Proceedings
from

NETWORK MANAGEMENT WORKSHOP

Sponsored by

Defense
Advanced Research
Projects Agency

Held at
USC Mount Ada Conference Center
9 - 11 January 1983

Proceedings
from

NETWORK MANAGEMENT WORKSHOP

sponsored by
Defense
Advanced Research
Projects Agency

held at
USC Mount Ada Conference Center
9-11 January 1983

Table of Contents

4. Attendees

Barry Leiner	DARPA
Jon Postel	USC-ISI
Jil Westcott	BBN
John Jubin	Rockwell
Gil Falk	BBN
Jim Herman	BBN
Steve Kent	BBN
Jack Haverty	BBN
Keith Klemba	SRI
Cliff Weinstein	Lincoln Labs
Jim Forgie	Lincoln Labs
Dale McNeill	BBN
Dave Mills	Linkabit

Acknowledgment

Thanks are due to all the participants in the workshop for their time and effort resulting in productive discussions and proceedings. Particular thanks are due the staff of University of Southern California, and particularly Ms. Judie McIntyre, for providing the coordination and support which helped make the workshop a success.

1. Introduction

Computers and data processing have undergone a tremendous growth in the past several years, and their proliferation throughout the military has begun. With this growth comes the need for efficient, reliable, and robust computer communications. Recognizing this need, DARPA has sponsored the development of a number of computer networking technologies. Beginning with the ARPANET (the first packet-switched network), programs were begun to develop packet radio (a blend of packet switching and broadcast radio), satellite networks (to allow dynamic sharing of satellite channels) and interneting (to allow communications to take place over dissimilar networks). In addition, military requirements dictate the need for security, and therefore a program in network security was undertaken.

Considerable progress has been made in each of these technologies, and many similar questions have arisen within the different environments. This motivated the holding of a workshop to discuss these common concerns and share results. In January 1983, such a workshop was held at the USC Mount Ada conference center. Sponsored by DARPA and attended by researchers in each of the technologies mentioned above, discussions were held on the various areas of concern. To facilitate such discussions, presentations were first given on each of the networks under development, with the main focus being on their automated network management and control strategies and algorithms. Round table discussions were then held on three primary areas; routing, flow and congestion control, and the interaction of network and internet algorithms.

These proceedings consist of two sections. The first section provides write-ups by a number of the participants of various issues (of their choosing) resulting from the discussions. (note: it was felt that such write-ups would be more valuable to the community at large than conventional minutes). The second section provides the vugraphs of the initial network presentations. It is hoped that these proceedings will be useful to the reader in outlining several issues in computer networking and will stimulate further discussion.

2. Discussion Viewpoints

2.1 Overall Discussion

Barry M. Leiner
DARPA/IPTO

Discussions were held on a number of subjects after the initial presentations were given on the various network management and control strategies. It was clear that the exchange of information between the various network developers was extremely useful and productive. Several points came out of the discussions. First of all, it was clear that, although similar issues arise in the development of the different networks, the different environments that the network management strategies have to deal with dictated very different network management solutions.

Secondly, considerable discussion was held on the relationship between network management algorithms and the internet management algorithms. It became apparent that, although interneting could be done under the assumptions that little was demanded of each network other than a basic datagram service, improved performance could be achieved by allowing a stronger interaction between the network and internet algorithms. For example, more intelligent routing could be done by the internet algorithms if each network provided the gateways with information about the current state of delays through the network.

This point became even more apparent as the discussion turned to congestion and flow control. While it was perhaps possible to do congestion and flow control in a simple internet environment, a tight coupling between the congestion monitoring and control algorithms in a network and the algorithms used by the internet would allow much more effective use to be made of the available network resources.

The main result of the workshop, in my opinion, was twofold. First of all, the exposure of the various network developers and researchers to other networking technologies and approaches was valuable in giving them new insights into the roles that the different algorithms can play. Secondly, the realization that coupling between the internet and network algorithms can result in improved performance is likely to have impact on research activities over the next several years.

2.2 Notes on Network Management Issues

Jack Haverty
Bolt Beranek and Newman

The issues of Network Management are really the issues of how to design a network system that works well in the real world. There is a large gap between the sciences such as queueing theory and statistics, and the real-world environment which we have to deal with in deploying a network and maintaining it over some lifetime. The success of a network architecture, and of any specific implementation, will be dependent upon the degree to which the designers and implementors recognize and deal with the problems which are endemic to real systems, as well as the problems which are well-covered by the various scientific disciplines.

With this requirement in mind, we can consider several broad guidelines for the network designers and implementors:

o figure out what you are building

o be paranoid

o expect changes everywhere.

The first of the guidelines, "figure out what you are building" seems obvious, yet it is very often ignored at the system level. In any large system-building effort, many people will be involved. Competent builders will in fact always "figure out what they are building". The problem usually arises when the various separate people or teams have different ideas about what the individual components of the system are required to do.

The four diagrams of the "Internet Network" illustrate this point. Each diagram illustrates the structure of the Internet as it was viewed by some group of researchers, at some point in time. Each such group was intensely engaged in designing the appropriate network management mechanisms for the Internet Network, as they viewed it at the time, including techniques for dealing with addressing, flow control, error handling, fault isolation, redundancy, and the other problems of a network design activity. The basic problem with having such different views is that it is natural for each group to assume that some problems were being addressed by the designers of the other parts of the system. The pitfall that awaits in these situations is that the final integration of all of the pieces will uncover gaps between the various pieces, as well as some pieces which are internally

inconsistent if the various members of the design team had different views of what they were trying to build!

Another advantage of clearly defining the requirements of the system being designed is the tremendous savings in time during the design process. At a recent meeting, several camps were engaged in intense debate about the appropriate techniques to apply in an information retrieval system application, comparing the virtues and drawbacks of indexing, hash-coding, and various searching techniques. It was only after long discussions that everyone realized that one camp was focused on an application with a few hundred entries in the data base, and another on an application with a few hundred thousand entries in the data base. Each had an opinion about the right techniques to apply to the problem, and each was right, for the problem they were addressing!

The second guideline, "be paranoid", addresses reality. In any complex system, probability dictates that as a system becomes larger, it becomes more likely that the system is not implemented as was intended in the design. A design which carefully lays out the structure of the system, defines interfaces between components, and proscribes the detailed behavior of each component, is essential to the success of complex systems such as networks. An implementation of each piece, which does not rely on the various components behaving exactly as specified, is even more important to the success of the network in the real world.

There are many reasons for this difference between design and reality:

o something (hardware, software, firmware, etc.) may simply be broken, due to hardware or human error; in the worst case this may not bring the entire system down, but rather cause it to behave strangely;

o there may be a bug in the implementation of some piece (always the other guy's of course);

o there may be ambiguities in the design, or in the design specification documents (easy to happen, given the first guideline above);

o the overall system is in a constant state of flux, as the different players implement, tune, fix, and improve their pieces.

A successful network system is paranoid. It must continue to function, at the best possible level of functionality, even when something is wrong. Systems that don't follow this guideline tend to be characterized by a constant stream of problems and never provide the level of service which users

expect. The implementors race furiously to eradicate the 'last bug', which will allow them to release a system which they can be proud of, but it never comes to pass because the basic design cannot work well in an environment where everything isn't quite working exactly as expected.

At a system design level, this paranoia may influence design decisions. For example, two routing algorithms might be under consideration. One may have theoretical behavior which will result in an allocation of resources in a very efficient fashion, but be very sensitive to inaccuracies in the information fed to it as input. The other may be less optimal, but also less sensitive to anomalous events, such as loss of some routing information, or imperfect measurement of delays, and so on. In a system design from a theoretical viewpoint, the former, more optimal, mechanism would be chosen. In a real-world design, the latter, more robust, mechanism is preferable.

The third guideline, "expect changes everywhere", also reflects reality. Basically, this guideline recognizes the fact that it's virtually impossible to get a complex design correct the first time, or ever for that matter. Based on experience in the field, bugs, suboptimal design choices, and other problems will surface, and need to be addressed. Also, as the system is used, the users have a tendency to start applying it in new ways, which the designers didn't predict, and for which the original design choices increasingly become inappropriate. Sometimes this phenomena is known as "software rot", but it really reflects that fact that the environment surrounding a system changes, independently of whether or not that systems itself is being changed. Thus even if "nothing has changed" in a system, the world around it has changed, and may excite latent bugs, or simply drive the system outside its original designers' intent.

An important characteristic of a network system is therefore to be flexible, to be able to adapt to the changing world. Typically this impacts the protocols more than anything else. For example, a protocol should always have a 'version' field, or some mechanism whereby the 'next' version can be introduced, without causing enormous upheavals in the current system. In the Arpanet for example, the routing machinery has significantly changed over the lifetime of the network, in response both to increased understanding of the problems as well as changing requirements as the technology is applied to more problems. It was critical to the success of the Arpanet that this could be done in a fashion which did not disrupt current activity. Without this characteristic, the network managers are left with the untenable choice of having frequent major upheavals, versus living for long periods with various problems in between infrequent major upheavals. Neither of these choices is satisfactory.

Network systems are complex, distributed, multiprocessor, real-time systems. Management of these systems over a long life cycle requires attention to the realities of the environments in which

they exist, and of the changes that will occur over a time. A network must be dynamic not only from second to second, but also from year to year, in order to be successful. The three guidelines discussed above reflect some design principles which seem to address this kind of system architectural issue.

2.3 Comments on Workshop Issues

Jon Postel
USC-ISI

Three major topics emerged as the focus for discussion: (1) Congestion Control, (2) Routing, and (3) Monitoring and Control.

Each of these topics became first a focus for an exchange of ideas between the different network types, then soon led to discussions of what more the networks could do to either hide the problem from the hosts, or involve the hosts in a complex resolution of the problem.

My view is that we should make sure we keep things as simple as possible. In many cases we can avoid the difficult problems by increasing capacity instead of devising complex procedures to get the last 10% of use out of an overloaded system.

An interesting notion came out in thinking about the alternative to what we have in the current Internet. We now have an internet of networks where each network is built of a specific media (ARPANET = 50Kb lines, PRNET = radio, WBNET = satellite, etc.). One could instead build networks of "mixed media". What impact would that have on the major topics?

Another contrast that came up was the difference between a "loosely coupled" internet and a "tightly coupled" internet. It seems that the "loose" design allows more flexibility and generality, while the "tight" design allows more control and perhaps better performance.

One needs a statement about the basic assumptions of the Internet. Is there any fixed set of requirements that the rest of the design must support?

Here is my set of statements:

(1) No change required to component networks.

(2) The Internet should be robust.

(3) There should be end-to-end reliability.

There are a bunch of other statements that are derived from these.

(1) The Internet should be a datagram network.

(2) Gateways should be simple, cheap, and fasts.

(3) Gateways should get only the information essential for their task from the hosts; that is, IP should include only what gateways need to see.

(4) The end-to-end reliability has to be done in the hosts; that is, put all the end-to-end stuff in TCP.

(5) Since datagrams are not reliable, error reports about datagrams are not reliable either. The hosts can not count on receiving any feedback on errors.

Then there are some other goals and policy statements.

(1) There will be about 1000 networks in the Internet.

(2) There will be several different suppliers of gateways.

There seemed to be a fair amount of sentiment for making modifications to the underlying networks to make them do a better job given that they are serving IP/TCP using hosts. The possible changes range from changing packet sized to those called for by IP (thereby eliminating the need for IP fragmentation), to giving feedback on congestion problems, and getting involved in routing decisions. Again, let me call for keeping things simple.

There seemed to be many stories of really bad performance, in many cases traced to (1) not enough memory, or/and (2) not enough cpu speed. I think this should be a point of concern in planning the development of any new network or the expansion of existing networks. The hardware of interfaces, IMPs, GWs, etc. has to have enough capacity to do the job. If it does not,
clever programming still won't help.

2.4 A Maximum-Likelihood Algorithm for Filtering EGP Updates

Dave Mills
M/A-Com Linkabit

A scruffy old EE who has fumbled in the signal-processing game for some time might come up with the following analysis of what is needed to make IGP/EGPs work in our networks of mismatched plumbing.

The Model

Consider a set of autonomous systems, each containing an almost-copy of the routing data base for N networks. Each system maintains a local copy using its native IGP, which is optimized for the particular network dynamics. These systems are connected together via EGP, which we have asserted copes with gross mismatch in these dynamics. The problem is to design the protocols so to maintain the local copy in each system with the best accuracy in the shortest time.

First, consider what the ideal (from a signal processing point of view) gateway G would do to process routing updates received from other gateways in its system. Its routing data base R contains N entries, each of which is a random variable X that can take one of M values (perhaps a (hops, port) code) and is updated by IGP on average once every T seconds. We can assume the system does the best job possible in maintaining R consistent, but expect glitches to occur from time to time due to changes in connectivity.

Next, assume G exchanges routing information via EGP with a number of neighbors, each marching to its own dynamics which may be faster or slower than T. Let G' be one of the neighbors and assume it belongs to a system marching to T'. The dynamics of EGP itself are limited to the lesser of T and T', which we will call t, and may be considerably less. The mechanisms used to smooth and filter routing information from G to G' via EGP is the topic of this note.

Computing Updates

An algorithm often mentioned for determining connectivity counts the number of I-Heard-You replies to the last n Hello messages and declares an UP transition if the count is greater than j and a DOWN transition if the count is less than k. This is in fact an implementation of the matched filter familiar in signal processing, where the "signal" is a "square wave" of period 2*n. This would seem a good algorithm to deal with short connectivity glitches while maintaining good system stability.

A faithful implementation of this algorithm uses an n-bit shift register and a counter that increments when a one is shifted into the register and decrements when a one is shifted out. The value of the counter is then compared to the j and k thresholds to determine the transition. The following paragraphs describe how this algorithm can be applied to the problem of coupling the routing data base of one autonomous system to another.

The Algorithm

During each interval of t seconds duration information is collected over one or more IGP intervals of T seconds. Note that the actual interval between EGP NR messages may be considerably longer than t. An algorithm of the Maximum Likelihood (ML) class is proposed here. It uses the routing data base R of gateway G, together with lists of 3-tuples (X,s,r), where X is the code, s an n-bit shift register, where n equals the greatest integer contained in t/T, and r a counter. There is one list associated with each entry of R for each set of neighbors with the same n. In the natural implementation all lists for all neighbors would be maintained in a threaded fashion for storage economy. The total number of 3-tuples necessary could potentially reach g*n*N, where g is the number of neighbors; however, in practice far fewer will be required.

The algorithm is run once during each IGP update interval, producing at most one 3-tuple for each entry and possibly deleting one or more 3-tuples. It starts with an empty list for each entry and operates subsequently as follows:

1. For every (X,s,r) on all lists shift s to the right, filling zeroes from the left. If a one bit is shifted off the right end, decrement r by one. If r = 0 delete the entry.

2. For each entry in turn search the list of 3-tuples for an occurrence (X,s,r) where X matches the value of the entry in R. If found replace the first bit of s by a one and increment r by one. If not found add (X,s,1) to the list, where s contains a one bit in the first position and zeroes elsewhere.

3. When the EGP update is computed and stored in the output buffer, examine each list for that neighbor in order. The 3-tuple with the largest r represents the ML candidate and its value X is copied into the output buffer as the entry.

Discussion

This algorithm is designed to filter glitches from the local routing data base and deliver clean data to the set of EGP neighbors. Each neighbor may have a different aperture (update interval) in its own

IGP, so for optimum performance each different aperture requires a separate set of lists. The EGP updates themselves may be sent at intervals much longer than t, with the understanding that the most recently filtered good data are delivered at whatever time the update is sent.

In common with similar algorithms used in signal processing, this algorithm tries to keep track of all possible outcomes of the filtering operation simultaneously. In principle, there is no UP-DOWN-UP transition where old routing data have to be purged before new data can be believed. In practice, the required storage (number of 3-tuples) can be greatly reduced by discarding low-probability 3-tuples (low values of r) should the number attached to any particular entry exceed a threshold.

The number of bits in the shift registers depends on the ratio t/T, which can be interpreted as the inverse ratio of the "bandwidths" of the routing systems of the autonomous systems connected by the EGP link. If the ratio is near unity the algorithm degenerates so that only a single outcome is possible for each update interval and no lists are required. Systems like GGP, which "count to infinity" when a gateway goes down, exhibit an inherent instability which can last for an interval approximately equal to the product of the mean update interval times the network diameter. The cleanest routing information available in the shortest time will result if the number of bits is equal to the network diameter.

Remarks

This is a nice exercise to gain some insight in the optimum behavior of a system of autonomous systems and coupling dynamic routing algorithms in general. The particular algorithm suggested here is probably best suitable for implementation in high-performance systems which place a premium on dynamics, in particular, systems designed for partitionable nets.

2.5 Thoughts on the Network Management Workshop

Jim Herman
Bolt, Beranek and Newman

I found the conference on Network Control Algorithms extremely stimulating and informative. I learned a great deal about the Internet project and the Packet Radio project. Until that conference I had concerned myself almost exclusively with the problems of the ARPANET and paid little attention to the Internet that was growing up around it. At the conference, and the Internet meeting that followed it, I became aware of the need to consider the mutual interactions between the Internet system and the ARPANET. What the ARPANET does affects the Internet, and what the Internet does affects the ARPANET. Changes in the global control algorithms of either must take into account the effect on the other. Achieving efficient, effective performance from both systems will require interaction and cooperation between them.

This is a major change in perspective for both projects as far as I can tell. My own ignorance of the Internet project is perhaps the best evidence of this. How can it be that this was my first Internet meeting, my first opportunity to meet many of the important designers and architects of the Internet system, when I have been the manager of the ARPANET project at BBN for the past three years? In discussions at the conference it became clear that this division between subnetworks and the Internet had been an explicit goal of the Internet project. Gateways were supposed to be able to interface to networks as they were, without making special demands on them for special services or interface features. The ARPANET still thinks of itself as servicing regular hosts and makes no special provisions for gateways.

It is now clear to me that we must make a major change in this thinking. There is a clear trend towards local network technology which will result in many more gateways. It could be that five years from now the host that directly connects to the ARPANET will be the rarity and that most hosts on the network will be gateways into other networks, especially local area ones. If such is the case, a lack of specialized interaction between the IMPs and gateways seems like a needless constraint, one that will lead to poor performance and waste of resources.

What areas can be investigated first? I suggest looking at the area of congestion control as the first one in which to consider passing dynamically varying data about network conditions to a gateway. Gateways are as capable as IMPs at processing dynamic data and the IMPs should begin to take advantage of this. In congestion control the main problem is the propagation of information about network conditions back to the source of a data flow. This will require an interaction between IMPs and gateways to provide internet level congestion control.

Other areas of investigation include having the IMPs assist the gateways in finding each other. There could be some special interface message that identified a gateway as such. The IMPs could use the new logical addressing feature in a slightly augmented manner to table this information and make it available to any host, including other gateways. The operation of the gateways should also be considered in redesigning the ARPANET subnetwork end to end protocol. Lastly, the ARPANET host to IMP protocol and subnetwork end to end protocol should be tuned for operation with TCP now that mosts hosts use this protocol.

Further down the road, I see the possibility that the IMP and the gateway will actually merge into a single system. In thinking about this I see little reason to distinguish between the two. The optimal internet system would have information about all subnetworks it uses and would utilize algorithms that were tightly integrated with those of the subnetworks. This can best be achieved when the subnetworks utilize the same basic algorithms as the gateways themselves. THere are still reasons for specialized subnetworks when dealing with unique media such as satellites, radios or local area networks. These are different mostly because of their broadcast nature whereas the internet and ARPANET are based on a point to point topology. Integration of broadcast subnetworks into the routing schemes of an internet that is based on point to point routing is an interesting problem that I also hope to work on in the next few years.

Although the internet system and the ARPANET have developed during the past few years with little interaction and special treatment of each other, I see this changing now. I think that it will be necessary to adapt the ARPANET to the needs of the Internet and that the Internet will begin to look inside the ARPANET and deal more directly with its dynamic state in making decisions about global control. BBN's major involvement in both projects makes it ideally suited to pursue these goals. I hope to be able to foster new interactions and sharing of ideas between the programmers on these projects and to encourage them to think of ways in which the two systems can become mutually better adapted to each other.

2.6 Wideband Satellite Network

Gil Falk
Bolt, Beranek and Newman

GENERAL COMMENTS

The focus of the workshop was on control mechanisms employed in the constituent networks of the ARPA internet and the internet itself. Major topics addressed during the workshop were the related areas of routing, congestion control, and flow control. Network monitoring and control of the type carried out by the ARPANET Network Operations Center was also discussed briefly. Routing is always fundamental to network operation, that is, each data unit must be routed to its proper destination independent of the network traffic level or network type. Routing algorithms will, of course, be much more complex in networks requiring multi-hop paths from source to destination than in fully-connected or single-channel networks. Congestion control, while generally necessary for network operation, can often be ignored to the extent that the network is guaranteed to be lightly loaded. Local area networks, where bandwidth is relatively inexpensive, was identified as one specific environment where it is probably cheaper to buy additional bandwidth than implement a complex mechanism to manage more limited transmission capacity. It is interesting to consider how long the ARPANET has operated with limited mechanisms specifically designed to address the congestion control problem. In order to support operation at higher traffic levels, the ARPANET is only now actively developing a separate mechanism for global congestion control. Flow control needs to be provided on a node-to-node basis and an end-to-end basis within the communications network as well as being supported by end users themselves. A network without internal end-to-end flow control can become congested due to a rate mismatch between only a single sender/receiver pair. The implementation of such a mechanisms allows a network to protect itself and eliminate at least one source of global network congestion. The following paragraphs summarize mechanisms in place and planned to support routing, congestion control and flow control in the Wideband Network.

THE WIDEBAND NETWORK

The Wideband Network is an experimental 3 MBPS communications system being used to evaluate the use of satellite packet switching for efficient voice transmission, voice conferencing,

and the integration of voice, image, and data transmission. The Wideband Network is a direct descendant of the 64 Kbps Atlantic Packet Satellite Network (SATNET) modified to operate at the higher satellite channel rate. Both SATNET and the Wideband Network manage the allocation of a single broadcast channel via PODA, a priority oriented demand assignment packet reservation protocol.

A detailed comparison of SATNET and the Wideband Network implementations was presented at the workshop in the form a table comparing the two systems. The Wideband Network (as well as SATNET) provides a unique environment for routing, congestion control, and flow control mechanisms due to:

- Operation on a single broadcast channel
- Satellite introduces 1/4 sec delay on all transmissions
- Bandwidth is cheaper than terrestrial common carier links but not negligible (as in the case of LANs)
- Network designed to handle continuous (stream) as well as bursty (datagram) transmission modes
- Message priorities must be accounted for in channel allocation mechanisms
- Each station can hear the transmissions from every other station

ROUTING IN THE WIDEBAND NETWORK

Routing is trivial in the Wideband Network due to the single broadcast channel which supports fully-connected network operation. Any datagram message destined for a non-local host, any message addressed to a group, or any stream message at all is scheduled and routed over the satellite channel on the uplink. On the downlink, each message received is examined by each node and messages with destination addresses corresponding to local hosts are delivered. (By a filtering mechanism it is possible to eliminate the need to individually examine each message that a station itself transmits.)

CONGESTION CONTROL IN THE WIDEBAND NETWORK

The PSATs in the Wideband Network currently implement only a very limited type of local congestion control. Queue limits are currently established for a number of host and channel-related queues. If the size of these queues ever reach the defined limits, messages back up in the node and the node eventually stops accepting messages from the host. In the near future the node software will be modified so that data messages are discarded when this local congestion occurs. The Wideband Network does not guarantee delivery of messages and TCP or other higher level protocols must handle retransmission if it is

appropriate. In the future a global congestion control mechanism is planned which predicts future offered datagram load and throttles source hosts when predicted offered load exceeds available capacity.

The currently envisioned mechanism uses averaged reservation requests heard over some interval (e.g. a few seconds) as the predictor of future offered traffic load at each priority level. BBN Report No. 4179 describes the proposed algorithm where each node computes ALLOWED-TRAFFIC(priority K, interval T) = ACTUAL-TRAFFIC(priority K, interval T-1) + FRACTION*AVAILABLE-CAPACITY(priority K, interval T-1). The AVAILABLE-CAPACITY(priority K) is computed as Q*(C - Sk) - Dk where C is the raw satellite channel capacity, Sk is the allocated stream capacity at k or higher priority, Dk is the measured (averaged) datagram load at k or higher priority, and Q is a factor representing the desired utilization planned for the datagram portion of the channel. This scheme makes incremental changes in the allocations to the various Wideband Network nodes in an attempt to maximize satellite channel utilization without overloading. A recent review of this proposal has indicated that it still appears feasible in spite of various changes to the channel protocol which have occurred since the time that Report No. 4179 was written.

FLOW CONTROL IN THE WIDEBAND NETWORK

There are currently no end-to-end flow control mechanisms implemented in the Wideband Network. BBN Report No. 4179 also proposed an incremental end-to-end flow control mechanism similar in structure to the global congestion control scheme described above. Unfortunately, the proposed scheme involves each node monitoring the flows to each destination port in the network at each priority and has a number of additional shortcomings. An alternative scheme proposed more recently which seems attractive has flow control decisions made at the destination and broadcast to the other nodes in "throttle" and "unthrottle" messages. This proposal as well as the congestion control scheme described above require considerably more analysis prior to any decision with regard to implementation for the Wideband Network.

HOST ACCESS PROTOCOL FLOW CONTROL MECHANISMS

There are several mechanisms defined within the Wideband Network Host Access Protocol (HAP) to support current and future internal flow and congestion control mechanisms. An optional Acceptance/Refusal (A/R) mechanism can be enabled. Individual messages as well as groups of messages can be identified as accepted or refused by A/R exchanges. In addition, each message delivered to the host by the network contains a GO-PRI field

which identifies the lowest currently acceptable priority of the network. The host can use this information to avoid sending messages that will certainly be rejected. Finally, the current status messages exchanges which keep the host link alive contain information on the available stream capacity which may be of value to the host in deciding whether or not to attempt the establishment of stream traffic flows. As we refine the internal control mechanisms of the Wideband Network we will have to review the suitability of the existing HAP feedback to the hosts.

2.7 Notes on TCP Flow and Congestion Controls

Dave Mills
M/A-Com Linkabit

Introduction

Following are some notes on TCP flow and congestion controls which were prompted by some considerable experience trying to force fire hoses into tiny pipes while keeping the Internet plumbing from leaking. They describe a model believed generally applicable in the space of TCP/IP host implementations, but was developed with the DCN fuzzball implementation in mind. These notes describe the model, which includes a set of estimators for delays and flows used to manage resource commitments in the host and the network. A procedure for calculating these estimators is presented along with algorithms using their values to effect per-connection flows.

System Model

We are concerned with a single virtual circuit consisting of a pair of TCP peers connected by an Internet path involving some number of local nets and gateways. The most interesting cases occur when either host can generate flows well in excess of that acceptable to its local net and when at some point in the path a flow mismatch of one or two orders of magnitude exists between two local-net neighbors.

Of primary interest is the control of flows for bulk data transfer, such as would characterize an FTP data connection. In such cases the primary concern is for throughput, rather than delay. Throughput is degraded by retransmissions due to lost packets, which in turn is due to insufficient resources, primarily packet buffers. While end-to-end flows are controlled in TCP by a window mechanism which is responsive at the level of individual octets, it can (and often does) happen that insufficient resources at gateways along the path cause packets to be lost anyway.

The problem addressed in these notes is to identify a set of estimators which can be used to predict the resource demands implied by a particular flow of data (octets or packets) from moment to moment as transmission proceeds. The estimators can be used to predict delays and flows at several points along the path. From them can be derived such things as the retransmission timeout (zero and non-zero window cases), along with flow strategies designed to minimize resource demands throughout the system while sustaining high throughputs.

The Host

The host can support a number of simultaneous TCP connections, each with its own set of state variables and estimators (see below). An interval timer is assumed with resolution in the order of a millisecond. The host operating system is assumed to have some sort of internal flow controls as part of its internal resource management system. The controls act to deny a request for a packet buffer if the quota assigned that connection has been exhausted. Once a request has been denied another is not made until after an system-dependent interval.

As each packet is filled it is sent to the net using some sort of IPC message. The packet may be multiplexed along with others on a queue for transmission into the net. When the packet has been transmitted an IPC message is returned to the TCP process and used to update the estimators for the particular connection (see below).

The Net

The local nets may or may not have intrinsic mechanisms to control flows between the hosts and gateways. In the case of the present ARPANET flows are controlled by RFNMS and blocking. For the purposes here, we can assume control by RFNMs avoids blocking and that the flow controls back up into the host, which can determine which connection is involved and update the estimators accordingly. This information is conveyed to the TCP process via the IPC message mentioned above.

The Gateway

The gateways typically allocate packet buffers on the basis of input interface and output queue threshold. If a particular output queue threshold is exceeded for a packet arriving at a particular input interface, it is discarded. If some number of packets are discarded in this manner during a system-dependent interval, the gateway returns a source-quench packet to the originator. A host receiving such a packet sends an IPC message to the TCP process, which then updates the estimators for that particular connection.

This mechanism is generally agreed to be inadequate on the basis of several shortcomings. One is that the information arrives too late to effect a useful modification in behavior on the connection. Another is that it is effectively bang-bang in nature and can lead to undamped flow-rate transients. A third is that it is sent only after considerable numbers of packets have been lost and, presumably, long delays for TCP retransmissions.

Several suggestions to improve the effectiveness of source-quench have been made. One is to keep an LRU stack for each output queue to help isolate those hosts claiming excessive resources. Another is to send source-quench messages before packets have to be dropped and to include additional information, such as could be derived from the LRU stack. A third is to send source-quench messages to the destination host, as well as the source host. It is not the intention of these notes to explore these issues further; however, it is assumed that the host receiving a source-quench message will in fact use whatever information is in fact contained to update the estimators for that connection.

Estimators

Each estimator contains a value computed from past behavior and which can be used to predict future behavior. Typically it is an average of past samples of a random variable, such as round trip delay, with newer samples weighted more heavily than older ones. In recursive-filter averaging a new sample value weighted by w is added to the running average weighted by (1-w). In matched-filter averaging a new sample is entered in a shift register containing n of the most recent samples and the average computed as the sum of the n sample values divided by n. There is some reason to believe the matched-filter method may work better than the recursive-filter method for some estimators, but this will not be an issue in the following.

The accuracy of an estimator depends on the number of samples included in the average and the sample variance. An estimator is typically updated for every received ACK packet or IPC message. Since bulk-data transfers typically use large packet sizes, it is important to gain as much information as possible in each sample. A typical problem occurs when the sample value correlates strongly with the length of the packet, which can occur if the degree of aggregation on the path is small. Thus, differences in the length of the packets can show up as a larger sample variance, which can lead to increased retransmissions with some choices of TCP parameters. This is the principal cause of performance degradation now observed on tiny-pip nets.

The estimators suggested in these notes are all constructed in the same way. When some event happens, such as sending a packet, the time of the event and the current sequence number are recorded in a FIFO stack. When an ACK packet or IPC message arrives the FIFO stack is searched for the entry with sequence number less than or equal to the sequence number ACKed. The elapsed time is then computed and adjusted by linear interpolation between the entry found and the next later entry.

The elapsed time computed in this way can be used to update a delay estimator directly by using

one of the averaging methods above. In addition, the elapsed sequence numbers can be used to update a related estimator. By simply dividing the second by the first a flow-rate estimator can be derived and used directly to control flows on the connection.

There are three estimators that are suggested naturally by the model, the net-input, TCP-ACK and TCP-window estimators. Each is described below:

Net-input. Two events update this estimator. The first is an IPC message when a packet buffer has been sent to the net. The second is an IPC message as the result of a source quench. There is not much that can be done with the current source-quench mechanism other than to fiddle the estimator value in an ad-hoc way, such as reducing it arbitrarily by half. The flow-rate estimate indicates the rate the sender should use for most efficient use of the Internet path to the receiver.

TCP-ACK. This is the classic "RSRE Algorithm" refined by the FIFO stack and interpolation technique described above. This technique reduces sample variance and probably should have some correction for packet length. The delay estimate is used in calculating the initial TCP retransmission delay. The flow-rate estimate indicates the rate the sender should use to fill the window.

TCP-window. This is computed in the same way as the TCP-ACK estimator, but the sequence number used is the right-window edge, as determined from the window field in the TCP ACK packet. The flow-rate estimate indicates the rate data are being delivered to the end user and thus the net rate of the end-to-end circuit. The sender should try to send at a somewhat higher rate and rely on the TCP window for fine tuning.

Flow Management

Let's examine some strategies that use the values of the delay and flow-rate estimators described above. First, in the case where the sender flow is small compared to the net and end-user capabilities, the TCP-ACK and TCP-window rates should be about the same. In the interesting case mentioned above with fire hoses and tiny pipes, the net-input rate is likely to be much higher than either of the others. In the case where the end user is much slower than the sender or net, the TCP-window rate will decrease, possibly going to zero.

The current DCN fuzzball implementation incorporates the first two of these estimators along with a simple recursive-filter averaging method. Flow control is based on the net-input rate, which follows the local-net back pressure and responds to source-quench messages. This is done by throttling the

actual flow rate into the net so as not to exceed the net-input rate. The effect of this is to avoid tying up packet buffers unnecessarily, as well as provide a control point for flow modulation.

In the fuzzball implementation the TCP initial retransmission delay is calculated in the classic way from the TCP-ACK estimated delay, with subsequent delays adjusted for back off. Our experience shows that packet length should be factored in this estimate; however, this would involve estimating two quantities simultaneously, the absolute delay and the end--to-end flow rate. In point of fact, this is not too hard to do and may be considered for future implementation. Nevertheless, the performance is good with the present net configuration of up to five links (not themselves flow controlled) and from speeds of 100 Kbps or so down to 1200 bps.

Our experience indicates that considerable improvement can be had by incorporating more highly developed estimation methods, such as suggested above, and by coupling the flow-management algorithms more closely. This is planned as the implementation is refined. Some of the possible mechanisms might be:

1. Use the TCP-ACK and TCP-window estimators to implement a strategy designed to avoid trying too hard to keep the window closed, which leads to silly-window syndrome.

2. Couple the TCP-ACK and TCP-window estimators into the packet-generation strategy mentioned in connection with the net-input estimator. This would avoid pumping lots of packets into the net well before the end user is ready for the data.

3. Develop a complementary set of estimators for use at the receiver. These could be used to control the ACK strategy and avoid buffer fragmentation, while minimizing traffic on the reverse direction.

4. Investigate the feasibility of sending additional flow-control information (say, in the URGENT field of packets when the URGENT condition is not in effect) to help the sender and/or receiver improve its strategy.

One of the current pressures in the internet project is to get improved performance. An important subject of the meeting was how the current nets could assist in this, by providing information to gateways, for example, of cross-net delay and throughput. Then it would be the job of the gateways to control internet routing and congestion. Several points were made on this strategy. First, good performance probably implies tightly interacting gateways, which is a problem if numerous different organizations are to build gateways. Second, since the nets were developed independently, it is unlikely that performance equivalent to the "multimedia net" can be achieved, even if the gateways are uniform and tightly interacting. Third, the role of hosts in interacting with gateways to determine a total route (including the hop from source host to first gateway and last gateway to destination host) complicates the problem, requiring more host/gateway interaction. Despite these difficulties, the pressure to improve performance in the internet will not go away. A challenge will be to select improvements which are reasonable in terms of cost vs payoff.

There is certainly a large spectrum of intermediate positions between the loosely-coupled internet and the tightly-integrated multimedia net. The wideband system is an interesting example in this regard. The primary focus of the wideband project has always been performance, i.e., efficient multiplexing of packet voice and data onto long haul wideband links. Motivated by the need to achieve access from different types of local nets, and by the goal of achieving a degree of generality and commonality with the standard internet protocols, the wideband system has actually been configured as an experimental internet. The wideband internet includes local cable nets (LEXNETs) and radio nets (PRNETs) linked by common gateways through the WB SATNET. The gateways

implement the stream (ST) protocol, which is aimed at performance for real-time voice traffic. ST has the effect of helping to bind the individual nets together to perform more like a multimedia net. In fact, the gateways (PDP-11 miniconcentrators and Voice Funnels) have efficient traffic concentration and access to satellite streams as major functions. The gateways also include IP, which allows interoperability, but not as much performance.

Routing in the wideband internet is generally quite straightforward, since the host on a local net has only one gateway to choose, and there is only one long-haul net (at least for voice). In the discussion at the meeting on the role of host/gateway interaction in routing, it was suggested that the problem would be considerably simplified if we forced hosts to reside only on local nets, and gateways to provide the only access paths to long haul nets. This is generally the case in the wideband system. This somewhat hierarchical structure should be quite reasonable for a large class of communication requirements. However, local nets need not be restricted to have access to only one gateway and one long-haul net. Both for flexibility and survivability, access to a mix of long-haul nets is desirable. The gateway routing algorithm should be designed to select the best route, depending on load and network conditions.

2.9 Routing and Congestion Control

John Jubin
Rockwell International

Two of the major topics discussed at the Network Management Workshop were routing control and congestion/flow control.

ROUTING CONTROL

One of the issues in routing control discussed was the ability to update the distributed database in a timely-enough or synchronized-enough way to prevent route loops and other anomalies. (An analogous issue - indeed, the "worst problem" in SATNET, according to Dale McNeill - is keeping synchronized the schedule database that is distributed among the access nodes.) The way the ARPANet's SPF algorithm attempts to provide an always-consistent distributed database, as Jim Herman stated, is to broadcast every routing change throughout the network as fast as possible. The packet radio net, on the other hand, uses an algorithm - "tiered rings" - similar to the old ARPANet algorithm in that changes accumulate at a node and then are distributed at a slow rate as a part of the local-repeater-on-packets (LROPs) (analogous to the ARPANet's up-down packets). The tiered ring algorithm, however, does have checks on inter-node data consistency that the old ARPANet algorithm apparently did not have.

As an outgrowth of this discussion, I mused on what the (as yet unimplemented) algorithm should be for distributing tier data that won't all fit in every LROP. Previously I thought that distance should be the main criterion - the larger the tier value (i.e., the farther away from the destination), the less accurate the data has to be, and therefore the fewer the LROPs it has to go in. Now I recognize distance as the secondary criterion; whether there was a change should be the main criterion. Moreover, changes should be ranked according to importance as follows:

1. change from good to bad - Bad news should be spread fastest; having spread bad data is a prerequisite for accepting new (good) data in most cases.

2. increase in tier value - The goal is to have incremental increases spread; an increase in tier value by more than one causes a discontinuity in the route.

3. decrease in tier value - This is more of an optimization aid than a means to provide a good route when there otherwise would be no route available.

FLOW CONTROL

I also mused on what an ideal internetwork framework for flow control might entail....

Each low-protocol-layer, say (N)-layer, entity (e.g., an ARPANet IMP) quantifies the effective maximum allowable flow rate for each source-destination peer entity (IMP) flow through it. The flow rate for each flow is reported back to the source entity (IMP). The (N)-layer source entity (IMP) takes the lowest rate of those reported back to it from all the entities between it and the destination peer entity, remembers it, and reports it to the next higher (N + 1)-layer entity (gateway). The (N + 1)-layer entity (gateway) either knows the association between the (N)-layer's destination entity (IMP) and the (N + 1)-layer's entity (gateway or host) or is told the association by the (N)-layer's source (IMP); this identifies a certain (N + 1)-layer "link". So the flow rate reported by the (N)-layer's source (IMP) is the maximum allowable flow rate for every (N + 1)-layer flow that traverses the previously identified (N + 1)-layer "link". The (N + 1)-layer entity (gateway) reports this flow rate back to the source peer entity (host internet layer) for the flow. Each (N + 1)-layer source entity (host internet layer) takes the lowest rate of those reported back to it from all entities (gateways) between it and each destination entity (host internet layer), remembers it, and reports it to the next higher (N + 2)-layer entity (host transport layer). This cycle repeats as many times as is necessary to reach the user layer.

The salient requirements of this framework as presented so far are:

1. Every entity must be able to quantify an allowable flow rate. There can be a different rate per destination or per link, or there can be one rate for all flows through the entity.

2. Flows must be identifiable by source for the particular layer so that they can be reported, and flows must be identifiable by destination for the particular layer so that the sources can associate the values properly. If none of the preceding requirements is met, at the very least source entities must have an estimate for the generic flow rate through their network.

3. Flow rates must be reportable from each protocol layer to the next higher layer, all the way up to the user.

Once flow rate values have been distributed to all the sources at all protocol layers, they can be used for flow control. The user, of course, should offer traffic at a rate less than the allowed value. If the flow rate is detected to be exceeding the allowed flow rate at any-layer source protocol entity, that entity is warranted in discarding enough packets from the flow to bring the rate down to the allowed.

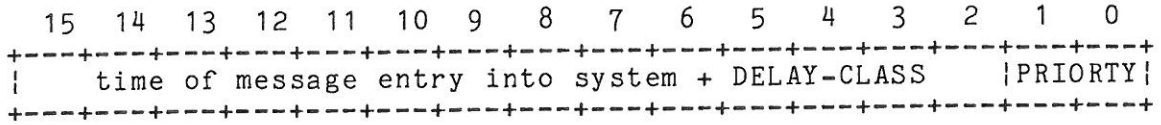
Theoretically, every layer shouldn't have to do flow control but probably ought to in order to protect its own layer from congestion. Ideally, each user source-destination flow would enter the internetwork through one spigot whose flow rate could be controlled by the spigotee, in which case no packets should have to be discarded.

2.10 Comments on Network Manager's Workshop

Dale McNeill
Bolt, Beranek and Newman

One of the lessons to be learned from the workshop is that network control mechanisms have considerable impact throughout the entire internet system and are immensely difficult and not well understood. Routing, congestion control, and flow control issues are pervasive. In broadcast networks, such as SATNET and WIDEBAND network, routing is not an issue but channel management is. Other discussed topics were local area networks, where increased bandwidth is used to avoid control mechanisms, and multimedia nets, where deficiencies with internet routing of intranet traffic are circumvented. Because SATNET is primarily a transport network, its design has a far-reaching effect; some of its details were presented at the workshop and are also presented below.

In SATNET, a message has two independent attributes which are taken into account for the scheduling of its transmission over the satellite channel; namely, DELAY-CLASS and PRIORITY. The former, which is a measure of the timeliness of a message, corresponds to the acceptable time span for a message to remain within the system; it is undesirable for a message to remain beyond its designated time span. DELAY-CLASS has the values {1, 2, 5, 20} seconds; speech traffic is expected to assume the smallest value, and bulk data traffic is expected to assume the larger values. PRIORITY is a 2-bit quantity to specify four levels. The two attributes are catenated into one word to form the urgency of a message as shown below.



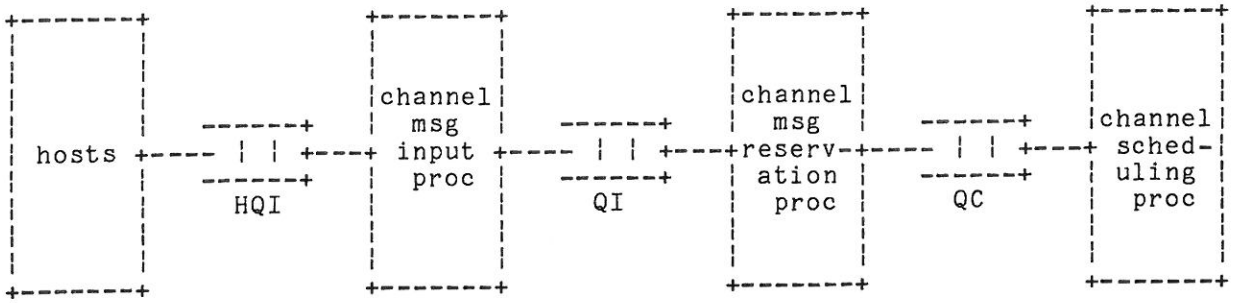
URGENCY

Channel scheduling is done by most urgent messages (lowest value of urgency) first; thus, as messages remain longer within the system, they get preferential treatment. PRIORITY is significant only for messages having equal remaining timeliness. Clearly PRIORITY has the lesser significance of the two attributes. Consequently, speech traffic of lower priority can have preferential treatment over bulk data traffic of higher priority. For messages of equal urgency, a round-robin service is provided to promote fairness.

Reservations for channel time can be sent either in a special time slot (designated the control subframe) or piggybacked on a previous message already having an entry in the scheduling queue and ready to be scheduled. Because of piggybacking, sites with considerable traffic have more opportunity to make reservations than sites with

little traffic. Hence, the channel can be dominated although not captured by heavy users. (In MATNET, where channel scheduling is by PRIORITY only, the round-robin service has greater effect to provide improved fairness over SATNET; no one site can dominate the channel.)

Congestion control is rudimentary at best; it is based on queue length throttling to provide backpressure in the system, despite susceptibility to oscillatory behavior and despite sub-optimal overload control. There are three important queues involved in congestion control; these are shown below in a simplified representation of the system operation.



The three queues are:

HQI--contains messages from the different hosts ready for submission to the channel messages input process;
QI--contains messages ready for submission to the process to generate channel reservations;
QC--contains the messages waiting to be scheduled after their channel reservations have been heard by all sites.

All three queues are serviced by the most urgent messages first. The way congestion control works is that

[1] QC throttles QI;
[2] QI throttles HQI;
[3] HQI and QI throttle the hosts.

[1] The QC throttling of QI is divided into four categories, where in each category reservations can be made only for messages such that within a specified amount of time they will have been held in the system longer than their DELAY-CLASS values. The categories are as follows:

QUEUE LENGTH	RESERVATION RESTRICTION
0 < QC <= 5	no restrictions
5 < QC <= 10	only messages with 0.5 seconds remaining
10 < QC <= 15	only messages with 0.25 seconds remaining
15 < QC <= 20	only messages with no time remaining
20 < QC	no reservations made whatsoever

[2] If QI exceeds 20, then transfers from HQI to QI are disallowed.

[3] If QI + HQI exceeds 50, then all non-control messages are refused from the hosts.

In a steady state operation where the system is overdriven, then we would expect the queues to have the following lengths:

```
   QC  =  20 messages
   QI  =  20
  HQI  =  30
          ---
total  =  70 messages delaying all traffic.
```

Flow control is minimal; the host interface is non-blocking. A message will be refused from a host if:

- The destination host is attached to a dead SIMP.
- The destination host is dead and is attached to the same SIMP as the source host.
- The SIMP to which the source host is attached is out of buffers.
- Congestion control is putting enough backpressure on to cause messages to be refused from the hosts.

In the current Honeywell 316 SIMP, where buffers are in short supply, it is not clear whether reservation throttling or lack of buffers is more important in restricting the hosts.

2.11 Comments on the Network Manager's Meeting

Jim Forgie
MIT Lincoln Labs

COMMENTS ON THE NETWORK MANAGERS' MEETING

by

J. W. Forgie

I came away from the meeting on Catalina Island with a number of thoughts about the topics that were discussed. For this brief report I have chosen some that were of particular interest to me and that appear relevant to work in the Wideband Project. They are:

1. I think the Internet project must evolve from an experiment in interconnecting "given" networks into an effort to achieve a multi-media net with performance adequate to meet the needs of its users. The IP/TCP transition has forced many ARPANET users into an "Internet" environment even though their packets rarely, if ever, pass through a gateway. This environment is exacting a performance penalty, and BBN is considering modifications to the ARPANET to minimize that penalty. Such modifications violate the internet concept of using "given" nets but appear to be inevitable and should be encouraged. I also favor experiments such as using the WB SATNET for "expressway" routing. (The WB SATNET is not sufficiently reliable at the moment to support any real service, but it should be adequate for experiments.) We can easily do experiments using source routing to evaluate the performance advantages, if any, in using the WB SATNET as an overlay to the ARPANET, but such an approach puts the smarts in the hosts rather than in the net where we as networkers would prefer to see them. To take full advantage of the overlay, the IMPs would have to be aware of the

possibility of routing through gateways to get to other IMPs. An IMP could easily encapsulate an ARPANET message in an Internet header, hand it to a gateway, and have it come popping back into the ARPANET addressed to a fake host at some other IMP, where the Internet header would be stripped, and the message would continue on its way as an ordinary ARPANET message. From a protocol layering point of view, a lower level would be drawing on the services of a higher level to deliver a packet. To make it work with the current Internet, the IMPs would have to have a priori knowledge of the potential extra-ARPANET paths and use source-routing to get around the Internet routing algorithm which would normally return a packet with an ARPANET address to the sending IMP instead of forwarding it to some other gateway.

2. I think we should explore the possibility of extending EGP to allow communication not only between mutually suspicious gateways but between such gateways and their adjacent similarly suspicious networks. I suspect that it would be necesary to use network-specific messages to exchange routing and capacity information since the task of coming up with a network-independent format would be very challenging. I don't see network-specific messages as a major problem since network-specific code would be needed in gateways to deal with the data in any case. Basically, the needs of the nets and the internet are similar. A gateway wants to find out about the connectivity that a network provides to other gateways on the same net. A network node wants to find out about paths that the Internet may provide to other network nodes. As things stand today, networks have the information the gateways need, but

the reverse is not true. Gateway routing information is not sufficiently detailed to allow a network node to determine whether or not the gateway could provide a path to some other network node. Since it is probably not reasonable to extend gateway routing information to provide the necessary detail, the node would have to probe to find out what connectivity existed via the Internet.

3. The discussion on congestion control strengthened my belief that virtual circuits have significant advantages relative to datagram nets in coping with heavy loads. Fairness considerations require that individual flows be identified and regulated. The required information can be gathered by watching datagram flows and building up the necessary state information in nodes, but the mechanisms to do so are rather complex, and the information will be noisy as a result of upstream routing decisions as well the natural fluctuations in the offered traffic relative to its mean value. Also, current datagram protocols lack any explicit mechanisms for communication between nets and users about flow rates. The "quench" message provides an indirect means of communication about a flow problem, but we have seen that its interpretation and usage have been controversial. A virtual circuit (vc) protocol such as ST can provide a reliable out-of-band mechanism for communication between the users and the net about user requirements for flow capacity and the network's ability to satisfy those requirements. The same mechanism can be used to adjust flows to changing network conditions. In addition, the flow control messages are seen and understood by nodes up stream of a congestion point at which they are generated. Consequently,

enforcement of flow limitations can take place at the periphery of the net thereby minimizing wastage of resources in forwarding packets that will later be dropped at a down-stream point of congestion. Of course, propagation delays make it necessary to change flow parameters relatively slowly, and momentary peak flows can cause congestion that will result in packet loss. For satisfactory performance, nodes must schedule flows on links with enough margin so that the probability of such loss is small. Even for speech, which is relatively tolerant of packet loss and for which retransmission is not necessary, a loss rate exceeding a few percent will result in user complaints.

One of the principal objections to vc protocols has been the need to retain state information in nodes, but it appears that that may be an asset rather than a liability. The other major objection has been the time required to set up a connection before data can flow and the consequent increased cost of a brief interaction. An ideal net would offer both datagram and virtual circuit services allowing the use of datagrams for short interactions and virtual circuits for sustained ones. The IP/ST system we are using for speech is just such a net. We use IP datagrams for initial call negotiation and for interaction between packet voice terminals and the conference access controller is setting up conferences. We use ST virtual circuits for the speech itself. We expect in the near future to try some data experiments over the WB net using TCP on ST virtual circuits and to compare performance relative to TCP with IP. We will design the experiments to be independent of SATNET streams which are normally associated with ST connections but not with IP datagram flows.

2.12 Thoughts and Comments on the DARPA Net Management Workshop

Keith Klemba
SRI International

This report documents some reflective thoughts and comments resulting from the exhilarating discussions and technical intensity of the DARPA net management workshop. It focuses on two topics discussed at the meeting which I feel are of importance to the DARPA program areas that SRI is involved in. The first topic deals with logical/generic addressing within the internetwork. Secondly, the issue of broadening the cost parameters considered in internet routing is discussed.

Logical/Generic Addressing

There are several possible ways to establish a 32 bit internetwork address. The process usually involves one or more levels of name to address mapping recursively applied until at last a full internetwork address is established. These addresses have in the past been represented either a one-to-one or many-to-one mapping to physical devices somewhere within the internet. It is the one-to-many type of address that logical or generic addressing deals with. Within this discussion, logical addressing is a technique whereby a single internetwork address may represent several different physical devices within the internet.

It is important to consider the design and implication of logical addressing within the internet at this time. At the network level (ARPANET, PRNET) design considerations are already being given towards the support of logical addressing. This development at the network level is best preceded by a similar design development at the internetwork level, since all network level addresses must be derivable from internetwork level addresses. While the significance of internet logical addressing may be clear to some, this discussion continues by means of example to stress the importance of developing logical addressing capabilities for the internet.

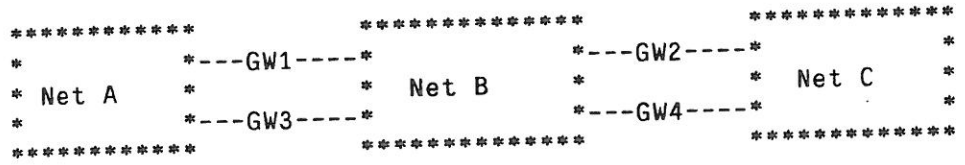


Figure 1

Figure 1 shows an internetwork configuration where three networks are connected via gateways.

Each of the gateways have unique internetwork addresses as well as unique network addresses. Let's suppose for the sake of this example that each of the networks (A,B, & C) can support logical (one-to-many) addressing at the network level. What could the gateways, as the heart of the internetwork system, do to make use of the logical addressing offered by the networks? One possibility is they recognize that there are multiple gateways serving between two networks and assign each of them a logical internetwork address in addition to their unique internetwork address as shown in figure 2.

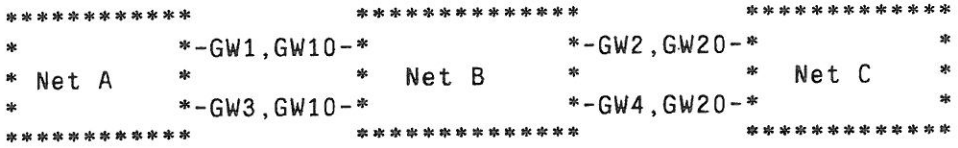


Figure 2

With this type of logical internetwork addressing, a host on net A can be advised either by EGP query or ICMP redirect to use GW10 to get to Nets B or C. Should it be necessary to use a specific gateway the unique internetwork address could still be used. For an even broader use of logical addressing all gateways as members of the internal gateway set could also assign themselves a common logical address such as GW0 as seen in figure 3.

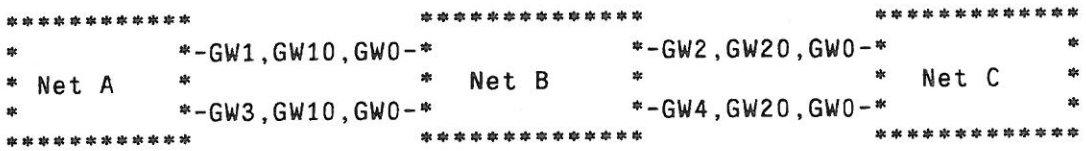


Figure 3

A host on net A that wishes to contact any internetwork gateway could do so by using the internetwork address GW0. Using logical addressing this broadly, is also known as functional or service addressing. There are common services offered by many hosts within the internetwork that could be coupled via use of internetwork logical addressing. One such service might be an internetwork name server.

The logical addresses in the above gateway example were chosen and assigned dynamically by the gateway level processes. It would be wrong to make these assignments at the network level and in my opinion wrong to make them static.

As distributed processing begins to characterize functions and services offered by the internetwork

the need for logical addressing capabilities will intensify. If a coherent design is developed by the internet, a much richer and complementary development of logical addressing support will be developed by the individual networks.

Internetwork Routing Cost Parameters

As with the maturity of most network routing algorithms, the internetwork routing is now broadening its metrics beyond minimum number of network hops. These metrics might include such things as delay, throughput, packet size, transport cost (media rates), priority, reliability, and datagram capabilities. At the meeting these topics were most interesting and will no doubt require a great deal of research and development in the near future to understand how to use these metrics.

It is clear that these discussions of increased internetwork routing parameters predicts the need for networks to provide increased performance and characteristic metrics to some element of the internetwork. Of particular interest at this point, as network developers are considering future network protocol and management development tasks, would be the establishment of a list of metrics and the procedure for collecting them. If the internet developers could produce this specification now and provide it to network developers, it would help assure that the metrics can be provided when someone has figured out how to use them. There is also the argument presented by measurement designers for the need to understand the variations and deviations that real world data all too often generates. Getting the metrics out in the networks will allow some preliminary review of data ranges and accuracies.