CIRCUIT EMULATION OVER IP NETWORKS

Raffaele Noro, Maher Hamdi and Jean-Pierre Hubaux
*Institute for computer Communications and Applications,
EPFL-DI-ICA, 1015 Lausanne, Switzerland
{noro, hamdi, hubaux}@epfl.ch*

Abstract

Recent developments in transmission and switching technology has made available low-cost gigabit network capacity for packet-based WANs (e.g., IP over WDM), that will not require the intermediate SONET/SDH multiplexing layer. For competitive operators that want to minimize their investment by deploying *native* IP over WDM networks, Circuit Emulation over IP gives support to the important class of legacy circuit-based equipment (e.g., T1/E1 multiplexers), which otherwise will require the deployment of an expensive telecommunication infrastructure, including SONET/SDH equipment. At the user interface, Circuit Emulation is required to maintain the same service of a circuit-based network, including synchronization and structured data delivery. In this paper we design the algorithm and the protocol that provide these two features. First, we identify the protocols and mechanisms used in the Internet and their ability to be used for Circuit Emulation. Second, we address the problem of synchronization and we propose an algorithm for efficient source clock recovery and handling of delay variations. Finally, we show through simulations that Circuit Emulation over IP provides the needed service performance at the user interface.

1. INTRODUCTION

In recent years, important progress has been done in transmission and swithing technology and is becoming possible to deploy low-cost, high-speed *packet-based backbones* for Wide Area Networks (WAN), for example by running the Internet Protocol (IP) directly over Wavelength Division Multiplexing (WDM) layers. Native IP over WDM will be the preferred solution for the large majority of new network operators: in-

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35580-1_16](https://doi.org/10.1007/978-0-387-35580-1_16)

deed, it allows to more efficiently transport the increasing amount of Internet traffic across their high capacity transmission infrastructure (e.g., CATV and/or fiber), and without investing in expensive telecommunication equipment used for Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) networks.

However, given the increasing demand for circuit-based leased lines (i.e., T1/E1 and T3/E3), these new operators would maximize their benefits by providing support to legacy *circuit-based equipment*. Circuit Emulation over IP permits to map T1/E1 traffic (as well as other digital signals of the Plesiochronous Digital Hierarchy- PDH), in IP packets to be transported over an Internet-optimized infrastructure.

The mapping of isochronous traffic, like T1/E1, into IP packets presents several challenges which are only partially addressed in the literature. In particular, as already highlighted in the case of Circuit Emulation over ATM [8], the service offered by Circuit Emulation at the user interface (i.e., the edge between the network and the user equipment) must include *synchronization* and *structured data delivery*.

On the contrary of PDH signals transported in SONET/SDH networks, synchronization cannot be transferred end-to-end in a transparent way, since IP does not provide any mechanism for synchronous transport, such as in the Time Division Multiplexing (TDM) of PDH and SONET/SDH. As a consequence, *clock recovery* and *dejittering* must be performed in the receiver of the IP packets. While the problem of dejittering is somehow similar to the case of packet audio and packet video applications (e.g., [1, 14]), the problem of clock recovery cannot be solved with conventional algorithms based on Phase Locked Loops (PLLs). Under normal conditions of jitter in the Internet, the duration of the convergence period become prohibitively long with PLLs (in the order of minutes or more [9]) for good performance in the steady state, thus causing loss due to overflow/underflow of the dejittering buffer.

The contribution of this paper is twofold. The first contribution consists of a *method for clock recovery* that relies on a Least-square Linear Regression (LLR) algorithm that, unlike PLLs, achieves accurate synchronization within few processing cycles, thus overcoming the limitation of PLLs. The second contribution consists in the *design of the protocol* for supporting the encapsulation and synchronization of PDH traffic in an IP packet stream. The protocol is based on existing features of the Real-Time Protocol/Real-Time Control Protocol (RTP/RTCP [13]), such as the Time Stamp (TS) field -used for dejittering-, and also some extended functionality of the RTP/RTCP, such as the Source Clock In-

dication (SCI) and the Spacing fields -used for clock recovery and data structure delivery, respectively.

The paper is organized as follows. In Section 2 we define the role of the Circuit Emulation adapter, which is the device implementing the interface between the IP network and the user equipment. In Section 3 we describe the problem of synchronizing a circuit-emulation receiver, we survey past work in this area and we propose a solution based on the LLR algorithm. In Section 4 we describe the protocol designed for Circuit Emulation. In Section 5 we evaluate the performance of the protocol through simulations and in Section 6 we give some concluding remarks.

2. DESIGN OF CIRCUIT EMULATION

Circuit Emulation is implemented as a network adapter which permits to the PDH user equipment to plug in the IP network as it was a conventional circuit-switched network (Fig. 1).

Figure 1 System under study; Circuit Emulation over IP (top), compared to the conventional circuit-switched case (bottom)

In the case of Fig. 1, the T1/E1 multiplexer is able to send/receive data units in *structures*, which correspond to groups of bits to write/read simultaneously. The *Transmitter* collects data units from the T1/E1 equipment and write them in packets. The *Receiver* receives packets out of the IP network interface, writes them in a receiver buffer and delivers each data unit at a precise delivery instant, under the control of the receiver clock that needs to be properly synchronize.

Designing Circuit Emulation consists of designing the functionality allocated to both Transmitter and Receiver. To accommodate the requirements of different synchronous applications [8], the circuit emulation service should provide:

- structuring of the data units to be output (any group of bits and bytes - including a whole packet)
- delivery of data units at the specified, varying rate (the traffic generated by the multiplexer may vary, according to the number of input lines that are simultaneously active)¹
- constant end-to-end delay (of individual data units)
- handling of data losses (this point is not treated in this paper and is left for further study)

It can be observed that the need for constant *end-to-end delay* puts stringent requirements on the *synchronization* between Transmitter and Receiver. Indeed, to provide constant end-to-end delay, the Receiver must be equipped with a buffer that absorbs early arrivals of packets. The stability of this buffer (i.e., no overflow/underflow) is guaranteed only when the clock rates of the Transmitter and of the Receiver are strictly the same, that is, when clocks are synchronized.

Source clock recovery is therefore the first functionality to consider for Circuit Emulation.

3. SOURCE CLOCK RECOVERY

Clock recovery (or synchronization) is a critical issue in any synchronous system. Fig. 2 shows the effect on the receiver buffer occupancy when the receiver clock is 1% slower than the transmitter clock. The x-axis is the time seen by the source, and the y-axis is the level of

Figure 2 Overflow due to a slower receiver clock

a receiver buffer. Buffer overflow will occur since the rate seen by the receiver is different of the rate seen by the sender

The skew between source and receiver clock is responsible for the long term variation of buffer level observed in Fig. 2. The short term fluctuations of the buffer level are due to the varying network delay and play-out rate.

Although synchronization schemes without any synchronized clock have been proposed, optimal performance is clearly achieved when the receiver clock is somehow synchronized to the source clock. Clock synchronization schemes consists of a protocol and a mechanism. The protocol is based either on information contained in the data stream itself (e.g., SRTS of AAL type 1 [8]) or on a synchronization flow independent of the main data stream (e.g., NTP [9]). In the last case the synchronization is either global (e.g., such as provided by the GPS) or point-to-point.

There are numerous synchronization mechanisms that are described in the literature. PLLs are extensively used to synchronize telecommunications equipment [4] as well as multimedia systems [12]. In [5] a probabilistic method for reading remote clocks in distributed systems subject to unbound communication delay is proposed. The method becomes more accurate as the round-trip time becomes shorter. Moon et al. [10] have recently proposed a scheme for removing clock skew in traces of network delay measurements. This scheme, based on a linear programming algorithm, seems most appropriate for post-processing of network delay measurements.

In the Internet, the Network Time Protocol (NTP, [9]) consists of a

set of time servers dispersed worldwide. NTP messages are processed by servers and clients in order to synchronize clocks with an absolute time base, the Coordinated Universal Time (UTC). NTP is built on top of IP and uses UDP. A digital PLL is employed as synchronization mechanism. PLL design is such that in the presence of an high amount of jitter, such as in the Internet, synchronization is maintained only at the expense of an intolerable long transient period [11]. It may be therefore difficult to maintain synchronization of dispersed host with the required accuracy, even if it looks substantially appropriate for LAN applications.

We propose an alternative synchronization method based on a Least-square Linear Regression (LLR) algorithm, which slaves the receiver clock to the transmitter clock with greater accuracy. We define the following terminology used for the LLR:

- clock indication: sample of the source clock,
- source clock: master time measured at the source,
- receiver clock: free-running time measured at the receiver,
- recovered clock: slaved time obtained with synchronization,
- t_s : measurement of the source clock,
- t_r : measurement of the receiver clock when source clock is t_s ,
- $\hat{t}_s$: estimate of the recovered clock at t_s ,
- t_{si} : value of the i -th clock indication issued by the source,
- t_{ri} : receiver clock when the i -th clock indication is received,
- a : actual clock skew between source clock and receiver clock (i.e., frequency ratio),
- b : initial clock offset between source clock and receiver clock,
- $\hat{a}$: estimate of the clock skew,
- $\hat{b}$: estimate of the clock offset.

The time model [5] that relates source and receiver clock is of the form

$$t_s = a \cdot t_r + b \quad (1)$$

a and b are typically unknown and the goal of any clock recovery method consists in estimating $\hat{a}$ and $\hat{b}$ such that the error between t_s and the estimate $\hat{t}_s = \hat{a} \cdot t_r + \hat{b}$ is minimized. Our approach is based on the observation that the points (t_{si}, t_{ri}) are distributed close to a straight line (i.e., on the straight line if network delay is constant) whose slope and offset are, respectively, the estimate of clock skew and the estimate of clock offset (minus the average of the network delay). The values of $\hat{a}$ and $\hat{b}$ are computed by minimizing the Mean Square Error (MSE) with least-square linear regression

$$\hat{a} = \frac{m \sum t_{si}^2 - (\sum t_{si})^2}{m \sum (t_{si} t_{ri}) - (\sum t_{ri} \sum t_{si})}, \quad (2)$$

$$\hat{b} = \frac{\sum t_{si} \sum (t_{ri} t_{si}) - (\sum t_{ri} \sum t_{si}^2)}{m \sum (t_{si} t_{ri}) - (\sum t_{ri} \sum t_{si})} \quad (3)$$

Only the m last received clock indications are processed by the algorithm. This is known as the Durbin algorithm and is based on the assumption of stationary distribution of jitter. Similar methods for non-stationary distribution also exist. The method discussed here has a shorter duration of the transient period, since m (the upper bound) does not need to be very large (in the order of 100). It is also much faster than a PLL with equivalent steady-state accuracy of the recovered clock [11]. This avoids severe losses due to overflows and underflows of the receiver buffer during the transient period. Furthermore, the method can be as simple as a second-order PLL when recursion is used [11].

Performance

The source clock recovery method is simulated in the following conditions:

- Simulated duration of 160 s;
- Source clock indication generated at 100 ms rate;
- $m = 1000$;

For the network jitter, we refer to the model of [2] which is further discussed in the Section 5 for the simulation of the network delay. Fig. 3 shows the distribution of the network delay around its average value.

Figure 3 Jitter during the experiment

Fig. 4 (left) and Fig. 4 (right) illustrate the performance of the clock recovery method in the presence of the network jitter. The x-axis represents the master time measured at the source. In Fig. 4 (left) we plot the accuracy of the recovered clock, defined as the difference $t_s - \hat{t}_s$ and measured in ms. In Fig. 4 (right) we plot the stability of the estimate clock skew, defined as the difference between the output of the Eq. 3 and its actual value (expressed in parts per million, ppm).

Figure 4 The accuracy of the recovered clock (left), and the stability (right)

The comparison of Fig. 3 and Fig. 4 (left) shows that the residual jitter of the recovered clock is 10 times smaller than the network jitter in the steady state. We can also observe a transient period of around 40 s. The steady-state is reached when a sufficient number (i.e., 400 in this case) of source clock indications have been processed by the LLR

algorithm. As shown in [11] the transient period is around 100 times shorter than the transient period of a PLL with an equivalent steady-state accuracy.

4. CIRCUIT EMULATION FUNCTIONALITY

In this section we analyze in greater detail the functionality needed for Circuit Emulation and we define the use of RTP/RTCP for this application. In the context of packet audio and packet video over the Internet, the RTP/RTCP [13] is well established as the standard for transporting time-sensitive data. Its native functionalities include timestamping of packets and sequencing. A number of implementations of real-time applications is based on RTP as the transport layer of audiovisual data (e.g., [6]).

In our case, we extend the features of the RTP in order to implement the functionalities of Circuit Emulation: handling of data structures, the handling of delay variation, and the clock recovery at the receiver.

Functional Blocks

The Fig. 5 illustrates both Transmitter and Receiver. We define the *spacing* as the time (corresponding to the emission rate) between two consecutive structured data units which enter the system, the *timestamp* as the time at which a packet leaves the source, *structure* as the number of bits that compose a data unit and *d_max* as the maximum packing delay of a data unit.

The maximum network delay influences the design of the receiver buffer. This bound can be a guaranteed value, a stochastic value (i.e., via a reservation mechanism [3]) or an estimated value (e.g., in best effort Internet). Separate reservation mechanisms are clearly necessary to optimize the end-to-end performance of circuit emulation.

The payload of a packet is filled-up by consecutive data units and is transmitted either when its fullness has reached a predefined level (i.e., a bound of the packing delay) or when a change in the spacing of arrival units has been detected². Therefore, packets can be of variable size.

While packets suffer from *network jitter*, individual data units suffer also from *packing jitter*. The purpose of the receiver buffer is thus to absorb both transmission and packing delay [8], while the role the clock recovery method is to guarantee that the spacing between two data units at the output is the same of the input (which does not mean

Figure 5 Functional blocks of Circuit Emulation adapters

that all data units have the same spacing - the rate can be variable). The receiver clock can optionally generate a synchronization signal, if specifically required by some of the equipment attached to the output lines (e.g., video decoders) of the demultiplexer.

The data structure is identified at the beginning of the session and must be communicated during the session setup phase via an RTCP message. Even if under normal conditions is not supposed to change, it can optionally be dynamically modified during the session with a successive RTCP message. Data structures can be any group of bits or bytes and even whole packets³ that are exchanged between the circuit emulation adapter and the application. The spacing between consecutive data units is expressed in units of the Transmitter clock and is encoded in each RTP packet header. The spacing is combined with the timestamp to obtain the delivery instant of every individual data unit.

The receiver buffer handles the variability of delay. The design of the buffer depends on several factors: the network delay, the packing delay, the input rate and the maximum packet size. The design of the receiver buffer is a well studied problem [1, 14] which is not tackled here. The maximum packing delay is provided through RTCP messages, while the characteristics of the network delay (e.g., average and maximum value) has to be provided by the reservation protocol (or estimated in the case of best effort).

The source clock recovery method has been described in Section 3. The source clock indications are carried in an extension to the RTP packet header. For interoperability with existing implementations, we

consider that the common timebase has a frequency resolution f_n equivalent to the 4.3 GHz resolution of the NTP (at most). The source frequency is expressed as f_n/x , where x is a rational. Information about the source clock is periodically inserted by the source to allow the receiver to adjust its clock drift. It is important to note that the receiver can ignore this information and operate with the local clock or NTP (i.e., reducing the accuracy of the clock synchronization).

RTP/RTCP Extensions

Circuit Emulation with RTP/RTCP is supported by two types of *RTCP messages*, to indicate *d_max* and *structure*, and by two types of *RTP packet extensions*, for carrying source clock indications and/or spacing.

A 32-bit field of a Sender Report (SR) or Application-defined (APP) RTCP message supports this function. It is partitioned in two 16-bit fields: the first field carries the value of *d_max* in units of the source clock, the second field indicates the length of data units in bits. Fig 6 illustrates both cases.

- V: version of RTP
- P: padding, set to one if padding octets are included at the end of packet
- RC: number of report blocks in the packet
- PT: payload type
- length: length of packet in 32-bit words minus one
- SSRC: synchronization source originating the packet
- subtype: to allow subtypes or application-dependent data
- name: 4 octets to identify the name of the APP
- d_max: maximum packet delay in source clock units (zero if undefined)
- structure: length of data units (in bits, zero means one bit)
- packet number: first RTP data packet subject to d_max and structure

Figure 6 Usage of RTCP packets

RTP packet header extensions contain the spacing, the source clock indication and the timestamp. The RTP packet payload contains the

user data. In the header extension, a first 16-bit field is used to encode the spacing in units of the Transmitter clock, a second 64-bit field (if present) carries the source clock indication. The overhead introduced by the latter must be kept within a reasonable limit (e.g., not exceed the 1 % of the session traffic, for example one source clock indication field each 64 ms for a 1 Mbit/s session). This is illustrated by Fig. 7

- profile: defines the CES
- length: length of header extension in 32-bit words minus one (either 1 or 3)
- spacing: temporal distance between data units in Transmitter clock units
- source clock indication: absolute value of source clock in units of 4.3 GHz
- reserved: for future use

Figure 7 Usage of RTP packets

5. RESULTS

We have carried out simulations of Circuit Emulation for the transport of coded audiovisual information in MPEG-2 Transport Stream [7] format. The goal is to show that the delay jitter of data units, i.e., the end-to-end performance, is below the threshold of ± 4 ms defined for this particular type of applications. The parameters of the simulation are the following:

- Source emission rate varying between 1 Mbit/s and 250 kbit/s;
- Size of data units of one byte;
- Size of RTP packet payload of 188 bytes;
- Geometric distribution of the network jitter with 100 ms of minimum delay, 780 ms of maximum delay and 154 ms of average delay (Fig. 3);
- Window length of the source recovery method $m = 1000$;

- Source clock indication generated every 100 ms;
- Simulated duration of 160 s

The network jitter model is consistent with the results presented in [2]. The inter-arrival process of packets emitted at regular intervals δ fits with a geometric distribution

$$\text{Prob}\{(t_r^{i+1} - t_r^i) = k \cdot \delta, k \geq 0\} = p \cdot (1 - p)^k \quad (4)$$

where p is a decreasing function of δ . In our experiment we have taken $\delta = 3.4$ ms and $p = 0.52$ ⁴. We also assumed lossless transmission.

directly affected by the residual jitter of the recovered clock (see Section 3). Fig 8 (left) shows the delay experienced by each delivered data unit during 16 s of the experiment.

Figure 8 Effect of residual jitter in (left), and buffer occupancy (right)

Fig. 8 (left) shows that circuit emulation provides the required end-to-end performance by maintaining the jitter below ± 4 ms.

The receiver buffer occupancy reflects the fact that data units are generated and consumed at different rate. As shown by Fig. 8 (right), occupancy increases during bursts and tends to be lower during low-activity periods.

It is important to note that the overhead induced by the source clock indication has been maintained below the 1 % of the average rate.

6. CONCLUSION

In this paper we described the design of Circuit Emulation over IP. The Circuit Emulation allows legacy circuit-based equipment to be sup-

ported by IP-based networks. Two purposes are served: network operators has to deploy and maintain only one network infrastructure while users with traditional network access does not need to upgrade their equipment. We identified three key features of Circuit Emulation: handling of data structure, handling of delay variations and clock recovery. We discussed the ability of protocols and mechanisms of the Internet to support the circuit emulation service.

The clock recovery method is the second contribution of this paper. As an improved alternative to NTP, we proposed a method for clock recovery which slaves the Receiver clock to the Transmitter clock in the presence of clock skew and network jitter. The method is based on a Least-square Linear Regression algorithm. Our simulation results show its ability to compensate for severe amounts of network jitter in a short convergence period.

Finally, at the protocol level, we fit the functionality of Circuit Emulation with the features of RTP/RTCP. Clock indications, data rate, and data structure information are some additional feature added to RTP/RTCP. The proposed protocol has a number of characteristics in common with the circuit emulation in ATM. In fact, RTP over UDP/IP plays the same role of AAL type 1 on top of ATM (while the Least-square Linear Regression mechanism plays the same role of the SRTS mechanism). One additional feature is the ability to support variable rate streams.

We emphasize that these conclusions are based on the assumptions used in this paper. Factors not taken into account can alter the conclusions. Examples of these factors are the effects of packet loss on the clock recovery process, the handling of data loss at the delivery process and the appropriate interface of the protocol to external reservation protocols (i.e., the user-to-control plane interface). The impact of these factors on Circuit Emulation are interesting subjects for future research.

Notes

¹ This represents an optional feature. However, the adapter itself can be arranged in order to not transmit the idle time slots of the PDH signal, since they can be easily regenerated by the Receivers. The usage of network bandwidth results optimized

² assumption is made on the fact that emission rate is changed only at packet boundaries

³ in RFC2250 [6] a single Transport Stream packet can be encapsulated in an RTP packet (although it is not an efficient scheme). In this case it is the whole payload of the RTP packet that is delivered to the Transport Stream decoder

⁴ This means that 52 % of packets are assumed to reach the receiver in bursts, as a consequence of network overload. δ is the average of the packet emission interval

REFERENCES

- [1] H. M. Ahmed, "Adaptive Terminal Synchronization in Packet Data Networks", in *Proc. of Globecom'89*, pp. 728–732, Dallas, Texas, Nov. 1989.
- [2] J.-C. Bolot, "Characterizing End-to-End Packet Delay and Loss in the Internet", *Journal of High-Speed Networks*, vol. 2, pp. 305–323, Dec. 1993.
- [3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReServation Protocol (RSVP) - Version 1 Functional Specification", IETF RFC 2205, Sept. 1997.
- [4] S. Bregni, "A Historical Perspective on Telecommunications Network Synchronization", *IEEE Communications Magazine*, vol. 36, pp. 158–166, June 1998.
- [5] F. Cristian, "Probabilistic clock synchronization", *Distributed Computing*, vol. 3, pp. 146–158, Dec. 1989.
- [6] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video", IETF RFC 2250, Jan. 1998.
- [7] ISO/IEC, *Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems*, Oct. 1994, DIS 13818-1.
- [8] ITU-T Study Group 13, "B-ISDN ATM Adaptation Layer Specification, Type 1 AAL", ITU-T Recommendation I.363.1, Aug. 1996.
- [9] D.L. Mills, "Internet Time Synchronization: The Network Time Protocol", *IEEE Transactions on Communications*, vol. 39, pp. 1482–1493, Oct. 1991.
- [10] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements", in *Proc. of Infocom'99*, pp. 227–234, New York, Mar. 1999.
- [11] R. Noro and J.-P. Hubaux, "Clock Synchronization of MPEG-2 Services over Packet Networks", *Telecommunication Systems*, vol. 11, pp. 3–16, Mar. 1999.
- [12] M. Perkins and P. Skelly, "A Hardware MPEG Clock Recovery Experiment in the Presence of ATM Jitter", ATM Forum contrib. #94-0434, Apr. 1994.
- [13] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", IETF RFC 1889, Jan. 1996.
- [14] H. Schulzrinne, "Voice Communication Across the Internet: A Network Voice Terminal", Technical Report 92-50, Univ. of Massachusetts, July 1992.