Inter-Office Memorandum

To	File	Date	September 13, 1981
From	L. Stewart	Location	Palo Alto
Subject	Voice vs. Data: Internet Issues	Organization	PARC/CSL

XEROX

Filed on: [Ivy]<Audio>Doc>VoiceVsData.press, .bravo

Abstract

Data communications and real-time communications, in particular, packet voice, put substantially different demands on a datagram based internet. Relevant facts about the requirements of voice communications are presented. The differing needs of real-time (voice) users and data users are discussed. Some possible ways of managing the operation of a combined voice and data internet are described. A proposal for incorporation of voice and other real-time applications into the OIS Communication Protocols is made.

Facts about voice and telephony

Medium-bandwidth

Telephone quality digital voice transmission can be achieved with rates as low as 8000 bits per second, but at this writing, the required techniques are computationally expensive. Intermediate bit rates are a possibility, but 64,000 bits per second represents the present telephone industry standard. For this reason, we restrict our attention to 64 Kbps telephone industry compatible voice. Such digital voice signals consist of sampling the analog voice waveform 8000 times per second and representing the sample amplitude as an 8 bit quantity.

Real-time

Voice communication from human to human (telephony) is a real time communications problem. The perceived delay must be fairly small and constant. Tolerable delays are generally below 100 milliseconds. [Notes on the Network].

TASI advantage

While a conversation between people is usually full duplex (both people *can* talk at once), usually only one participant at a time is speaking. In addition, when a person is speaking, there are often gaps between words and sentences. On the other hand, both participants occasionally speak at once. Over conversations in general, something like 47% of the full-duplex channel is used.

The laws of large numbers apply to these statistics. Useful data points derive from the telephone industry use of *Time Assigned Speech Interpolation* (TASI), in which a certain number of trunk circuits (e.g. transoceanic cable circuits) are overcommited. If 24 full duplex trunks are available, usually 36 conversations can be supported, for a ratio of 1.5. If 150 circuits are available, 300 conversations can usually be supported, for a ratio of 2.0. [Notes on the Network, BSTJ] These statistical effects are usually referred to as the *TASI advantage*.

Error-tolerant

To a certain extent, the human ear is tolerant of distortion in speech. For digital speech this means that, to a certain extent, the ear is tolerant of errors in the digital representation of speech.

Consistant service

Once set up, a voice connection should maintain an adequate quality. In the presence of network overloading it is better to reject (block) connection attempts altogether than to offer poor quality to the new caller. A corollary is that it is certainly better to block new calls than to degrade old ones.

Other possible real-time applications

Connection to non-flow controlled data circuits

Suppose a medium bandwidth (9600 bits per second to 56,000 bits per second) asynchronous (start-stop) serial line with no flow control is connected to an OIS gateway. If the internet side of the gateway cannot keep up, the gateway buffers will eventually overflow and data will be lost. This example exists today with the Research Internet Data Line Scanner (DLS); if an internet client cannot keep up with the speed of the DLS line, data is lost as the DLS buffers overflow.

This example is not strictly real-time. There are no particular *delay* requirements, but there is a *bandwidth* requirement. As a somewhat contrived example, suppose an internet is used to link *two* such non-flow controlled circuits of the same speed. If the input line is full, the delay introduced by the internet by buffering and transit delay can only increase. Once capacity on the outgoing circuit is left idle, the time lost can never be made up.

Connection to slow speed printers

It might be desired to transmit the bit-map for a raster printer in "real-time" through an internet, with some finite (less than full page) buffering at the exit from the internet.

The general idea of a real-time protocol

Suppose there is a producer of data for the real-time application that delivers data at a constant rate. The data is collected at the originating end until a full packet is accumulated. The packets are sent (at a constant rate) to the receiver, where the data is doled out (at a constant rate) to the consumer of data. Some amount of data, perhaps less than a packet's worth, perhaps more, is buffered at the receiver to smooth out jitter in the arrival of packets.

Naturally there must be adequate *average bandwidth* to support the application. There must also be sufficiently low variation in the rate at which packets arrive at the receiver so that the receiver's buffer never becomes empty.

For a printer application, an empty buffer might mean a missed scan line. For a voice application, an empty buffer might mean a momentary hiccup in the conversation. Probably neither case is absolutely catastrophic, but such hiccups must not occur at more than some acceptable rate.

To the extent that the types and numbers of errors in the internet are acceptable to the application, a real-time protocol does not need acknowlegements or retransmissions.

A voice protocol, in order to benefit from the TASI advantage, might detect periods of silence and utilize reduced bandwidth while silence is present.

The needs of real-time vs. data users

Some general principles:

1) Data users must make progress.

2) Voice users must get the bandwidth they need, or none. It is better to block a phone call than to offer a bad connection. (Similar principles hold for other real-time applications.)

To illustrate the operation of the first principle, we offer two examples, the Ethernet and point to point links between gateways. On the Ethernet, everyone is equal. When the load is low, everyone gets the bandwidth they need and the issue is moot. When the offered load exceeds the bandwidth of the net, the contending users share equally [Hupp & Shoch]. (In fact, the contending users share the actual bandwidth in proportion to the rate at which they become ready.) For the point to point line case, the current gateway program allocates the line first come first served, and maintains a queue of packets to transmit on the line. If the queue exceeds a certain limit, packets are dropped. By symmetry, everyone's packets are dropped with equal probability. (Actually, gateways promote small packets; by generating great numbers of small packets, a client could get 100% of a phone line, locking out other users. I think this is a bug.)

The effect of all this from the standpoint of a data-only internet, is that even when the communications capacity is greatly overcommitted, all users get at least some of it, thus all users make forward progress (if you wait long enough, your file transfer *will* finish).

Real-time communications (including voice) are fundamentally different. Once a connection is set up, it should get the bandwidth it needs. It is better to refuse service altogether than to consume internet resources providing useless service. Consider what would happen if an "equal-sharing" network were slowly loaded with telephone calls. As the first users pile on, everything works fine; there is enough capacity for all. At some point, the demand exceeds the supply and the sharing property of the network allocates the available bandwidth equally to all contending users. *All* the telephone calls fail at once! It would have been better to refuse service to the "last-straw" phone call, thus limiting the outrage.

These matters can be interpreted as optimizing an objective function. The objective function for data users might be the sum of the logarithms of the bandwidths per user: more bandwidth is better, the channel is shared equally, and getting zero bandwidth is infinitely bad. The objective function for real-time users might be the sum of step functions with the various jumps at the required bandwidths for the various users: more than a certain amount is ok, less than that amount produces nothing. The maximum of this function is achieved by allocating the requested bandwidth to each user until capacity is reached, other users get nothing (but may try again later). It is not clear how to combine voice and data users within this model without adding information on the relative worth of data and voice.

There is an interesting analogy between data vs. voice users of communications and time sharing vs. personal computers. The capacity of a time shared computer is allocated equally (usually) among contending users, the capacity of a collection of personal computers is allocated in "sufficient size" chunks up to the limit of the number of computers and none thereafter. In one case the advantage of adding capacity is that more people can work, in the other, everyone can still work, but their work gets done faster. It is possible to "successfully" overload a timesharing computer by piling on users. The same is not true of a collection of personal computers.

Traffic Engineering

In the telephone industry, there is the notion of *probability of blocking*. Given a certain number of physical trunks between A and B, and certain statistics of the numbers and durations of calls placed, there will be a certain probability that all the trunks will be busy when a call arrives. Traffic engineering is the business of providing enough trunks so that the probability of blocking is acceptably low, subject to economic constraints. (Typically, users are charged more during periods of high load than at other times. This tends to even out the loading and raise the *average* utilization of the trunks.)

So far in our construction of internets, the notion of traffic engineering is one of persuading an organization to invest in capacity when their data communications become *too slow* rather than on any objective grounds.

The advantage of combined real-time (voice) and data networks

In a network, a single link is more efficient than two half-speed links. Separate links suffer from two disadvantages: one link can be overcommitted while the other has idle capacity, and, in a datagram network, the separate links will have higher delays simply because it takes longer to transmit a packet over a slow line than over a fast one. (But remember that separate links may be more reliable than a single link. One of them can go down without breaking communications altogether.) Similar advantages accrue to the use of a single network for both voice (real-time) traffic and data traffic. Both voice and data networks are (should be) engineered to handle the peak loads expected. If the voice load peaks at different times than the data load, then both kinds of traffic can use the same network capacity at different times. [Refs somewhere]

Problems

Our existing networks cannot reliably handle real-time traffic.

Our existing networks have only rudimentary notions of congestion control.

Our existing routing cannot handle class-of-service notions.

Proposals

How can the differing needs of voice users and data users be reconciled? In general, the system must recognize that the classes of users have differing needs and apply different "objective functions".

Basic Proposals

Class-ofService. Stray from the ideologically pure notion of a stateless datagram network and build a system that understands some semantics of the kinds of traffic using it. We have already departed from purity by recognizing "interactive traffic" and promoting small packets to the head of queues. Legitimizing these activities will require a *class-of-service* field in our internet packets.

Employ traffic engineering. In our present datagram-only internet, we have escaped with only rudimentary traffic engineering because we had only one class of users. With the addition of voice traffic and with larger internets in general, we will have to keep loose track of "blocking probability", line utilization, and user populations and add capacity as appropriate.

Mechanisms

Load Control. At least for real-time applications, users should be turned away once the load on a network or link has reached capacity. The same information used on a minute by minute basis to handle loading can be used in the longer term to guide traffic engineering.

Hints. Although routers, gateways, and other load control points must keep track of who is using how much bandwidth for what, they can do so in a nearly stateless fashion by using hints. We want the advantages of centralized control without the reliability problems. The same bandwidth and delay requirements that cause real-time or voice packets to pass fairly often permit the "state" information in routers to time-out rapidly. Bad information will not persist long enough to disturb the internet.

Counterproposal

Dan Swinehart has suggested that sufficient traffic engineering may remove the need for load control. Most of our networks and systems behave quite well until just below overload. At the overload point, there is a sharp "knee" and above it our systems behave quite badly. (On the Ethernet, the "badly" is in terms of delay; network utilization remains high.) Suppose we kept track of average and peak loading on networks and links, and used the information to keep capacity ahead of demand. If we arrange that enough capacity is in the right places by the time it is needed, we might be able to hold the number and durations of overload periods to an acceptable level with no minute by minute load control.

Anti-counterproposal arguments

The required degree of success at traffic engineering is too hard to achieve.

We (and our customers) don't have the money to obtain endless network capacity, we will almost always be operating near overload in order to be economically competitive. (It is not that Ethernet cable is expensive; routers and long haul circuits are expensive.)

Internet traffic is too bursty to handle on a statistical basis. While I do think we should "think big" and plan on linking 10 Mbit Ethernets with 1.5 Mbit links, we already have *individual machines* which are capable of overloading such links. The laws of large numbers are not very effective for small numbers of users.

Class of Service proposal for OISCP

The class of service for a packet should provide an indication of in which class the packet belongs plus whatever class-specific information seems useful. The purpose of class-of-service is to give hints to the internet to aid it in routing the packet. More and better quality information will permit the internet to do a better job of meeting everyones needs.

There are at least five classes of traffic: ordinary datagrams (e.g. packet exchange protocol), file transfers (e.g. Sequenced Packet Protocol with large packets), interactive traffic (e.g. SPP with small packets), real-time traffic, and voice traffic. The general idea for class of service is that it indicate in which group a packet belongs and that it provide an indication of *how much* traffic is involved. (Is this packet the only one or are more expected in the near future?)

Proposal

A *Class-of-service* field wide enough to encode the classes of traffic mentioned, with room for expansion. A *how-much* field, to indicate, variously, that the given packet is part of a stream requiring so much bandwidth, or that (loosely) so much traffic is expected in the near future. The how-much field need not be very precise but it should identify the exact requirements of heavily used real-time applications.

> Ed Taft has suggested that a single bit in the present Transport Control field may be enough. Protocols using this bit would place bulkier class information in the data portion of a packet. No present software would need to be changed and new routers could be taught about the new bit. One drawback of this approach is that real-time users are precluded from using any of the existing protocols -- which might otherwaise make good sense.

Semantics

Ordinary packets. The how-much field could be used to indicate whether more packets are expected.

File Transfers. File transfers are not usually delay sensitive, but the sooner they are completed, the happier the clients. Route along the path with maximum excess bandwidth. The how-much field

might indicate the rough amount of data being transferred, so that the routers can make more intelligent decisions.

Interactive traffic. Interactive traffic (keystrokes etc.) is highly delay sensitive, low bandwidth, and intermittent. Route for minimum delay.

Real-Time traffic. Real-time traffic requires consistant delay below some maximum, plus guaranteed bandwidth. It is better to refuse such traffic that to offer less than the requested performance. The how-much field might indicate the required bandwidth. If the best achieveable delay is too long, the end users will find out after the first few packets.

Voice. Voice is not quite the same as "real-time". In order to obtain the TASI advantage, silence detection must be used. When silence detection is used, the voice stream becomes intermittent. For voice messages, the duty cycle might be quite high. For conversations, the duty cycle is somewhat below 50%.

Examples

Managing the bandwidth of a point to point line

Consider the case of two 10 Mbit Ethernets connected by a point-to-point 1.5 Mbit link. There is plenty of bandwidth around, but it is not infinite. A pair of routers connected by a 1.5 MBit line would have a parameter indicating that up to 1 MBit of line capacity may be used for voice (or other real-time traffic), with the remainder reserved for data. When there is less than 1 Mbit of real-time traffic flowing, the idle capacity can be used for data datagrams: (and the data queue empties faster), but when there is real-time traffic around, it gets reserved capacity. The routers keep an eye on packets coming in. Suppose the router sees a real-time, how-much=64 Kbit packet for a new source-destination pair. The router takes this as a hint that a new "stream" is being set up and makes a table entry "reserving" capacity for the connection. By using the how-much field together with the packet length, the router can predict when the next packet of the connection is expected. The table entry can be deleted (timed-out) if the next packet doesn't show up. (Thus there is no "stream setup" protocol, it is all done with hints.) When it happens that the n-th+1 apparent stream shows up, the router drops the packet and sends an error reply "no capacity now".

Now consider the case of "voice" traffic rather than just "real-time". A typical phone call uses each half-duplex path slightly under 50% of the time. A voice connection would send 50 160-data-byte packets per second while talking and would also send small packets at a lesser rate during silence in order to let the routers know (via the hint mechanism) that the 'connection' was still there. The router could actually get away with allowing, say, 20 'connections' over the 1 Mbit of capacity rather than only 16. Only for brief periods would the offered load from the 20 conversations exceed 1 Mbit. When that happens, the router could intrude momentarily into the "data" bandwidth.

Managing the bandwidth of an Ethernet

Consider the use of an Ethernet for telephones. So long as the total offered load is below the "knee" in the delay curve, the Ethernet works very well. Much above the knee, its performance may not be adequate for voice. The exact position of the knee is dependent on the distribution of packet sizes and on the average number of stations contending for the channel but it is in the 50% to 80% area for voice packets.

If too many people attempt to make calls at the same time, the Ethernet delays would grow rapidly, disrupting service for all. One solution is to register calls with a server -- callers would not get dial-tone if the Ethernet could not handle their call. Another solution is to monitor the general levels of Ethernet traffic and to split the network into two parts (adding capacity) well before the loading reaches dangerous levels. (This is just a localized version of the counterproposal described above. Its successful application might depend on separate Ethernets for voice and for data.)

More complex is the problem of using an Ethernet as a transit network in an internet. While a telephone server might register calls and perform load control on a local basis, who could take the responsibility for internet traffic? One approach might have the routers (perhaps using special hardware), watch *every* packet on the Ethernet and keep track, by hints, of the traffic levels. Transit connections could be blocked before entering a congested region.

Internet Issues

Routing

These hint schemes have the flavor of fixed routing. Once a call is set up along a particular path, the path is hard to alter. There is nothing wrong with this! Think big: there is a *lot* of traffic in our hypothetical internet. That particular connections are not rapidly rerouted is fine, load-sharing can be done by routing some connections one way and some another. If a particular connection's bandwidth requirement cannot be met without load sharing, the internet is probably operating too close to overload. Our present routing mechanisms can not deal effectively with these concepts, but then, they cannot, at present, handle excess bandwidth or delay either.

Accounting

The same hint based measurement machinery that is used to perform load control can also handle accounting.

Traffic Engineering

The load information can be used to prepare summaries of line and net utilization for traffic engineers. The information might be detailed enough to identify new candidate networks in the internet for direct connection.