PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :		(11) International Publication Number:	WO 00/45581
H04M 11/00	A2	(43) International Publication Date:	3 August 2000 (03.08.00)

(21) International Application Number: PCT/US00/02266

(22) International Filing Date: 28 January 2000 (28.01.00)

(30) Priority Data:

60/117,916	29 January 1999 (29.01.99)	US
09/318,785	25 May 1999 (25.05.99)	US

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(71) Applicant: DATA RACE, INC. [US/US]; 12400 Network Boulevard, San Antonio, TX 78249 (US).

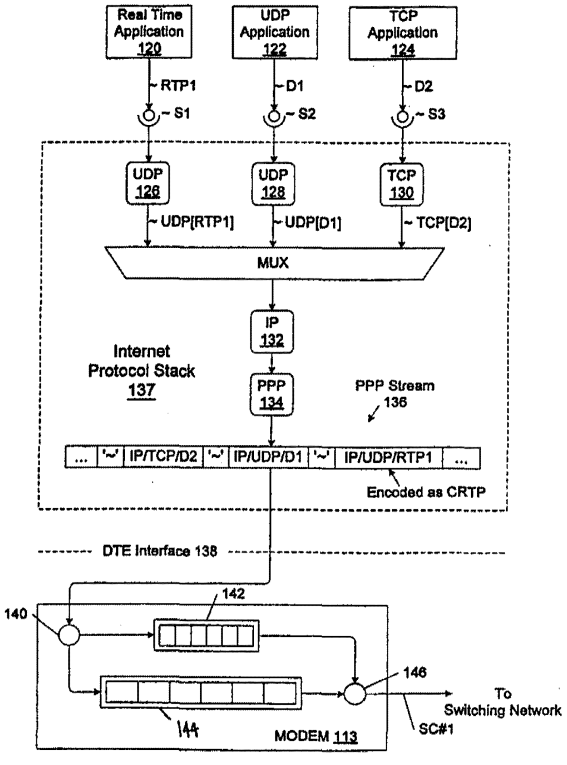
(72) Inventor: OLIVER, David, C.; 8721 Mountain Top, San Antonio, TX 78255–3528 (US).

(74) Agent: CONLEY, ROSE & TAYON, P.C.; P.O. Box 398, Austin, TX 78767–0398 (US).

(54) Title: MODEM TRANSFER MECHANISM WHICH PRIORITIZES REAL–TIME DATA TRANSFER OVER REGULAR DATA TRANSFERS

(57) Abstract

A modem transfer mechanism which provides for multiplexed transfer of real–time data and regular data across a communication medium (e.g. a dialup modem link) where real–time data is prioritized over regular data. Real–time latency is minimized. When the modem receives and parses a data stream comprising real–time data and regular data from a computer (or other DTE), data conforming to a real–time protocol is stored in a real–time buffer and transmitted expeditiously using a special protocol which is described herein. Optionally, real–time data is compressed before transmission onto a communication medium (e.g. an analog phone line). Data not conforming to a real–time protocol may be buffered for transmission in due course. The modem is also configured to receive a data stream including multiplexed real–time data and regular data from the communication medium. The modem demultiplexes the real–time data and the regular data. The real–time data is stored into a second real–time buffer and the regular data is stored into a second regular buffer. Voice frames are delivered to the computer (or other DTE) with priority over regular data frames. Using a pair of such modems to establish a modem link across a switching network (e.g. the PSTN) allows real–time data to be transmitted with a minimum of real–time delay while simultaneously transmitting regular data.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Title: Modem Transfer Mechanism which Prioritizes Real-Time Data Transfer over Regular Data Transfers

Field of the Invention

The present invention relates to the transmission of real-time data (such as voice data) and non-real-time data, and more particularly to a system and method for minimizing the delay of a real-time data transfer while simultaneously performing a non-real-time data transfer over a communication link (such as a dialup modem link).

Description of the Related Art

In the prior art, a modem link, such as that depicted in Fig. 1, may have been used to transfer real-time data or non-real-time data between two digital devices. Any device capable of sourcing and/or sinking digital data will be referred to herein as a data terminal equipment (DTE). In the modem link of Fig.1, a first DTE, i.e. DTE#1, provides a data stream to modem #1. Modem #1 transmits the data stream to modem #2 across a switching network, such as the Public Switched Telephone Network (PSTN) or the Integrated Services Digital Network (ISDN). Modem #2 receives the data stream from the switching network and forwards the data stream to a second DTE, i.e. DTE#2. Similarly, DTE#2 may provide a second data stream to modem #2 for transmission to modem #1 across the switching network. Modem #1 forwards the second data stream to DTE#1. Modem #1 couples to the switching network through a first subscriber connection SC#1, and modem #2 coupled to the switching network through a second subscriber connection SC#2. For example, subscriber connection SC#1 may be an analog telephone line, i.e. a Plain Old Telephone Service (POTS) line, and subscriber line SC#2 may be an ISDN line.

DTE#1 and DTE#2 may be computers, in which case Figs. 2A&2B illustrate a typical arrangement for the software organization of the first computer DTE#1. Fig.2A depicts the transmission of data originating from software applications running on the first computer DTE#1. Fig. 2B depicts the reception of data by the same software applications. The software applications transfer data to/from an Internet Protocol Stack which may be part of an operating system. The Internet Protocols Stack comprises a multi-layered system of software protocol modules. The Internet Protocol Stack communicates with modem #1 through a DTE interface. The DTE interface typically has a much higher bandwidth than the bandwidth attainable through the switching network.

As shown in Fig. 2A, software applications running on the first computer DTE#1 submit data to "sockets" S1-S3. Each socket passes the application data to a corresponding transport protocol module. The transport protocol modules implement transport protocols such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). TCP is a reliable protocol, and thus, is well suited for applications such as file transfer or email which require reliable transfer. UDP makes no provision for reliability, and thus, it executes with lower latency (delivers the data more quickly) than TCP. Therefore, UDP is typically the choice for real-time applications.

The first computer DTE#1 includes a central processing unit and memory for executing software applications. For example, a real-time application may generate a stream of Real Time Protocol (RTP) packets of which a representative packet RTP1 is shown. The real-time application is shown supplying the packet RTP1 to socket S1. Internet Telephony, slow scan TV and telemetry are examples of real-time applications.

In addition to the real-time application, the first computer DTE#1 may execute other UDP applications or TCP applications: one of each is shown in Fig. 2A. The UDP application is shown providing a data packet D1 to socket S2. The TCP application provides a bit stream D2 to the socket S3.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 3 of 38

A first UDP module receives packet RTP1 from socket S1 and encapsulates this packet in a UDP packet. The resulting UDP packet is denoted as UDP[RTP1]. A second UDP module receives the data packet D1 from the second socket S2 and encapsulates the data packet D1 in a UDP packet. The resulting UDP packet is denoted as UDP[D1]. A TCP module encapsulates the bit stream D2 (or a portion thereof) in a TCP segment denoted TCP[D2]. The UDP and TCP packets are multiplexed and presented to an Internet Protocol (IP) module. The IP module encapsulates the UDP and TCP packets according to the Internet Protocol. The resulting IP packets are presented to a Point-to-Point Protocol (PPP) module. The PPP module encodes the stream of IP packets into a stream of PPP frames. Each PPP frame is separated by a tilde character. A portion of the PPP stream is shown. For example, the PPP stream includes (a) a first frame which encapsulates a packet denoted IP/TCP/D2 whose payload D2 originated from the TCP application, (b) a second frame which encapsulates a packet IP/UDP/D1 whose payload D1 originated from the UDP application, and (c) a third frame which encapsulates a packet IP/UDP/RTP1 whose payload RTP1 originated from the real-time application. The PPP stream is supplied to modem #1 through the DTE interface. Modem #1 transmits the PPP stream to modem #2 (not shown) through the switching network.

It is noted that the PPP module may perform header compression on the IP/UDP/RTP encapsulated packets. For example, the Compressed Real Time Protocol (CRTP) specifies a scheme for performing such header compression. In addition, the PPP module may perform TCP/IP header compression as well as TCP/IP data compression.

As shown in Fig.2B, modem #1 receives from modem #2 (not shown) a second PPP stream. The second PPP stream is supplied to a PPP module in the Internet Protocol Stack. A portion of the second PPP stream is shown at the input of the PPP module. The second PPP stream may include (a) a first frame which encapsulates a packet IP/TCP/D4 whose payload D4 is targeted for the TCP application, (b) a second frame which encapsulates a packet IP/UDP/D3 whose payload D3 is targeted for the UDP application, and (c) a third frame which encapsulates a packet IP/UDP/RTP2 whose payload RTP2 is targeted for the real-time application. The PPP module recovers a stream of IP packets from the second PPP stream. The IP module recovers UDP and TCP packets from the IP packets. The UDP and TCP packets are demultiplexed according to their destination port numbers. Each socket is associated with a unique port number. UDP packets which encapsulate RTP, e.g. UDP[RTP2], are sent to a first UDP module which recovers the RTP packet RTP2. The RTP packet RTP2 is supplied to the real-time application through socket S4. Another UDP encapsulated data packet UDP[D2] is supplied to a second UDP module. The second UDP module recovers the embedded data D2 and passes this data to the UDP application through socket S5. TCP encapsulated data TCP[D4] is sent to a TCP module. The TCP module recovers the embedded data D4 and passes this data to the TCP application through socket S6.

While it is desirable to be able to execute a real-time application and a non-real-time application simultaneously, the non-real-time data communicated by the non-real-time applications may compromise the performance of the real-time data transfer, especially in the situations where subscriber connection SC#1 is a low-bandwidth connection such as an analog telephone line. Fig. 3 illustrates how real-time performance may be compromised by the presence of non-real-time data transfers. Subgraph (A) depicts a PPP data stream comprising real-time frames and non-real-time frames in transit across the DTE interface from the Internet Protocol Stack to modem #1. In particular, a first non-real-time data frame D1 is followed by a first real-time frame V1. The first real-time frame V1 is followed by a second non-real-time data frame D2. The second non-real-time data frame D2 is followed by a second real-time frame V2. The first non-real-time frame is transmitted across the

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 4 of 38

DTE interface at time t_0. The real-time data frames V1 and V2 are transmitted at times t_1 and t_2 respectively. Quite often, real-time applications generate real-time frames with a predetermined periodicity. For example, the G.729 speech compression standard prescribes that frames be generated periodically with a 10 millisecond period.

Subgraph (B) depicts the same real-time frames and non-real-time frames as they are being transmitted onto the subscriber connection SC#1 by modem #1. It is noted that the time duration of the frames is significantly lengthened due to the low bandwidth of the subscriber connection SC#1 as compared to the bandwidth of the DTE interface. A large non-real-time data frame such as D1 may significantly delay succeeding real-time frames. For example, real-time frame V1 is delayed from time t_1 to time t_4 because non-real-time frame D1 is elongated in transmission onto the subscriber connection SC#1. Furthermore, the real-time delay may accumulate when a succession of large non-real-time frames occurs in the transmitted stream. Observe that the second non-real-time packet D2 adds to the delay already developed by non-real-time data frame D1, and thus, time delay t_5-t_2 is larger than delay t_4-t_1. In other words, real-time frame V2 is delayed more than real-time frame V1. Such accumulating delays typically compromise the quality of the reconstructed real-time signal at the real-time destination. For example, a human auditor at the real-time destination may find the reconstructed voice signal to contain annoying pauses. When these pauses occur with sufficient frequency and/or duration, the human auditor may not be able to make sense of the reconstructed voice signal. It is noted that DTE#2 may not necessarily be the real-time destination. For example, DTE#2 may serve as a gateway to the Internet in which case the real-time destination may be yet another computer coupled to the Internet.

Furthermore, the time interval between successive real-time frames may be greatly increased due to intervening non-real-time frames. For example, non-real-time frame D2 is large enough to stretch the time interval between frames V1 and V2 to the significantly larger value t_5-t_4 during transmission onto the subscriber connection SC#1. This lengthening of the time interval between successive real-time frames due to the low bandwidth subscriber connection SC#1 compromises the quality of the reconstructed real-time signal at DTE#2. For example, a real-time voice stream may manifest a broken (i.e. discontinuous) sound to the human ear when non-real-time data is simultaneously transmitted.

It is noted that buffering within modem #1 induces a time delay on the PPP stream. For example, the time difference t_3-t_0 may be attributed to buffering delay within modem #1. For example, the v.42 protocol typically involves buffering data in packets of size 128 bytes.

Subgraph (C) depicts the same real-time frames and non-real-time frames as they are being received from subscriber connection SC#2 by modem #2. It is noted that the frames of subgraph (C) are delayed in time as compared to the corresponding frames of subgraph (B) due to propagation through the switching network. Subgraph (D) depicts the same real-time frames and non-real-time frames as they are delivered to DTE#2 from modem #2.

It is noted that modem #2 performs data buffering before transmitting data to DTE#2. This induces an additional time-delay in the PPP stream. For example, the time difference t_7-t_6 may be attributed to buffering delay.

The deleterious effects discussed above are so pronounced that non-real-time data streams must typically be disabled as, e.g., by closing the non-real-time applications in order to obtain an acceptable level of performance in the real-time data transmission. For example, an Internet Telephony correspondent who accesses an Internet Service Provider (ISP) through the switching network with an analog phone line SC#1 must typically

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 5 of 38

close/suspend other software applications which might transmit and/or receive data across the modem link in order to obtain acceptable voice quality.

Conventional modems contribute to the problem of real-time frame delay because they naively transmit data frames in the order they are presented by DTE#1. Thus, a substantial need exists for a modem which could concurrently transfer real-time data and non-real-time data over a low-bandwidth subscriber connection such as an analog telephone line without compromising the performance of the real-time data transmission. In particular, it would be very desirable to have a modem which could transmit voice data with minimum latency (i.e. time delay) while concurrently transferring non-real-time data.

Internet Telephony is a popular example of a real-time application. Internet Telephony applications (such as NetPhone®) use Voice over Internet Protocols (VoIP) with RTP to make telephone connections between computer users. A standardized Voice over Internet Protocol is described in ITU Recommendation H.323 (02/98) - Packet-Based Multimedia Communications Systems. RTP is elaborately described in ITU Recommendation H.225.0 (02/98) - Call Signaling Protocols and Media Stream Packetization for Packet-Based Multimedia Communication Systems. Internet Telephony applications work well on direct local area network (LAN) connections, or over a segment of the Internet which is able to ensure an acceptable Quality of Service (QoS) level. In other words, the LAN or Internet segment needs to be able to provide a minimum bandwidth and maximum latency for packet transmissions between Internet Telephony correspondents. Bandwidth reservation protocols such as the Resource Reservation Setup Protocol (RSVP) have been developed to allow applications to reserve resources along the path from data source to data destination.

However, over a low-speed dial-up (modem) connection to the Internet, which normally uses the Point-to-Point Protocol (PPP), there are a number of problems with Internet Telephony (IT) as follows.

(1) The typically small voice packets generated by the IT application are triply encapsulated: first in RTP, then UDP, then IP. This RTP/UDP/IP protocol stack adds a large overhead to the small voice packets.

(2) As alluded to above in conjunction with Fig. 3, modem buffering can delay voice packets.

(3) Some modems perform data retransmission upon error. However, this facility is not suitable for the voice packet transmissions. Voice packets which are transmitted with error should be dropped (i.e. ignored). Therefore, modems which perform data retransmission upon error needlessly consume transmission bandwidth and induce added voice delay.

(4) As alluded to above, voice packets may be delayed by concurrent data transmission, as packets are sent in the order they are received.

(5) Digital Simultaneous Voice & Data (DSVD) Modems address the problem of data packets delaying voice packets, but only handle special-purpose voice sources, such as a locally terminated analog voice circuitry.

If a user desires to transfer both real-time data and non-real-time data simultaneously, one solution is simply to use two connections (such as analog telephone connections) to the switching network, one reserved for voice traffic and the other reserved for data traffic. However, this solution suffers because of its expense and because of its inefficiency - the user may not always be performing real-time and non-real-time transfers concurrently. There is also the expense of requiring two modems to handle the two connections.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 6 of 38

Conventional modem protocols for transmitting voice and regular data together (e.g., DSVD and the V.70 suite) provide a way to transport locally originated (e.g. from a connected telephone handset) voice packets across a modem with some level of latency control, depending upon the protocol. However, they do not provide a mechanism to accept and deliver general purpose real-time data packets, such as Voice over IP. To modify the conventional methods would require significant change in the software running on the host (user and server) computers to direct voice packets to a high priority port on the modem. No standardized way of directing the data exists, and such changes would require access to multiple sources of software (IP protocol stacks, Windows sockets, etc.), which may be proprietary and often unavailable.

It is noted that there exist a class of modems called Digital Simultaneous Voice and Data (DSVD) modems which are configured for transferring voice and non-voice data across analog telephone lines. Fig. 4 illustrates a DSVD modem link. A first DSVD modem, i.e. DSVD1, couples to the PSTN through a first analog telephone line L1, to DTE#1 through a DTE interface, and local telephone TEL1. A first user situated at DTE#1 speaks into the handset of local telephone TEL1. The first user's voice is digitized and compressed by circuitry in the first DSVD modem DSVD1. The first DSVD modem DSVD1 receives non-voice data from DTE#1, multiplexes the compressed-voice data with the non-voice data, and transmits the multiplexed stream to a second DSVD modem, i.e. DSVD2, through the PSTN. The second DSVD modem DSVD2 demultiplexes the multiplexed stream. The recovered voice data is decompressed and then converted to analog form for presentation to the handset of local telephone TEL2. The non-real-time data recovered from the multiplexed stream is transmitted to DTE#2. The same process occurs in the opposite direction, thus enabling the two subscribers to concurrently engage in voice and data communication over analog telephone lines L1 and L2.

The multiplexed voice/data stream includes separate voice frames and data frames. A frame typically includes markers which define the start and end of the frame. A frame may additionally contain information about the data format, etc., in a frame header. The separate frames are multiplexed so that bandwidth is statistically divided between voice frames and data frames. For example, a 28.8 Kbps DSVD modem may transmit non-voice frames at 19.2 Kbps and voice frames at 9.6 Kbps. However, DSVD modems have several disadvantages. By inserting a substantial amount of regular data between voice frames, the latency (i.e. time delay) between voice frames is increased. This increase in voice latency may cause the reconstructed voice signal to appear "jerky" (i.e. discontinuous).

The increase in voice latency may be avoided by transmitting voice frames as often as they are generated by the voice compression circuitry, and transmitting non-voice data in the time intervals between voice frames. The size of data frames must be controlled so as to fit between successive voice frames. Segments of non-voice data which are larger than a critical size may be fragmented into two or more frames. The critical size is determined by the time interval between voice frames and the signaling rate across the analog phone line L1. While this multiplexing scheme may control voice latency, it increases overhead in the non-voice transfer due to the overall increase in the number of non-voice frames. Each frame includes a header. Increasing the number of voice frames to transmit a given segment of non-voice data implies a loss of available transmission bandwidth. Therefore, controlling the non-voice frame size to improve voice latency has the effect of decreasing overall data throughput. Several ad hoc DSVD implementations have been advanced by telecommunication vendors. However all of these implementations suffer from the above noted disadvantages.

Another disadvantage of DSVD is that a more complicated modem is required having at least two ports or interfaces, one for the DTE and another for the phone.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 7 of 38

One technique for transferring simultaneous voice and non-real-time data is displayed in the Integrated Services Digital Network (ISDN). ISDN is an international standard which defines a digital interface between subscribers and the PSTN. The digital interface provides increased transmission rates as compared to the rates attainable over analog telephone lines. A subscriber must be equipped with a special ISDN modem which performs low-level time-division multiplexing of voice and regular data. The ISDN modem allocates a fixed portion of the transmission bandwidth to the voice data. Unfortunately this allocation scheme implies that the regular-data bandwidth cannot expand during those time when the full voice bandwidth is not being used, and vice versa. ISDN service must be subscribed to from the phone company and may not be available in all areas. ISDN service is typically more expensive than analog telephone service. In addition, ISDN modems are typically more expensive than conventional modems. Thus, users desirous of performing simultaneous real-time and non-real-time data transfers are often discouraged from considering ISDN.

DSVD is addressed in the International Telecommunication Union (ITU) v.70 standard. This uses a technique for implementing simultaneous real-time and non-real-time data transmission by interrupting regular data frames in a modem connection using non-standard "abort" tokens in order to introduce real-time data. The suspend/resume tokens of the ITU v.76 recommendation is such an approach. However, using "abort" tokens has the disadvantage of requiring special hardware to create and detect the abort tokens. Furthermore, overhead is increased by the use of the tokens and additional frame information.

As described above, all the existing solutions for providing simultaneous real-time and non-real-time communication suffer from one or more of the following limitations:

(a) they require special hardware;

(b) they involve increased complexity and/or expense;

(c) they can only be applied to special-purpose voice sources; or

(d) they negatively impact voice latency.

It is therefore desirable to provide a low latency technique for communicating real-time and non-real-time data over a conventional modem connection with minimal impact on performance.

Summary of the Invention

The problems described above are largely solved by a communication system and method according to the present invention which provides for multiplexed transfer of real-time data and regular (e.g. non-real-time) data across a communication medium (e.g. a dialup modem link). The multiplexing scheme employed by the communication system prioritizes real-time data over regular data, and thereby minimizes transmission latency for the real-time data stream.

The present invention contemplates a modem which is configured for coupling to a host computer across a DTE interface. The modem is configurably aware of real-time protocols such as CRTP or RTP. When the modem receives a data stream comprising real-time data and regular data from a high-speed computer port, it parses the data stream for real-time data. Data which conforms to a real-time protocol is stored in a real-time buffer and transmitted expeditiously using a special protocol which is described herein. Optionally, real-time data is compressed before transmission onto a communication medium (e.g. an analog phone line). For example, compression may involve throwing away framing information which may be easily reconstructed by a complementary receiving modem. Data which does not conform to a real-time protocol may be buffered for transmission in due

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 8 of 38

course. If the regular data buffer fills up, the DTE port is not "flow controlled", but instead packets of data are discarded. Thus, the real-time data is unimpeded.

The modem described above is also configured to receive a data stream including multiplexed real-time data and regular data from the communication medium. The modem demultiplexes the real-time data and the regular data. The real-time data is stored into a second real-time buffer and the regular data is stored into a second regular buffer. Voice frames are delivered to the DTE with priority over regular data frames. If the DTE port is not flow-controller by the host computer, the modem may implement a data discard procedure when the second regular buffer and/or second real-time buffer are full. The modem may discard newly arriving regular data frames and retain the regular data frames already stored in the second regular buffer. In contrast, the modem may discard old real-time frames stored in the second real-time buffer and overwrite these old real-time frames with newly arriving real-time frames.

Using a pair of such modems to establish a modem link across a switching network (e.g. the PSTN) allows real-time data to be transmitted with a minimum of real-time delay while simultaneously transmitting regular data. One of the modems may connected to a user computer while the other may be situated at an ISP's Point-of-Presence. Alternatively, the pair of modems may serve as a bridge between two data networks. For example, the pair modems may bridge a corporate office network to a branch office LAN.

The present invention also contemplates a software modem client which executes on the host processor. The software client transfers data to/from a conventional modem. The software client implements the forwarding of real-time data over regular data according to the principles of the present invention in both transmit and receive directions.

Brief Description of the Drawings

A better understanding of the present invention can be obtained when the various embodiments of the following detailed description is considered in conjunction with the following drawings, in which:

Fig. 1 illustrates a modem-based communication link according to the prior art;

Fig. 2A&2B illustrate a software architecture for transmission and reception of data respectively according to the prior art;

Fig. 3 illustrates real-time frames and non-real-time frames in various stage of delivery across a modem link;

Fig. 4 illustrates a Digital Simultaneous Voice and Data (DSVD) modem link;

Fig. 5 illustrates modem-based communication according to present invention;

Fig. 6A illustrates the transmission architecture of a modem according to the present;

Fig. 6B illustrates a state diagram for transmission logic 146 of modem 113;

Fig. 6C illustrates an escape sequence protocol for real-time data injection into an output data stream;

Fig. 6D illustrates further features of the escape sequence protocol;

Fig. 6E illustrate the initiation of transmission in response to the availability of real-time data;

Fig. 7A illustrates the reception architecture of modem 113;

Fig. 7B illustrates a state diagram for transmit logic 170 which mediates transmission from modem 113 to first computer (or DTE) 112;

Fig. 8A illustrates a Point-to-Point Protocol frame format;

Fig. 8B illustrates the format of a PPP header field; and

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 9 of 38

Fig. 9 illustrates an embodiment of the present invention where some of the modem functions are implemented in a software client.

While the invention is susceptible to various modifications and alternative forms, specific embodiments are shown by way of example in the drawings and will herein be described in detail. It should be understood however, that the drawings and detailed description thereto are not intended to limit the invention to the particular forms disclosed. But on the contrary the invention is to cover all modifications, equivalents and alternatives following within the spirit and scope of the present invention as defined by the appended claims.

Detailed Description of the Preferred Embodiments

Fig. 5 depicts a communication system which employs modems 113 and 117 configured according to the principles of the present invention. Modem 113 is coupled to a first computer or DTE 112 through any of a variety of interconnection technologies such as, e.g., an RS-232 serial port. Modem 113 may be configured for insertion into an expansion slot of first computer 112. In an alternate embodiment, modem 113 may be considered part of first computer 112 as, e.g., when modem 113 is incorporated on the motherboard of first computer 112.

The term modem is used broadly herein to refer to any type of communication device including devices such as analog (e.g. POTS) modems, Integrated Services Digital Network (ISDN) modems, Digital Subscriber Line (DSL) modems, cable modems, etc.

First computer 112 may be any of a variety of computing devices such as a personal computer, a notebook computer, etc. First computer 112 may be coupled to a data network 100 such as, e.g., a local area network. In addition, first computer 112 may also be coupled to a real-time signal device 110 such as, e.g., a telephone.

Modem 113 is also configured for coupling to switching network 115 through a subscriber connection SC#1. Switching network 115 may be the Public Switched Telephone Network (PSTN). Subscriber connection SC#1 may be an analog telephone line.

Modem 117 is configured for coupling to the PSTN through a subscriber connection SC#2. The subscriber connection SC#2 may also be an analog telephone line. Modem 117 is also configured for coupling to a second computer or DTE 118. Second computer 118 may be coupled to a data network 119. Data network 119 may also be a local area network, or perhaps, a wide area network such as the Internet.

Modem 113 receives from first computer 112 a first data stream including real-time data and regular data. The real-time data may originate from real-time signal device 110 or from a source coupled to data network 100. Similarly, regular data may originate from an application running on first computer 112 or from a data source coupled to the data network 100. The first data stream typically comprises a PPP stream. However, the principles of the present invention do not rely on the specific features of the Point-to-Point Protocol, and thus are broadly applicable to any stream of multiplexed real-time and regular data.

Modem 113 receives the first data stream and performs reordering of the real-time data with respect to the regular data. The real-time data is locally forwarded ahead of the regular data. The resultant output data stream is transmitted onto the subscriber connection SC#1. The reordering minimizes the delay experienced by the real-time data.

Real-time data takes priority over regular data in transmission. Thus, within modem 113 a regular data transmission may be temporarily interrupted in order to transmit newly arrived real-time data. When the real-time data has been transmitted, the regular data transmission may be resumed. The injection of the real-time data transmission occurs according to a protocol which is known by modem 117. Regular data is transmitted only if

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 10 of 38

no real-time data is available for transmission. It is noted that real-time data may be injected at locations which correspond to the interior of regular data packets in cases where the regular data is packet oriented.

The output data stream transmitted by modem 113 is transmitted through the switching network to modem 117. Modem 117 receives the output data stream from subscriber connection SC#2, and may forward real-time data ahead of non-real-time data. Real-time data may be transmitted to second computer 118 as soon as it becomes available, while regular data may be transmitted to second computer 118 only if real-time data is not available.

It is noted that the subscriber connection SC#1 is not necessarily a wired connection. For example, the subscriber connection SC#1 may be achieved by a radio link.

In the opposite direction, modem 117 may receive a second data stream which includes real-time data and regular data from second computer 118. The real-time data may originate from a real-time source coupled to data network 119. For example, a Voice over IP (VoIP) correspondent coupled to data network 119 may be the source of the real-time data. Similarly, the regular data may originate from a data source coupled to the data network 119. The real-time source and data source are generally distinct nodes of data network 119. Data network 119 may include the Internet.

Modem 117 forwards real-time data ahead of regular data, and thereby generates a second output stream. In other words, real-time data is transmitted onto subscriber connection SC#2 as soon as it becomes available, while regular data is transmitted only if real-time data is not available. Furthermore, if real-time data becomes available during a regular data transmission, the regular data transmission is temporarily interrupted so that the real-time data may be injected. When real-time data is no longer available, the regular data transmission may be resumed. The real-time data is injected according to a protocol which is recognized by modem 113. The injection protocol deposits information in the second output stream which allows modem 113 to detect the location and length of real-time injections in the second output stream.

Modem 117 transmits the second output stream onto the subscriber connection SC#2. Again is it noted that subscriber connection SC#2 is not necessarily a wire-based connection.

Modem 113 receives the second output stream from the switching network 115 through subscriber connection SC#1. Modem 113 may forward real-time data ahead of regular data in transmissions to first computer 112. Real-time data may be transmitted to first computer 112 as soon as it becomes available. In contrast, regular data may be transmitted to first computer 112 only if real-time data is not available.

First computer 112 preferably includes (or is coupled to) hardware which converts the real-time data into a real-time signal. The real-time signal is presented to real-time device 110. Alternatively, first computer 112 may forward the real-time data to a real-time destination on data network 100. In addition, a software application executing on first computer 112 may serve as the destination for regular data sent provided by modem 113. Alternatively, first computer 112 may forward the regular data to a data destination coupled to data network 100. In general, the real-time destination and data destination may be distinct nodes of data network 100.

Since modems 113 and 117 forward real-time data ahead of regular data, real-time latency in transmission is minimized. With regard to transmissions onto the subscriber connection SC#1 or SC#2, large regular data packets do not contribute to real-time latency since real-time data transmission is given priority of regular data transmission. The injection of real-time data into the transmitted stream in not constrained to respect regular data packet-boundaries.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 11 of 38

Fig. 6A illustrates a preferred embodiment for the transmission architecture of modem 113 according to the present invention and a typical software arrangement for first computer 112. The elements depicted in Fig. 6A above the dotted line labeled DTE interface 138 are software modules which execute on first computer 118.

First computer 112 includes a processor and a memory which provide for the execution of an arbitrary number of software applications. For example, first computer 112 may execute real-time applications, UDP applications, TCP applications, etc. Real-time application 120 generates a stream of RTP packets. One of these packets RTP1 is especially denoted in passage to socket S1. UDP application 122 is illustrative of UDP applications which are non-real-time in nature. UDP application 122 is shown passing a data packet D1 to socket S2. TCP application 124 generates a bit stream. A portion of this bit stream is shown in transit to socket S3.

An Internet Protocol Stack 137 also executes on first computer 112. The Internet Protocol Stack 137 comprises a multi-layer system of protocol modules. The Internet Protocol Stack 137 may be part of a conventional operating system running on first computer 112. The Internet Protocol Stack mediates data transfers for software applications running on first computer 112.

UDP module 126 encapsulates the RTP packet RTP1 received from socket S1. UDP module 128 encapsulates the packet D1 received from socket S2. TCP module 130 encapsulates the data D2 received from socket S3. The UDP and TCP encapsulated packets generated by the transport layer module including modules 126-130 are multiplexed and provided to IP module 132. IP module 132 encapsulates the UDP and TCP packets as IP packets. The resulting stream of IP packets are provided to PPP module 134. PPP module 134 encodes the IP packets stream as a stream of PPP frames. A portion of the PPP stream 136 is shown. Observe that frames in the PPP stream are separated by a tilde character, i.e. 0x7E.

Fig. 8A depicts the frame format for the Point-to-Point protocol. The generic PPP frame 15 includes a header field 151, a protocol field 152, a data field 153 and a check sum 154. Fig. 8B depicts the format of header field 151. Header field 151 includes an address byte 151B and a control byte 151C. The address byte 151B may take the value 0xFF. The control byte 151C may take the value 0x03. Check sum 154 provides for error detection at the PPP receiver. The protocol field 152 defines the protocol(s) used to generate data field 153. Data field 153 contains the data payload of the PPP frame 15.

The PPP stream 136 generated by the PPP module may include TCP/IP encapsulated data. One such frame denoted IP/TCP/D2 encapsulates data D2 which originated from TCP application 124. The PPP stream 136 may also include UDP/IP encapsulated data. One such frame denoted IP/TCP/D1 encapsulates data D1 which originated from the UDP application 122. The PPP stream may additionally include IP/UDP/RTP encapsulated data. One such frame denoted IP/UDP/RTP1 encapsulates RTP packet RTP1 which originated from real-time application 120.

The PPP module may compress the redundant header information of IP/UDP/RTP encapsulated frames using a protocol such as, e.g., the Compressed Real Time protocol (CRTP). The PPP module provides the PPP stream 136 to modem 113 through DTE interface 138.

Modem 113 comprises demultiplexing logic 140, real-time buffer 142, regular buffer 144, and transmit logic 146. It is noted that functions performed by demultiplexing logic 140 and transmit logic 146 may be implemented by a microcontroller (or processor) situated within modem 113.

Demultiplexing logic 140 receives the PPP stream 136 from the PPP module 134, and demultiplexes the PPP stream in real time. Demultiplexing logic 140 parses the PPP stream for the header field 151 to determine the beginning of a new PPP frame. Once a header field has been detected, the subsequent protocol field 152 is ex-

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 12 of 38

amined. If the protocol field indicates that the newly detected PPP frame encapsulates real-time data (such as RTP or CRTP data), the PPP frame is stored into the real-time buffer 142. If the protocol field indicates an encapsulation corresponding to non-real-time data protocol(s), i.e. protocols other than RTP or CRTP, the newly detected PPP frame is stored into regular buffer 144. It is noted that characters which make up the PPP stream 136 are received by demultiplexing logic 140 and very quickly stored in either real-time buffer 142 or regular buffer 144.

Real-time buffer 142 may comprise Random Access Memory (RAM) such as Dynamic RAM (DRAM). Regular buffer 144 may also comprise RAM. Real-time buffer 142 and regular buffer may comprise distinct sections of a common memory space accessible by a microcontroller. Note that no particular size is mandated for buffers 142 and 144.

Transmit logic 146 generates and transmits an output character stream (see item 92 of Fig. 6D) based on the condition of real-time buffer 142 and regular buffer 144. Fig. 6B illustrates a state diagram for transmit logic 146. In an idle state 150, transmit logic 146 is not engaged in active transmission.

In response to real-time buffer 142 being empty and regular buffer 144 attaining a non-empty condition, transmit logic 146 moves into the "regular data transmission" state 155. In this state 155, transmit logic 146 transmits regular data from regular buffer 144 until regular buffer 144 is exhausted, i.e. attains the empty state, or until the real-time buffer attains the non-empty state. If real-time buffer 142 remains empty and regular buffer 144 attains the empty state due to the transmission activity of transmit logic 146, transmit logic 146 returns to the idle state 150.

In the preferred embodiment, real-time buffer 142 is said to be empty or non-empty with respect to complete real-time frames. Thus, real-time buffer 142 is said to be empty when it contains no complete real-time frames, and non-empty when it contains one or more complete real-time frames. In contrast, regular buffer 144 is declared to be non-empty when the number of bytes stored in regular buffer 144 exceeds a threshold value, and declared to be empty when the number bytes in regular buffer 144 reaches zero. The threshold value is chosen to be a small integer such as two or three to guarantee that transmit logic 146 does not run out of regular data to transmit. If regular data to be transmitted is queued in the first computer 112 awaiting transfer to modem 113, two or three byte transmission times at the low bandwidth of the subscriber connection SC#1 is generally enough time for the regular data to be transferred across the DTE interface 138 and into real-time buffer 142.

If the real-time buffer attains the non-empty condition during a regular data transmission (i.e. state 155), transmit logic 146 temporarily interrupts the regular data transmission, and moves to the "inject real-time data transmission" state 160. In state 160, transmit logic 146 transmits real-time data (e.g. RTP or CRTP frames) from real-time buffer 142 until real-time buffer 142 attains the empty state, i.e. is exhausted of complete real-time frames. It is noted that demultiplexing logic 140 may store real-time frame data into real-time buffer 142 while transmit logic 146 concurrently reads real-time frame data from the real-time buffer 142 for transmission onto subscriber line SC#1. The functions of demultiplexing logic 140 and transmit logic 146 may be implemented in software on a processor (e.g. a microcontroller) situated within modem 113.

Transmit logic 146 accesses real-time data from real-time buffer 142 and injects the real-time data into the output character stream. In the preferred embodiment, the real-time data stored in real-time buffer 142 is organized into frames (such as PPP frames). In this case, transmit logic 146 may repackage the contents of a real-time frame into a more efficient form referred to herein as a real-time capsule. For example, when forming a real-time capsule from a real-time PPP frame, transmit logic 146 may throw away the header field 151, the protocol

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 13 of 38

field 152, and the checksum field 154 of a real-time PPP frame. See Fig. 8A&8B for the format of a PPP frame. The real-time capsule includes the data field 153 of the real-time PPP frame and a length field which defines the length of the data field. The length field may precede the data field in the real-time capsule so that a receiving device (e.g. modem 117) may discern the length of the data field prior to its arrival in the output character stream.

Transmit logic 146 generates the real-time capsule on-the-fly, i.e. with minimal buffering. It is noted that demultiplexing logic 140 may compute the length field for the real-time capsules as it is writing a corresponding real-time data frame into real-time buffer 142. Thus, transmit logic 146 may avoid additional buffering of the real-time data to compute the length field.

When the real-time buffer 142 attains the empty condition, i.e. when no more complete real-time frames reside in real-time buffer 142, transmit logic 146 (a) moves to the idle state 150 if regular data buffer 144 is empty, or (b) moves to the "regular data transmission" state 155 if regular data buffer 144 is non-empty.

As noted above, the transmission of regular data may be interrupted in order to inject real-time data into the output data stream. In the state diagram this interruption corresponds to the transition from state 155 to state 160 in response to the real-time buffer 142 attaining a non-empty condition. When the real-time data injection is complete, transmit logic 146 resumes regular data transmission, i.e. returns to state 155.

It is noted that transmit logic 146 transmits an output data stream to modem 117 (of Fig. 5) through the switching network. The data transfer protocol between transmit logic 146 and modem 117 may be a synchronous or an asynchronous protocol.

When real-time buffer 142 is full, demultiplexing logic 140 may be configured to discard one or more real-time frames previously stored in real-time buffer 142 to make room for newly arriving real-time frames. A newly arriving real-time frame from the PPP stream 136 may overwrite the old real-time frames previously stored in the real-time buffer 142. Preferably the discarded real-time frames are the ones which are oldest chronologically, i.e. the ones which are resided in real-time buffer for the longest duration.

In contrast, demultiplexing logic 140 is configured to discard any regular frames arriving in the PPP stream 136 when regular buffer 144 is full. Regular frames previously stored in regular buffer 144 are retained.

Fig. 6C depicts an escape sequence protocol for injecting real-time data into the transmitted output stream. Fig. 6C includes several graphs. The topmost graph illustrates the arrival of PPP stream 136 across the DTE interface 138. PPP stream 136 is illustrated with four frames in succession, i.e. regular frame D1, real-time frame V1, regular frame D2 and real-time frame V2. The second graph illustrates the arrival of regular data frames into regular buffer 144. The third graph illustrates the arrival of real-time frames into real-time buffer 142. Recall that demultiplexing logic 140 parses the PPP stream so as to separate the real-time data frames and regular data frames.

For simplicity, it is assumed that real-time buffer 142 and regular buffer 144 are empty prior to time t_{10} when the first bytes of regular frame D1 are stored into regular buffer 144. After a threshold number of bytes of regular data frame D1 have been stored into regular buffer 144, transmit logic 146 moves into the "regular data transmission" state 155 and starts to transmit regular data frame D1. The setup time is defined as the time required for the threshold number of bytes to arrive into regular buffer 144. Transmit logic 146 may create an HDLC frame H1 in order to contain the regular data frame D1. It is noted the choice of HDCL as a data transport protocol is arbitrary, and the present invention contemplates any of a variety of transport protocols.

The transmission of regular frame D1 is interrupted at times t_{11} and t_{12} when real-time frames V1 and V2 respectively become available for transmission. Thus, regular frame D1 is transmitted as three disjoint compo-

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 14 of 38

nents D1-1, D1-2, D1-3 which are separated by injected real-time capsules V1' and V2'. At time t_{11} demultiplexing logic 140 completes the storage of real-time frame V1 into real-time buffer 142. In response to real-time buffer 142 now being non-empty, transmit logic 146 injects real-time capsule V1' corresponding to real-time frame V1 into the transmitted output stream. In order to signal the introduction of real-time data to the receiver (i.e. modem 117), transmit logic 146 inserts an escape character into the output stream. The escape character defines the position of the real-time injection to the receiver. The real-time capsule V1' which includes a length field and the real-time payload data is inserted into the output stream after the escape character. The length field defines the length of the real-time payload data. Thus, the length field allows the receiver to detect the position in the transmitted output stream where regular data transmission resumes.

For more information concerning the escape sequence protocol, please refer to U.S. Patent Application Serial No. 09/238,819 entitled "Escape Sequence Protocol for Multiplexing Real-Time Data with Non-Real-Time Data", filed on January 28, 1999, invented by David C. Oliver, assigned to Data Race, Inc., which is hereby incorporated by reference.

After the insertion of real-time capsule V1' into the output stream has been completed at time t_{13}, real-time buffer 142 is empty, i.e. contains no real-time frames. Thus, transmit logic 146 returns to the "regular data transmission" state 155, and thus, resumes transmission of regular frame D1. At time t_{12} demultiplexing logic 140 complete the storage of real-time frame V2 into real-time buffer 142. In response to real-time buffer 142 again being non-empty, transmit logic 146 injects real-time capsule V2' corresponding to real-time frame V2 into the transmitted output stream. The escape character is inserted prior to real-time capsule V2' and after the interruption of regular data at time t_{12}. After transmit logic 146 completes insertion of real-time capsule V2' into the output stream at time t_{14}, real-time buffer 142 once again attains the empty condition, and thus, transmit logic 146 resumes transmission of regular data.

When transmission of regular data frame D1 is completed at time t15, real-time buffer 142 is empty and regular buffer 144 is non-empty. Thus, transmit logic 146 remains in the "regular data transmission" state 155, and immediately begins to transmit regular frame D2 onto the communication medium (i.e. subscriber connection SC#1). Transmit logic 146 may generate a new HDLC frame H2 for regular data frame D2. In general, transmit logic 146 may generate a new HDLC frame for each regular data frame transmitted. As mentioned above, the HDLC frame may contain any number of real-time data injections according to the escape sequence protocol.

Note that other protocols for multiplexing real-time data and regular data into the output data stream may be used. For more information on one such multiplexing scheme, please refer to U.S. Patent Application Serial No. 09/100,778 entitled "System and Method for Low Overhead Multiplexing of Real-Time and Non-Real-Time Data", filed on June 10, 1998, invented by David C. Oliver, and assigned to Data Race, Inc., which is hereby incorporated by reference.

Transmit logic 146 is further configured to parse characters of regular data frames prior to transmission. The parsing operation treats the regular frames as a character stream, and injects a secondary character after solo occurrences of the escape character in the character stream. The secondary character is different from the escape character. The parsing operation further includes detecting adjacent occurrences of the escape character in the character stream, and replacing the second of the adjacent occurrences of the escape character in the character stream with a tertiary character. The tertiary character is different from the escape character and the secondary character. This allows data that has the same pattern as the escape character to be distinguished from the escape character.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 15 of 38

Turning now to Figure 6D, data streams are provided illustrating the replacement of regular data characters according to the mechanism described above. A regular data stream is shown at 90. The regular data stream 90 includes characters equal in value to the primary escape sequence hex 1A. Note that while the value hex 1A is shown for the primary escape sequence (the ASCII SUB character), any value may be used as the escape character as long as the value is known to the transmitting and receiving devices. As shown in data stream 92, the regular data sequences equal to the primary escape sequences are replaced with the second or third type of escape sequence before the data stream is transmitted. Single occurrences of the escape character are replaced with the second escape sequence hex 1A80, and double occurrences of the escape character are replaced with the third escape sequence hex 1AC0. The receiving device (i.e. modem 117) receives the data stream 94 containing the substituted second and third escape sequences. The receiving device replaces the second escape sequence with a data value equal to the escape character hex 1A and replaces the third escape sequence with a data value equal to a pair of the escape characters, i.e. hex 1A1A, as shown in the data stream 96. Data stream 96 is then used as the received data stream.

Fig. 6E illustrates the situation where one or more real-time frames become available for transmission while regular buffer 144 is empty. The topmost graph in Fig. 6E illustrates PPP stream 136 as it is received by demultiplexing logic 140. PPP stream 136 includes a succession of frames, i.e. real-time frame V1, regular frame D1 and real-time frame V2. Demultiplexing logic 140 separates the PPP stream 136 so that regular frames as sent to regular buffer 144 and real-time frames are sent to the real-time buffer 142. The second graph depicts the arrival of regular frames in regular buffer 144. The third graph depicts the arrival of real-time frames in the real-time buffer 142. The fourth graph depicts the transmitted output stream generated by transmit logic 146.

It is noted that real-time frames V1 and V2 become available for transmission at times t_{17} and t_{18} respectively, i.e. as soon as demultiplexing logic 140 completes storage of the respective real-time frame into real-time buffer 142. In Fig. 6E, it is assumed that transmit logic 146 is in the idle state 150 prior to the arrival of real-time frame V1. This implies that real-time buffer 142 is empty, i.e. devoid of real-time frames, and regular buffer 144 is empty. At time t_{17} real-time frame V1 becomes available for transmission, and transmit logic 146 enters the "inject real-time data transmission" state 160. In order to support the injection of real-time data, transmit logic 146 generates an initial portion (not shown) of a containing frame H3. The contents of the initial portion may be defined by the data transport protocol. The containing frame H3 may be an HDLC frame if HDLC is being used as the data transport protocol. Transmit logic 146 inserts the initial portion of the containing frame into the transmitted output stream. After the initial portion, transmit logic 146 transmits the primary escape sequence, and then transmits real-time capsules until real-time buffer 142 attains the empty condition, i.e. is devoid of complete real-time frames. In this case, real-time buffer 142 is exhausted after real-time capsule V1' corresponding to real-time frame V1 is transmitted.

At time t_{19} real-time buffer 142 is again empty. However, in the time interval between time t_{17} and t_{19}, regular buffer 144 has become non-empty due to the arrival of regular frame D1. Thus, at time t_{19} transmit logic 146 enters the "regular data transmission" state 155. Regular data transmission is interrupted at time t_{18} in order to inject real-time capsule V2' corresponding to newly arrived real-time frame V2. After real-time capsule V2' is transmitted using the escape sequence protocol, transmit logic 146 may resume transmission of the remaining portion D1-2 of regular frame D1 from regular buffer 144. When regular buffer 144 is exhausted, transmit logic 146 may transmit a terminating portion (not shown) of the containing frame if real-time buffer 144 is concurrently in the empty state, i.e. if real-time buffer 144 contains no complete real-time frames.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 16 of 38

It is noted that the terminating portion of containing frame H3 may be transmitted from the "inject real-time data transmission" state 160. Namely, when the real-time buffer 142 is attains the empty state, i.e. is exhausted of complete real-time frames, the terminal portion of a containing frame will be transmitted if the regular buffer is also empty.

In summary, modem 113 comprises a first real-time buffer 142. a first regular buffer 144, and control logic. The control logic comprises demultiplexing logic 140 and transmit logic 146. The control logic may be implemented by a microcontroller. In this case, demultiplexing logic 140 and transmit logic 146 are implemented in software on the microcontroller. Alternatively, demultiplexing logic 140 and transmit logic 146 may be implemented as separate hardware devices optimized for their respective functions. Demultiplexing logic 140 is configured:

(a) to receive a first stream of frames from a first data interface,

(b) to demultiplex the first stream of frames into a second stream of real-time frames and a third stream of regular frames, and

(c) to store the second stream of real-time frames into the first real-time buffer and third stream of regular frames into the first regular buffer.

Transmit logic 146 is configured:

(d) to initiate transmission of a first regular frame from the first regular buffer onto a communication medium,

(e) to temporarily interrupt transmission of the first regular frame in response to the real-time buffer attaining a first non-empty condition,

(f) to transmit one or more real-time capsules (which are generated from real-time frames stored in the first real-time buffer) onto the communication medium until the first real-time buffer attains a first empty condition,

(g) to resume transmission of the first regular frame in response to the first real-time buffer attaining the first empty condition.

In one embodiment, the following mechanism is employed to manage the empty or non-empty status of the real-time buffer. Demultiplexing logic 142 is configured to increment a real-time frame count in response to completing storage of a real-time frame into real-time buffer 142. Demultiplexing logic 142 may parse the PPP stream 136 for the PPP header bytes, and thereby detect the end of one frame and the beginning of the next frame. Thus, demultiplexing logic 142 is able to detect the end of a real-time frame. The real-time frame count increases by one for real-time frame stored into the real-time buffer 142. In addition, transmit logic 146 is configured to decrement the real-time frame count in response to completing transmission of a real-time frame from the real-time buffer 142. Real-time buffer 142 is said to attain a non-empty condition when the real-time frame count attains a value greater than zero. Transmit logic 146 monitors the value of the real-time buffer count, and moves into the "inject real-time data transmission" state 160 when the real-time frame count attains a value greater than zero. Correspondingly, the real-time buffer 142 is said to attain the empty condition when the real-time frame count attains the value of zero.

In a second embodiment, demultiplexing logic 142 maintains an real-time input count by incrementing the real-time input count in response to completing storage of a real-time frame into the real-time buffer 142.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 17 of 38

Transmit logic 146 maintains a real-time output count by incrementing the real-time output count in response to completing transmission of a real-time frame from real-time buffer 142. Transmit logic 146 determines the empty or non-empty condition of real-time buffer 142 by subtracting the real-time output count from the real-time input count. A positive resultant value indicates that the real-time buffer 142 contains one or more complete real-time frames. A resultant value of zero indicates that real-time buffer 142 contains no complete real-time frames.

In the preferred embodiment, modem 113 is further configured for data reception as shown in Fig. 7A. Modem 113 includes a second real-time buffer 166, a second regular buffer 168, demultiplexing logic 164 and transmit logic 170. The functions of demultiplexing logic 164 and transmit logic 170 may implemented in software on an embedded microcontroller.

Demultiplexing logic 164 is configured to receive a fourth data stream from the communication medium (i.e. subscriber connection SC#1). The fourth data stream is transmitted from modem 117 through the switching network. Demultiplexing logic 164 demultiplexes the fourth data stream into a fifth stream of real-time data and a sixth stream of regular data. Demultiplexing logic 164 stores the fifth stream of real-time data into real-time buffer 166 and the sixth stream of regular data into regular buffer 168.

Demultiplexing logic 164 may parse the fourth data stream for the primary escape sequence to detect the initiation of a real-time capsule in the fourth data stream. When the primary escape sequence is detected, demultiplexing logic 164 receives the real-time capsule which follows the primary escape sequence in the fourth data stream. The real-time capsule includes a real-time length field and real-time payload data as described above. The real-time length field specifies the length of the real-time data following the length field in the fourth data stream. Demultiplexing logic 164 may repackage the real-time payload data in a frame format such as the PPP frame format before storage into real-time buffer 166. For example, the real-time payload data may be prefixed with a header field and a protocol field, and postfixed with a checksum as defined by PPP, in order to generate a real-time PPP frame.

Demultiplexing logic 164 is also configured to scan the sixth stream of regular data, and to replace each occurrence of the second escape sequence in the sixth stream with the escape character, and each occurrence of the third escape sequence in the sixth stream with a pair of escape characters, i.e. hex 1A1A.

Transmit logic 170 transmits data to first computer 112 from real-time buffer 166 and regular buffer 168. Fig. 7B depicts a state diagram for transmit logic 170. In an idle state 180 transmit logic 170 is inactive. Transmit logic 170 stays in the idle state until either real-time buffer 166 or regular buffer 168 becomes non-empty. Real-time buffer 166 is said to be empty when it contains no complete real-time frames, and non-empty when it contains one or more complete real-time frames. Similarly, regular buffer 168 is said to be empty when the number of complete regular frames stored in regular buffer 168 equals zero, and non-empty when this number is greater than zero.

Demultiplexing logic 164 and transmit logic 170 implement a mechanism for indicating the number of complete real-time frames stored in real-time buffer 166, and the number of complete regular frames stored in regular buffer 168. In Fig. 7B these numbers are denoted RealCount and RegCount respectively.

Transmit logic 170 transitions from the idle state 180 to the "transfer a regular data frame" state 190 in response to regular buffer 168 attaining a non-empty condition (i.e. RegCount>0) while real-time buffer 166 remains empty (RealCount=0). In state 190, transmit logic 170 transfers a regular data frame from regular buffer 168 to first computer 112 across DTE interface 138. Upon completing transfer of the current regular data frame, transmit logic 170 checks the condition of the buffers. If real-time buffer 166 is still empty (i.e. RealCount is

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 18 of 38

zero) and regular buffer 168 is still non-empty (i.e. RegCount is positive), transmit logic 170 transfers another regular data frame from regular buffer 168 to first computer 112.

Upon completing transfer of the current regular data frame, transmit logic 170 transitions to the idle state 180 if both buffers are found to be empty (i.e. RealCount=0 and RegCount=0).

Upon completing transfer of the current regular data frame, transmit logic 170 transitions to the "transfer a real-time data frame" state 185 in response to real-time buffer 166 being non-empty (i.e. RealCount>0).

In the "transfer a real-time data frame" state 185, transmit logic 170 transfer a real-time data frame from real-time buffer 166 to first computer 112 through DTE interface 138. Transmit logic 170 repeatedly transfers real-time data frames from real-time buffer 166 until real-time buffer 166 is emptied (i.e. RealCount=0). When real-time buffer 166 has been emptied, transmit logic returns to the idle state 180 if regular buffer 168 is empty (RegCount=0), and transitions to the "transfer a regular data frame" state 190 if regular buffer 168 is non-empty (i.e. RegCount>0).

When real-time buffer 166 is full, demultiplexing logic 164 is further configured to discard one or more real-time frames previously stored in real-time buffer 166 to make room for the storage of a current real-time frame (i.e. one that is just arriving) from the fourth stream. The discarded real-time frames are preferable the oldest real-time frames residing in real-time buffer 166. In contrast, when regular buffer 168 is full, demultiplexing logic 164 is configured to discard any arriving regular data frames, and to retain regular frames already stored in regular buffer 168.

Fig. 9 illustrates a second embodiment of the present invention where the functions associated with transmission elements 140-146 and reception elements 164-170 in modem 113 are implemented in software client 200. Software client 200 executes on the host processor, i.e. the processor of first computer 112. Software client 200 may include transmission module 204 and reception module 202. Software client 200 communicates with Internet Protocol Stack 137 through a virtual DTE interface. Internet Protocol Stack 137 communicates with software client 200 as if software client 200 were a physical modem. In other words, the virtual DTE interface behaves like a physical modem interface so far as Internet Protocol Stack 137 is concerned.

Software Client 200 transmits and receives data from conventional modem 210. Conventional modem 210 may be any of a variety of existing modems such as, e.g., a v.80 modem (i.e. a modem conforming to the v.80 standard).

For more information on implementing special protocols using host software in conjunction with a conventional modem, please refer to U.S. Patent Application Serial No. 09/238,820 entitled "Implementing Standard Protocols Using Standard Modems", filed on January 28, 1999, invented by David C. Oliver, and assigned to Data Race, Inc., which is hereby incorporated by reference.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 19 of 38

Claims.

1. A communication system comprising:

a first real-time buffer;

a first regular buffer; and

control logic, wherein the control logic is configured (a) to receive a first stream of frames from a first data interface, (b) to demultiplex the first stream of frames into a second stream of real-time frames and a third stream of regular frames, (c) to store the second stream of real-time frames into the first real-time buffer and third stream of regular frames into the first regular buffer, (d) to initiate transmission of a first regular frame from the first regular buffer onto a communication medium, (e) to temporarily interrupt transmission of the first regular frame in response to the real-time buffer attaining a first non-empty condition, (f) to transmit one or more real-time frames from the first real-time buffer onto the communication medium until the first real-time buffer attains a first empty condition, (g) to resume transmission of the first regular frame in response to the first real-time buffer attaining the first empty condition.

2. The communication system of claim 1, the control logic is further configured (h) to increment a real-time frame count in response to a completion of storage of any one of the real-time frames of the second stream into the real-time buffer, (i) to decrement the real-time frame count in response to completing transmission of each of said one or more real-time frames from the first real-time buffer onto the communication, wherein the real-time buffer attains the first non-empty condition when the real-time frame count attains a value greater than zero, wherein the real-time buffer attains the first empty condition when the real-time frame count attains the value of zero.

3. The communication system of claim 1 or 2 further comprising:

a second real-time buffer;

a second regular buffer;

wherein the control logic is further configured (h) to receive a fourth stream of frames from the communication medium, (i) to demultiplex the fourth stream of frames into a fifth stream of real-time frames and sixth stream of regular frames, (j) to store the fifth stream of real-time frames in the second real-time buffer and the sixth stream of regular frames into the second regular buffer, (k) to transmit one or more real-time frames from the second real-time buffer to the first data interface in response to a second non-empty condition of the second real-time buffer and until the real-time buffer attains a second empty condition, (l) to transmit a second regular frame from the second regular buffer to the first data interface in response to the second empty condition of the second real-time buffer and a third non-empty condition of the second regular buffer.

4. The communication system of any of claims 1-3, wherein the first stream of frames conforms to the Point-to-Point Protocol (PPP), wherein the control logic demultiplexes the first stream of frames based on values of a protocol field in each of the frames of the first stream.

5. The communication system of claim 4, wherein each of the real-time frames in the second stream comprises an IP packet which encapsulates a UDP packet which encapsulates an RTP packet.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 20 of 38

6. The communication system of any of claims 1-5, wherein the real-time frames comprising the second stream are generated by one or more real-time applications executing on a first computer, wherein the first computer supplies the first stream of frames to the control logic through the first data interface.

7. The communication system of any of claims 1-6, wherein the control logic comprises a central processing unit (CPU) of a first computer, wherein (a) through (g) are implemented by a first software module executing on the CPU, wherein the first data interface comprises a software interface between the first software module and a second software module executing on the CPU, wherein the communication medium is configured for coupling to a conventional modem.

8. The communication system of any of claims 1-6, wherein the control logic comprises a microprocessor configured within a modem, wherein the first data interface comprises a physical interface between the modem and a first computer.

9. The communication system of any of claims 1-8, wherein the control logic is further configured to discard one or more old real-time frames previously stored in the first real-time buffer and to store a current real-time frame of the second stream into the first real-time buffer in response to the first real-time buffer being full.

10. The communication system of any of claims 1-9, wherein the control logic is further configured to discard one or more old real-time frames previously stored in the second real-time buffer and to store a current real-time frame of the fifth stream into the second real-time buffer in response to the second real-time buffer being full.

11. The communication system of any of claims 1-10, wherein the control logic is further configured to discard any regular frames from the third stream when the first regular buffer is full.

12. The communication system of any of claims 1-11, wherein the control logic is further configured to discard any regular frames from the sixth stream when the second regular buffer is full.

13. The communication system of any of claims 1-12, wherein the control logic is further configured to transmit an escape character onto the communication medium after said temporarily interrupting transmission of the first regular frame and prior to said transmission of said one or more real time frames from the first real-time buffer.

14. The communication system of claim 13, wherein the control logic is further configured to parse characters of first regular frame prior to transmission, wherein said parsing operates on the characters of the first regular frame as a character stream, wherein said parsing includes injecting a secondary character after solo occurrences of the escape character in the character stream, wherein said secondary character is different from the escape character.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 21 of 38

15. The communication system of claim 14, wherein said parsing further includes detecting adjacent occurrences of the escape character in the character stream, and replacing the second of the adjacent occurrences of the escape character in the character stream with a tertiary character, wherein said tertiary character is different from the escape character and the secondary character.

16. The communication system of claim 1, wherein the first control logic is further configured (h) to generate an initial portion of a first containing frame in response to the first real-time buffer attaining the first non-empty condition and the first regular buffer concurrently being empty, (i) to transmit the initial portion of the first containing frame onto the communication medium, (j) to repeatedly transmit a real-time frame from the first real-time buffer onto the communication medium until the first real-time buffer attains the first empty condition.

17. The communication medium of claim 18, wherein the first control logic is further configured to transmit a terminating portion of the first containing frame in response to the first real-time buffer attaining the first empty condition and the first regular buffer concurrently being empty.

18. The communication system of claim 19, wherein the first control logic is further configured (k) to initiate transmission of a second regular frame from the first regular buffer in response to the first real-time buffer attaining the first empty condition and the first regular buffer concurrently being non-empty.

19. A communications device, comprising:
a first real-time buffer configured to store real-time data;
a first regular buffer configured to store non-real-time data;
control logic comprising:
interface logic configured to receive a first data stream from a data terminal equipment, wherein said first data stream comprises real-time data and non-real-time data;
routing logic configured to store real-time data from said first data stream in said first real-time buffer and non-real-time data from said first data stream in said first regular buffer; and
first transmit logic configured to transmit a second data stream onto a communication medium, wherein said second data stream comprises said real-time and non-real-time data from said first data stream, wherein said first transmit logic is configured to receive said real-time data from said first real-time buffer and said non-real-time data from said first regular buffer, wherein said first transmit logic is further configured to insert said real-time data into said second data stream ahead of said non-real-time data when both real-time data and non-real-time data are pending in said first real-time buffer and first regular buffer respectively.

20. The communication device of claim 19, further comprising:
a second real-time buffer;
a second regular buffer;
said control logic further comprising:

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 22 of 38

receiving logic configured to receive a third data stream from said communication medium, wherein said receiving logic is further configured to separate real-time data from non-real-time data in the third data stream, and to store the real-time data in said second real-time buffer and said non-real-time data in said second regular buffer;

second interface logic configured to transmit a fourth data stream to said data terminal equipment, wherein said fourth data stream comprises said real-time data and non-real-time data from said third data stream, wherein said second transmit logic is configured to insert real-time data from said second real-time buffer into said fourth data stream ahead of said non-real-time data when both real-time data and non-real-time data are pending in said second real-time buffer and said second regular buffer respectively.

21. A communication device, comprising:

first interface logic configured to receive a first data stream from a data terminal equipment, wherein said first data stream comprises real-time data and non-real-time data;

a first regular buffer configured to buffer non-real-time data from said first data stream;

a first real-time buffer configured to buffer real-time data from said first data stream;

first forwarding logic configured to forward real-time data pending in said first real-time buffer ahead of non-real-time data pending in said first regular buffer; and

multiplexer logic configured to receive said real-time data and said non-real-time data from said first forwarding logic and multiplex said real-time data and said non-real-time data into a second data stream; and

first transmit logic configured to transmit said second data stream on a communication medium.

22. The communication device of claim 21, further comprising:

receiver logic configured to receive a third data stream from said communication medium, wherein the third data stream comprises real-time data and non-real-time data;

demultiplexer logic configured to receive the third data stream from said receiver logic and to demultiplex said real-time data in said third data stream from said non-real-time data in said third data stream;

a second regular buffer configured to buffer non-real-time data from said third data stream;

a second real-time buffer configured to buffer real-time data from said third data stream;

second forwarding logic configured to transmit a fourth data stream to said data terminal equipment, wherein said second forwarding logic is further configured to forward real-time data from said second real-time buffer ahead of non-real-time data from the regular buffer into the fourth data stream when real-time data and non-real-data are pending in the second real-time buffer and second regular buffer respectively.

IPR2022-00833
CommScope, Inc. Exhibit 1013
Page 23 of 38

Fig. 1
(Prior Art)

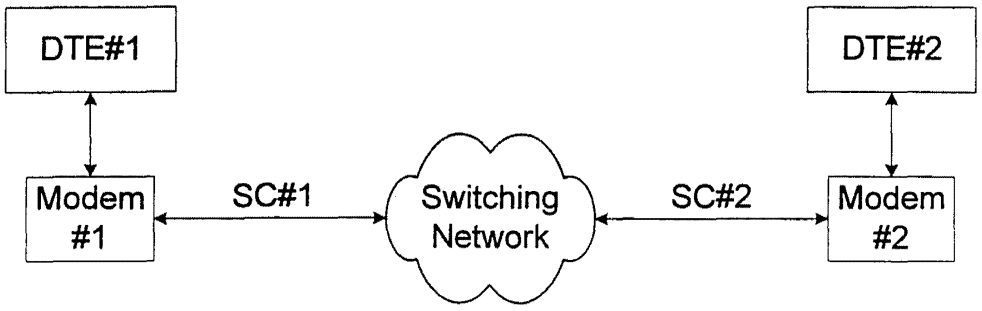


Fig. 2A
(Prior Art)

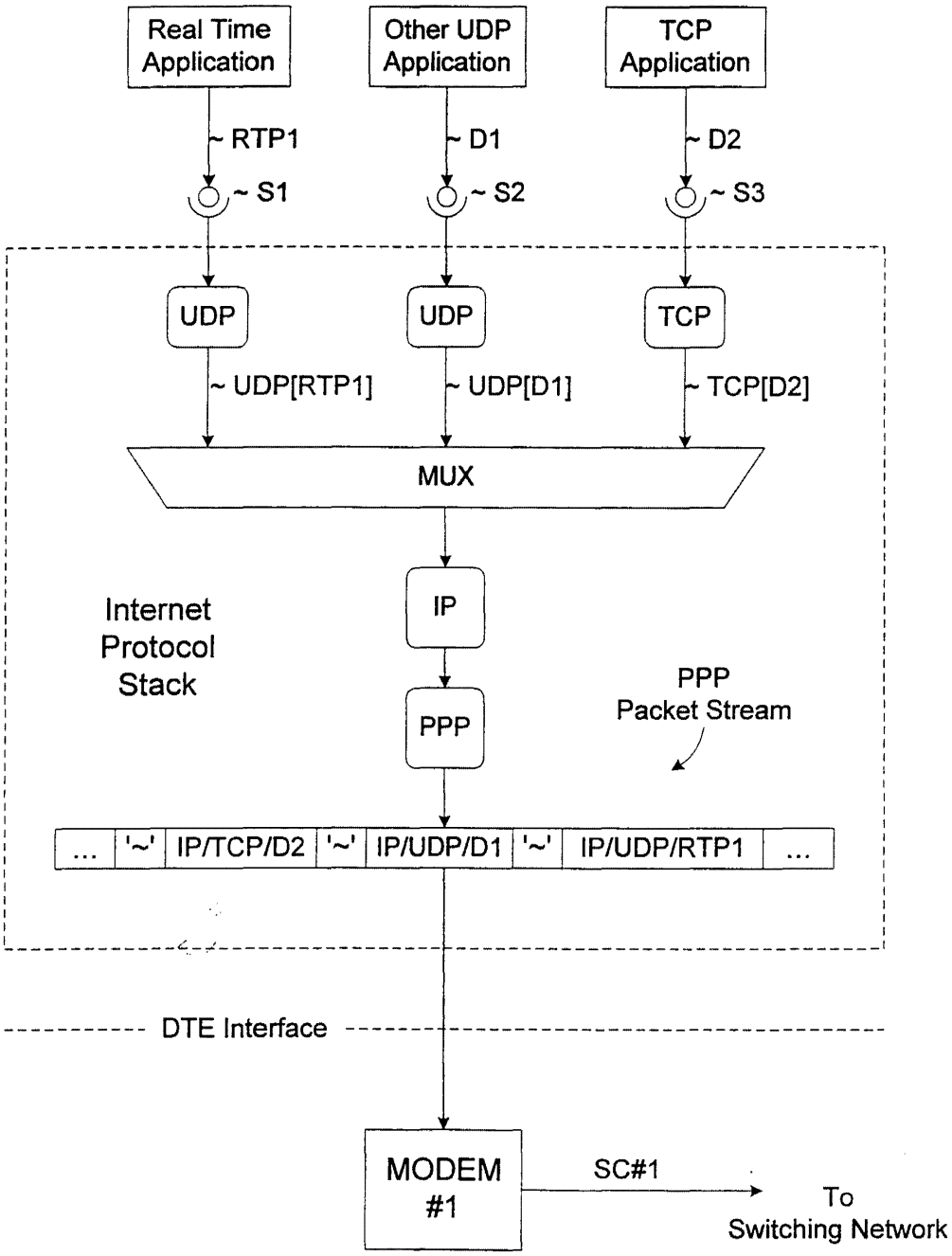
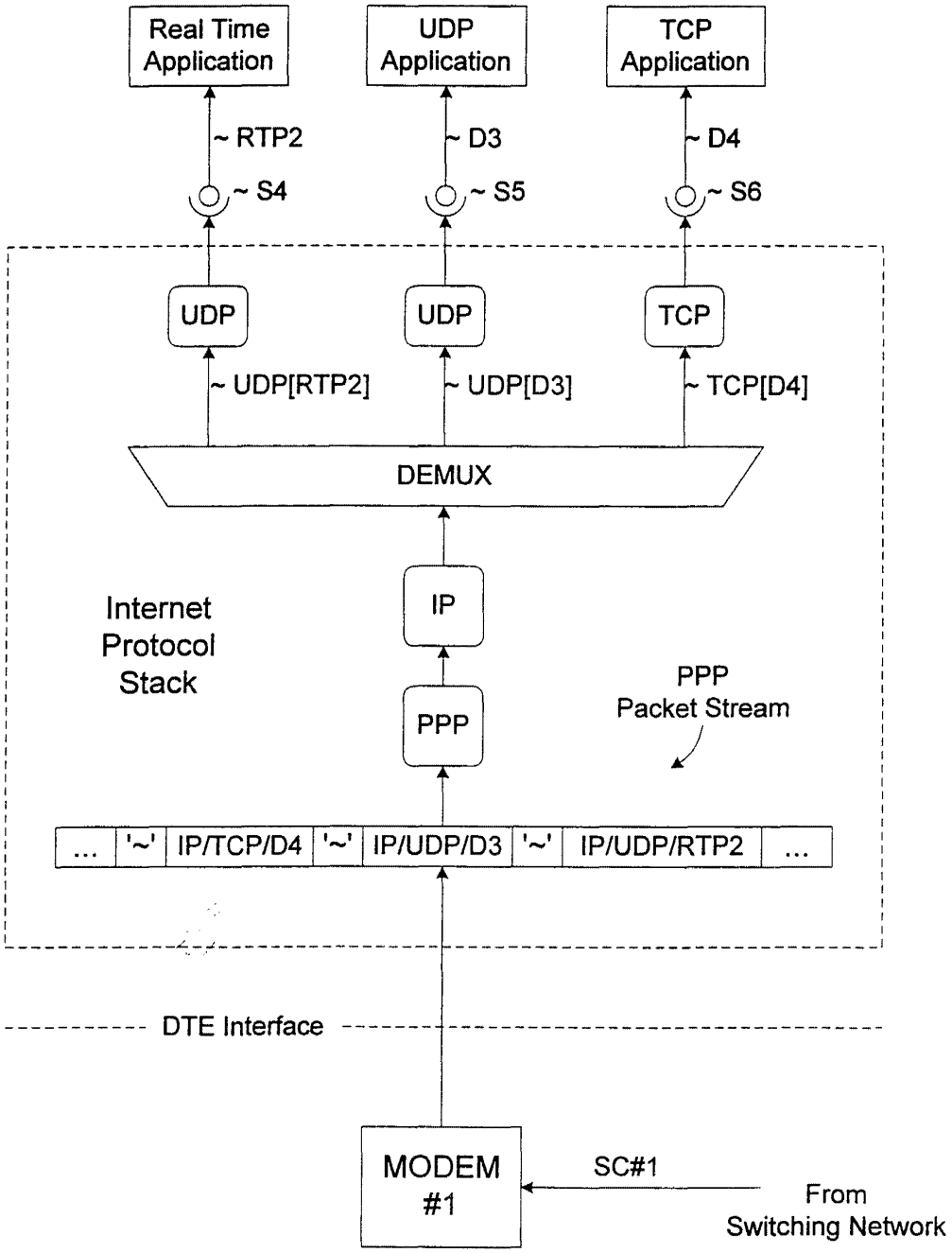


Fig. 2B
(Prior Art)



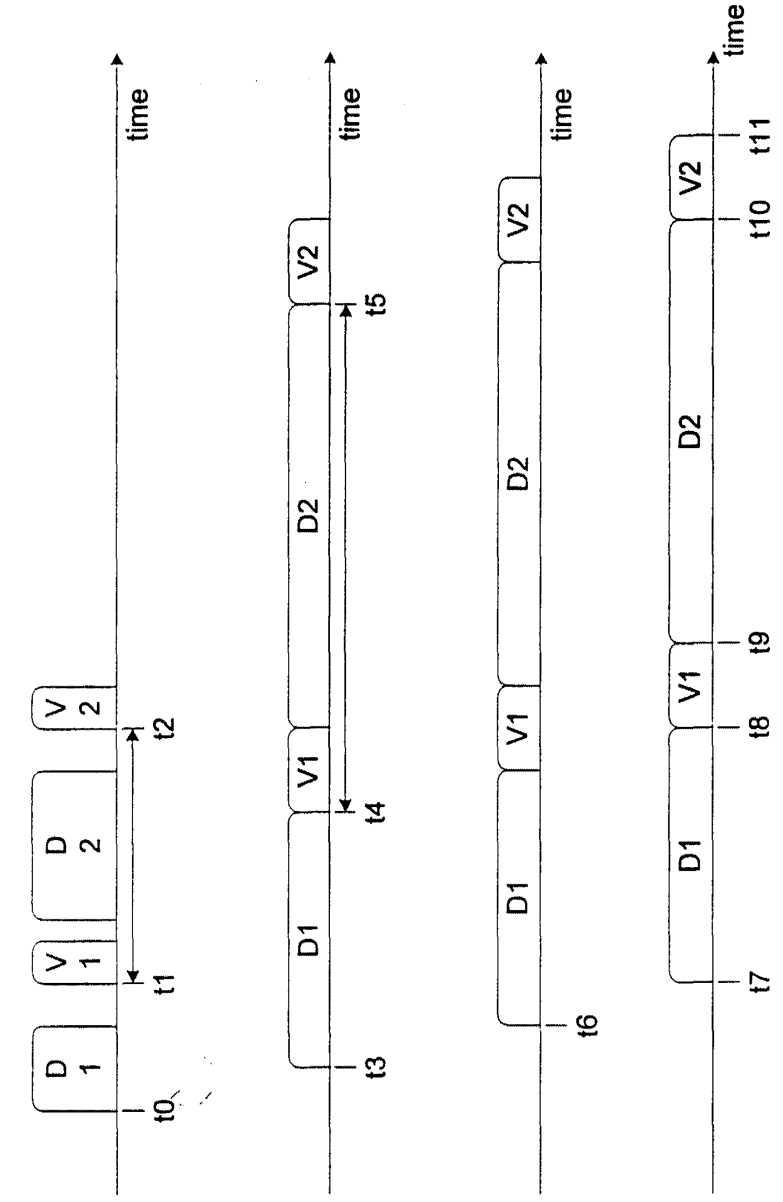
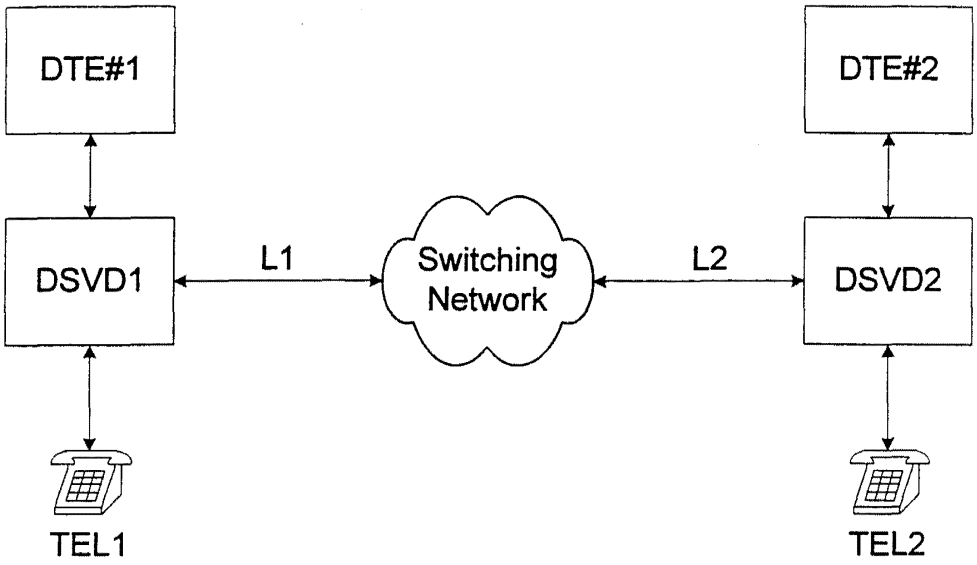


Fig.3
(Prior Art)

Fig. 4
(Prior Art)



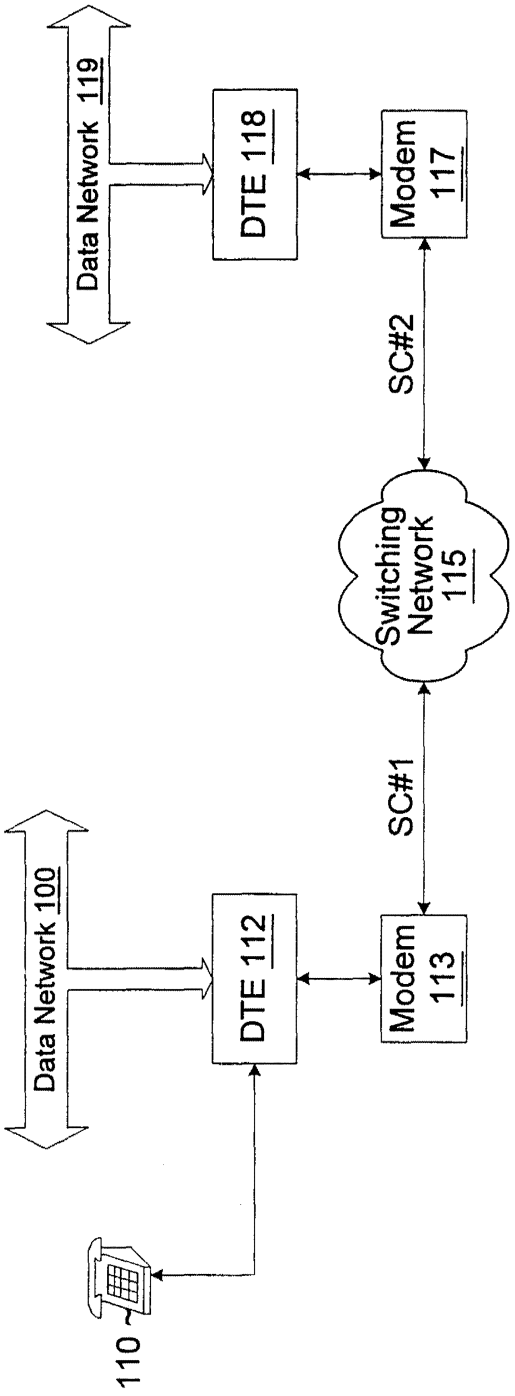
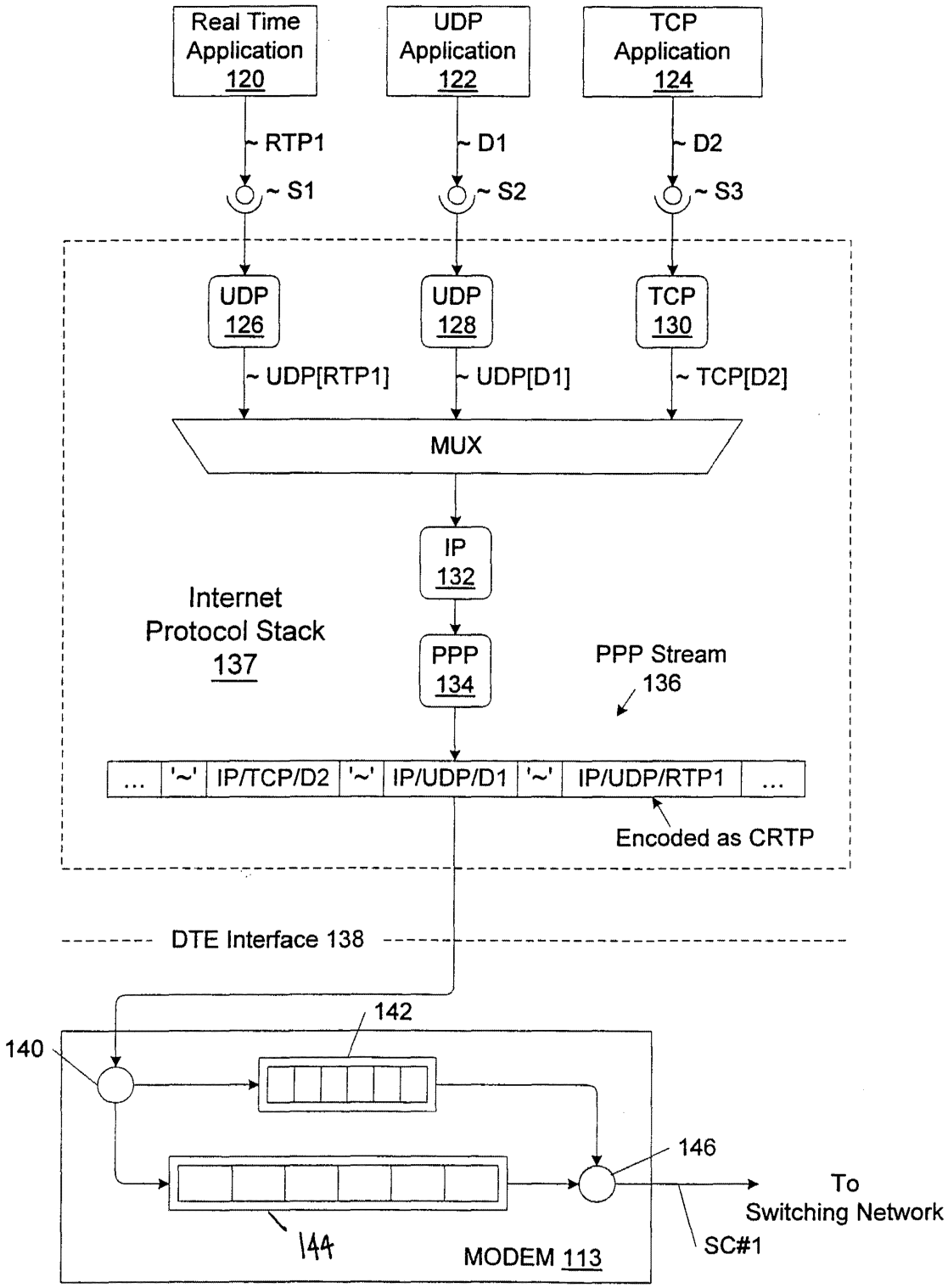

Data Network 119
DTE 118
Modem 117
SC#2
Switching Network 115
SC#1
Data Network 100
DTE 112
Modem 113
110

Fig. 5

Fig.6A


Real Time Application 120
UDP Application 122
TCP Application 124
RTP1
D1
D2
S1
S2
S3
UDP 126
UDP 128
TCP 130
UDP[RTP1]
UDP[D1]
TCP[D2]
MUX
IP 132
Internet Protocol Stack 137
PPP 134
PPP Stream 136
... '~' IP/TCP/D2 '~' IP/UDP/D1 '~' IP/UDP/RTP1 ...
Encoded as CRTP
DTE Interface 138
140
142
144
146
MODEM 113
To Switching Network
SC#1

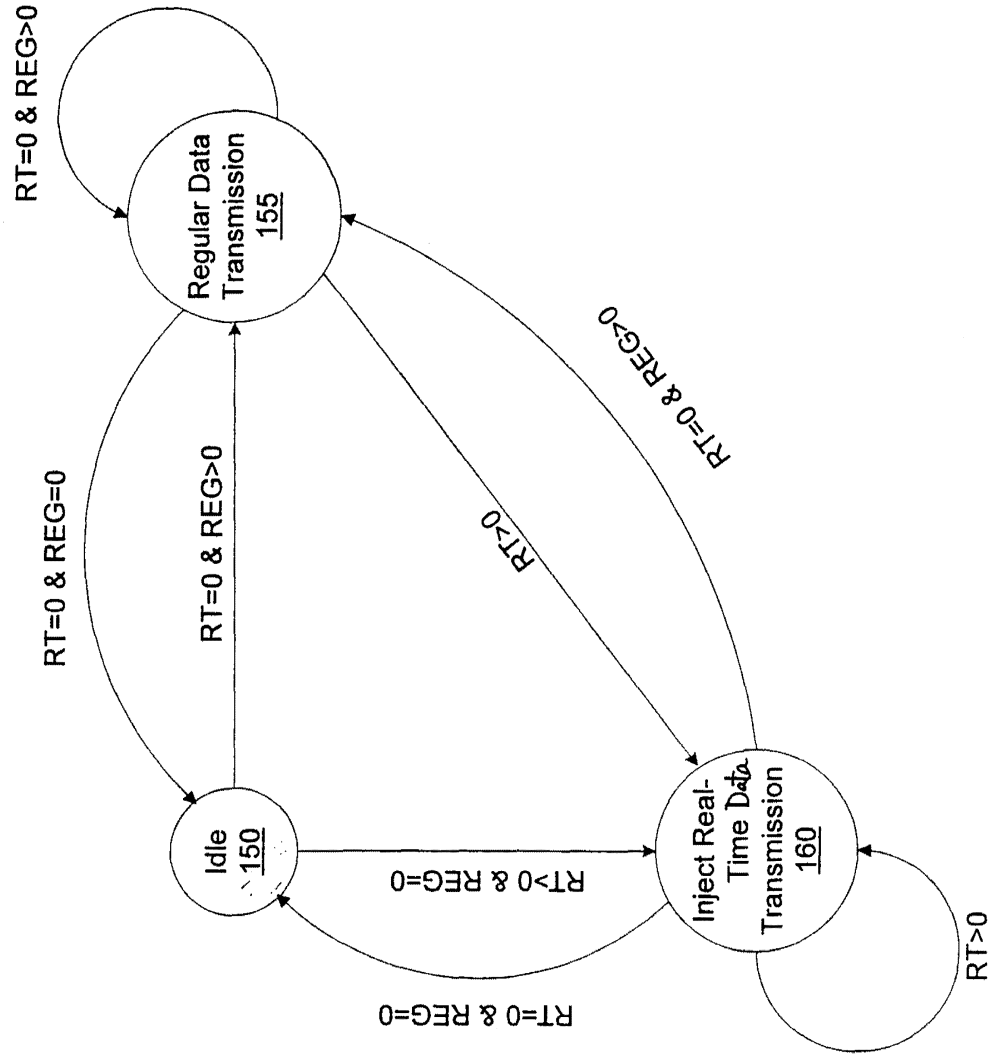
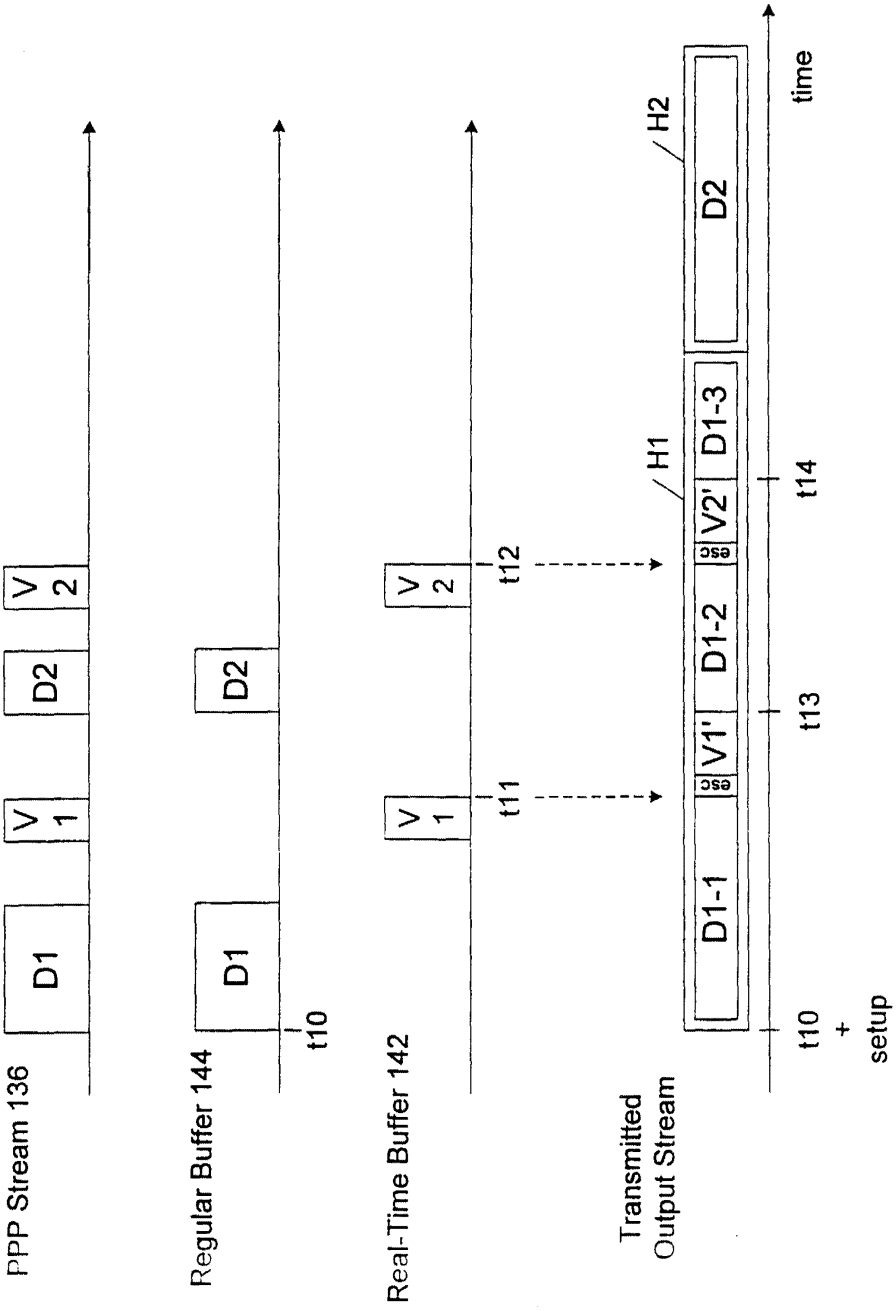

RT=0 & REG>0
Regular Data Transmission 155
RT=0 & REG=0
RT=0 & REG>0
RT>0
RT=0 & REG>0
Idle 150
RT>0 & REG=0
Inject Real-Time Data Transmission 160
RT=0 & REG=0
RT>0

Fig. 6B

Fig. 6C


PPP Stream 136
D1
V 1
D2
V 2
Regular Buffer 144
D1
D2
t10
Real-Time Buffer 142
V 1
V 2
t11
t12
Transmitted Output Stream
D1-1
esc
V1'
D1-2
esc
V2'
D1-3
H1
H2
D2
t10 + setup
t13
t14
time

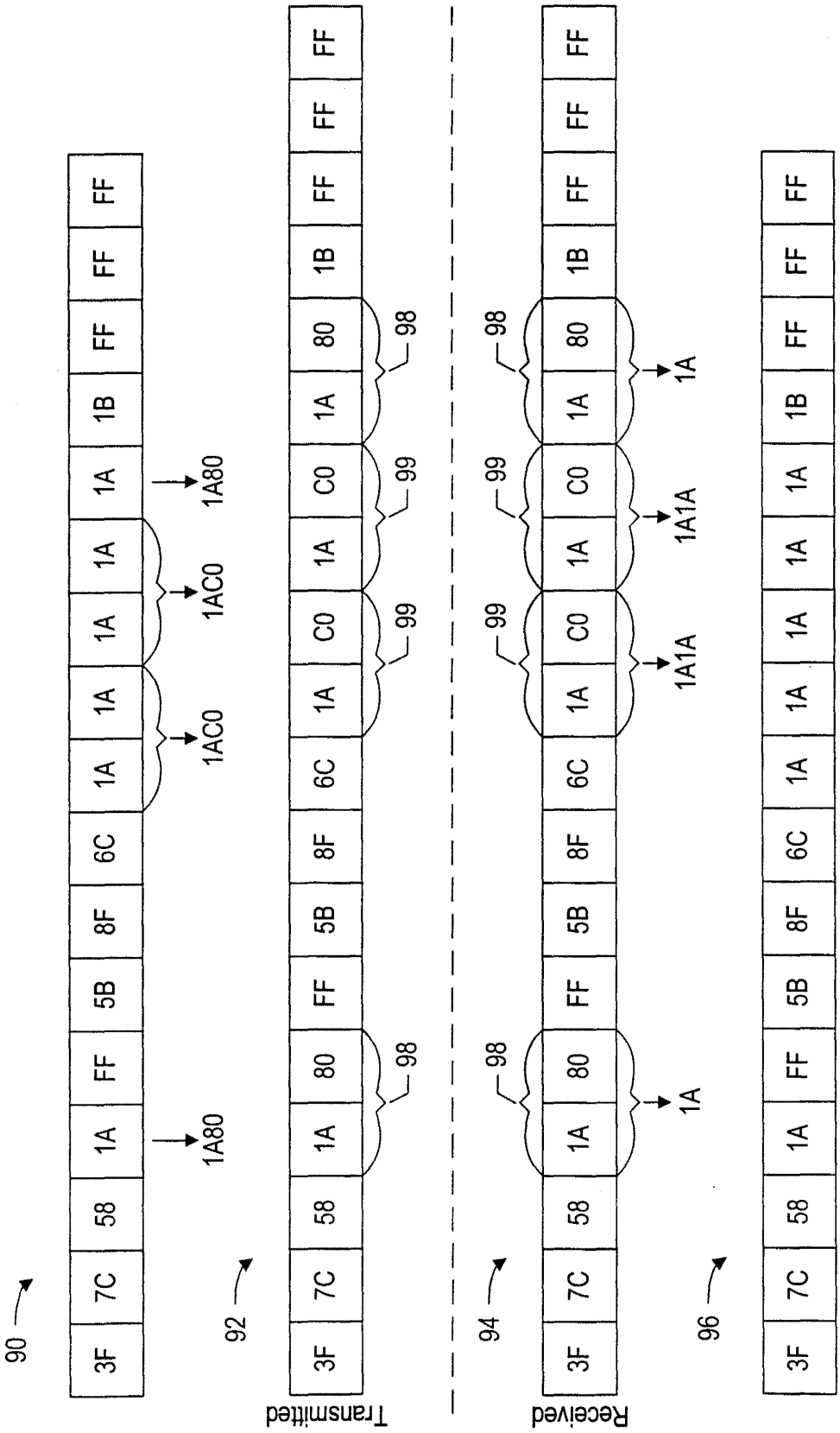

90
92
94
96
Transmitted
Received
3F 7C 58 1A FF 5B 8F 6C 1A 1A 1A 1A 1B FF FF FF
1A80
1AC0
1AC0
1A80
98
99
99
98
1A
1A1A
1A1A
1A

Fig. 6D

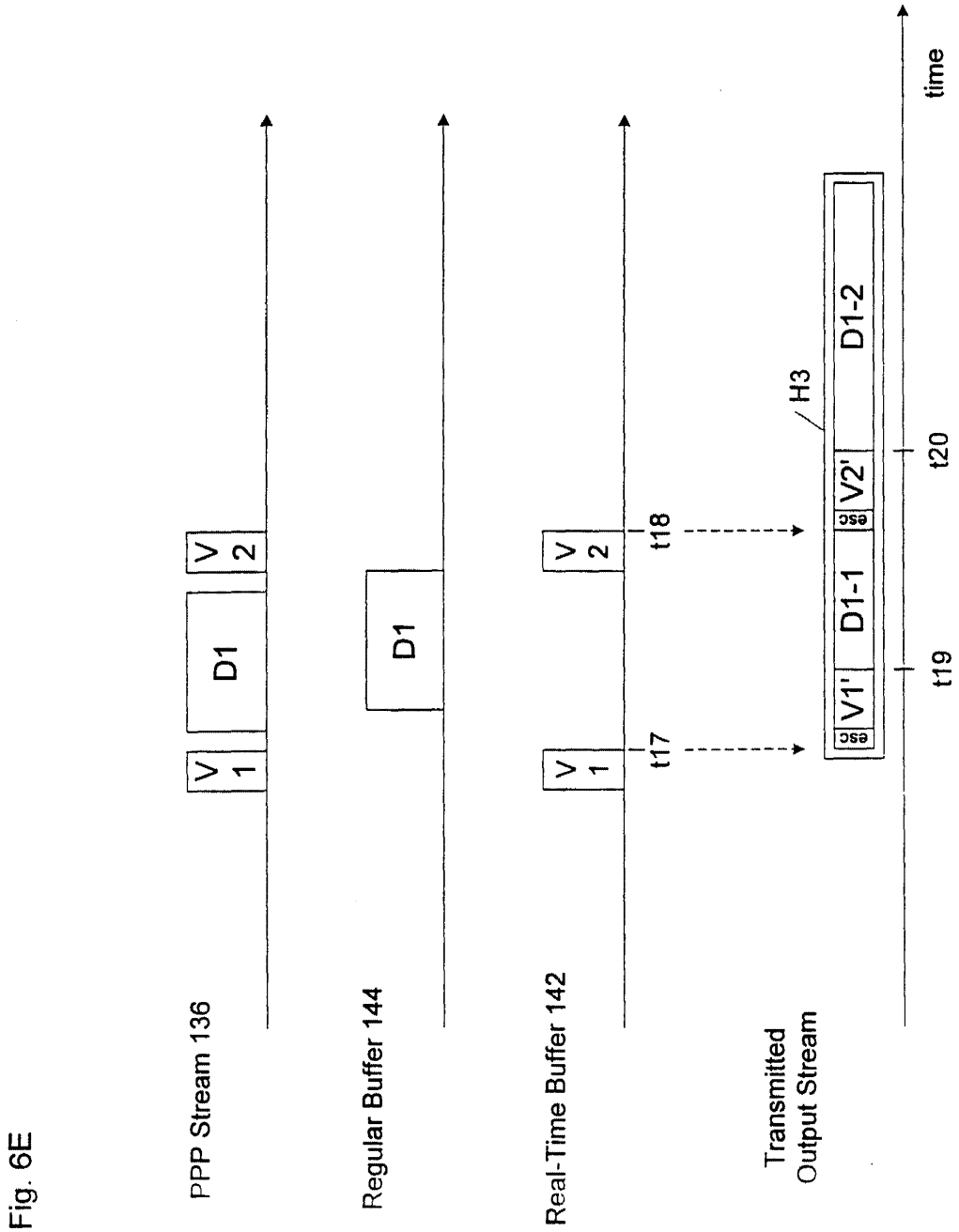

PPP Stream 136
V 1
D1
V 2
Regular Buffer 144
D1
Real-Time Buffer 142
V 1
V 2
t17
t18
H3
Transmitted Output Stream
esc
V1'
D1-1
esc
V2'
D1-2
t19
t20
time

Fig. 6E

Fig. 7A

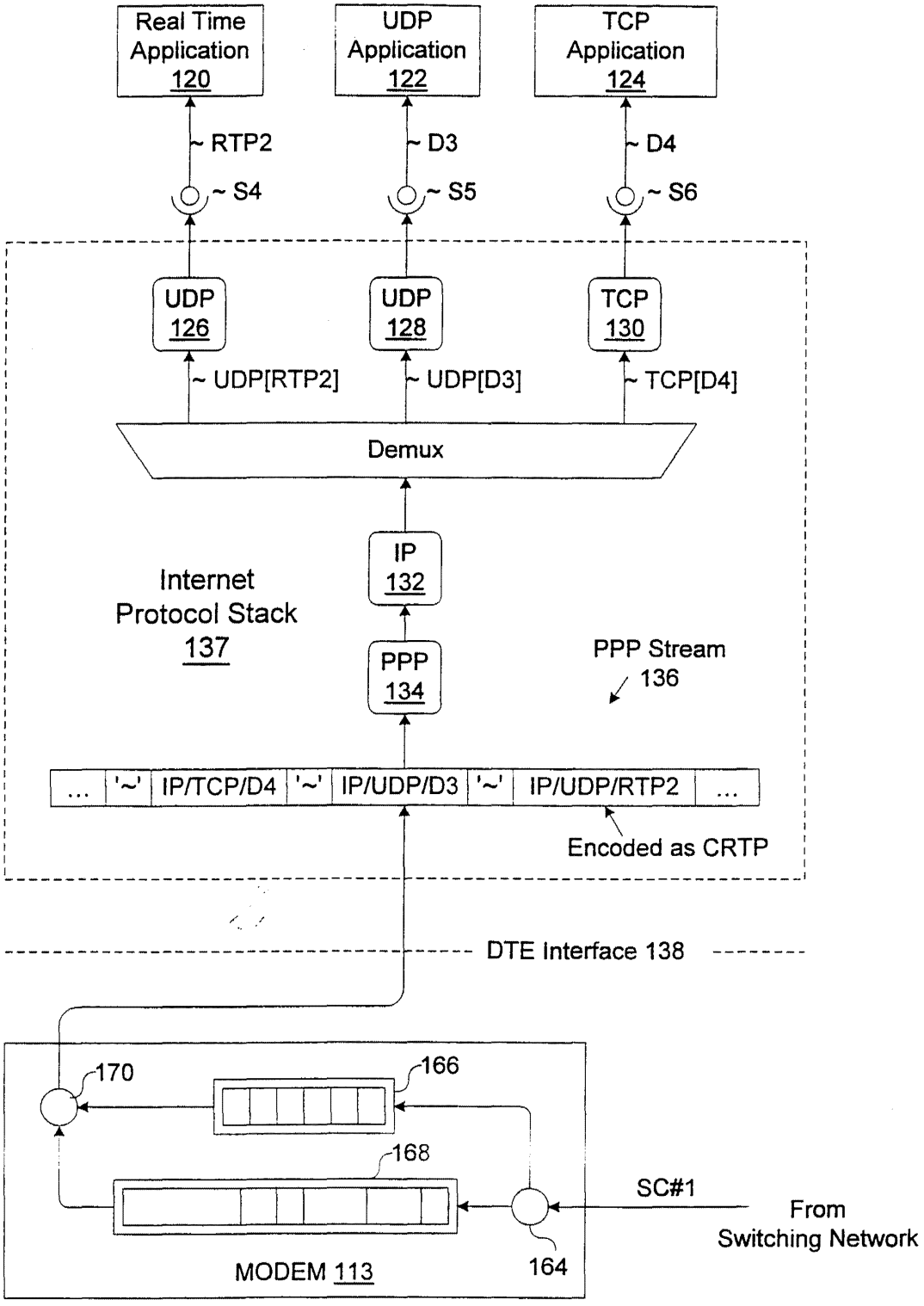
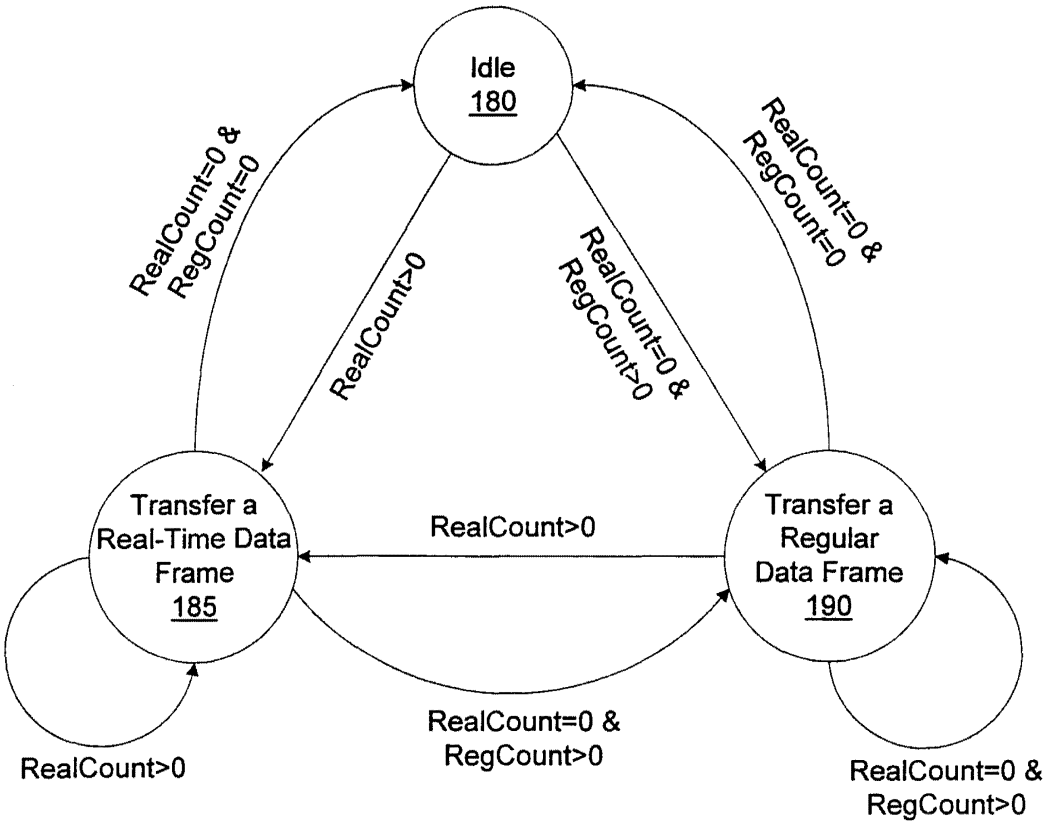

Real Time Application 120
UDP Application 122
TCP Application 124
RTP2
D3
D4
S4
S5
S6
UDP 126
UDP 128
TCP 130
UDP[RTP2]
UDP[D3]
TCP[D4]
Demux
IP 132
Internet Protocol Stack 137
PPP 134
PPP Stream 136
... '~' IP/TCP/D4 '~' IP/UDP/D3 '~' IP/UDP/RTP2 ...
Encoded as CRTP
DTE Interface 138
170
166
168
SC#1
From Switching Network
MODEM 113
164

Fig. 7B


Idle
180
RealCount=0 &
RegCount=0
RealCount>0
RealCount=0 &
RegCount>0
RealCount=0 &
RegCount=0
Transfer a
Real-Time Data
Frame
185
Transfer a
Regular
Data Frame
190
RealCount>0
RealCount=0 &
RegCount>0
RealCount>0
RealCount=0 &
RegCount>0

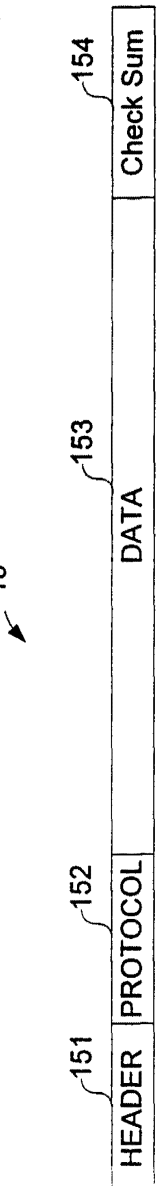

15
151
HEADER
152
PROTOCOL
153
DATA
154
Check Sum

Fig. 8A

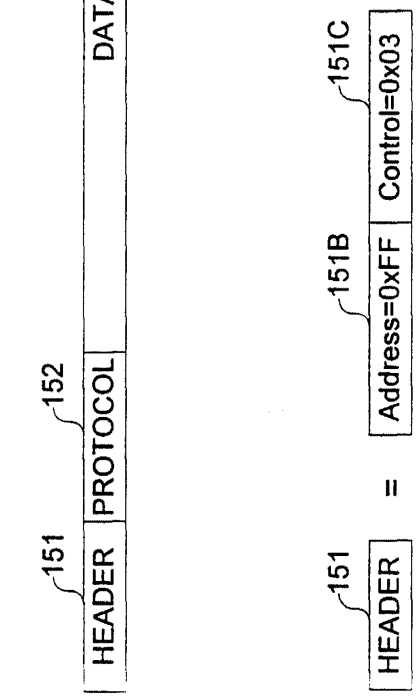
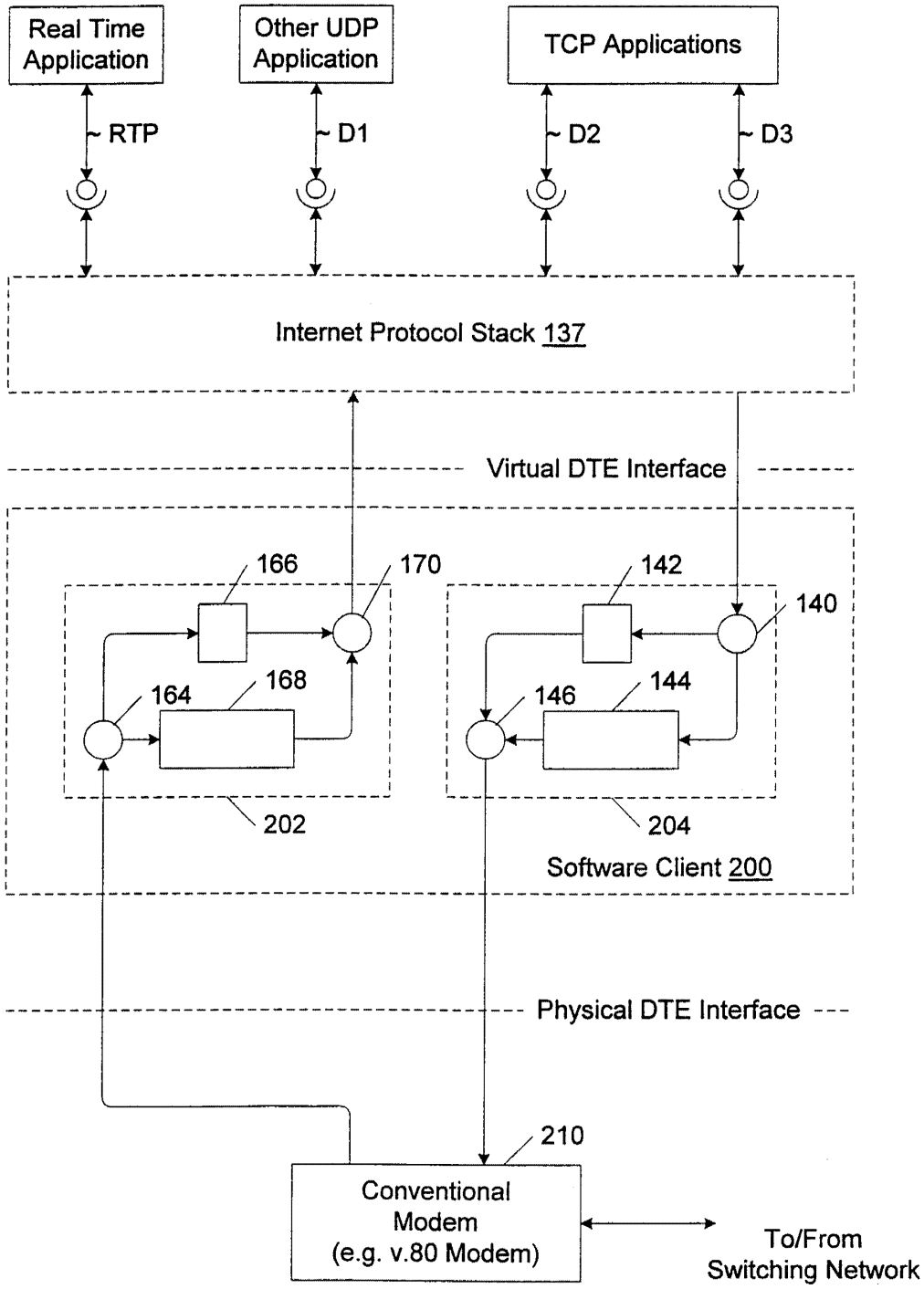

151
HEADER
=
151B
Address=0xFF
151C
Control=0x03

Fig. 8B

Fig. 9


Real Time Application
Other UDP Application
TCP Applications
RTP
D1
D2
D3
Internet Protocol Stack 137
Virtual DTE Interface
166
170
142
140
164
168
146
144
202
204
Software Client 200
Physical DTE Interface
210
Conventional Modem (e.g. v.80 Modem)
To/From Switching Network