Delay Compensation Protocols for Synchronization of Multimedia Data Streams

K. Ravindran and Vivek Bansal

Abstract—**The paper focusses on temporal synchronization of various data streams in multimedia information (e.g., voice, video, graphics and text) exchanged between users over a high speed network. During delivery of such data, maintaining the required association between data units across various streams in real-time is necessary to sustain quality of service in the presence of data loss and/or delay in the network. Solving this synchronization problem requires *framing of data streams* whereby various points in the data streams deliverable simultaneously to a user are identified ("lip-sync"). In our solution approach, the temporal axis of an application is segmented into intervals, where each interval is a unit of synchronization and holds a data frame. Simultaneous data delivery involves delivering all data segments belonging to an interval within a certain real-time delay, as specifiable by the application. Based on the approach, the paper describes end-to-end transport protocols to compensate for the data skew that may arise due to data loss/delay. Simulation experiments confirm the viability of the protocols. Major uses of the protocols are in broadband ISDN's and metropolitan area networks.**

Index Terms— **Data framing, data slippage, network delay models, quality of transport service, skew recovery, temporal intervals.**

I. Introduction

A MULTIMEDIA application is structured as a set of user entities communicating with one another over a network (e.g., ethernet, fiber-optic networks), with one or more users generating information consisting of multiple data streams for consumption by other users. A transport system collects the multimedia data generated by source entities, moves the data through the network and delivers the data at destination entities for consumption. In digital TV for example, video and audio sent from the TV station are transported through, say, a fiber optic network and delivered to the display screen and sound speaker in TV receivers. In general, each media causes a distinctly perceivable effect at various entities that persists for a certain real-time duration, with the composite effects due to various media manifesting as progress of the application along the temporal axis. So an acceptable quality of multimedia data presentation requires maintaining the temporal association between the various data streams in real-time during their transport. In the TV example, the picture image seen on the screen and the voice heard on the speaker cause distinct effects in the human viewer for 30–40 ms. So perceiving a continuous change in the picture and voice requires the delivery of successive screen data and sound data at fixed real-time intervals of 30–40 ms. Thus unlike conventional data transport such as remote file transfers, a multimedia data transport has an additional dimension that associates the flow of real-time with the presentation of data streams to the application.

Multimedia synchronization involves segmenting each data stream into a timed sequence of application perceivable data units and identifying the various segments deliverable to the destination(s) at a given interval in real-time. These temporally related segments constitute a *data frame*. See[1] Fig. 1. The framing characteristics, viz., length of an interval and temporal association between data segments in a frame, typically depend on the persistence duration of data and the allowed sequences in which various data can be seen in the application (temporal ordering). The real-time data delivery constraints are stronger for continuous multimedia than for noncontinuous multimedia since the latter are less sensitive to real-time delays. For example, a videophone may allow video and audio segments to occur within a range of 70–150 ms [1], whereas a catalogue browsing service may allow audio, graphics and text to occur, say, within about 2 s. The paper focusses on providing a set of end-to-end transport protocols that solve the above "data skew" problem (also known as "lip-sync") in the presence of random delays and/or loss of data in the network, referred collectively as *delay compensation protocols*. The protocols are largely useful for real-time multimedia data transport where data is generated "live" by sources and is consumed "live" by destinations.

A solution approach in which synchronization may be achieved by virtue of enforcing tight guarantees on data delays in the network (e.g., no delay jitter allowed) is feasible [2]. In our view, this approach may reduce transport efficiency and flexibility because of the insulation of the application specific synchronization requirements among data streams from the transport protocols (but may provide simpler protocols). Accordingly, our paper provides an approach in which the transport system enforces synchronization in application specific manner. Using this approach, the paper describes new protocol techniques for synchronized data delivery. The techniques encapsulate recovery from the effects of data delay/loss based on the application level tolerance to variability in data delivery delays and to data misses. Exploiting the application characteristics in the protocol basically manifests in weakening the delay guarantees expected of the network,

Manuscript received June 1, 1992; revised December 1, 1992.

The authors are with the Department of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506.

IEEE Log Number 9209935.

[1] A data segment may be realized as a sequence of packets or ATM cells, depending on the underlying network.

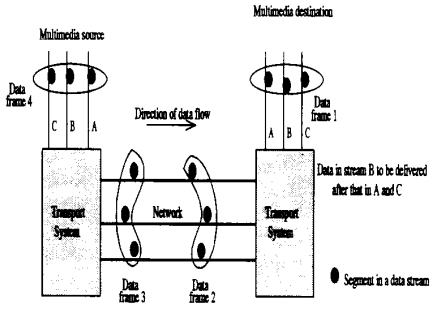


Fig. 1. Illustration of synchronization issues in multimedia data streams.

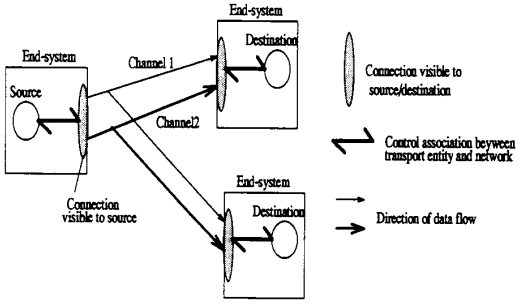


Fig. 2. Transport level components for multimedia applications.

which can result in better utilization of the network internal resources. For instance, a delay tolerant data stream allows the network to assign less priority for scheduling the transport of data segments through the network than in the case of a data stream with higher delay sensitivity. The paper also describes simulation studies on protocol performance to illustrate the feasibility of our approach.

The potential contribution of this paper is in providing a canonical model of compensating for network delays during multimedia synchronization. This research will be useful primarily in the evolving application technologies for broadband ISDN and metropolitan networks.

The paper is organized as follows: Section II describes a canonical model of multimedia data transport from an application perspective. The model is used to crystallize the issues in real-time data framing, viz., skew in data arrivals caused by delays and its effects on applications, in Section III. Sections IV and V present a protocol that achieves synchronization in the presence of random data delays in the network. Sections VI and VII present a simulation study on functional feasibility and performance of the protocol. Section VIII compares our model with other related works. Section IX summarizes the paper.

II. A Canonical Model Multimedia Transport

In an object-oriented view of a multimedia application, each media affects a distinct part of the "application level state," with the application progressing along the temporal axis by virtue of a state change due to each media occurring within a certain real-time duration. The data segments of various media that cause these state changes are said to be *temporally related*, and the real-time range in which these segments occur constitutes a *temporal interval*. Thus the temporal axis of the application indicates the passage of real-time (the source and destination entities maintain local clocks to perceive the flow of real-time). For example, the visual and aural cues in the human viewer due to video screen images and sound from voice speakers constitutes the "state" in a TV application, with each screen image and the voice data occurring within an interval of 33 ms corresponding to the persistence duration of visual/aural cues.

The data of each media in an application may be transported on a separate *communication channel* set up through the network. That the media transported on various channels belong to a single application is not known to the network. So the network treats these channels as independent data paths, each possibly with different characteristics. See Fig. 2. For example, a video telephone has two channels, viz., one for (compressed) video data at 2 mb/sec and the other for audio at 64 kb/sec. These channels may possibly take different paths through the network based on bandwidth availability. A *user level connection* enfolds one or more channels along which the various data streams in a multimedia information flow. The transport system synchronizes the data received on various channels of a connection for presentation to the user [3,4].

2.1. Quality of Transport Service

The synchronization requirement is specifiable in the form of a *quality of transport service* description (QOS_{transp}) on multimedia data. The parameters considered for QOS_{transp} in this paper relate to, among other things, allowed data latency between source and destination, data generation rate, data persistence duration and data delay tolerance, represented in some canonical form. The QOS_{transp} originates from the application based on inherent temporal relationships among various data segments. It is used by a synchronization protocol in the transport system architecture in two ways (see Fig. 3):

- mapping to QOS_{net} at the network interface below to support real-time movement of data segments through the network;
- generating a *data play* schedule over real-time for delivering the data segments arriving through the network at destinations.

With appropriate QOS_{net} specification, data may arrive at a destination before their delivery deadlines. In the catalogue browsing example, QOS_{transp} may specify the allowed skew between audio, graphics and text segments (viz., less than 2 s) which may be mapped to a QOS_{net} indicating the average and peak bandwidth required on channels carrying these data, the acceptable level of delay variation across various channels, and the acceptable end-to-end delay between source and destination. The QOS_{net} typically influences the allocation of resources internal to the network for implementation of the channel abstraction.

A data play schedule controls the delivery times of various data segments arriving from the network, typically by buffer-

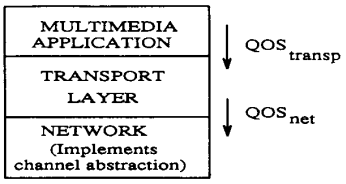


Fig. 3. Functional layers in a multimedia application.

ing and sequencing the data across the transport–application interface. For instance, a data segment that is scheduled for delivery at time T based on QOS_{transp} but arrives at time $T - k$ is buffered by the protocol for a duration of k before delivery. How temporal ordering parameters may be specified in QOS_{transp} to control the data play schedule is part of our current research at Kansas State University [5].

As can be seen, the design of a delay compensation protocol requires studies on what list of parameters need to be specified in QOS_{transp}, how QOS_{transp} may be mapped to a QOS_{net} and a set of procedures to support this mapping. The transport system strives to provide the desired level of QOS_{transp}. If QOS_{transp} cannot be met, the system generates an exception whereupon the user may reset the connection by initializing its channels or may abort the connection.

2.2. Transport System View of Network

We assume packet switching mode of data transport through network channels since the variable bit-rate and high speed data traffic generated by applications are elegantly supportable in this mode (however, our transport model can be used with other types of switching modes as well such as ATM cell switching). The issue relevant to this paper is how end-to-end inter-packet delay (or simply, delay) in the network affects the real-time transport of data streams[2].

2.2.1. Delay Specification of Network: The delay behavior desired of the network by the transport system, represented as $\overline{D}_Q$, may be captured with the following specification:

$$\mathrm{QOS}_{net} = \{(\overline{D}_{Q:\max}, \overline{D}_{Q:avg}), |\frac{d\overline{D}_Q}{dt}|_{\max}, G(\overline{D}_Q)\},$$

where the pair $(\overline{D}_{Q:\max}, \overline{D}_{Q:avg})$ indicating the maximum and average delay values specifies the degree of randomness in packet delays, and $|d\overline{D}_Q/dt|_{\max}$ indicates the maximum rate of change in delay over time. The parameter $G(\overline{D}_Q)$ indicates the level of guarantee on the delay behavior expected of the network. This is specified in the form of a probability with which the boundary conditions are expected to be enforced by the network, with $G(\overline{D}_Q) \rightarrow 1.0$ indicating a deterministic channel and $G(\overline{D}_Q) \rightarrow 0.0$ indicating a totally random channel. The delay attributes specified in QOS_{net} from the transport system may be used by the network to control its actual delay, represented as $\overline{D}$.

[2]Our analysis of end-to-end inter-packet delay in the network does not include the time gaps that may be introduced between succesive packets at the source.

In general, the actual delay behavior of a network is highly stochastic, and may be captured by nonlinear stochastic models of $\overline{D}$ to varying levels of accuracy [6]. From this viewpoint, the QOS_{net} specification indicates the boundary conditions on delay variability and the type of delay guarantees provided. However, QOS_{net} does not specify any particular form of delay model to be exhibited by the network nor does it specify the recovery actions needed should the network exceed the boundary conditions.[3]. The QOS_{net} is thus a canonical macro level description of the delay behavior desired of the network by the transport system.

In the strongest form, the transport system requires deterministic channels in which there is no variability in the delay. Requirements weaker than this result in networks providing probabilistic delays, and range between deterministic and probabilistic guarantees of the boundary parameters.

2.2.2. Demand on Network Internal Resources: The degree of randomness in delay, as given by $(\overline{D}_{Q:\max} - \overline{D}_{Q:avg})$, and the extent of change in average delay over time, as given by $|d\overline{D}_Q/dt|_{\max}$, specified for a channel represent the delay constraints imposed on the network by the transport protocol. The delay constraints have a direct relationship to the demand put on the network internal resources, such as allocation of packet buffers and assigning packet scheduling priority, in the channel implementation. Typically, the constraint of a fixed and deterministic delay (indicated by "0" values of delay parameters), as with high fidelity audio, represents the strongest form of constraint which puts a heavy demand on resources such as pre-allocation of packet buffers and assignment of higher packet priority. On the other hand, a large variability in delay (indicated by large values of delay parameters), as with catalogue browsing, represents a weaker constraint which puts less demand on resources such as buffer allocation upon packet arrival and assigning low packet priority.

Details of how various types of channels may be realized and the policies for network internal resource control are outside the scope of this paper. See [7] for such details.

2.3. Data Framing

Consider an N-channel connection to transport a set of data streams $\underline{X} = \{x_1, x_2, \ldots, x_N\}$ from one or more sources to one or more destinations, with stream x_r flowing on rth channel, where $r = 1, 2, \ldots, N$. Segmenting the temporal axis of the application into a sequence of intervals, the sources generate data in each interval along various channels while the destinations consume the data [8]. When the application is positioned at ith interval ($i = 1, 2, \ldots$), the data segments belonging to this interval, denoted as $\underline{X}^i = \{x_1^i, x_2^i, \ldots, x_N^i\}$, constitutes a data frame presentable at the user(s). The duration of ith interval is determined by the persistence durations of various x_r^i and the allowed sequences in which these x_r^i can be seen in the application. When the entire frame $\underline{X}^i$ is presented, the application progresses to the $(i + 1)$th interval along the temporal axis.

[3]The stochastic nature of the actual delay behavior in the network is too complicated to be concisely captured in QOS_{net}

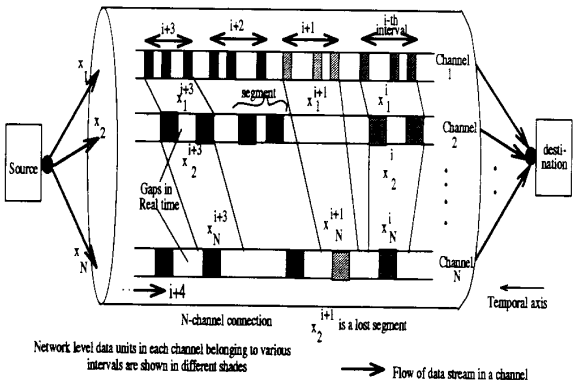


Fig. 4. Temporal segmentation of data for inter-channel framing.

The framing may occur at application specified points in the data streams, as indicated in QOS_{transp}. For example, 256 samples from a 8-Khz voice sampler corresponding to human hearing persistence of 1/32 s may constitute a segment. If a segment is delayed or lost, the voice system may continue by replacing the lost one by the immediately preceding segment without perceivable loss in quality. As another example, 480×640 samples corresponding to a picture in a 480×640 pixel digital TV may constitute a data segment. Thus a segment is a structured unit of information in a channel, realized as a sequence of packets, that is perceivable by the application through a media in real-time. A frame is a collection of segments carried on various channels in a temporal interval. See Fig. 4.

2.3.1. Flow of Time: All notions related to time in the application such as "simultaneity" in data delivery, "forward progress" in the application and "duration" of data exchange are expressed in terms of frame intervals. In other words, a frame interval represents the unit of real-time perceivable to the application. Thus a frame interval constitutes the granularity of synchronization ("coarse" or "fine"), with things such as length of the interval and whether the length is fixed or variable being application dependent (i.e., specified as part of QOS_{transp}). A timeout period for the frame interval may be agreed upon among transport entities during connection setup. The expiry of timeout period signals the progress of the application from the current interval to the next interval.

2.3.2. Labeling of Frames: Markers may be used to associate various points in the data streams with appropriate synchronization intervals [9]. Accordingly, each synchronization interval is marked with a unique sequence number. A marker thus acts as delimiter (or label) for a collection of data packets in an interval. The sequence numbers assigned across multiple data sources may be guaranteed to be unique by a systemwide algorithm such as assigning global clock values and prefixing unique source names to locally generated sequence numbers.

We have thus far described an application-oriented model of the transport system and data framing. Based on this model, we now discuss the issues involved in multimedia data delivery.

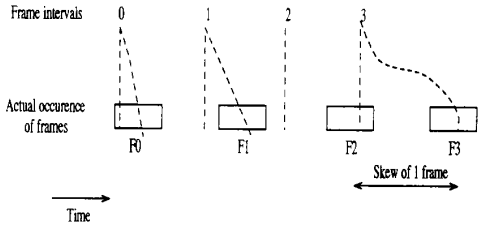


Fig. 5. Illustration of data skew phenomenon.

III. Data Skew Phenomenon

We assume that the underlying network provides FIFO delivery (i.e., packets arrive at a destination in the same sequence in which they were sent from a source), but does not recover from packet loss. Also in some cases, the network can only guarantee probabilistic bounds on the packet delay and packet loss rate specified in QOS_{net} with the delay and loss variability arising due to stochastic nature of the control mechanisms internal to the network [7]. With such a network, data skew can occur in ith interval of an application when data segments off by one or more intervals arrive in place of those belonging to ith interval (e.g., data in second channel being x_2^{i-l} instead of x_2^i, for some $l > 0$). See Fig. 5. In these cases, the end systems should resort to recovery as required by the application. We discuss below how the skew arises and how it affects data delivery.

3.1. Clock Synchronization

A destination may perceive data skews due to asynchrony of its local clock with respect to the clock at the source, which may arise due to network delays and/or drifts in the clocks. In the absence of synchronized clocks, the time interval at a source (say) may have drifted to a value much larger than that at the destination. So the latter may persistently miss the last packet in one or more segments of each frame even though these packets may well be on their way.

We assume that clocks are synchronized using some form of an asynchronous protocol between the transport level entities in the presence of network delays compounded by clock drifts [10]. Most clock synchronization protocols require the entities to asynchronously exchange their local clock values through the network and agree on a common clock value. These protocols use knowledge of the network delays in reaching agreement. For instance, a protocol based on the Internet Time Service Model requires the entities to receive their clock values from a central time server that maintains a highly stable and accurate clock and to correct the received clock values by offsetting the network delays [11]. Details of such protocols are outside the scope of this paper.

For clock synchronization protocols to function correctly, it is desirable that the network delay is deterministic, i.e., the degree of randomness in the delay is small and the average delay does not change significantly during execution of synchronization protocol. Accordingly, the transport protocol may create a deterministic channel with high loss and delay sensitivity to exchange clock control information.

With synchronized clocks, all transport entities perceive the flow of real-time identically with the aid of frame timeouts. And any data skew may be perceived only due to delay variability on data channels.

3.2. Data Slippage

A strong delivery requirement is that a frame presented on a connection in ith interval is always $\underline{X}^i$. Suppose an interval cannot be so completed, say because the data segment of a channel has not fully arrived yet while that of other channels have arrived in a timeout period for the frame. In this case, the interval is skipped without delivering the missing data segment and the temporal position is advanced to the next interval. We refer to this as a *data slip.*

The main reason for data slips is the differential delays experienced by packets in the frames across various channels. Suppose $\overline{D}|_{avg} + d_1$ and $\overline{D}|_{avg} + d_2$ are the actual delays experienced by the frames on two different channels of the connection. Then if the differential delay between these frames $|d_1 - d_2|$ is larger than the frame interval, a slip may occur between these channels.

3.2.1. Glitches in Data Delivery: A data slip manifests as the occurrence of a glitch during the presentation of data frames to the application. A glitch may disrupt data flow in the corresponding frame interval. The effects of a glitch such as how the glitch changes user level activities and what the persistence duration of the glitch is on users are specific to the application. In an application requiring video delivery for example, a glitch is observable as a blank image interspersed in a sequence of picture images on the screen. The effect of the blank image may persist in the minds of the human viewer for a few seconds.

3.2.2. Recovery Requirements: If the defaulting channel has weak delivery requirements, the data of other channels can be delivered within the limits of acceptable QOS_{transp}. Suppose in the TV example, a picture is lost. It is often not required (and also not feasible) to recover the lost picture. Either subsequent pictures may be displayed with the audio segment corresponding to the lost picture discarded or the audio segment may be delivered along with a re-display of the picture previous to the lost one [12]. In an example of scientific visualization that uses graphic images and text descriptions, the loss of an image may cause the end-system to re-request the missing image as part of a higher level protocol in the visualization system and then display the image and text. Thus in some applications, a segment loss in one channel may cause data slip in one or more other channels.

In general, if data slip cannot be tolerated, the transport system generates an exception which causes the connection to be aborted or reset. Given that an application can tolerate data slippage to some extent, one needs to specify how to handle the skewed segments. This may reduce the number of connection aborts.

3.3. Specifying Tolerance to Skew

We identify the transport attributes (or parameters), indicatable in QOS_{transp}, that characterize the skew tolerance in the application. These attributes are discussed below (see Fig. 6).

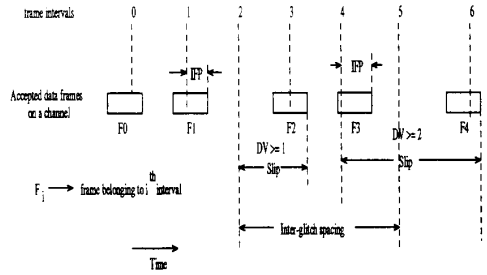


Fig. 6. Illustration of DV, IGS and IFP.

3.3.1. Divergence Vector: Given that an application is at ith interval, the constraint is how far apart (or divergent) the data segments can be separated along the temporal axis and be still acceptable to the application. This may be specified by a *divergence vector* (DV) $\{l_1^i, l_2^i, \ldots, l_N^i\}$ where l_j^i is the maximum temporal divergence allowed on jth channel, given in terms of the number of intervals $(0, 1, 2, \ldots)$. In other words, the range of acceptable data values are $[x_j^i, x_j^{i-l_j^i}]$ and hence data slip within this range can be tolerated. A vector with only $\underline{0}$'s indicates the strongest constraint while that with larger divergence values indicate weaker constraints. The choice of l_j^i may be based on loss/delay sensitivity of data on jth channel: higher the loss/delay sensitivity, lower the l_j^i and viceversa. In digital TV for example, DV can be {3,2} assuming that the user can tolerate up to three consecutive picture loss on the video channel and sound loss for a duration of 0.067 s on the audio channel.

3.3.2. Inter-glitch Spacing: A minimum acceptable spacing between two consecutive glitches, given in terms of the number of intervals $0, 1, 2, \ldots$, is also specified for each channel. The *inter-glitch spacing* (IGS) is related to the persistence level of a glitch in the application, i.e, how long the effects of a glitch linger on in the application. In the TV example, the human viewer can tolerate glitches occuring no more frequently than 1 in 300 pictures (say), i.e., IGS = 300. The IGS is specified as a vector with one entry for each channel in the connection.

3.3.3. Inter-frame Pause: An application can tolerate certain amount of pause in the flow of real-time when the application advances from the current ith temporal interval to the next $(i + 1)$th interval. The maximum length of the pause period measured in real-time between successive frame intervals in a channel is referred to as *inter-frame pause* (IFP). In the TV example, the IFP can be as high as 5 ms before the human viewer starts perceiving a chatter in the successive play of pictures on the screen. The concept of IFP can be illustrated with a "virtual clock" that indicates passage of time in the application after nullifying the effects of intervening pauses between frames.

Thus we attempt to characterize user tolerance to skewed data segments in terms of divergence vector, inter-glitch spacing and inter-frame pause, specifiable in the QOS_{transp}.

3.4. Cross-Channel Slips

In certain applications, complex inter-media relationships concerning data slips may also exist. For instance, simulataneous data slips in multiple channels may not be tolerated by the application even though the slips may individually be within the specified divergence limits, such as segment loss being acceptable in voice or text channels but not in both. In the example of scientific visualization, a loss in the image channel can affect the text channel. This may require capturing the interactions among various channels of the connection. These relationships can also be specified in QOS_{transp}, say, using AND, OR, and NOR connectives on the transport attributes DV, IGS, and IFP. Studying the effects of data skew in a variety of applications to ascertain the adequacy of these attributes in specifying the skew compensation requirements is a future research issue.

We now describe how the QOS_{transp} parameters may be used in the transport level protocol.

IV. Generating Delay Specifications from Transport QOS

The parameters specified in QOS_{transp} are used in two ways in the transport system:

1) to map the parameters into a specification of the delay behavior required of the network; such a specification will be encapsulated in QOS_{net} at connection setup;
2) to perform compensatory actions to handle slips that may occur during protocol execution; this is needed to maintain the connection at acceptable level of quality.

These functions are incorporated as distinct components in the transport protocol. This section describes 1) while Section V describes 2).

As discussed in Section 3.3, the skew tolerance information specified by the application at the transport layer interface is given by

$$\text{QOS}_{transp} = \{\text{DV}, \text{IGS}, \text{IFP}\}.$$

The transport protocol maps this information to the acceptable delay variations for each channel, specifiable in QOS_{net} during connection setup. We derive this mapping below.

Consider a data stream transported on two different channels X and Y of a connection, synchronized at ith frame interval.

4.1. Delay Rate

If packets in channel X experience a fixed delay and if δ is the additional delay for each packet in channel Y relative to that in channel X, then the skew in the arrival of $(i+1)$th frame on channel is given by $(N_{pkt} * (N_{pkt}+1) * \delta)/2$ where N_{pkt} is the number of packets in a frame. This skew should be within the limits of acceptable divergence between channels X and Y to avoid connection aborts, as shown by the relation $(N_{pkt} * (N_{pkt}+1) * \delta)/2 \leq \text{DV} * T$, where T is the frame interval. If g is the rate at which packets are generated at the source, then the rate of change of delay in channel Y that induces a differential delay of δ is given by $d\overline{D}/dt = g * \delta$. From these relations, the allowed values of the maximum acceptable rate of change in delay speciable in QOS_{net} may be given by

$$\left|\frac{d\overline{D}_Q}{dt}\right|_{\max} \leq \frac{2 * \text{DV} * T * g}{N_{pkt} * (N_{pkt}+1).} \tag{1}$$

In a video application for instance, DV = 3, T = $33.5 * 10^{-3}$, $N_{pkt} = 480$, $g = 14400$. So the QOS_{net} can indicate a value of $|d\overline{D}_Q/dt|_{\max} \leq 12.5 * 10^{-3}$. An average acceptable rate of change in delay $|d\overline{D}_Q/dt|_{avg}$ may also be specified, with a value less than $|d\overline{D}_Q/dt|_{\max}$.

4.2. Maximum Delay

Suppose the delay in the channels of a connection are randomly distributed in the range $[0,\alpha]$. Then[4] α indicates the maximum differential delay possible between channels X and Y. This delay should be within the divergence limits to avoid connection aborts, as indicated by the relation $N_{pkt} * \alpha \leq \text{DV} * T$. So the allowed values of the maximum acceptable delay speciable in QOS_{net} may be given by

$$\overline{D}_{Q:\max} \leq \frac{\text{DV} * T}{N_{pkt}}. \tag{2}$$

For the previous example of video application, $\overline{D}_{Q:\max} = 0.2$ ms.

4.3. Average Delay

Suppose β is the average of the delay, where $0 \leq \beta \leq \alpha$. The parameter $(\alpha - \beta)$ is an indication of how frequent the differential delay between channels X and Y reaches α, and hence the rate of slip occurrence. Assuming that the network delays are symmetrically distributed around β, we can assume a linear relationship between the average slip rate S and $(\alpha - \beta)$ in the form

$$S = S_{\max} - \frac{S_{\max}}{\alpha} * |(\alpha - 2 * \beta)|, \tag{3}$$

where $S_{\max}$ is the slip rate when α is set at $\beta/2$. To avoid connection aborts, it is necessary that

$$S \leq \frac{1}{\text{IGS} * T}. \tag{4}$$

Combining the relations (3) and (4), we obtain $|1-2*(\beta/\alpha)| \geq 1 - 1/(\text{IGS} * T * S_{\max})$. Suppose $S_{\max} = 2$ and IGS = 300 in the earlier example of video application. Then we have $|1 - 2 * (\beta/\alpha)| \geq 0.95$. For $S_{\max}$ 0.5 and 0.2, we have $|1 - 2 * (\beta/\alpha)| \geq 0.801$ and 0.503 respectively.

After choosing a value of α to be specified as $\overline{D}_{Q:\max}$, an appropriate value of β may be chosen satisfying the above relation for specification as $\overline{D}_{Q:avg}$ in QOS_{net}. For $\overline{D}_{Q:\max}$ = 0.2, $\overline{D}_{Q:avg}$ = 0.1801 (or) 0.0199 indicating that the delays are more accentuated toward $\overline{D}_{Q:\max}$ and toward 0.0 respectively.

[4] A more general analysis specifying a delay range $[\alpha_1, \alpha_2]$ where $\alpha_1, \alpha_2 \geq 0$ will basically be the same as our present analysis.

4.4. Some Observations on the Mapping

As can be seen, the mapping that projects the DV and IGS into $(\overline{D}_{Q:\max}, \overline{D}_{Q:avg})$ is to specify the desired degree of randomness in the delay behavior of the network, while the projection into $|d\overline{D}_Q/dt|_{\max}$ is to specify the degree of change in average delays in the network.

As mentioned in Section 2.4, the parameters $\overline{D}_{Q:\max}$, $\overline{D}_{Q:avg}$ and $|d\overline{D}_Q/dt|_{\max}$ specify the boundary conditions on the delay behavior of the network. However, it is quite likely that the conditions are often enforced probabilistically by the network, depending on G($\overline{D}_Q$) and how faithfully the network is able to enforce the delay constraints on the channels in a sustained manner. The situations where the exceeding of the boundary conditions causes unacceptable slips potentially lead to connection aborts by the protocol.

Given the probabilistic nature of the delay guarantees, data slips may occur now and then in various channels. With a conservative choice of the parameter values in QOS_{net} whereby the values are chosen to be well below the allowed limits (such as$\overline{D}_{Q:\max} = (0.5 * \text{DV} * T)/N_{pkt}$), the extent of data slippage may be less. With an optimistic choice of parameter values however, i.e, choosing values close to or above the allowed limits, the extent of slippage may be high. In any case, it is necessary for the transport protocol to recover from data slips, i.e , take compensatory actions in the presence of missing data segments.

V. Skew Compensation Mechanisms

We describe how the protocol uses the DV, IGS, and IFP to recover from data slips that may occur during execution.

Consider the ith interval. The normal case is the arrival of frame $\underline{X}^i$ completely—as indicated by $(i+1)^{\text{th}}$ frame beginning to arrive—before frame timeout. Suppose the timeout expires, indicating a data skew. The recovery for jth channel is as below (see Fig. 7).

5.1. Frame Interpolation

The following recovery is enabled only if the most recent data slip (if any) occurred farther than the IGS. This is ascertainable by keeping a count of the number segments delivered since the last slip.

$l^i_j = 0$: The connection is aborted since data slip cannot be tolerated (e.g., as in compressed video).

$l^i_j \geq 1$: The complete segment is not available.[5] So the most recent segment in the range $[i-1, i-l^i_j]$, available in a *replay buffer*, may replace the incomplete one for delivery as ith segment.

When intermedia realtionships exist, the frame interpolation involves an analysis of the channel interactions, as specifiable in QOS_{transp}, before replacing a segment. The segment level interpolation is a generalized mechanism for multimedia transport, though specific variants of this mechanism have been used elsewhere for packetized voice transport [13].

[5] A packet loss in the network may be treated as a loss of the segment to which the packet belongs.

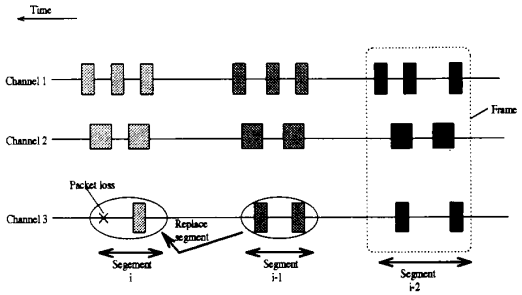


Fig. 7. Illustration of frame recovery and data slip.

5.2. Handling of Persistent Slippage

A frequent data slip on jth channel indicates persistent packet loss and/or delay in the path through which the channel is routed. In this case, the destination can feedback a request to the source to re-establish this channel with tighter bounds on the delay and loss rate.

Re-establishing a data channel amounts to modifying the bounds on the delay and/or loss rate specified in the QOS_{net} earlier for the channel. The glitch due to this activity may be more noticable to the application than that due to infrequent data slips. A failure to re-establish the channel forces the end systems to abort the channel. If the application mandates the presence of the channel to retain the entire connection, the channel abort results in aborting the connection. Otherwise, the application may continue with the remaining channels in the connection (graceful degradation) [3]. In video telephone for example, failure of the video channel may allow the conversation to continue with the audio.

5.3. Advancing to Next Temporal Interval

After so completing the ith interval, the temporal position is advanced to the $(i+1)$th interval by restarting the frame timeout, retaining the segments in the range $[i, i+1-l^i_j]_{\forall j=1,2,\ldots,N}$ in the replay buffer and delivering the data. Any subsequently arriving packet which belongs to ith interval is retained for a possible playout in the $(i+1)$th interval if the frame for the latter interval does not arrive.

For noncontinuous media applications (e.g., graphics and text in catalogue browsing), the next frame timeout may not be known *a priori*. In such cases, the arrival of the next frame starts off the $(i+1)$th interval.

5.4. Control of Frame Timeouts

The inter-frame pause IFP specified in QOS_{transp} can be used to compensate for the clock drift between the source and destination by controlling the frame timeout at the destination. If the network delay decreases, packets in a frame arrive before the frame interval times out. In this case, the play out period specified in QOS_{transp} can be met. If, on the other hand, the network delay increases, the destination frame interval times out before all packets in that frame arrive. In this case, the probability that subsequent frames would also miss the frame time intervals increases. So delaying the

starting point for the next frame timeout by an appropriate compensatory period increases probability that subsequent frames are received within their frame intervals. The shift in the starting point of frame intervals can be based on some policy, subject to the limits imposed by IFP. Two possible policies that we suggest are below.

Policy 1: After observing K slips, shift the starting point of frame intervals by an amount equal to the average of these slip periods. The average value should be less than the IFP.

Policy 2: An extra margin is added to the average shift generated in Policy 1. This combined value of the shift should be less than the IFP.

One may intuitively observe that Policy 2 may result in less number of slips than Policy 1 because of the additional margin introduced in the timer shift. However, Policy 2 may cause more delay in frame playouts at destinations.

5.5. Effects of Network Characteristics on Transport Level Recovery

With deterministic guarantees on channel delays and appropriate choice of maximum and average delay values, data slippage may never cause the transport connection to be aborted. With probabilistic guarantees however, connection aborts may occur because the possibility of slips beyond the limits specified by DV and IGS still exists. In cases where the network does not accept the QOS_{net} specified at connection setup, the transport system may reconfigure by specifying a QOS_{net} value for degraded service if possible. This minimizes the possibility of connection aborts.

We now describe simulation studies to evaluate the protocol described thus far in various sections.

VI. Evaluation Model for Transport Protocol

This section discusses i) how the functional feasibility of the transport model in the presence of random network delays and losses can be studied, and ii) how the effectiveness of the protocol mechanisms can be analyzed. An actual building of the proposed transport system itself is planned as future work.

6.1. Protocol Execution Model

Since each of the data streams in a multimedia information is transportable independently, the end system protocol handling the streams is decomposable into separate modules. In each temporal interval, these modules communicate with one another to execute the synchronization protocol. See Fig. 8.

At a source, the **media decomposer** module distributes each data stream to a **media dispatcher** module that sends the data over the corresponding channel.

The **media decomposer** also generates the control information for transport over a separate synchronization channel. At a destination, the **media receiver** modules field the data streams to the **media composer** module. The latter composes the data streams for delivery to the destination as per the delay constraints carried in the synchronization channel.[6]

[6]As may be seen, there is potential for parallelism in copying of data streams between the transport/application buffers and the network channels and in evaluation of data delay constraints.

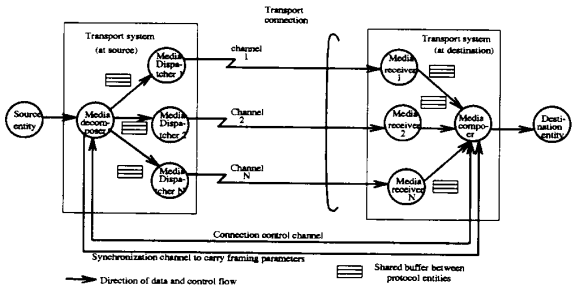


Fig. 8. Execution model of multimedia transport protocol.

6.2. Simulation of Protocol Execution

The execution model of the protocol is embedded into various simulation nodes, with each node implementing a functional module. The inter-process communication among various modules and network activities are realized as processing of discrete events flowing across simulation nodes. Packet delay in the network is simulated by increasing the simulation time stamp of a scheduled packet arrival at **media receivers** by the delay value. A packet loss is simulated by not queueing the packet arrival event at **media receivers**.

The **source** module simulates the generation of data into the transport buffer at the rates specified for each media. The **media decomposer** segments data in the buffer and enqueues events carrying these segments to **media dispatchers**. The segments are affixed with segment sequence numbers and media type information. A **media dispatcher** receives events from the **media decomposer** indicating the availability of segments. Depending on the media type, the **media dispatcher** forward the events to the **network** module which simulates the packet level delay characteristics of the channel for that media. The **media dispatchers** also receive timeout events from a **timer** module after every frame interval, upon which it increments the frame number affixed to each segment.

A **media receiver** stores the incoming packets belonging to a segment and ascertains if any packets are missing. It then notifies the **media composer** about the arrival of a segment, along with an indication of the number of packets lost in the segment, if any. Upon receiving a command from the **media composer**, a **media receiver** copies the segment to the destination. Upon receiving a segment arrival notification, the **media composer** updates its tables keeping track of segment notifications in a frame interval. It also receives timeout events from the **timer** after every frame interval, upon which it sends a command to all the **media receivers** to copy the segments belonging to the frame number to the **destination** module.

6.3. Network as a Random Delay Element

For our purpose of simulation, the network is modeled as a "black box" that introduces random delays on the packets passing through it. The model basically attempts to approximate the actual delay behavior of the network in the normal ranges of operation. The stochastic nature of the delay behavior can be due to a variety of reasons such as traffic flows through the network, network internal buffer management and delay

control policies adopted by the network. Since it is outside the scope of this paper to study such behavior, we have taken the "black box" approach.

The actual delay behavior of a network may be captured by nonlinear stochastic models involving the variables $\{\overline{D}, d\overline{D}/dt, d^2\overline{D}/dt^2, \ldots\}$ [6], where each variable is associated with a pair of values (maximum,average). In many cases, the models are too complex to derive a concise representation of the delay behavior that is required in the simulation. So we approximate the delay behavior with simple stochastic models involving $\overline{D}$ and $d\overline{D}/dt$.

Given a steady stream of packets, the delay $\overline{D}$ is controlled to vary slowly in relation to the inter-packet arrival rate. In other words, successive packets in a steady stream experience less variation in delay in comparison to packets spaced further apart. The desired rate of variation of delay is incorporated in a mathematical function in the simulation. Different delay functions model different network behaviors. The delay function used in the simulation includes the following:

- $d\overline{D}/dt = 0$, i.e., constant delay network;
- $d\overline{D}/dt = C(> 0)$, i.e., ramp delay network with rate ON-OFF control;
- $d\overline{D}/dt + k_1\overline{D} = k_2$ $(k_1, k_2 \leq 0)$, i.e., first order delay network with rate ON-OFF control.

The bounds of the delay average are also controllable in the simulation so that any desired network delay behavior can be achieved by incorporating higher order variations in the mathematical functions. How these delay functions are incorporated in the simulation [14] is beyond the scope of this paper.

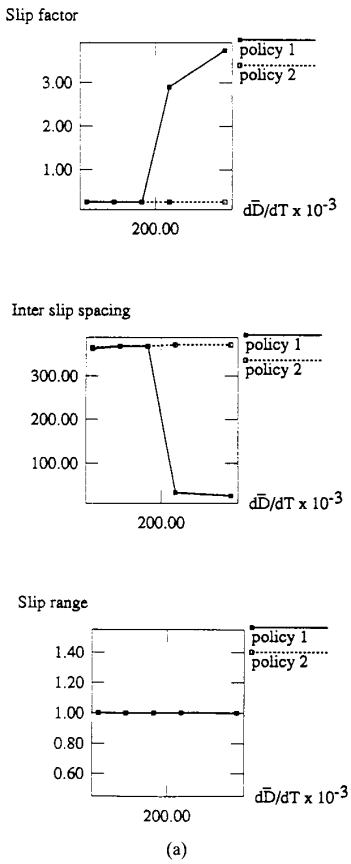
6.4. Performance Indexes

To simplify the performance studies, we assume that each station generates the following traffic types: video, audio, and text data. We categorize performance indexes into two types: those that are perceptible at the application level and those that reflect the protocol execution behavior. Note however that there is a strong correlation between the two sets of performance indexes.

6.4.1. Indexes for Application Perceivable Performance:

1) Slip factor = Ratio of the number of slips in an observation interval to the number of frames exchanged in this interval (expressed as a percentage)
2) Interslip spacing = Average number of frames exchanged between two succesive slips in the observation interval
3) Slip range = Average number of frames missed in a slip

A smaller value of the slip factor, a larger value of inter-slip spacing and a smaller value of the slip range are indicative of good protocol performance. These indices are studied against $|d\overline{D}/dt|_{\max}$ for each of the networks, namely, constant delay, ramp delay and first order delay networks. Each of the networks was studied with the policy 1 and policy 2 to control the frame timeouts,viz., shifting the receiver clock by the average of observed delay in the frame misses over the last K frames (K = 3), and shifting the clock by the observed delay and an extra margin limited by the application specified IFP ($\approx$ 2 ms).



(a)

Fig. 9. Application level performance indexes. (a) On first-order networks.

Index for Protocol Execution: This index describes the execution behavior of the protocol by providing timing profile of the relevant components, and is given by "ΔT(t)" which is the cumulative adjustment (or shift) to the **timer** value at time "t" since the start of execution. The ΔT(t) is a monotonically increasing function of "t." This is because a compensatory shift in the frame timeout intervals is absorbed into the "virtual clock" in the application, and hence any further compensation is considered by the applications as a fresh shift introduced by the transport protocol. The ΔT(t) establishes a timing relationship between the delay behavior of the network and the slips seen by the application.

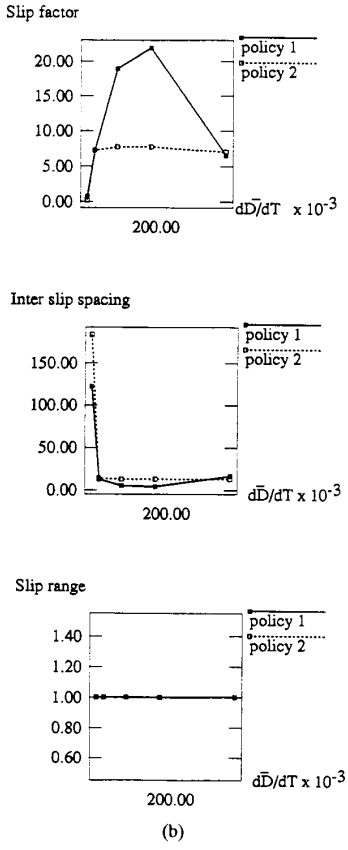
The following section discusses the simulation environment and the performance results.

VII. Simulation Study

Simulation studies were performed on Sun Sparc workstations using a simulation testbed, called REAL, developed at the University of California [15].

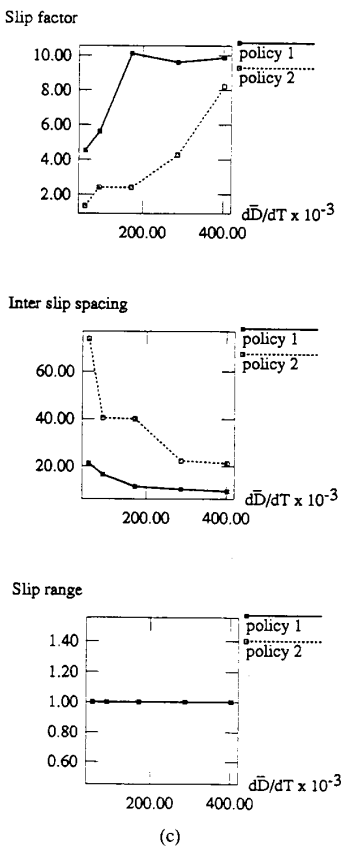
7.1. Simulation environment

The scope of the simulation is to evaluate the skew compensation part of the protocol. The protocol parameters were chosen as: IFP = 2 ms, K = 3, T = 33.5ms, corresponding



(b)

Fig. 9. Application level performance indexes. (b) On ramp delay networks.



(c)

Fig. 9. Application level performance indexes. (c) On constant delay networks.

to a video channel. The delay in the network is uniformly distributed around the chosen average delay values. Functionality testing of the protocol was done to validate the recovery components such as frame interpolation and clock shift, media synchronization and connection aborts.

The simulation is aimed at exercising the protocol under strenuous delay behaviors of the network. For this, we choose $\overline{D}_{\max}$ and $|d\overline{D}/dt|_{\max}$ to be much larger than $\overline{D}_{Q:\max}$ and $|d\overline{D}_Q/dt|_{\max}$, namely by a factor of 20–50. For instance, the normal operating range for a video channel is given by $\overline{D}_{Q:\max}$ = 0.2 ms, $\overline{D}_{Q:avg}$ = 0.18 ms, and $|d\overline{D}_Q/dt|_{\max} = 12.5{*}10^{-3}$. Besides testing the protocol in this region, we increased $\overline{D}_{\max}$ and $|d\overline{D}/dt|_{\max}$ to the ranges of 5–12 ms and $100-400{*}10^{-3}$.

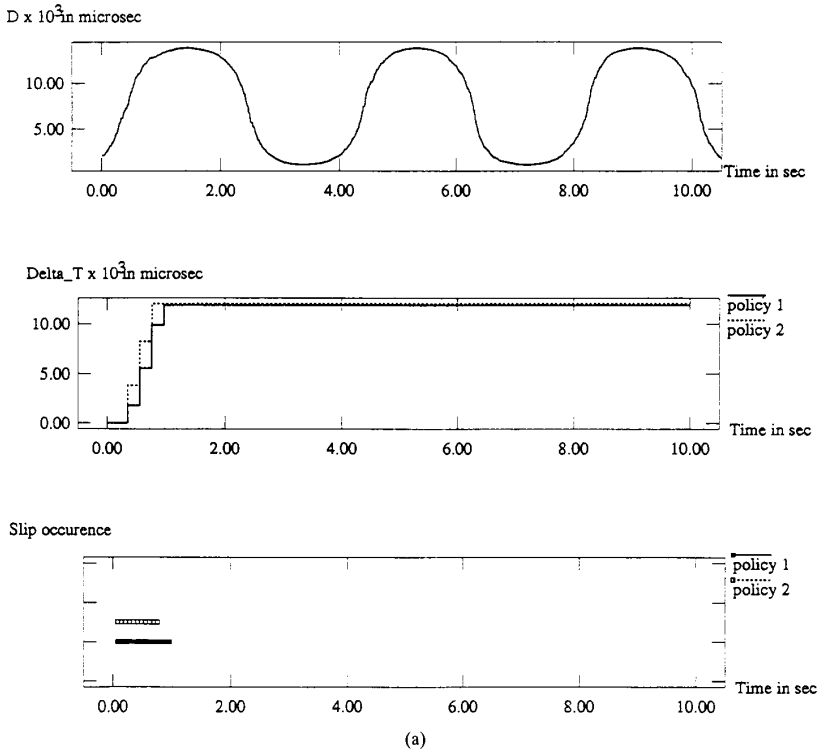
Testing the protocol under strenuous conditions requires that the connection abort mechanisms be turned OFF in the simulation so that the slip phenomenon can be observed without any interference. In other words, even though the conditions that cause a connection abort arise during execution, the protocol was not aborted to enable uninterrupted monitoring of the slip behavior. We believe that such a worst case testing of the protocol is necessary in validating the protocol under different networks exhibiting a variety of delay behaviors such as providing deterministic and random delay channels.

The simulation results are presented in the form of graphs depicting the application level performance indices in Fig. 9 and the protocol execution indexes in Figs. 10–12. The former indexes are studied as to how they vary with $d\overline{D}/dt$, while the latter indexes are given in the form of timing behaviors during specific execution instance of the protocol. The results are analyzed below.

7.2. Observations on Application Level Effects

As can be observed in Fig. 9, the slip factor increases with $d\overline{D}/dt$, and saturates as $d\overline{D}/dt$ increases beyond a certain value. The rate of increase is higher with Policy 1 than with Policy 2. The reason is that Policy 2 provides an extra cushion in shifting the timer starting point, which in turn reduces the probability of slips in subsequent frame intervals. The interslip spacing decreases as $d\overline{D}/dt$ increases since the packets experience more delay variation, which in turn results in an increased probability of a packet belonging to a frame not arriving in the frame interval. For the same reason discussed earlier, the Policy 2 results in larger interslip spacing than the Policy 1. The slip range remains almost at 1.0 in the observation interval. However, we expect that as $d\overline{D}/dt$ is increased to a larger value, slip ranges beyond 1.0 are likely to occur.

The actual delay behavior assumed in the network also influences the application level performance, as can be seen



(a)

Fig. 10. Protocol execution profile on first-order delay networks.

by comparing the performance indexes in the case of constant, ramp and first delay networks. For instance, the minimum value of inter-slip spacing (with Policy 2) is about 20 frames on constant delay networks and 13 frames on ramp delay networks in the extreme ranges of protocol operations, i.e, $0.1 \leq d\overline{D}/dt \leq 0.4$ The reason is that the delay variation is more in ramp delay networks, thereby resulting in a lower value of the minimum inter-slip spacing. At lower values of $d\overline{D}/dt$, however, the inter-slip spacing is higher in ramp delay networks. Over the range of $d\overline{D}/dt$ simulated, the slip factor increases from 2% to 8% with $d\overline{D}/dt$ in constant delay network while the slip factor remains more or less unchanged at 6% and at 0.1% in ramp delay and first-order networks respectively. With first-order delay network behavior, however, the inter-slip spacing remains large ($\approx$ 350) and slip factor is small ($\approx$ 0.1) over a wide range of $|d\overline{D}/dt|_{\max}$, implying that the protocol is able to sustain more delay variation in these networks because of slower changes inherent in first order delay behavior.

These observations substantiate or premise that it is the delay variability and not the absolute delays in networks that affects media synchronization in a significant manner.

To analyze the results from an application perspective, consider the example of digital TV. The inter-slip spacing is observed to vary roughly between 300–400 frames. We expect that with protocol optimizations such as using previous segments from the replay buffer, the performance will improve.

7.3. Observations on Protocol Execution Behavior

As can be seen from the ΔT(t) curves in Figs. 10–12, the protocol that incorporates Policy 2 has its ΔT(t) tracking the $\overline{D}$(t) curve more closely. This is due to the incorporation of an additional cushion in the timer shifts for delay compensation. As can be seen from the corresponding timing of slip occurences, the Policy 2 results in less number of slips.

The ability of the protocol to track a given change in $\overline{D}$(t) is an indication of the effectiveness of the protocol in preventing the delays from affecting the application in the form of slips. So large changes in $\overline{D}$(t) in a given interval, when accompanied by significant changes in ΔT(t), result in reduced number of slips. Accordingly, Policy 2 provides a better tracking ability than Policy 1, and correspondingly results in a better application level performance.

At start, ΔT(t) begins to track the $\overline{D}$(t) and then settles down at a higher value. The transients indicate the behavior of a "first-order control system" with slower settling time. Policy 2 results in a faster settling time because the extra cushion allowed in the timer shift provides a better response to changes in $\overline{D}$(t). During both the startup transients and the transients that occur later, the slips occur more frequently than the case when there are no transients.

The effects of the delay behavior assumed in the network can be seen by comparing the graphs in Figs.10–12. With constant delay networks, the startup transients persist for a longer duration ($\approx$ 2 s) than with ramp delay networks ($\approx$ 0.5

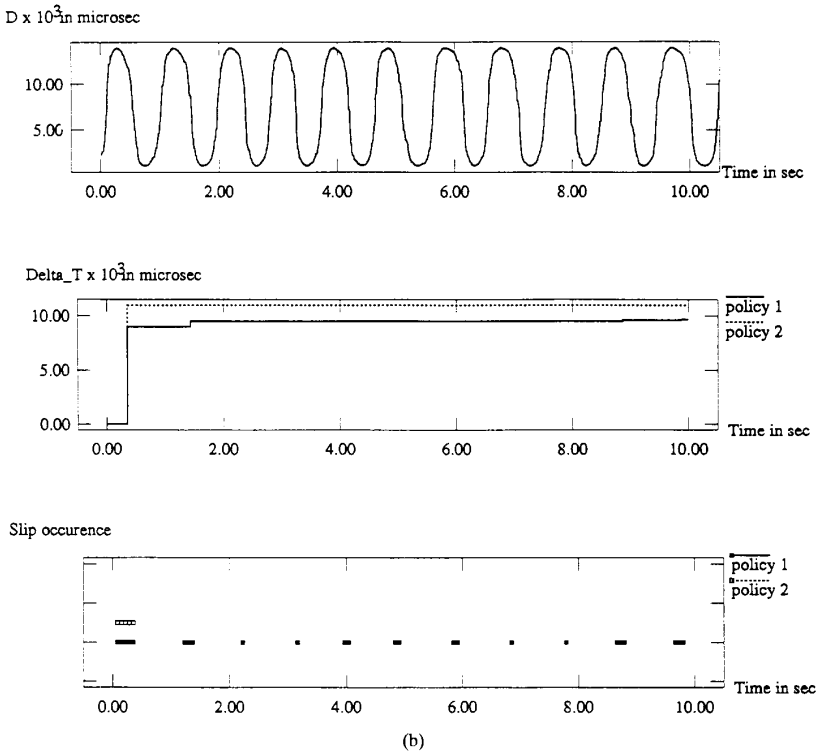


Fig. 10. Protocol execution profile on first-order delay networks.

s) and with first order delay networks (≈ 0.25 s). The reason is that the larger delay variability in ramp delay and first-order delay networks trigger the delay compensatory mechanisms in the protocol sooner than with constant delay network, which results in faster settling time. In all types of networks, Policy 2 performs uniformly better than Policy 1 as expected.

In a sample application such as digital TV, the startup transients may cause frequent glitches in reception of video/audio before the system settles down to a glitch-free reception. The initial glitches may persist between 0.25 to 0.5 s.

VIII. Related Works

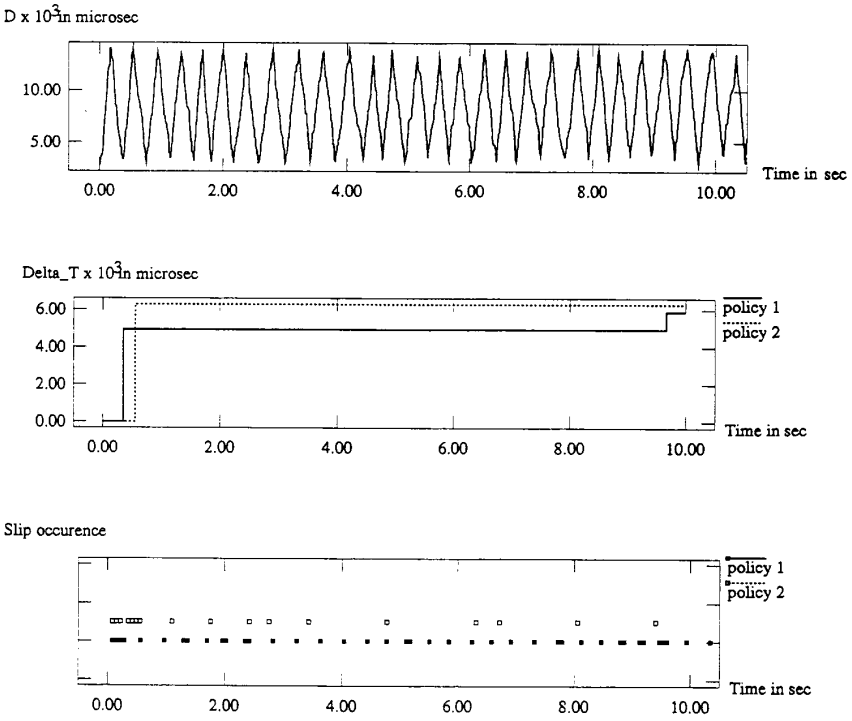
Many network level models developed for high-speed communications [2], [16] basically propose enforcing deterministic delays in a network. Such a network is a special case of deterministic channels with $\overline{D}_{avg} \simeq \overline{D}_{\max}$ and $d\overline{D}_{avg}/dt = 0$ in terms of our characterization of networks. With these tight delay guarantees by the network, providing clock synchronization in the transport layer is sufficient for multimedia synchronization because all the media data packets sent on deterministic channels experience fixed and nonrandom delays, thereby arriving at destinations for play at the scheduled points in real-time. Thus synchronization support may be viewed as implicitly built into the network.

Since the above models normally treat data with different synchronization requirements in the same way, the delay constraints on the network can be a overkill (i.e., more than necessary) for applications that inherently exhibit weaker synchronization requirements. For instance, the network acts as a "perfect delay line" for both continuous and bursty data streams. In contrast, our approach is more flexible in that it allows QOS_{transp} to be systematically exploited to relax the delay constraints on the network to varying levels, with strong synchronization, viz., DV $= \underline{0}$ and IGS $\rightarrow \infty$, being only an extreme case.

Some works propose network models that are capable of providing deterministic and probabilistic channels with different delay parameters [7], [17], [18]. These works basically deal with how various delay constraints can be enforced in the network, but they do not deal with the multimedia synchronization problem *per se*. In our view, our synchronization protocols may be realized as transport level protocols on top of these networks.

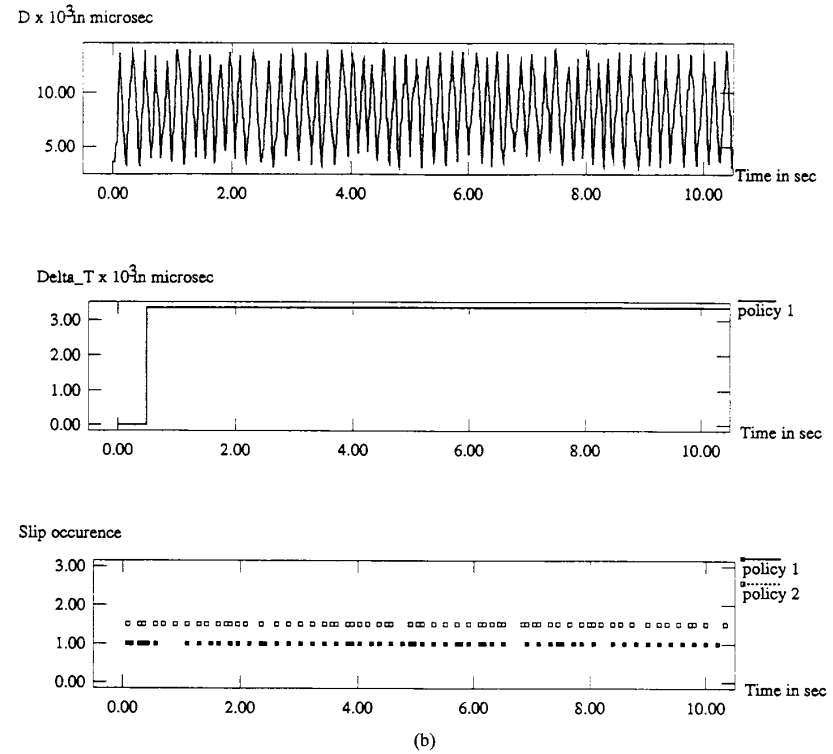
The multimedia synchronization protocol described in [19] is basically derived from a specification of the temporal relationships between multiple data streams. In comparison to this work, our protocols are at a lower level in that they rely on a systematic use of the information on skew tolerance in the application which, in itself, may be derived from the information on how strong the different media data are temporally related.

The work in [12] demarcates data synchronization semantics from data transport semantics and deals with the former by describing temporal relationships among data at a higher level.



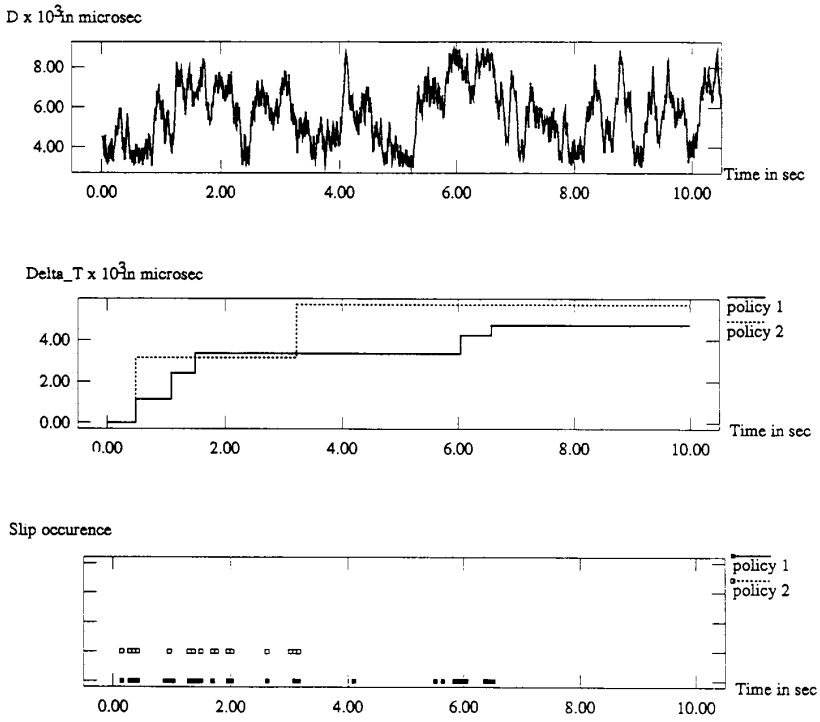
(a)

Fig. 11. Protocol execution profile on ramp delay networks.



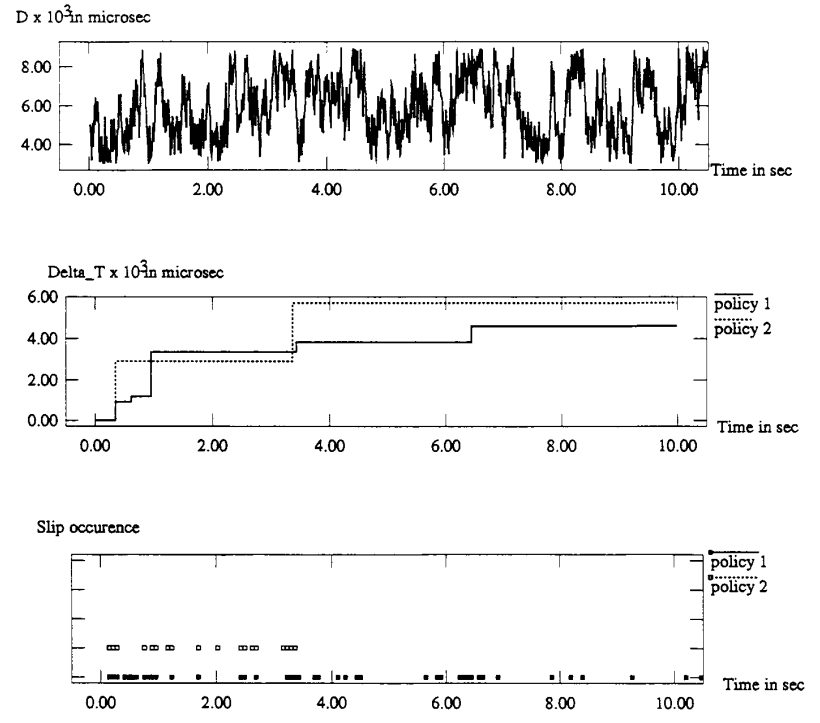
(b)

Fig. 11. Protocol execution profile on ramp delay networks.



(a)

Fig. 12. Protocol execution profile on constant delay networks.



(b)

Fig. 12. Protocol execution profile on constant delay networks.

The work in [20] proposes an architecture in which distributed applications interface with the transport system through a set of synchronization operations. However, both these works do not deal with the protocol issues, viz., the effects of packet delay and/or loss on synchronization, are not directly addressed. In contrast, our work deals with the protocol mechanisms in the transport system for synchronization.

IX. Conclusions

The paper described protocols for delivering multimedia data streams at destinations, typically through a high-speed network, with the streams synchronized in real-time in the presence of network induced delays. The basic premise in our approach is that the user level transport system takes the burden of synchronization instead of the network, which introduces efficiency and flexibility in the data transport protocols.

The synchronization problem is to maintain the required association between multimedia data across various streams in real-time. Our solution to the problem requires framing of data streams whereby various data units in the streams, i.e., data segments, deliverable simultaneously to the user need to be identified ("lip-sync"). The notion of simultaneity in data delivery is meaningful only to the application in that the latter determines: i) the length of a data frame, and ii) how far a data segment in a frame can be skewed in time in relation to other segments and still be acceptable for delivery. Furthermore, the acceptability of previous segments in the place of a current expected segment that does not arrive in the stipulated time manifests as a skew tolerance in the application. The skew tolerance is related to the real-time persistence levels of data in the application, and is derivable from the temporal relationships among data streams inherent in the application. The skew tolerance is systematically exploited in the data transport protocols to effectively compensate for the network induced delay and/or loss during transport.

Basically, the temporal axis of an application is segmented into intervals where each interval is a unit of synchronization and holds a data frame. Simultaneous data delivery involves delivering all data segments belonging to an interval within a certain real-time delay. A data segment basically depicts an application-specifiable granularity of data, often realizable as a sequence of packets exchanged through the network. The skew tolerance in the application is translated into relaxing the strong packet level delay constraints that otherwise need to be imposed on the network. The weaker delay constraints manifest in putting less demand on network internal resources such as allocation of packet buffers and assignment packet scheduling priority.

Using the above approach, the paper described protocols for synchronized data delivery, meeting application specific real-time constraints (e.g., digital TV, scientific visualization, industrial control systems) in the presence of data loss/delay in the network. Specific parameters that describe the skew tolerence in applications were identified as transport level QOS requirements, and procedures to generate delay constraints on the network from the QOS parameters were described. The paper also studied the viability of the protocols by functionality testing in various delay situations and by performance evaluation using specific skew measures. The study is based on simulating the protocol execution for typical applications that use audio/video, both under normal and extreme delay behaviors exhibited by the network.

The contribution of the paper is in formulating a canonical and unified data transport model that provides delay compensation during delivery of multimedia data streams. We believe that the application specific handling of skew in the model increases flexibility and efficiency of data transport. Higher level synchronization protocols such as temporal ordering of data segments [5], [19] may execute on top of the services provided by the transport model.

In summary, we believe that the research described in the paper will shed useful insight into the evolving application technologies for broadband ISDN and metropolitan networks.

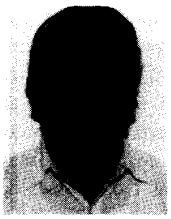
Acknowledgment

The authors acknowledge J. E. Butler of Kansas State University for valuable discussions on analyzing the delay behavior of high-speed networks in simulation models.

References

[1] CCETT,*Multimedia Synchronization*, in *CCETT Int. Note: AFNOR ad-hoc group on AVI standardization*, July 1988.
[2] D. Ferrari, "Design and applications of a delay jitter control scheme packet switching internetworks," in *Proc. 2nd Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 72–83, Nov. 1991.
[3] S. E. Minzer, "Signaling protocol for complex multimedia services," in *IEEE J. Selected Areas in Commun.*, vol. SAC-9, pp. 1383–1394, Dec. 1991.
[4] K. Ravindran and R. Steinmetz, "Transport level abstractions for multimedia Communications," in *Conf. Multimedia Communications*, ICCC, Apr. 1993.
[5] K. Ravindran, "Real-time synchronization of multimedia data steams in high speed packet switching networks," in *Workshop on Multimedia Information Systems*, IEEE Communication Society, Feb. 1992.
[6] S. Chowdhary, "Distribution of the total delay of packets in virtual circuits," in *Conf.Computer Communications*, IEEE INFOCOMM, vol. 2, pp. 911–918, 1991.
[7] D. Ferrari and D. C. Verma, "Real-time communication in a packet switching network," in *Second Int. Workshop on Protocols for High Speed Networks*, IFIP WG6.1/WG6.4, Nov. 1990.
[8] R. G. Herrtwich, "Timed data streams in continuous-media systems," in *Tech. Rep. TR-90-017*, International Computer Science Institute, Berkeley CA, May 1990.
[9] D. Shepherd and M. Salmony, "Extending OSI to support Synchronization required by multimedia applications," in *Computer Communications*, Butterworth-Heinemann, vol. 13, pp. 399–406, Sept. 1990.
[10] P. Ramanathan, D. D. Kandlur, and K. G. Shin, "Hardware-assisted software clock synchronization for homogeneous distributed systems," *IEEE Trans. Computers*, vol. 39, pp. 514–524, Apr. 1990.
[11] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, Oct. 1991.
[12] R. Steinmetz, "Synchronization properties in multimedia systems," *IEEE J. Selected Areas in Commun.*, vol. SAC-8, pp. 401–412, Apr. 1990.
[13] J. Suzuki and M. Taka, "Missing packet recovery techniques for low bit-rate coded speech," *IEEE J. Selected Areas in Commun.*, vol. SAC-7, pp. 707–717, June 1989.
[14] J. F. Kurose and H. T. Mouftah, "Computer-aided modeling, analysis and design of communication networks," in *IEEE J. Selected Areas in Commun.*, vol. SAC-6, pp. 130–145, Jan. 1988.
[15] A. Dupuy, J. Schwartz and Y. Yemini, "NEST: A network simulation and prototyping testbed," *Commun. ACM*, vol. 33, pp. 63–74, Oct. 1990.
[16] G. J. M. Smit, P. M. Havinga, and M. J. P. Smit, "Rattlesnake: A network for real-time multimedia communications," *Tech. Rep.*, University of Twente, Netherlands.

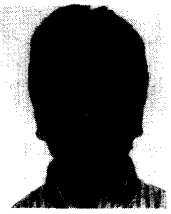
[17] G. Anastasi, M. Conti, and E. Gregori, "TPR: A transport protocol for real time services in a FDDI environment," in *Proc. Int. Workshop on Protocols for High Speed Networks*, IFIP, 1991.
[18] J. M. Hyman, A. A. Lazar, and G. Pacifici, "Real-time scheduling with quality of service constraints," *IEEE J. Selected Areas in Commun.*, vol. 9, pp. 1052–1063, Sept. 1991.
[19] T. D. C. Little and A. Ghafoor, "Multimedia synchronization protocols for broadband integrated services, *IEEE J. Selected Areas in Commun.*, vol. SAC-9, Dec. 1991.
[20] C. Nicalaou, "An architecture for real time multimedia communication systems," *IEEE J. Selected Areas in Commun.*, vol. SAC-8, pp. 391–400, Apr. 1990.



K. Ravindran received the B.Eng. degree in electronics and communications engineering and the M.Eng. degree in automation, both from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in computer science in 1987 from the University of British Columbia, Canada.

Currently, he is on the faculty in the Department of Computer Science, Kansas State University. Previously, he has held faculty position in the Department of Computer Science and Automation at the Indian Institute of Science, and also held research positions in communications industries in Canada and in the space research organization in India. His current research interests are in distributed systems modeling and design, distributed protocols, high-speed network architectures and protocols, and real-time systems.

Dr. Ravindran is a member of the ACM and IEEE Computer Society.



Vivek Bansal received the B.E. degree in computer science from M.S. University, Baroda, India, in 1990, and the M.S. degree in computer science in 1992 from Kansas State University, Manhattan.

He joined NEC Research Institute, Princeton, NJ, in 1992. His current research interests include transport protocols for multimedia systems, and congestion control and performance evaluation of ATM networks.