IMPLEMENTING REAL-TIME COLLABORATIVE INTERNET INTERFACES

Larry McGourty and Kevin McGourty
McGourty Associates, LLC.
1229 Vine Street
Paso Robles, CA 93446
(805) 239-4834/4836 fax
larry@mcgourty.com

There is tremendous interest in using the Internet as a real-time collaborative interface. However, because of limitations in the current infrastructure and in the response time of the Internet, not all applications are suitable candidates for Internet control. Timeliness of exchange of data is the determining factor in the suitability of using IP as means of implementing collaborative interfaces. It is possible to define which applications are and are not suitable for real-time collaboration depending upon the nature of how a system changes its overall state in response to data exchanges between collaborating sub processes.

COLLABORATIVE APPLICATIONS.

Imagine a live Internet community where musicians of all levels and interest can plug into the Internet and jam live with anyone in the world, at any time, with any type of instrument including voice, in a simple and natural way for personal entertainment, expression, and learning. (Original vision of StickyJam.com, Inc, a music recreation site)

Reality or Fantasy?

What are "Collaborative Interfaces"? Very simply, collaborative interfaces are data exchanges between two or more linked and concurrent processes running in a single application. Interfaces are collaborative in applications where the output state (the collaborative effort) is driven by a process where control is shared between at least two concurrent and interacting sub processes (the collaborators). The output state is thus not just a composite of the individual sub processes that drive the application. For true collaboration, the sub processes must also be able to interact which each other in such a manner that the current state of any individual sub process is also dependent upon the current state of the other collaborating processes.

The state of each sub process in turn is controlled by its own set of local state variables, which are continuously changing in reaction to the current state of the other collaborating processes. The better the communication path, and the more unconstrained and large the set of local state variables at each process, the richer and more unpredictable the final output state of the combined process. Conversely, the poorer the communication path, and the more constrained and small the set of local state variables at each process, the simpler and the more predictable the final output state of the collaborative process. This is the difference between true collaborative effort and simple feedback control.

We are focusing only upon the communication path of the collaboration process. To support collaborative processes mutually interacting in real time, communication between the processes must encapsulate both information and time. Ideally, each collaborating process has complete and instantaneous access to information about the states of all the collaborating processes as they are changing.

The problem is however, to implement a suitable communication path with sufficient bandwidth (complexity) and synchronicity (timeliness) to support a complex set of state variables at the local processes. This is the basis of the tradeoff off between a Switched Network and a Packet Network. The switched network creates a quick response direct link, where bandwidth is optimized for maximum timeliness of delivery at the expense of the amount of information (payload). In contrast, the Packet Network is less responsive time-wise because the link is less direct and bandwidth is optimized for maximum payload delivery at the expense of timeliness.

The applications classes, and why time is of the essence.

Even the slowest communication link, given sufficient time can receive any arbitrary amount of data we care to wait for. Therefore timeliness of the communication path is the quality which best determines which class an application falls into. The most severe case is when time itself has intrinsic meaning and contributes to the output of the process, such as music. The least severe case is where timeliness of delivery has no intrinsic meaning and has no contribution to the final output of the process, such as online data searching – content is neither gained nor lost depending upon the rate and sequence of data transmission.

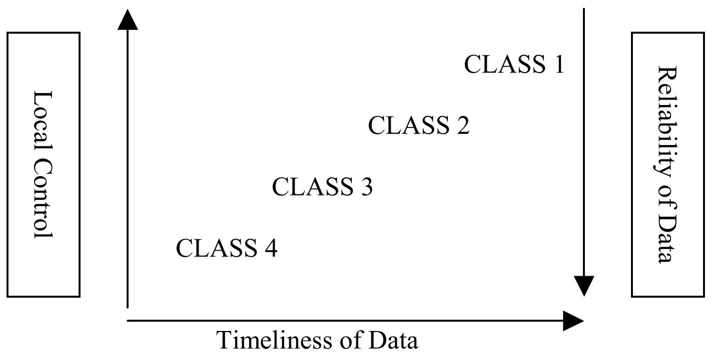

Figure 1. The relationships between reliability, local control and timeliness and the four application classes.

Somewhat arbitrarily we can split the applications into four classes, but in reality, the distinctions between any two classes can blur depending upon the application.

1. Interactive and simultaneous: (Isochronous and Bi-directional) This is the class of applications where (i) it is difficult or impossible to stop or "wait" an individual process and (ii) complete state information of each of the local sub-processes must virtually coexist and be known simultaneously by all collaborating sub processes to maintain information validity. These are tightly coupled processes that require tight data coupling to operate correctly. They operate as coupled asynchronous state machines or like neural nets. The information exchange is generally isochronous data (information must maintain order and time continuity) and is simultaneously exchanged between all connected nodes. Even small delays in data transport between linked processes can result in asymmetries in processing and can be annoying enough to spoil the interaction. At any instant of time an outside observer can determine the exact state of the system.
2. Interactive and near simultaneous: (Isochronous and unilateral) This class of application still requires that the information be presented as streamed data (the data has time continuity and order) but a slight delay may occur between the sending and receiving nodes since the processes do not have to act on the information simultaneously- a process can be "waited" between synchronized state changes. These processes operate more akin to linked synchronous state machines, which change state only upon predetermined intervals, or at least upon deterministic state conditions. Only at the instant of the state change "event" can an outside observer determine the exact state of the system.
3. Interactive but not necessarily simultaneous: (Batched Isochronous and Asynchronous) This class of interactivity can allow for greater delays and still have acceptable performance. The information can be streaming data or "burst" data (information can be send in short burst and can be received out of order). State changes at individual nodes are not necessarily synchronous. An individual process can run independently until it needs knowledge of the system wide state, then "waited" until

resynchronized by an update event. Only at the instant of the update event can an observer determine the exact state of the system.

4. Non-interactive and batch control: (Asynchronous) Any batch application where a large amount of time (greater than a few seconds to several hours) is acceptable. The information can be sent in bursts, though the illusion of streamed data can be achieved through buffering. The content of a data exchange is completely deterministic and unaffected by the system wide state. At no point can an observer determine the exact state of the system, but the final outcome is predetermined.

Examination of various classes of applications.

Using our classification defined above, Figure 1. depicts the relationship between timeliness of transmission local control and reliability of data for the different classes of applications. Depending upon the application, the reliability of the data received may or may not be as important as its timeliness of receipt. In general, the less reliable the data, the more timely (and frequently updated) the data required, and the more local control needed. Classes 1 and 4 are antithetical cases.

Class 1 Applications. In the case of a pure class 1 application, it is important to note that data is very ephemeral, and cannot be cached, or pre-computed. It must describe the state of the system at any instant of time. State changes are continuous and instantaneous system wide. In the case of missing data at any particular process, the error effectively becomes part of the state of the system, and the rest of the system immediately reacts to whatever response is made. Locally the decision about what to do must be made immediately, the rest of the system cannot wait. In some cases a simple no change in state for missing data may be a valid reaction for the affected process. Other cases may require more local intervention.

Some examples of class 1 application would include:

Live music jamming and true interactive gaming, where the presence of time is intrinsic in the information, and where relatively small delays in the order of 25 ms[*] between interacting participants (nodes) can disrupt communications.

"Hard" time control systems like remote controlled automobiles and other remote controlled time & position critical applications. A familiar example is the LCD touch screen signature application. Though this application is usually local in nature, it demonstrates clearly why the information at the sending node (the touch screen sensors) and the receiving node (the LCD display) must be virtually simultaneous for the information to be of value for human hand-eye coordination to function correctly.

Class 2 Applications. The essential difference between class 1 and class 2 is the reduction of the required response time of the system to change, and setting a fixed system wide rate of change. In effect this limits the rate of state change and bounds the amount of change possible at each state change. Even though Class 2 applications exhibit to some extent isolation (hence uncertainty) between the individual processes running at each node, a Class 2 application will synchronize all nodes on a state-to-state change. Any individual node will be cognizant of the current state of the other of the nodes of the system only at the moment of the state change.

Even in the case of a pure class 1 application, it is highly unlikely that the local response to missing or incorrect data would be completely unbounded. Generally some sort of constraint would kick in, or reversion to autonomous control. It is the extent of the amount of autonomous control that further distinguishes the differences between class 1 and class 2 applications. By structuring the data exchange to constrain it and restricting the rate of exchange if effects, the amount of local autonomous control required can be significantly reduced. Control requirements can be further reduced by only allowing state changes at specific periods of time (to synchronize changes and to limit the number of state changes over a period of time), or to restrict interactions by transferring information between processes as discrete stand-alone

[*] Obtained in a conversation with MIT's Multi-Media Labs Eric Scheirer. Precise data on how much delay can be tolerated between musical interactions is unavailable, but generally 25 ms is thought to be the critical limit.

chunks (to quantize and structure the data in a way to reduce the permissible number of resultant states). We are implicitly trading off the need for complex local control by providing more accurate and less variable data.

Some examples of class 2 applications would include:

Voice over IP (VOIP) and teleconferencing. Slight delays between the transmission and reception do not destroy the interaction since in normal human communication, one waits her turn before speaking. However delays exceeding 200ms (Held 2000) become noticeable and disruptive.

Tangible and co-presence interfaces. Interactive interfaces which manipulate tangible objects in physical or simulated space with digital augmentation across networks. Arguably, some classes of tangible interfaces may belong in class 1.

Class 3 Applications. Class 3 applications allow even more decoupling between the individual processes, but at specific points the system will synchronize. This is more akin to branching and forking operations in parallel programming rather than discrete state changes in a state machine. The difference between Class 3 applications versus Class 2 applications is that any point in time an individual node may not have complete knowledge of the exact state of all the nodes of the system, but it is aware that they exist. Upon resynchronization a class 3 application will modify its next series of states (opposed to just the next state) based upon the update of the current states of the other nodes. The system is not sharing information in real time, but information is being created in response to received data. Even so, for that response to be meaningful and useful, some delivery and sequence constraints need to be placed upon the received information.

Some examples of class 3 applications would include:

Interactive presentation applications such as "live chat", white boards and PowerPoint presentations. Although often called "real-time collaboration", the industry misclassifies these applications as real-time. They really do not need to exchange data in real-time. Data just needs to be sent within a reasonable time between requests. It is less time critical then VOIP since the data does not need to be isochronous.

Accessing web presentations and databases where the user expects the requested information to be presented in a specific sequence, within a reasonable time, say less than 2 seconds before the interface becomes problematic.

Security monitoring and other "soft" real-time applications. Building environment control systems fall into this class. Information just needs to be predictably presented within a fixed period of time (just what that time is, is application dependent).

Class 4 Applications. In class 4 applications, other than triggering a request, the individual states of the collaborating processes do not influence the information exchange. The individual processes are not aware that any other exists in the system; there is complete de-coupling between the nodes. These applications are very deterministic in nature, and function like preprogrammed state machines. The content and order in which data is requested and sent will have no effect on the content of the next transmission. The information when received is fixed and complete, although not necessarily in the exact order sent. In the case of class 4 data, it does not matter at all when the data arrives (other than being annoying to the receiver – but that is a human interface issue). If data is missing, waiting for retransmission is an option.

An essential difference between Class 4 and the others above is how the system responds to data exchanges. For a class 4 application, the appropriate response to received data at an individual node must be completely encapsulated in the data. The sending process is not aware of the receiving process so it cannot direct the receiving process' response. The key issue is that the data must be fixed and complete since it will be batch executed in a specific fashion with no corrective or adaptive action (other than what corrective response which may or may not be encapsulated in the data itself, or preprogrammed into the application running at the node). This is the antithesis of collaboration.

Some examples of Class 4 applications are:

Batch data retrieval and system updating, status monitoring and system updates, slow information like news wire and data downloads like MP3 files.

On demand and broadcast video and audio. Although this can be classified as group 3 depending on the application requirements. A one-way non-interactive simulcast can handle a lot more delay between the original signal and the rendered signal. Typically the receiver requires no interaction. Adding click-on-ads and other interactions with the video stream as allowed in SMIL (Synchronous Multimedia Integration Language) reclassifies this application as group 3.

It is important not to confuse local control with local computing requirements. Class 4 applications like MP3 may require a lot of computing power to process the application. But, both the final output as well as all the individual states of the data transform process are known before any processing even begins. With so few options, very little or no local control is needed.

IMPLEMENTIING THE COLLABORATIVE INTERFACE USING INTERNET PROTOCOL

Using Internet Protocol as means of data exchange.

The problem with the Internet Protocol (IP) is that, like Ethernet, it is a connectionless technology and does not guarantee bandwidth. Unlike "pure virtual circuit" technologies like ATM (Asynchronous Transfer Mode) and Frame Relay, IP does not make hard allocations of resources. This provides much more efficient use of the available bandwidth. The Internet Protocol is based on the simple concept that datagrams with source and destination addresses can traverse a network of IP routers independently, without the help of their sender or receiver. This also allows IP to adapt more flexibly to applications with varying needs. However, it also leads to some unpredictability in service. The capacity to tolerate this unpredictability relates to the level of guarantee that the applications require.

The reason IP is flexible is because it doesn't provide many services. IP provides addressing and that enables the independence of each datagram. Since IP can fragment datagrams in routers and reassemble them at the receiver, this allows parts of the same datagram to traverse different network media and paths to the final destination. But IP does not provide reliable data delivery. Routers are allowed to discard IP datagrams en route, without notice to sender or receiver. IP relies on upper-level transport protocols to keep track of datagrams, and retransmit as necessary. These "reliability" mechanisms can only assure data delivery; neither IP nor its high-level protocols can ensure timely delivery or provide any guarantees about data throughput. IP provides what is called a "best effort" service. It makes no guarantees about when data will arrive, or how much it can deliver.

This limitation has not been a problem for traditional Internet applications like web, email, file transfer, and the like. But as discussed in this paper, class 1, 2 and 3 applications can have both high data throughput capacity (bandwidth) and low-latency requirements (some class 3 applications can tolerate more latency). So the question is how can the service level of the Internet Protocol and its infrastructure be increased, without adding reliability requirements and overhead that could break it? A number of promising technologies are under development by the IETF (Internet Engineering Task Force) to address the quality of service issue (see QoS below).

Effect of delays in the data exchange.

Modeling the delays in an IP link An appliance is a device that performs a special function. If we exclude devices like routers that implement the Internet itself, a simple Internet appliance uses the Internet as means of connection between the appliance device (what ever that may be) and another appliance device or a data server. A simple model for a data exchange timing between two devices is as follows:

$$\text{Time} = \mathbf{t}_{\text{conversion}} + \mathbf{t}_{\text{packetize}} + \mathbf{t}_{\text{transmit}} + t_{\text{depacketize}} + t_{\text{deconversion}}$$

Where $\mathbf{t}_{\text{conversion}}$ = Conversion time from native format to communication format

$\mathbf{t}_{\text{packetize}}$	= Time required to create the data packet
$\mathbf{t}_{\text{transmit}}$	= Time to transmit the data
$\mathbf{t}_{\text{depacketize}}$	= Time to un-package the data packet
$\mathbf{t}_{\text{deconversion}}$	= Conversion time from communication format to native format

In the case of conversion times, even for analog input and output data, conversion times and de-conversion times are typically very fast, well under 1ms. Flash converters typically convert at Mega-sample per second speeds, and are capable of running at video rates.

The packetizing and de-packetizing processes are a function of the response time and switching granularity of the operating system. In the case of an interrupt driven RTOS (Real Time Operating System), the context switching times can be very quick, in the sub-millisecond range.

In any event, data conversion rates and RTOS performance are areas of active investigation, and are improving with each generation of electronics and software. There are so many options available today that with the possible exception of very demanding high-speed applications like streaming HDTV Video, the selection of an particular implementation option is primarily a cost issue, not a performance issue. Whatever solution is chosen, its response time and overall performance will be deterministic for a particular set of input state conditions.

The bulk of the transmission delays are incurred by routing delays, and there in lies a problem. At a minimum, transmission times are a function of physics. If the speed of light is 186,000 miles-per-sec., in 25 ms light will have traveled 4,672 miles. In the case of very demanding class 1 applications, there is even a physical distance limitation depending upon the responsiveness required of the system – but at least it is deterministic! It is the processing of the IP link itself that contributes most of the delay and probably all of the state uncertainty in the system.

<u>Effect of the delay in the IP link.</u> Table 1 shows typical delay components in transmission times. This model holds true for a data exchange in one direction only. However by definition, collaborative applications are bilateral in operation. In the case of bilateral communication, the actual time for a complete system state change, is doubled - if A is transmitting its state to B, A will not know how B's state is changing in response to A's transmission until B has received information about A's state, made its change, and then transmitted its response back to A. Because of this there will always be a lag time of at least one transmission delay in some part of the system. This especially affects class 1 applications since the underlying assumption is that information is present at all nodes simultaneously. This effectively means that the system node with the slowest communication link limits the speed of the overall system's response to change. Simply put, the slowest link determines the overall system performance.

Conversion	1ms
Packetize	10ms
Routing	10ms to 100ms
De-packetize	10ms
De-conversion	1ms
Total	32ms to 122ms

Table 1 Typical delay components in transmission times

Finally, a Packet Network is optimized for maximum payload delivery at the expense of timeliness. There is no real guarantee of transmission time, and bi-directional transmission times are not guaranteed to be symmetrical. This precludes high-speed, quickly changing applications as good candidates for control by the Internet (as currently implemented –see below).

Quality of Service

If the IP link ultimately determines the system performance, understanding how that link will perform is essential to determine what applications may work with an IP link. The performance of a network is measured by its Quality of Service (QoS). In the simplest sense, Quality of Service means providing consistent, predictable data delivery service. In other words, satisfying customer application requirements. It means providing a network that is transparent to its users. A useful and relevant analogy might be phone service that is so clear that one cannot tell the difference between a long-distance call and one to the office next-door or to a cell-phone. There are a number of characteristics that qualify QoS:

1. Minimizing delivery delay
2. Minimizing delay variations
3. Providing consistent data throughput capacity

The focus of QoS is providing predictable service (service defined as the share of available capacity) during periods of congestion. It is the periods of congestion that are the target of QoS. Being able to measure and report on service quality is also an important attribute of QoS solutions.

New services contemplated for the future

The main difference between a Circuit Switched network and a Packet Network is that the former utilizes a dedicated connection while the latter a shared connection. To improve QoS, in Packet Networks, a number of proposed control technologies have been introduced which at some level create a virtual dedicated connection. Some of the QoS technologies being drafted by the IETF (www.ietf.org/html.charters/rsvp-charter.html) include:

Resource Reservation Protocol (RSVP) enables non-QoS technologies such as IP to make QoS requests of the network. The protocol lets end stations request specific QoS levels for QoS enabled applications. For example, video conferencing applications, a request could include information that defines the maximum frame transmission rate, the maximum frame jitter, and the maximum end-to-end delay. When an end station makes an RSVP request for an application, each router situated between it and the source makes note of the QoS request and attempts to honor it. As the request travels through the routers to the source, it merges with requests from other end stations. If a router can't comply with the request, the requesting station receives an error message, the equivalent of a busy signal.

RSVP could lead to a perceived degradation of some network services. In a non-RSVP network, an application might run slowly or badly – but at least it will run. In an RSVP network, the application might not run at all if there's not enough bandwidth to support it. Similarly, if high-priority requests consume all allotted bandwidth, a router could dump non-prioritized flows. If you end up with an "arms race," where many non-bandwidth sensitive applications claim to be high priority anyway, you'll see no real benefits.

Another criticism of RSVP is its inability to scale. Because each and every router along a particular path must support it, using RSVP successfully over the public Internet could prove challenging. The requirement for each router to understand and support QoS requests may prove too costly to be fully supported.

Multiple Queues/ Differentiated services (Diffserv) is a router-based method for handling QoS. Routers that support this scheme identify flows that are tagged as high priority and move them to fast queues to expedite transmission. While still under debate as to it's use, the eight bit differential service or DS field (formerly named Type-of-Service (ToS) field) in an IP packet header is used to tag traffic with different service levels. Currently this field prioritizes traffic according to which end station initiated the flow, rather than the type itself, by telling network devices, that certain types of packets should receive a certain level of service.

Policy mechanisms The idea behind this concept is to make the network more intelligent by incorporating policy-based management. With this method, network managers could define policies that spell out which levels of service certain types of traffic should have and what particular routing paths they should use. For example, with voice traffic, a path with the minimal number of hops would reduce delay and maintain the highest possible quality. Policy-based management systems that major networking vendors are starting to support, give IT managers and service providers a way to take predefined policy and apply it across the board, rather than have to configure each network device with QoS support.

The Directory-Enabled Networks (DEN) initiative, a policy-based management method, allows network managers to create rules so they can provide services based on users applications. With DEN, directory services and networks are integrated, which mean that network elements and services are represented in a directory. This scheme allows a centralized policy for network services to be associated with particular end-user stations and applications. Because network devices and services are integrated with a directory, the network can be customized to provide a predictable level of service for users no matter where they are.

Ipv6. The next version of IP, includes inherent provisions for QoS. Ipv6 will allow applications to request different levels of service, and will guarantee these service levels even when a request goes over the WAN. This inherent QoS will be a big boost for real-time applications such as voice and video.

Design considerations

Rate of change. The speed of the overall application's system response to change is determined by the slowest link. This effectively sets the "granularity" of the system ability to change. In general, the coarser the granularity of the system, the more coarse and extreme the system changes at update; the finer the granularity, the finer and controlled the system changes at update. When analyzing the system, it is important to identify exactly where the slowest links exist, and optimize those links to achieve the best overall system performance.

Reliability and Protocol issues. The effect on system performance by a lost datagram is highly system dependent. If the granularity of the system is fairly fine and system state changes incremental in nature, there may be little consequence to occasionally losing a datagram. In any event it is important to design the system with sufficient robustness to adequately perform its application at the expected minimum level of QoS.

Since network topologies in use today (and coming in the future as discussed above) are already fairly reliable, UDP/IP, although considered "unreliable", in most cases is the best choice for collaborative interfaces. A UDP (User Datagram Protocol) datagram is likely to reach its destination anyway.

UDP has a number of attributes that recommend it over TCP (Transmission Control Protocol). Because of reduced overhead (at the expense of reliability) UDP is the higher performance protocol. UDP has the ability to broadcast system wide, unlike TCP/IP's virtual circuit which requires dedicated end-to-end connections. Finally, UDP deals only with one datagram at a time. Even if the underlying IP datagram is fragmented for transport, the destination will not receive any portion of the message until all of the fragments have been received and reassembled by IP (Hall 2000). For collaborative interfaces, reassembling a lost data packet may not be an option - the data will be "stale" anyway. A better option is to keep the data packets small (and perhaps partially redundant) and transmission times frequent, so nodes can re-sync on receipt of the next valid datagram.

Finally, it also possible to improve reliability by embedding controls so that the application can provide its own reliability services as required. These can vary from simple acknowledgement messages as part of the UPD datagram, much like TFTP (Trivial File Transfer Protocol) does, to -if the application can tolerate the delays- implementing some of the QoS technologies discussed above. In both cases reliability is improved by trading off faster and simpler data exchange for more complex data and slower overall transmission rates.

Class 1 Applications. Considering the current implementation state of the Internet today if data exchanges need to be very quick, it is reasonable to conclude that high-speed Class 1 applications – such as our cyber jam session proposition above, are difficult to implement using the Internet. Data cannot be buffered (other than what is needed by IP to reassemble the datagram if fragmented during transport), it must be acted upon as received. The imposed system state change rate (granularity) due to physical packets being routed, and the unpredictability of receipt times of the packets may make it difficult to properly co-ordinate the processes at multiple sites if there is any skew or disruption of the data stream. However, granularity is a relative concept. There are applications where the state changes occur slowly relative to the data transmission rate. These may actually be good candidates. For example, controlling heat, light and humidity in IP linked greenhouses. Reaction times to state changes are measures in tens of minutes. In these cases, data transmission times are negligible relative to meaningful state transition times.

Depending upon the granularity of the system, Class 1 systems appear to be more or less continuously adaptive. Local data storage and buffering requirements are minimal since the data is acted upon as received. Processing response time however must be fast enough to match the data reception rate. Processing cannot fall behind since this could cause a system wide skew.

Class 2 Applications. Class 2 applications work better with IP interfaces because of the ability to "wait" between state changes. Based upon the expected QoS, the data and the permissible states can be bounded to work properly within a specified performance range and state change rates can be relaxed to at least the granularity of the system. Using the example of the cyber-jam site, if one of the sites, A is essentially performing a pre-programmed "loop", a second site, B can improvise over the loop since the loop will go through predictable changes. A third site, C receiving the combined data would perceive the result as a co-present process, even though there is probably too much delay in the system making it impossible for A to react to what B is doing.

Class 2 applications are also adaptive, but are limited in response to a fixed system state change rate. Again, just how smooth the response to change appears, is a function of the rate of change of the underlying controlled processes relative to the granularity of the system. Local data storage and buffering only needs to be large enough to hold information for a single state change. However, processing capacity must be sufficient to execute the data within a state change.

Class 3 Applications. Class 3 applications are a good match for collaborative IP interfaces because of their ability to "wait" between system update events. The biggest impact is a limitation of the systems ability to quickly adapt to system changes. This is entirely dependent upon the rate of update events. Because of the potentially long periods between update events, to maintain overall system integrity, local autonomous control of the "system task" needs to be constrained depending upon the criticality of the system. For example, allowing a limited set of options such as falling back to a failsafe operation or shut down if a critical event must be responded to sooner than the next system update.

Class 3 applications are adaptive only to the extent of the rate of system update. The faster and more frequent the updates, the more class 3 applications approach class 2 performance. Local data storage requirements can be vary large depending upon the rate of update. In general, the slower the rate of update and the more intermediate states, the greater the local storage requirement. Processing capacity must be sufficient to execute the data between updates.

Class 4 Applications. Although Class 4 systems are generally unsuitable for collaborative interfaces, they may have a place as part of a system for things like program maintenance for updating supervisor information etc…

Class 4 applications are not and do not need to be adaptive. Local storage is strictly a local performance issue because data receipt and execution is not time dependent from a system point of view.

CONCLUSIONS

Collaborative interfaces are data exchanges between two or more linked and concurrent processes running in a single application. Timeliness of exchange of data is the determining factor in the suitability of using IP as means of implementing collaborative interfaces. Applications that use collaborative interfaces can be classified into four classes depending upon the sensitivity of the system to delays in the exchange of data. Class 1 applications are the most sensitive and class 4 the least sensitive.

The extent that IP can be use for interfacing collaborative processes is dependent upon how the rate of state change of the controlled processes is affected by the timeliness of the data exchange in the communication links. The more granularity in state changes an application exhibits, the more its suitability for using IP as a communication link. But, the greater the granularity of the application, the less responsive the application is to changes, and more local storage required.

Finally, the slowest communication link defines the minimum system skew between collaborating sub processes in the system. Transport delays may increase this value. At best, the fastest rate of change of state in a system cannot exceed the slowest reliable communication link in the system.

REFERENCES.

Hall, Eric A., Internet Core Protocols,
Sebastopol, CA: O'Reilly, 2000

Held, Gilbert., 2000. "Emerging Technologies: Reducing Voice Over IP Latency" Network Magazine, July 10, 2000

For information on the IETF Working Group addressing RSVP and an index to the Internet Drafts, go to **www.ietf.org/html.charters/rsvp-charter.html**.

For several informative reports on policy management and quality of services, go to The Burton Group's Web site **www.tbg.com**

___

Larry McGourty is president of McGourty Associates, LLC., a mechatronics design company in Paso Robles, CA. He has BS degrees from the University of Notre Dame and California Polytechnic State University, and has done post-graduate study at the University of California, Irvine, and Santa Clara University. McGourty Associates, LLC. was founded in 1982 to integrate the then new technology of the microprocessor with physical control interfaces. McGourty Associates continues its work in this area to embrace the Internet which figures to be as profound an influence in the future as the microprocessor was in the past.

Copies of the final presentation are available at **http://www.mcgourty.com/IAW_2001**