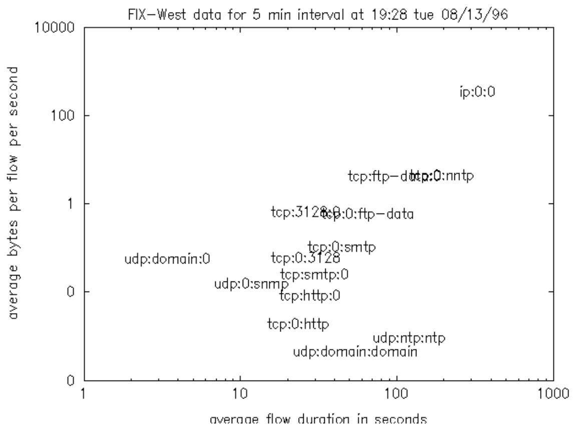

Figure 18: Byte-duration product of several popular applications during one 5-minute interval (FIX West)

network interexchange point among several providers, both national and regional, both commercial and federal. Although most commercial access has now moved to MAE-West, there are still several network providers that actively use the FIX: e.g., PACCOM, NSI, Sprint, MCI, ANS, and ESNET. All the following graphs reflect the second week of August 1996. Clearly there is still a commodity left (IP packets) for which the United States is a net exporter to Japan.

Figure 19 shows IP traffic between the US and Japan throughout the second week of August 1996.

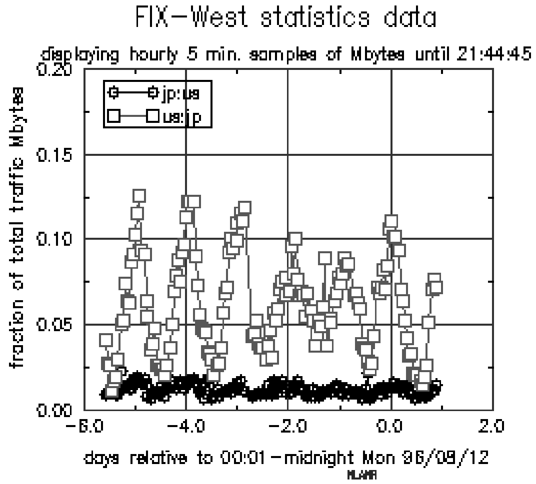

Figure 19: Japan-US trade balance in IP packets (FIX West, 7-12 August 1996)

Finally, one can graphically display traffic matrices by a specified granularity. We so far have only implemented support for country-by-country matrices, using the Internic database to map from IP address to country code. Figure 20 shows a snapshot of the top bandwidth-consuming countries from a single five minute sample from FIX West.

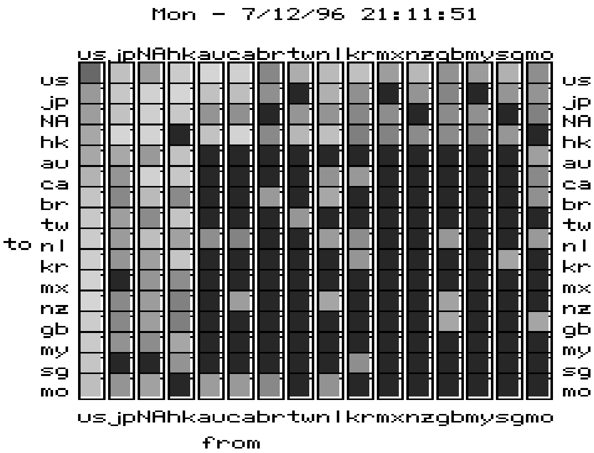

Figure 20: Country by country traffic matrix for FIX West five minute sample at 2100PDT 12 August 1996 (NA represents traffic of network numbers whose country mappings were not available from the InterNIC database.)

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. What we would most like to do next is to enhance OC3MON to use IP-to-AS path tables to support flow conversion at the autonomous system (AS) granularity, which will allow for assessment of traffic flow at a convenient macroscopic level.

Apart from the firmware on the NIC cards, OC3MON is not tied to the OC3 ATM interface. One can add any other interface type available for the PC, including FDDI and Ethernet. We are actively pursuing OC12c interface cards for the next generation of the monitoring platform. Our design goal is to be 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 OC3MON, such as flows extraction from the packet header trace onto the interface card, in order to offload the host processor. This optimization may become 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:

- 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 original prototype for web query engine, written by Hans-Werner Braun, is currently housed at http://www.nlanr.net/NA/ . 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.zip .

Acknowledgments

The data collection at FIX West is the result of a collaboration with NASA, MCI, NSF, and Digital Equipment Corporation. We are grateful to Hans-Werner Braun for prototyping the original flow statistics software, making it freely available, and coordinating its deployment at FIX West.

Author Information

Joel Apisdorf is a vBNS staff engineer at MCI. He dreamed up OC3MON in July of 1995 and has been implementing it ever since. He has developed communications and testing software for a variety of companies including Cable & Wireless, IBM, and GTE. Contact him via e-mail at apsidorf@mci.net .

kc claffy is a research scientist with the distributed National Laboratory for Applied Network Research, and based at the San Diego Supercomputer Center. Contact kc electronically at kc@nlanr.net .

Kevin Thompson is presently a senior engineer in the Internet Engineering department at MCI. He was employed as an engineer at MITRE Corporation in the Networking Center until 1995. He supports statistics collection architecture and implementation for the vBNS. His e-mail address is kthomp@mci.net .

Rick Wilder is an internet engineer at MCI. He manages MCI's engineering activities in the National Science Foundation's vBNS network and is involved in the planning and evolution of other IP services including internetMCI. His e-mail address is rwilder@mci.net .

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*.