D. Aleksandrov
December 2001

RTTP: Properties of a real-time protocol

<draft-aleksandrov-rttp-prop-01.txt>

Status of this Memo

Copyright Notice

Abstract

The purpose of this document is to provide ideas for real-time communications in Internet. This document doesn't expand any pre-defined standards or practices, it also doesn't conform ones. This document contains a separate idea for real-time data delivery.

Table of Contents

1. Introduction

The purpose of this memo is to focus discussion on real-time data transmission in the Internet and give a possible method of solution.

No proposed solutions in this document are intended as standards for the Internet. Rather, it is hoped that a general consensus will emerge as to the appropriate solution to such problems, leading eventually to the adoption of standards.

"Real-time data transmission" means audio and visual information. The closest by analogy with radio broadcasting.

The offered idea expects this imformation to be interpreted by a human. It is not suitable for real-time transmissions between machines only.

This document doesn't only propose ideas but also descibes the way some of them have been reached. The memo's origin is the "Protocol for real-time transmission in a packet-switched computer network" work first published and still available in HTML at http://rttp.over-ground.net .

The core idea of the document is described in sections 7 and 8.

Section 2 defines principles valid for analogue audio and video broadcasting. The conclusion is that they are much different than the principles Internet is constructed on. Sections 3, 4, 5 and 6 of this document develop ideas how to apply the principles of an ordinary tv and radio to a computer network. Section 9 compares the basic idea of the document with some existing ideas for real-time transmission.

2. Definition of basic principles

Two principles concerning data transmission are defined here. They are used for comparison between TCP/IP transmission and radio broadcasting.

The following definition of "broadcasting" is made here: transmission of data independently from the recipients. A broadcaster operates in the same way no matter the receivers. This definition is valid for the whole memo.

Principle for importance of reliability û transmitted data must be received as full as possible, no matter the time it will be taken for. The reliability is more important than the time the transfer will take.

Principle for importance of time û transmitted data must be received as synchronously as possible. The perfect way is to receive the data with the speed ot its transmission. If the whole data cannot be received synchronously some part of it may be eliminated but there must be no obstruction to receive the following actual data.

According to these definitions IP follows neither one nor the other of the principles. The time to live each packet has can be assigned to the second principle but its purpose is to preserve the network from overloading.

Basicly TCP/IP is constructed according to the first principle. The packed switching gives some level of reliability and the acknoledges TCP uses cause nearly every packet to reach its destination.

If a radio or tv program must be transmittes we must refer to the second principle. It is more important to receive data synchronously than to receive it full.

Broadcastings on air are constructed this way. The receiver is surely in touch with the transmitter, the transmitter is also a broadcaster, according to the definition. If the program is not received properly, for example if the antenna is too small,

usually there is enough data received for a human to understand what is broadcasted. This is a huge advantage of broadcastings on air.

This advantage is due to the fact that the most importaint peices of data are in the surest zone of the information stream.

The surest zone is the one in the center of the freqiency band width. Examples from FM radio and tv broadcastings will follow.

In the center of the frequency bandwidth there are the lowest frequencies which are enogh for speech or melody understanding. If a human receives with its radio only these lowest frequencies he is not supposed to enjoy the program but he will understand what is it about. If the bandwidth that is received is expanded there will be a good quality of the music received.

The stereo information is situated in the farest non-sure zone of the bandwidth. If the stereo information is received the quality gets better but there is a pretty good quality even without it.

Color and B&W tv can be compared by analogy. The color information is a little piece of the whole one and is in the farest zone of the bandwidth. Because of its lower frequencies the sound of a tv program is most surely received.

If some of the broadcasted data is lost the user will hear a distinguished sound and a picture with bad quality. If less data is lost there will be a brilliant picture but still B&W. In this case there is enough information a human can understand. Most of the information carriers are present, there is only an adition missing û the color.

Three characteristics of real-time broadcastings on air have been defined:

- guaranteed synchronous receiving;

- preserving of information as much as possible even if not the whole data is received;

- the broadcaster is independent from the count and state of the receivers.

To transfer data with these characteristics in a computer network the real-time mechanism must include data multiplexing according to its levels of importance.

Data must be devided into pieces with increasing and assigned

levels of importance. If the whole information cannot be transferred only the most important pieces should be.

3. "Streams" definition. Model of the Single Physical Line.

A model of real-time communication between a client and a server will be constructed. They are parts of a packet-swithing network. In particular we may thing for the data as audio-visual one although this is not obligatory.

In this model "client" is defined as a (process on) host requesting and receiving data. The "server" is a (process on) host which finally manipulates data before it is supplied to the client. This definition of "server" is not actually true in the common sense of the term but it will be used with its pre-defined meaning in the whole work.

Model description.

The client requests and receives data. The client is connected only with the server by a single line with defined maximal transfer rate. This kind of connection will be called "single physical line".

The server is not a generator of the packets it sends to the client. It is just a host of an interconnected network. The real executor of client's request is another host among the network and will be called "broadcaster". At this point the mechanism for transmission of data between the broadcaster and the server is not a subject of interest. It is accepted to be true that the server is able to receive all of the data generated by the broadcaster in its primary sequence.

The broadcaster generates every specified interval data with quantity k [B/s].

Let W [B/s] be the quantity of data which can be sent from the server to the client for the specified interval. Let W variates in the range [0; m], where m > k.

It is accepted that the client requests only the data of the broadcaster. It doesn't mean there are no other packets transfered by the single physical line that connects it with the server. This can be one of the reasons for W variation.

If W exceeds or equals k, evidently all of the data generated by the broadcaster will be received by the client.

If W < k the server must discharge some of the packets according to the principle for importance of time but according to the principle for importance of reliability the mustn't be chosen occasionally. The server must have a criteria which of the packets to be eliminated if they cannot all be transmitted.

The broadcaster must structure its data according to some levels of importance. It is to do this as the server is not oblidged to interpret broadcaster's data.

In this model the broadcaster generates n packets each unit of time. They are numbered from 1 to n independently for each unit and these numbers will be called "internal". Time intervals are numbered also form 1 to q and they form q so called ôstreamsö.

Every packet holds its stream number - i and the sequent number inside the stream - j. On the whole, the broadcaster generates packets each having two indexes (i, j) in the following sequence:

(1, 1), (1, 2)... (1, n), (2, 1), (2,2)...(2, n), (3, 1)... (q-1, n), (q, 1)...(q, n), (1, 1).....

In this model the information is first divided by time in units with equal length. For every time unit the data is structured according to its importance and the packets carrying the most important data are the packets with smallest internal numbers. The time to live for a packet (if there is one specified for the network) must not be greater than the time for turning through the whole stream numbers. This guarantees there will not be received two pakets with identical indexes (stream and internal number) without having idea which was first generated.

The indexes implemented in all of the packets must be used by the server in case W < k. The purpose is to obtain real-time transmission so it is unacceptable just to queue the newcoming pieces of data.

It is accepted for true that all of the packets that should be sent through the single physical line are stored in FIFO queue by the server.

This is not enogh for real-time data transmissions so broadcester's packets should be treated differently.

The sequence of streams is the sequence of their identifiers except for stream q followed by stream number 1.

The server acts as following:

1) Every newcoming real-time packet with destination the client is added to the queue for the client if in it there is no untransmitted packet with the same or smaller internal number and smaller stream identifier. Every time a real-time packet is added all of the broadcaster's packets are sorted by ascending indexes. They are stored in so sorted order on places of the queue already reserved by broadcaster's packets.

2) When a new packet arrives to the server and in the queue for the client there is untransmitted one with the same or smaller internal number and smaller stream identifier all the packets belonging to previous steams are discharged. The packets remaining in the queue are stored on the nearest to the exit places of the queue already reserved by broadcaster's packets.

These acts of the server asure:

1) The principle for importance of time. Discharging old packets guarantees that the client will always receive actual data. As a packet with same or bigger internal number from the next stream is received, the client's delay is bigger than one time unit of the broadcaster.

2) The principle for importance of reliability. The packets with smallest internal numbers are always stored on front positions into the queue.

3) Transmission equity. The real-time data doesn't bother the other. Each time real-time packets are sorted they are placed on already reserved by the real-time transfer places. These places are FIFO manipulated like non-real-time packets.

Example: Let's have packets with numbers (2, 4); (2,5) and so on in an outgoing queue of a server. There is a delay in transfer to the client and they are not transmitted for a period of time.

When (3, 1) packet arrives it will be stored at the end of the queue. After it (3, 2) and (3, 3) will be stored. If packet (2, 4) is still untransmitted and (3,4) arrives, evidently the client is behind time.

In this case packet (3, 1) will be stored on the place (2, 4) used to take; (3, 2) will replace (2, 5) etc. The old data will be discharged and the new will take its place for the purpose of compensating as much as possible the delay.

(end of the example)

The second term of transmission is according to the priciples for importance of time and reliability and also the queue operations are equitable. There is no danger to mix the arriving data. The basic problems are how information will be structured and transfered to the server.

The model does not guarantee synchronous receiving of data. We are not sure all of the packets will be received in the sequence they are generated but when a client receives a packet with stream number different than the one of the previous it may be sure there will be no more data from the previuos stream. If a client stores the received packets in memory after the stream changes the data from the previous one may be interpreted.

4. Structure of real-time data

An example of data multiplexing according to levels of importance is given here. No standard is specified in this point. The purpose is just to prove it is possible and combined with the real-time transfer mechanism sufficient result can be reached.

An example of stereo sound multiplexing will be used, the frequency responce is 20~20000 Hz and the dynamic range is 96 dB. The example is based on the charactreristics of the sound recording on a compact disc (digital audio) which is recognized as a high quality standard. Generally, in this point a sample mechanism for structuring the data from a compact disc is needed.

The first thing which can be done is to transform the stereo sound to mono. By making an average of each two corresponding samples from the two channels the mono channel is received. The received sound has enough information for sound understanding, it has lower quality than the original program but its data is half at size. If the mono channel is present together with one of the stereo channels, the other channel of the stereo sound can be reckoned. The whole size of the mono channel and one of the stereo ones is the same with the original size of the program. Even this elementary operation leads to two basic facts:

1) The data was divided, so a receiver able to receive only half of its size would interpret it correctly.

2) A recever of the complete information doesn't need to have bigger capacity than a receiver of the original program.

The received mono signal can additionally be devided by frequency ranges.

Let a new channel called K1/5 be created. K1/5 is made by calculating averages of each five sequent samples of the mono channel. The frequency responce of the new signal is 4 KHz, which is close to a phone call quality and enough for speech or melody understanding. The size of K1/5 is one fifth of the size of the primary mono sound. If any four of five samples from the mono channel are transmitted together with the respective K1/5 sample the all five samples of the mono signal can be received. Yet, a receiver having capacity 1/10 (it is 1/5 of the half) will be able to interpret some data. On the other hand the whole program, structured as K1/5 samples fist, followed by the complementary blocks of four samples from the mono channel, and then followed by one of the stereo channels has the size of the original program.

A working example for data structured by levels of importance must lie on channels with cascading expanding of frequency responce and dynamic range. This memo doesn't take up with offering such mechanism. This example was just for a provement it is possible to multiplex data this way. In the example compression is not foreseen but it can greatly reduce the data size. Surely there can be numberless algorithms for data structuring and if there is a working real-time mechanism for data delivery they will be undoubtlessly invented.

5. Fishbone model

At this point a model with many single physical lines will be considered. The Single Physical Line model was created with the purpose to find a mechanism for data structuring. Dividing the broadcaster from the server was not necessary. It was made because this model had to be easily extended.

As first extension of the model more than one client is added. All of the clients are connected by their own singe lines to one and the same server. All of the clients request one and the same broadcaster's program. ("Broadacster's program" is the real-time data generated by the broadcaster. The analogy is a "tv program" or "radio program".)

The server has different outgoing queues for all of the clients. In every queue the real-time packets are handled together with non-real-time packets, according to the principles for a single physical line defined in section 3 of the document. Evidently the server must operate with each queue individually. On the other hand there will be many coinciding packets in the outgoing queues. This is disadvantageous to send each client's request to the

broadacster as all of the data is sent through the server.

The Single Physical Line model doesn't discuss the request operation. When there are more than one clients the request method is important. There are two cases:

1) Every client sends its own request to the broadacster and the broadacster receives it. The broadacster sends packets with destination the client-requester. The server manipulates the real-time packets transmitted through it according to the rules for a Single Physical Line.

2) The server sends a request to the broadcaster. After a real-time packet from the broadcaster is received it is multiplied and stored in the queue for each client that requested the data. Each queue is manipulated according to the rules for a single physical line.

The second variant is more advantageous. The network traffic is diminished but it doesn't affect the quantity of data each client receives. The data for all of the clients is transmitted through the server so there is no need to do it as many times as the clients are.

The behaviour of each host of the network must be defined. It is defined that the server is engaged with the transfer. The server holds the balance between the principles for importance of time and importance of reliability. The single request for more than a client is close to the principle for independence of the broadcaster from the receivers and close to transmissions on air. This scheme works for clients connected by single physical lines but the purpose is to build a working scheme for the whole network.

It is possible a client to be connected than more than a line to more than a host of the network. If there is real-time data transmission there are three ways each host can act:

1) Lack of commitment:

   The server is the only host of the network dividing the real-time packets form the other. The other hosts route each real-time packet like any other. This scheme contradicts the principle for importance of time. The network can be engorged with data. These are enogh reasons to treat this variant as unacceptable.

2) Partial commitment:

   Every host recognizes the real-time packets. Every packet is

routed like any other but real-time ones are manipulated according to the principles of a single line in the outgoing queues.

3) Complete commitment (Fishbone model):

Every host commits with the real-time requests it is to transfer. Every host modifies the requests and transfers them with its address as receiver. When the host receives real-time data it sends it to each of the hosts that requested it.

Evidently the server from the previous model is completely commited with the transfer. If all hosts act this way any request's destination will be cascadingly modified. There will be constructed a transfer route through the network. When a request from another part of the network is sent it will either be performed by another route or will cause an existing transfer route to branch out into another direction. If the broadcaster is connected by d lines with d hosts of the network the maximum number of requests it will receive will always be d, and each will be executed by a different route through the network. The complete tree of routes through the network if there are many clients remains a fishbone which gives the model name.

The Fishbone model is near the principles for broadcaster's independance from the receivers, it observes the importance of time but has disadvantages about reliability.

It is based on a fixed route through the network and every host can become a weak point of the structure. Data losses between two hosts become current for all of the hosts relying on them.

A big problem appears when a host taking part in the transfer route drops out. If every request is resend after a period of time the transmission will be recovered by another route but after a period of silence.

On the other hand if a host is physically the only one able to serve multiple request ("server - client", according to the single line model) the Fishbone model seen to be the optimum one.

A network with lack of commitment of the hosts was categorically rejected so a medial variant between partial and complete commitment is needed.

The scheme for real-tme transfer must be applicable for a packet-switching network and what is the most important - all hosts of this network must follow same principles with others. In the

models yet described there were parts of the network (for example the server in the Single Physical Line model) acting differently than the others.

If different hosts of a network are able to act differently it must be only as a result of apllying same principles in different situations. It is assumed that all hosts must observe the following basic rules for real-time transfer:

1) every host distinguishes real-time packets from the others;

2) in every outgoing queue the real-time packets of any broadcasted program are manipulated according to the principles for a Single Physical Line.

6. Real-time requests

In the models already constructed requests are not discussed. These models accept there is a request and pay attention to data transfer as a result. A specified requesting mechanism is evidenly needed. The broadacster must receive at least one request for data to start transferring. The request can be sent by a real-time or non-real-time packet. The client won't need broadcaster's transfer ethernally so it better renew its request after a period of time. On the other hand the client mustn't be deprived of transfer if its renovating request is transferred too slow, therefore the broadcaster must generate data for a longer period than the one for request renovation.

Let Tz be the time interval after which the client will resend its request. The client will send its request and the broadcaster will receive it (no matter with or without modified client's address) after time represented by Tp. Let the broadcaster stop the transfer if a request is not re-confirmed in Ts time. In this case it is necessary to be true that:

$$Ts = Tz * k + Óp, \text{ as } k \text{ is a coefficient and } k > 1. \quad (1)$$

The reason for this is the assumption that the client will start counting out Tz imediately after the request is sent. The requesting packet will reach the broadcaster in time Tp, and if there's no a big change in network state the next (renewed) request will reach the broadcaster in time Tz + Tp. The coefficient k guarantees there will be no stop of transfer before the next request reaches the broadcaster. This coefficient should be big enough to cope with this task. If there is a mechanism for counting Tp, the broadcaster will be able to set different Ts for each request, even if k = const. It is necessary the time interval

for each client (Ts) to be reset by the broadcaster each time a request is renovated.

The mechanism of mutual time intervals guarantees there will be no constant high intensive request traffic in the network, also the presence of transfer for a while even if there is no request reconfirmation saves the network from unnecessary traffic in cases a client refuses the program. Evidently the time interval and the coefficient k must be carefully selected.

It is natural the time interval Ts to contain the time for turning out some series of data streams. If it elapses for the time of one data stream the requests will be too frequent. In equality (1) the most significant member should be Tz, i.e.

Tz >> Tp,

otherwise the variation of Tp will strongly affect the implementation of the term and a big enough value for k will be needed to prevent from transfer interrupting.

According to the client the real-time data will stop Tp + k * Tz + TpÆ time after its last request, TpÆ is the time for the last broadcaster's packet with destination the client to reach it. If the network has respectively constant parameters it is probably true that Tp approximately equals TpÆ.

As Tz >> Tp it is a good idea to provide a refusing request (refusal) which can be sent by the client if it no more needs broadcaster's transfer. It will preserve the network from a little unnecessary traffic. If a refusal is not sent, for example is the client just drops down, the traffic will be ceased too, but a little bit later.

7. Local Broadcasters Model

Request renovation applyed to the Fishbone model is able to minimish its disadvantages. Fishbone model presumes each host is to modify the request. The request will be sent by the most optimum route through the network, so the transfer will be sent by the same optimum route. When the request is renewed if the developed route is no more the most optimum another transfer path will be created. This solves the problem with dropping out a host of the network but doesn't solve the problem with the weak point each host can be.

The next development of the Fishbone model appears after a detailed study of a host signified as "server" in the Single

Physical Line model.

This host is always completely committed with the transfer. Real-time packets are always treated the same way no matter if it is the requester (according to the broadcaster) or a single client is. Committment of this host must be examined in details.

The first client's request reaches it and according to the Fishbone model this host should replace requester's address with its own. On the other hand there is no need to do this. The transfer between the server and the client will always follow one and the same scheme no matter which of them is the requester. According to the principle for network sameness (all hosts must follow same principles) if this host modifies the request all others must do this - it is the Fishbone model itself. To change the model it is accepted the host doesn't modify requester's address but just transmits the request-packet with its original contents. As there is network sameness all other hosts have to do this so the broadcaster receives a request for data with destination the client.

After a time the server receives another client's request for the program which data is already transmitted through it. If the server just transmits this request again it will bother the principle for mimimum transfer through the network. Evidently in this moment (or a little bit later) the server is to commit with the transfer, send its own request and then resend the real-time data to both his clients. The conclusion is that the server must remember all real-time requests transmitted through it and still active (with unelapsed time) and if another request for already ordered program appears, the server must send it as its own. The basic criterion is the number of still active requests passed through each host.

Let's have a detailed look at server's behaviour. By the time of the second request the first one is still active, so if the server immediately sends its request there will be a period of time with double identical data transfer. It bothers the principles for mimimum traffic and for broadcaster's independance. The second request is sent later than the first, so according to the subjection Tz >> Tp the second one will expire later. On the other hand the server can immediately comply with the real-time transfer just by copying and sending the packets with destination the first requester to the second one too. If the server is in charge of the minimum transfer there are two variants:

1) the server waits for the first request to expire and modifies its renovation with its own address;

2) the server sends its own request immediately and a refusal on behalf of the first client together with it.

Both variants are acceptable. If the first client doesn't renew its request the first variant seems to be better. Without renovation the server will pass second client's request without any modification but after the first client's request is elapsed. If the second variant is chosen in the above situation there will be for a short period of time:

```
server's REQUEST ->
first client's REFUSAL ->
second client's REQUEST ->
server's REFUSAL,
```

and as the second client's request was delayed there will soon be its next one.

The basic principle of this model is the following:

The server passes each request for a real-time program if there is no active request that passes through it for the same program. The server stops every request for a real-time program if there is at least one active request that passes through it for the same program. When the server stops client's request it send one for the same program with its own address as destination and takes up with multiplying the real-time data to all clients having active requests for this program.

As the server acts this way any other host of the network should act this way too.

A disadvantage - if a client's request is stopped it will start receiving the data which the server multiplies for the client. Anyway, it is possible that not the whole traffic for the previous client passes through the server, so the next one will not receive complete data but a partial one. It will be true until the previous client's request expires. After that the server will send the request and becoming a receiver of the whole data will act equally towards both clients.

The idea of the model is in constructing local broadcasters in network areas with enhanced interest in one and the same program. The model will be called "Local Broadcasters model" or just LBC model in future. There are no additional rules for packet routing except these for a Single Physical Line in the outgoing queues. As requests are renovated the local broadcasters will dynamically appear and die out. Each host can become a local broadcaster of a program for a time. Each host must also recognize not only the

packets containing real-time data but also the ones containing requests for such data.

Another aspect of LBC is that a refusal can reach not the local broadcaster but the real broadcaster which doesn't have idea about this client at all. Therefore a notification is necessary so each client should know whether it is served by the real or a local broadcaster. If a client was notified it should send a refusal to the local broadcaster.

Behaviours of the different components of a network, according to the LBC model must be explained:

1) Client. According to the client all Single physical line model, Fishbone model and LBC model are equal. Local broadcaster's notifications are a small and unessential difference between LBC and the other two models.

2) Host (which is not a client). Compared with the case in which each host is partially committed with the transfer there is an additional work-load as each host must listen for real-time requests. When a host becomes a local broadcaster it is completely committed with the transfer and still has to listen for other real-time requests.

3) Broadcaster. The maximum number of requests a broadcaster can receive equals the number of physical lines it is coonected to the network by. Each line is a beginning of a branch of lines and if more than a request is generated in any branch a local broadcaster will be created. The broadcaster is free to choose a route for each real-time packet. It is loaded like a completely commited with the transfer host.

4) Entire network. Network behavior can't be indicated without mathematical appliance. LBC seems to be better than the Fishbone model for there is no compulsory data route which can save the transfer from some losses of data. The whole data can be divided and transfered by lots of low-capacity lines which is not possible according to the Fishbone model. On the other hand there is a possibility of double traffic by one and the same line between two hosts. If host Á is a local broadcaster for host Ã all real-time data will have host A as distination. It is possible some of this data to pass through host B, reach A, and then be sent again to host B by host A. There will be equal transfer in both directions, one more outgoing queue with a possibility of data loss. Yet, this transfer will bother neither the broadcaster, nor the other hosts of the network.

Local Broadcaters Model is the core of this memo. The other ones

are constructed just for the purpose of reaching the idea of LBC easier. The author believes that a protocol based on LBC model can be sufficient for real-time data transmisions in Internet.

8. Mathematical evaluation of the created models

The created models for real-time data transmission need some mathematical evaluation. It is given at this point.

The first interesting index is the loading of network. The network load should be measured as the quantity of traffic transfered for a period of time, in particular B/s. The network load is a combination of all hosts' loads. There are three types of hosts while real-time data transmission. These are broadcaster, server and client. These designations are not really precise in their common meanings but they will be still used in the future in their redefined meanings.

At first broadcaster's load is to be discussed. Each time unit the broadcaster generates a real-time program which data's amount is t [B/s]. For our convenience it is accepted that the data has an even distribution in time, i.e. t=const for any time unit.

Let T [B/s] is the complete amount of data that the broadcaster sends to the network for a defined unit of time.

1) If hosts are partially commited with the transfer:

$$T = l1 * t + l2 * t + ... + lk * t = t * sum(l1; lk),$$

where li is the completeness of the transfer allocated for i-th receiver (0 =< li =< 1), according to the broadcaster; k - number of clients for the examined unit of time.

According to this model the broadcaster sends data concretely for each client, so there are k addends, it is possible some data to be lost in broadcaster's outgoing queues so coefficient for completeness are added.

Fictionally every li = 1.

2) Fishbone model:

$$T = n1 * t + n2 * t + ... + nm * t = t * sum(n1; nm),$$

where m is the number of broadcaster's physical lines used for real-time traffic; ni is the completeness of the transfer by the

i-th line (0 =< ni =< 1), according to the broadacster.

The Fishbone model permits only one real-time request by line, so the maximum value for m is the value of the physical lines that connect the broadcaster to the network. It is possible some data to be lost in the outgoing queues for the differenet lines, so coefficients ni for completeness are added. Fictionally every ni = 1.

3) Local Broadcasters Model:

T = p1 * t + p2 * t + ... + pq * t = t * sum(p1; pq),

where q is the number of received requests; pi is the completeness of the transfer allocated for i-th request, according to the broadcaster.

Only one request can reach the broadcaster by line, as any multiple requests will be stopped by local broadcasters, so the maximum value for q is the value of the physical lines that connect the broadcaster to the network.

It is possible some data to be lost in the outgoing queues, so coefficients pi for completeness are added. Fictionally every pi = 1.

The three formulas look rather alike, evidently with main inportance are the sums they contain.

On the other hand it mustn't be expected the index T to rise unlimited. The broadcaster has limited number of connections to the network and each connection has its maximum capacity.

Let c be the number of broadcaster's connections, and let the i-th has ri [B/s] as a capacity for real-time transfer, 1 =< i =< c.

Broadcaster's physical lines are numbered fictitiously but their numbers, once assigned, should be unchanged for the formulas.

Let R [B/s] be the maximum data that the network can accept from the broadcaster. In this case

R = r1 + r2 + ... + rc = sum(r1; rc).

Evidently T =< R, so there are the following inequalities for the three models:

1) sum(l1; lk) =< R/t for partial transfer commitment;

2) sum(n1; nm) =< R/t for Fishbone;

3) sum(p1; pq) =< R/t for LBC.

Both R = const, and t = const, so two facts are subjects of interest:

- Conditions making each inequality to an equation;

- The average value for the indexes of each sum whenever the expression it takes part in is an equation.

According to the principle for smallest traffic, the optimum inequality is the one which's left side is the smallest, compared with the other two, when the number of clients is one and the same for the three expressions, because the whole data transferred by the broadcaster is the product of each left part and t.

According to the principle for reliability, the optimum inequality is the one in which the average value of all indexes the left side consists of is the biggest, compared with the other two inequalities, when the number of clients is one and the same for the three expressions, because the left parts are all sums of coefficients giving completenesses of transfer. As bigger the average completeness is, as better the program will be received.

In both cases the least sum of bigger members is needed, so evidently the most optimum sum is the one with fewest members.

The first formula is the only containing the number of clients.

There is a need of mechanism for counting the number of clients in the formulas for Fishbone and LBC models. For ease, the following addmission is made: The broadcaster takes a central zone of the network, which means there are approximately equal numbers of hosts most shortly addressed by each line.

This is not a precised addmission but if it is true there are equal possibilities a request to reach the broadcster by any line. It will most load the broadcaster. If the network is not balanced in relation to the broadcaster, most of the requests will be served by some of the lines, which will be far of the borderline case (maximum load) that we are looking for. That's the reason the addmission mustn't be treated as a negative one.

If there is at least one client a transfer route will be created, so T = n1 * t.

If the second request reaches the broadcaster by another line the

whole transfer will be T = t(n1 + n2).

The value of T will grow until it reaches T = t * sum(n1; nm), realized for every i =< m, and then will stay constant for every i > m. Fictionally every ri >= t, i.e. R >= mt, so the biggest possible transfer is T = mt.

This was the sutuation according to the Fishbone model.

The LBC model is largely the same. The value of T will grow until it reaches T = t * sum(p1; pq), realized for every q =< m, and then will stay constant for every q > m. Fictionally T = mt.

For the model of hosts with partial commitment the fictional value of T is T = tk, and there is no limitation for the increase of k. This is a linear function which is practically impossible as the value of R can't be unlimited. That's the reason this model will no more be evaluated. It is to be rejected as unefficient. The more clients are, the average data each receives is less.

At the other two models the growth of transfered data stops no matter how many the clients are, and if R >= mt it is possible that no data is lost in broadcaster's outgoing queues.

The Fishbone model foresees each request is served by its own physical line, so the potential loss of data for the request depends entirely on this line.

The completeness of transfer by the i-th line is describes with the following function ni = F(ri):

   ni = ri/t, realized for ri =< t;
   ni = 1, realized for ri > t,
   t = const.

This dependence not only describes the completeness of transfer for the i-th request according to the Fishbone model, but generally the completeness of real-time transfer by the i-th line. If there are two or more requests served by a line the completeness of the data for each request will be ri/s * t, as s is the number of requests served by the line. The completeness of transfer is the ratio of the possible transfer and the desired by the host one, generally ri/Ri, as Ri [B/s] is the desired transfer (the one which if fully transmitted will lead to no loss). At the Fishbone model Ri = t realized for every i. The LBC model doesn't keep within this condition.

At LBC Ri is different than a constant, and wholly depends on network's condition.

Basicly, the data predestinated for the j-the request is to be divided in c parts, because c are the physical lines of the broadcaster. Each line will take gji part of the transfer which will be gji * t [B/s].

The following conditions are realized:

```
0 < j =< c;
0 < i =< c;
0 =< gji =< 1, realized for every i, j;
gj1 + gj2 + gj3 + .... + gjc = 1, realized for every j.
```

Each unit of time the desired transfer destinated to a host is:

Ri = t(g1i + g2i + ... + gci), and Ri is the quantity of data which if sent through the i-th host will be sent with no loss. Fictionally Ri =< ri.

The completeness of transfer through the i-th host - G, by analogy is the ratio of these two quantities:

```
Gi = Ri / ri = t(g1i + g2i + ... + gci) / ri,
realized for Ri =< ri;
Gi = 1, realized for Ri > ri.
```

In difference to F, G can't be defined as a function of one single index.

The coefficients showing the division of data among the hosts can be structured in a square matrix:

```
g11      g12      g13      ......      g1c
g21      g22      g23      ......      g2c
.........................................
gc1      gc2      gc3      ......      gcc
```

It is true that each line's sum equals 1. By multiplying the matrix by t it looks like:

```
g11t      g12t      g13t      ......      g1ct
g21t      g22t      g23t      ......      g2ct
.............................................
gc1t      gc2t      gc3t      ......      gcct
```

It is true that each line's sum equals 1 t. This is not totally true, as is the statement for 1 lines' sums. Some of the lines can equal 0 as there can be no request by every line. In the borderline case of maximum load the statement is true for each

matrix' line.

The received matrix can be multiplied by the one representing the completeness of transfer:

```
G1
G2
G3
.
.
.
Gc
```

The result looks as following:

```
g11G1 + g12G2 +......+ g1nGn
g21G1 + g22G2 +......+ g2nGn
.
.
.
.
gn1G1 + gn2G2 +......+ gnnGn
```

By analogy the Fishbone model can be presented by the following matrixes:

```
1 0 0 0 ... 0 0          F1          F1
0 1 0 0 ... 0 0          F2          F2
0 0 1 0 ... 0 0          F3          F3
0 0 0 1 ... 0 0    *     F4     =    F4
.............            .           .
0 0 0 0 ... 1 0          Fn-1        Fn-1
0 0 0 0 ... 0 1          Fn          Fn
```

In both cases the completeness of transfer for each request is expressed in one and the same way. But both formulas are still unrelated so the models can't be compared according to them. These formulas are convenient for simulating programs and statistical evaluation.

Another limitation must be introduced here, after the one for centered broadcaster. In LBC model it is assumed that the broadcaster acts intelligently in some measure.

The intelligent aspect of broadcaster's behaviour includes distribution of data according to the network load. Generally must be true the following:

$$Ri / Rj = ri / rj,$$

realized for

$$0 < j =< c;$$
$$0 < i =< c.$$

We are not interested in the concrete traffic distribution for each request but we know it is according to lines load. If the broadcaster doesn't behave this way the whole model is unefficient.

Evidently the last equality can't be entirely true in all cases but it must be approximately true. Anyway, if R >= ct, the loss for each request's data must lean towards 0 for the nearest to the broadcaster zone of the network. This must be true no matter which the physical lines are - the great advantage of the LBC model compared with the Fishbone.

As the maxumum possible transfer for both models is one and the same this advantage makes the LBC model the preferable one.

After a broadcaster's behaviour was examined the one of the hosts and the clients must be examined too. Any client is a host of the network but for ease "host" will be used only for ones that transfer real-time data but are no clients. The transfer route through the network is always unknown and there are numberless possibilities so it is impossible all variants of host behaviour to be embraced. Only conclusions based on statistics and probability can be made. Yet, there are two possible cases for a client's role in the network:

- final host, connected by a single physical line to another host of the network;

- intermadiate host, connected by multiple lines to the network. Other than the requested real-time data can be transmitted through this kind of client.

On the other hand if the client is an intermediate host a virtual final one can be added, so the host of the client will be examined just like any other. The capacity of the line between the real and the virtual hosts can be treated as unlimited and with no time delay. Evidently the quantity and quality of the data received by the virtual host depends entirely on the network. Virtual hosts can be added in any situation so client examination is not needed at all. Enough information can be obtained just by threshing out the hosts which transfer the data.

This is the reason client's examination to be abandoned. Only formulas describing the data transfer as a result of its transmission through the network are needed.

According to the Fishbone model the transfer is realized by k constant hosts (k > 0). The transfer route repeats the route of the request through the network. Relying on hosts' inteligence it can be expected the request to be sent by the freeest lines, so the created transfer route will be the one with lowest loss compared with the other variants for possible transfer paths. Anyway, the request is small enough to be directed on a route with capacity smaller than the real-time data needs.

As there are k hosts forming the route the data is sent by k+1 physical lines, and each has its own completeness of real-time transfer F - the same as broadcaster's lines have. This function will be called ôtransmissionö for ease. Let F(1) be the transmission of the first line of the data route - the one connected to the broadcaster, F(2) be the transmission of the next line data will flow through and so on, up to F(k+1).

The transmission of the whole system is presented by the mimimum value found among these functions.

Let P(x) be the probability a physical line to be able to transfer at least x [B/s]. (x >= 0) The probability to be no loss at the first line is P(1)(t). (We stick to the denotion that t [B/s] is the whole quantity of data generated by the broadcster for a unit of time.) The probability to be no loss at the second line is P(2)(t), etc.

The complete probability to be no loss after the data was sent by k+1 lines is

P = P(1)(t) * P(2)(t)....P(k+1)(t).

If the network is a regular one, i.e. there is same probability for each line, then

P equals P (t) multiplied k+1 times by itself.

In both cases the far form the broadcster the client is, the lowest probability for comlete transfer is, too. Figuratively, the signal "dies away" in the network.

The presentation for LBC model is more complex. The data is transfered by s physical lines, but the transfer by each line can be ti, realized for every ti =< t.

There must be no loss by any line, so the complete probaility to be no loss is

P = P(1)(t1) * P(2)(t2) * P(3)(t3).....P(s)(ts).

If the network is a regular one it is P = P(t1) * P(t2)....P(ts).

For better descriptions of the received formulas the variation of P(x) must be examined. P(0) = 1 for sure, as data with zero size can be sent even by unexisting line. As the material world has its limitations there exists j with a limited value, and every P(x) = 0, realized for x >= j.

On the other hand surely P(x) >= P(x + dx), realized for dx > 0.

P(x) beginning point is (0,1) and never growing the function reaches the point (j, 0).

As closer to the upward end of the shown zone P(x) is, better the real-time transfer is.

Generally, the formula describing the LBC model will consists of more but smaller members, than the formula describing the Fishbone model. Both models must be compared according to P(x).

The signal "dies away" according to the LBC model, too. The question is, is it more expressed at the LBC model than at the Fishbone? Author's opinion is that the answer is "No".

The reason for this is that P(x) will probably keep its value of 1 for values of x close to the zero point.

As P(x) variation may only be proved in practice and it will be much different for the different lines of a network, a final conclusion is not given here.

9. Requirements for a real-time protocol based on the Local Broadcasters Model

Overview of probable real-time data transmission in the Internet is made in this point. No standard is specified here. Properties of an imaginary protocol realizing data broadcasting are discussed. This non-existing imaginary protocol is called Real-time transfer protocol (RTTP) and doesn't conform to the specified in RFC 1889 Real-time transport protocol abbreviated as "RTP" [1]. The "RTTP" abbreviation and protocol's name are used just for ease, for example ôRTTP packetö means ôa packet containing real-

time data.

9a. Properties of an imaginary protocol

RTTP is based on LBC model. Some RTTP principles may be defined:

1) every host of the network can become a local broadcaster of any program;

2) every host of the network must recognize and check out any RTTP packet which passes through it.

The first conclusion based on these principles is that the whole network must support RTTP. The easiest way for introducing a new protocol is basing it on TCP/IP [2], [3], [4]. If RTTP is based on TCP/IP it is executable in Internet. On the other hand it is impossible to apply a new protocol to the whole Internet shortly, so mechanisms for compatibility between RTTP and non-RTTP hosts must be foreseen.

It must be marked off there's an essential difference between TCP and RTTP (based on LBC) conceptions. The whole network is engaged with the broadcasting in the meaning of ôit changes according to itö, so RTTP encroaches upon IP. TCP envolves the communicating hosts and they exchange data but the other hosts take part only as IP routers.

Yet, there are two assumptions that seem to be valid:

1) the broadcaster supports RTTP;

2) the requester (client) supports RTTP, otherwise it will not be able to interpret the received data.

The real-time transfer doesn't need some TCP functions like the acknoledge and the window size. If there exists RTTP, it will be based on, or even parallel to IP, not based on TCP, i.e. there will exist the independent subjections:

IP -> TCP, which can be IP/RTTP -> TCP;
IP -> RTTP,

and the subjection IP -> TCP -> RTTP will be no valid. ô->ö marks the protocol layering. RTTP abandones some of the ideas of TCP because of some real-time priorities - importance of time and broadcaster's independence. In this case the acknoledge is not desired. The temporary cease of transfer that happens in TCP when zero window size is reported is also unnecessary. Applying the principle of a sigle physical line for the outgoing queues

automatically stops or decreases the transfer when the network is loaded.

The conclusion is that RTTP can hardly be based on the existing TCP protocol.

If the future proves that an RTTP will work satisfactory the only variant for its network implantation is complete compatibility between IP and IP/RTTP hosts.

The things RTTP needs and IP doesn't include are pointed here:

The first is a port (identifier) of the broadcasted program. The broadcaster has an IP address but probably it generates more than one program. On the other hand there can be multiple clients on one IP, so client's port is needed, too.

Every packet must contain stream identifier and internal number. It is assumed packets are small enough and their fragmentation is not needed.

This data can easily be stored in the data field of an IP packet, so there are no problems about it. But a mechanism for recognition of RTTP packets is needed. On the other hands there are three types of packets, if RTTP exists:

1) non real-time (ordinary) IP packet;

2) a real-time packet. There are two kinds of RTTP packets:

- containig broadcasted data;

- containig request, refusals or other official information.

In every IP header there is a reserved for future use bit, there is an option class which is reserved, too, yet there are 152 unassigned values for the "protocol" field, so there are enough possibilities for marking an IP packet as an RTTP one.

RTTP packets must only be marked and all other necessary information can be stored in their data fields.

As real-time transfer can be carried out by IP packets the problem with host compatibility is solved. There appears the problem with the network behaviour.

IP hosts (host will be marked as IP and RTTP) will not manipulate their outgoing queues according to real-time requirements. RTTP packets will not be rejected earlier than their TTL is elapsed. It

will not reject the belated traffic and as there is some belated traffic the broadcaster generates more data than the network can handle. Also, the program will not be received satisfactory by the client.

It was stated above that the broadcaster is an RTTP host. As the network loads its outgoing capacity will fall down. The broadcaster will destroy more and more packets in its outgoing queues so the traffic it generates will also fall down. A balance will be reached, some IP zones will be much loaded, the clients divided by IP zones from the broadcaster will receive its program poorly, but the network will not be blocked up. These problems will be rarer if there are more RTTP hosts.

In a mixed network (IP & RTTP hosts) more difficulties the popular programs will experience. IP hosts will not engage as local broadcasters, so the broadcaster or the local ones will receive more requests than their physical lines are and there will be great losses even at their outgoing queues.

The statement that a mixed network will load but not block up sounds true but must be checked out statistically or by a simulating model. Neither statistics, nor a simulating model is offered here.

If a private program (produced for just a single client) in an RTTP network must be realized the easiest way is to modify its request method. In case the client sends its request to the broadacaster by non-real-time packet, another request sent to the same broadcaster will not be recognized by the hosts on its road, there will be no local broadcasters and no possibility for data confusion. The RTTP packets of the private program will be manipulated as RTTP ones on their road but only the host of the client is going to receive them.

9b. General principles of the chosen model for real-time data transmission

1) Data format:

- the real-time data is transfered through a packet-switching network. Each packet is routed separately;

- every host of the network recognizes the real-time packets;

- the broadcasted data is devided in time units with equal durations. The data for each unit of time is structured as a numbered stream, the packets inside each stream obtain sequent internal numbers. The packets carrying the most important information are the ones with smallest internal numbers;

- every real-time packet carrying data contains stream and internal numbers, and the address of the broadcaster.

2) Requests:

- a request for a public broadcasted program is made by a real-time packet sent to the broadcaster;

- every request is repeated after a specified time interval if the requester still wants to receive the program;

- every host includes all requests routed by it in its table of active requests;

- a request for a program not present in host's table of active requests is transmitted further through the network;

- every program is erased from host's table of active requests if it is not renovated, according to the host, after the specified time interval;

- a request for a program present in host's table of active requests is stopped by the host. It starts to multiply the packets of the program and send them to the new requester, too. After the time interval of the old requester elapses and if its request is renovated the host stops the renewed request and sends its own to the broadcster with its address as destination for the real-time data. The host notifies all clients for the program that it is their local broadcaster. It means that every host must count time intervals for all request in its table of active ones separately. As multiple requests are cascadingly stopped by hosts becoming local broadcasters, the maximum records for a program in a table of active requests equals the number of physical lines of the host that holds up the table;

- a refusal exists. A refusal is made by a real-time packet sent to the local broadcaster or the primary broadcaster if there is no a local one;

- a receive of refusal leads to erasing the client from all the tables of actice requests on the road of the refusal;

- a request for private program is send by non-real-time packet to the broadcster.

3) Real-time data transmission:

- every real-time packet is stored by a routing host in one of its

outgoing queues according to same mechanism with non-real-time packets;

- every newcoming real-time packet with destination the client is added to the queue for the client if in it there is no untransmitted packet with the same or smaller internal number and smaller stream identifier. Every time a real-time packet is added all of the broadcaster's packets are sorted by ascending indexes. They are stored in so sorted order on places of the queue already reserved by broadcaster's packets;

- when a new packet arrives to the server and in the queue for the client there is untransmitted one with the same or smaller internal number and smaller stream identifier all the packets belonging to previous steams are discharged. The packets remaining in the queue are stored on the nearest to the exit places of the queue already reserved by broadcaster's packets.

10. Comparison between the imaginary RTTP and other researches concerning real-time data transmissions

There are lots of works dedicated to real-time transfer. There are standartized protocols and practices. By the time researches concerning this item are too advanced this memo describes just basic ideas. That's because RTTP ideas are in a little different direction than the developed ones.

This section compares RTTP with existing protocols and practices for real-time data transmission. This section doesn't pretend to be comprehensive, its main point is to focus discussion on the need of resource reservation. This section's references are [6 - 10].

As most of the works dedicated to real-time transfer concern video and audio conferences, RTTP points mainly on broadcasting. "Broadcasting" according to its definition in section 2 of this document. Pointing on broadcasting RTTP doesn't provide any reliability which is a basic efford in other real-time researches. A common way for obtaining reliability is the resource reservation process ([5], [6], [7], [8]). Resource reservation is denied by the RTTP concept for some reasons:

- it doesn't seem to be democratic;

- it foresees the possibility of a "busi signal" when there is not enough bandwidth for the transfer;

- if the transmitted data is structured by levels of importance

(idea not present only in this memo, [9]) the bandwidth adjusts itself and it is always the optimum one.

Resource reservation lies on conception that a user has the right to order and receive a specified Quality of Service (QoS). Yet, resource reservation doesn't guarantee connection, it only guarantees it will be good enough if established. Let's look at the commercial aspect of the problem. If a user is often not able to establish connection because his lines don't have enough capacities he will just change his ISP, or pay for a better connection. All users will be satisfied only if they will always obtain the services (at fixed QoS) they will have ordered. It is possible only if all network's lines have enough capacities.

If the whole network consists of lines with big enough capacities, there is no need to reserve resources, is there? So, there is no need to develop complex protocols to take care of the transfer. The imaginary RTTP is very simple and it has another great advantage - it is self-adjusting. It adjusts the transfer without lots of data exchange, in fact without any data exchange. It can be disigned as fully compatible with IP ([2], [4]).

The commercial aspect of the Internet must be treated as a very important one. Currently, World Wide Web users receive transfer by TCP/IP which has no touch with the principle for importance of time. Yet, most of the users always obtain transfer equal or very close to the maximum transfer their network equipment provides. It means that user requirements force the market called Internet to provide features that used protocols do not really guarantee. Implementing RTTP in Internet will lead to the same effect. No matter the protocol will not stick to the principle for importance of reliability it will be in most of the cases reliable, as users will add RTTP reliability as one of their requirements.

If RTTP exists it will not be able always to asure quality real-time delivery but it will be able to asure all over delivery of real-time data. Any user, connected via modem to the Internet will be able to generate a low quality radio program, and if all other users among the network will want to listen to it, they will probably be able to do this.

RTTP's best effords are simplicity and democracity.

11. Plan for researches based on this document

A primary mechanism for data division by levels of importance must be created. An exemplary list of these levels is defined by the following sequence:

- low quality sound

- low quality picture with few frames per second

- increasing the number of frames per second

- reaching the necessary number of frames per second at low quality

- reaching a better resolution for the picture

- reaching a better quality of sound

- reaching the maximum sound quality

- reaching a better resolution for the picture

- stereo sound

- reaching a better resolution for the picture

and so on.

Then a model specification of RTTP must be creatred together with the software that will support it. After a simple mechanism for data division and a primary RTTP is created they must be tested in a small network but with many variations of its structure and different capacities of its lines. The primary mechanism for data division by levels of importance doesn't need to be complex, so the testing network will probably have lines with capacities higher than ordinary Internet users obtain.

Not until the network behaviour is examined RTTP must be specified. After a variant of a specified protocol is chosen it must be tested in a bigger than the previous network and if it proves its efficiency it may be proposed for implantation in the Internet.

Then the effords must be directed at mechanisms for real-time data structuring, requiring less traffic than the primary one.

12. Security Considerations

Security issues are not discussed in this memo.

13. References

[1] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[2] Information Sciences Institute, "Internet Protocol", RFC 791, September 1981.

[3] Information Sciences Institute, "Transmission Control Protocol", RFC 793, September 1981.

[4] Baccala, B., "Connected: An Internet Encyclopedia", http://freesoft.org/CIE/index.htm, April 1997.

[5] Borden, M., Crawley, E., Davie, B., Batsell, S., "Integration of Real-time Services in an IP-ATM Network Architecture", RFC 1821, August 1995.

[6] Schulzrinne, H., Rao, A., Lanphier, R., "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[7] Braden, R., Clark, D., Shenker, S., "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[8] ST2 Working Group, Delgrossi, L. & Berger, L. - Editors, "Internet Stream Protocol Version 2 (ST2) / Protocol Specification - Version ST2+", RFC 1819, August 1995.

[9] Gentric et al., "RTP Payload Format for MPEG-4 Streams", work in progress, draft-gentric-avt-mpeg4-multisl-04.txt, May 2001.

[10] Speakman, T., Farinacci, D., Lin, S., Tweedly, A., Bhaskar, N., Edmonstone, R., Sumanasekera, R., Vicisano, L., "PGM Reliable Transport Protocol Specification", work in progress, draft-speakman-pgm-spec-07.txt, September 2001.

14. Author's Address

Dimitar Aleksandrov
Vladislavovo, 7-12-93
9023 Varna BULGARIA
Phone: +359 98 425788
EMail: rttp@over-ground.net

15. Full Copyright Statement

This document expires June 19, 2002.