PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04J 3/24, H04L 9/18, 12/56, H04K 1/00, H04N 7/10, 7/12, 7/52

A1

(11) International Publication Number: WO 99/37048

(43) International Publication Date: 22 July 1999 (22.07.99)

(21) International Application Number: PCT/US99/00360

(22) International Filing Date: 7 January 1999 (07.01.99)

(30) Priority Data:

09/007,211	14 January 1998 (14.01.98)	US
09/007,212	14 January 1998 (14.01.98)	US
09/007,334	14 January 1998 (14.01.98)	US
09/007,203	14 January 1998 (14.01.98)	US
09/007,204	14 January 1998 (14.01.98)	US
09/007,210	14 January 1998 (14.01.98)	US
09/006,963	14 January 1998 (14.01.98)	US
09/006,964	14 January 1998 (14.01.98)	US
09/007,198	14 January 1998 (14.01.98)	US
09/007,199	14 January 1998 (14.01.98)	US

(71) Applicant: SKYSTREAM CORPORATION [US/US]; Suite B, 555 Clyde Avenue, Mountain View, CA 94043–2211 (US).

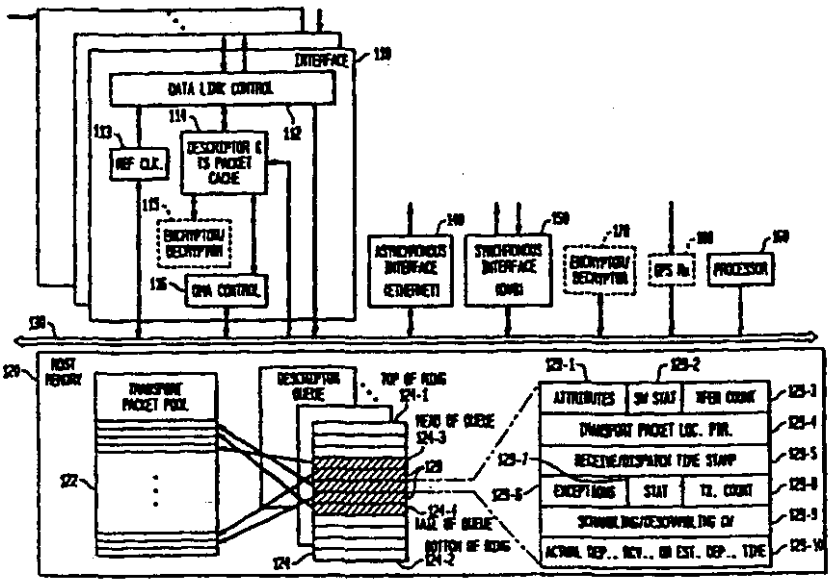
(72) Inventors: GRATACAP, Regis; 41 Ethyl Avenue, Mill Valley, CA 94941 (US). SLATTERY, William; 314 Almendra Avenue, Los Gatos, CA 95030 (US). ROBINETT, Robert; 485 Cotton Street, Menlo Park, CA 94025 (US).

(74) Agents: GUTTMAN, Charles et al.; Proskauer Rose LLP, Patent Dept., 1585 Broadway, New York, NY 10036–8299 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: VIDEO PROGRAM BEARING TRANSPORT STREAM REMULTIPLEXER



(57) Abstract

A method and system (30, 30', 100, 100', 100'', 100''') remultiplex video program bearing data (TS1–TS5, TS10–TS20), using a descriptor based system (122, 124, 129–4) for timely outputting transport packets, using a descriptor and transport packet caching technique (116, 122, 124, 114) for decoupling the synchronous receipt and transmission of transport packets from any asynchronous processing (160, 120, 130, S2, 402, S4, 404), using descriptors for managing scrambling and descrambling control words (129–9), optimizing bandwidth of transport streams by replacing null transport packets with transport packet data, and using a technique (180) for locking multiple internal reference clock generators (113).

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		

VIDEO PROGRAM BEARING TRANSPORT STREAM REMULTIPLEXER

Field of the Invention

The present invention pertains to communication systems. In particular, the invention pertains to selectively multiplexing bit streams containing one or more programs, such as real-time audio-video programs. Program specific and other program related information is adjusted so as to enable identification, extraction and real-time reproduction of the program at the receiving end of the bit streams.

Background of the Invention

Recently, techniques have been proposed for efficiently compressing digital audio-video programs for storage and transmission. See, for example, ISO\IEC IS 13818-1,2,3: Information Technology-Generic Coding of Moving Pictures and Associated Audio Information: Systems, Video and Audio ("MPEG-2"); ISO\IEC IS 11172-1,2,3: Information Technology-Generic Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/sec: Systems, Video and Audio ("MPEG-1"); Dolby AC-3; Motion JPEG, etc. Herein, the term program means a collection of related audio-video signals having a common time base and intended for synchronized presentation, as per MPEG-2 parlance.

MPEG-1 and MPEG-2 provide for hierarchically layered streams. That is, an audio-video program is composed of one or more coded bit streams or "elementary streams" ("ES") such as an encoded video ES, and encoded audio ES, a second language encoded

audio ES, a closed caption text ES, etc. Each ES, in particular, each of the audio and video ESs, is separately encoded. The encoded ESs are then combined into a systems layer stream such as a program stream "PS" or a transport stream "TS". The purpose of the PS or TS is to enable extraction of the encoded ESs of a program, separation and separate decoding of each ES and synchronized presentation of the decoded ESs. The TS or PS may be encapsulated in an even higher channel layer or storage format which provides for forward error correction.

Elementary Streams

Audio ESs are typically encoded at a constant bit rate, e.g., 384 kbps. Video ESs, on the other hand, are encoded according to MPEG-1 or MPEG-2 at a variable bit rate. This means that the number of bits per compressed/encoded picture varies from picture to picture (which pictures are presented or displayed at a constant rate). Video encoding involves the steps of spatially and temporally encoding the video pictures. Spatial encoding includes discrete cosine transforming, quantizing, (zig-zag) scanning, run length encoding and variable length encoding blocks of luminance and chrominance pixel data. Temporal coding involves estimating the motion of macroblocks (e.g., a 4x4 array of luminance blocks and each chrominance block overlaid thereon) to identify motion vectors, motion compensating the macroblocks to form prediction error macroblocks, spatially encoding the prediction error macroblocks and variable length encoding the motion vectors. Some pictures, called I pictures, are only spatially encoded, whereas other pictures, such as P and B pictures are both spatially and motion compensated encoded (i.e., temporally predicted from other pictures). Encoded I pictures typically have more bits than encoded P pictures

and encoded P pictures typically have more bits than encoded B pictures. In any event, even encoded pictures of the same type tend to have different numbers of bits.

MPEG-2 defines a buffer size constraint on encoded video ESs. In particular, a decoder is presumed to have a buffer with a predefined maximum storage capacity. The encoded video ES must not cause the decoder buffer to overflow (and in some cases, must not cause the decoder buffer to underflow). MPEG-2 specifically defines the times at which each picture's compressed data are removed from the decoder buffer in relation to the bit rate of the video ES, the picture display rate and certain picture reordering constraints imposed to enable decoding of predicted pictures (from the reference pictures from which they were predicted). Given such constraints, the number of bits produced in compressing a picture can be adjusted (as frequently as on a macroblock by macroblock basis) to ensure that the video ES does not cause the video ES decoder buffer to underflow or overflow.

Transport Streams

This invention is illustrated herein for TSs. For sake of brevity, the discussion of PSs is omitted. However, those having ordinary skill in the art will appreciate the applicability of certain aspects of this invention to PSs.

The data of each ES is formed into variable length program elementary stream or "PES" packets. PES packets contain data for only a single ES, but may contain data for more than one decoding unit (e.g., may contain more than one compressed picture, more than one compressed audio frame, etc.). In the case of a TS, the PES packets are first divided into a number of payload units and inserted into fixed length (188 byte long)

transport packets. Each transport packet may carry payload data of only one type, e.g., PES packet data for only one ES. Each TS is provided with a four byte header that includes a packet identifier or "PID." The PID is analogous to a tag which uniquely indicates the contents of the transport packet. Thus, one PID is assigned to a video ES of a particular program, a second, different PID is assigned to the audio ES of a particular program, etc.

The ESs of each program are encoded in relation to a single encoder system time clock. Likewise, the decoding and synchronized presentation of the ESs are, in turn, synchronized in relation to the same encoder system time clock. Thus, the decoder must be able to recover the original encoder system time clock in order to be able to decode each ES and present each decoded ES in a timely and mutually synchronized fashion. To that end, time stamps of the system time clock, called program clock references or "PCRs," are inserted into the payloads of selected transport packets (specifically, in adaption fields). The decoder extracts the PCRs from the transport packets and uses the PCRs to recover the encoder system time clock. The PES packets may contain decoding time stamps or "DTSs" and/or presentation time stamps or "PTSs". A DTS indicates the time, relative to the recovered encoder system time clock, at which the next decoding unit (i.e., compressed audio frame, compressed video picture, etc.) should be decoded. The PTS indicates the time, relative to the recovered encoder system time clock, at which the next presentation unit (i.e., decompressed audio frame, decompressed picture, etc.) should be presented or displayed.

Unlike the PS, a TS may have transport packets that carry program data for more than one program. Each program may have been encoded at a different encoder in relation to a different encoder system time clock. The TS enables the decoder to recover the

specific system time clock of the program which the decoder desires to decode. To that end, the TS must carry separate sets of PCRs, i.e., one set of PCRs for recovering the encoder system time clock of each program.

The TS also carries program specific information or (PSI) in transport packets. PSI is for identifying data of a desired program or other information for assisting in decoding a program. A program association table or "PAT" is provided which is carried in transport packets with the PID 0x0000. The PAT correlates each program number with the PID of the transport packets carrying program definitions for that program. A program definition: (1) indicates which ESs make up the program to which the program definition corresponds, (2) identifies the PIDs for each of those ESs, (3) indicates the PID of the transport packets carrying the PCRs of that program (4) identifies the PIDs of transport packets carrying ES specific entitlement control messages (e.g., descrambling or decryption keys) and other information. Collectively, all program definitions of a TS are referred to as a program mapping table (PMT). Thus, a decoder can extract the PAT data from the transport packets and use the PAT to identify the PID of the transport packets carrying the program definition of a desired program. The decoder can then extract from the transport packets the program definition data of the desired program and identify the PIDs of the transport packets carrying the ES data that makes up the desired program and of the transport packets carrying the PCRs. Using these identified PIDs, the decoder can then extract from the transport packets of the TSs the ES data of the ESs of the desired program and the PCRs of that program. The decoder recovers the encoder system time clock from the PCRs of the desired program and decodes and presents the ES data at times relative to the recovered encoder system time clock.

Other types of information optionally provided in a TS include entitlement control messages (ECMs), entitlement management messages (EMMs), a conditional access table (CAT) and a network information table (NIT) (the CAT and NIT also being types of PSI). ECMs are ES specific messages for controlling the ability of a decoder to interpret the ES to which the ECM pertains. For example, an ES may be scrambled and the descrambling key or control word may be an ECM. The ECMs associated with a particular ES are placed in their own transport packets and are labeled with a unique PID. EMMs, on the other hand, are system wide messages for controlling the ability of a set of decoders (which set is in a system referred to as a "conditional access system") to interpret portions of a TS. EMMs are placed in their own transport packets and are labeled with a PID unique to the conditional accesses system to which the EMMs pertain. A CAT is provided whenever EMMs are present for enabling a decoder to locate the EMMs of the conditional access system of which the decoder is a part (i.e., of the set of decoders of which the decoder is a member). The NIT maintains various network parameters. For example, if multiple TSs are modulated on different carrier frequencies to which a decoder receiver can tune, the NIT may indicate on which carrier frequency (the TS carrying) each program is modulated.

Like the video ES, MPEG-2 requires that the TS be decoded by a decoder having TS buffers of predefined sizes for storing program ES and PSI data. MPEG-2 also defines the rate at which data flows into and out of such buffers. Most importantly, MPEG-2 requires that the TS not overflow or underflow the TS buffers.

To further prevent buffer overflow or underflow, MPEG-2 requires that data transported from an encoder to a decoder experience a constant end-to-end delay, and that the appropriate program and ES bit rate be maintained. In addition, to ensure that ESs are

timely decoded and presented, the relative time of arrival of the PCRs in the TS should not vary too much from the relative time indicated by such PCRs. Stated another way, each PCR indicates the time that the system time clock (recovered at the decoder) should have when the last byte containing a portion of the PCR is received. Thus, the time of receipt of successive PCRs should correlate with the times indicated by each PCR.

Remultiplexing

Often it is desired to "remultiplex" TSs. Remultiplexing involves the selective modification of the content of a TS, such as adding transport packets to a TS, deleting transport packets from a TS, rearranging the ordering of transport packets in a TS and/or modifying the data contained in transport packets. For example, sometimes it is desirable to add transport packets containing a first program to a TS that contains other programs. Such an operation involves more steps than simply adding the transport packets of the first program. In the very least, the PSI, such as, the PAT and PMT, must be modified so that it correctly references the contents of the TS. However, the TS must be further modified to maintain the constant end-to-end delay of each program carried therein. Specifically, the bit rate of each program must not change to prevent TS and video decoder buffer underflow and overflow. Furthermore, any temporal misalignment introduced into the PCRs of the TS, for example, as a result of changing the relative spacing/rate of receipt of successive transport packets bearing PCRs of the same program, must be removed.

The prior art has proposed a remultiplexer for MPEG-2 TSs. The proposed remultiplexer is a sophisticated, dedicated piece of hardware that provides complete synchronicity between the point that each inputted to-be-remultiplexed TS is received to

the point that the final remultiplexed outputted TS is outputted--a single system time clock controls and synchronizes receipt, buffering, modification, ~~transfer,~~ reassembly and output of transport packets. While such a remultiplexer is capable of remultiplexing TSs, the remultiplexer architecture is complicated and requires a dedicated, uniformly synchronous platform.

It is an object of the present invention to provide a flexible remultiplexing architecture that can, for instance, reside on an arbitrary, possibly asynchronous, platform.

A program encoder is known which compresses the video and audio of a single program and produces a single program bearing TS. As noted above, MPEG-2 imposes very tight constraints on the bit rate of the TS and the number of bits that may be present in the video decoder buffer at any moment of time. It is difficult to encode an ES, in particular a video ES, and ensure that the bit rate remain completely constant from moment to moment. Rather, some overhead bandwidth must be allocated to each program to ensure that ES data is not omitted as a result of the ES encoder producing an unexpected excessive amount of encoded information. On the other hand, the program encoder occasionally does not have any encoded program data to output at a particular transport packet time slot. This may occur because the program encoder has reduced the number of bits to be outputted at that moment to prevent a decoder buffer overflow. Alternatively, this may occur because the program encoder needs an unanticipated longer amount of time to encode the ESs and therefore has no data available at that instant of time. To maintain the bit rate of the TS and prevent a TS decoder buffer underflow, a null transport packet is inserted into the transport packet time slot.

The presence of null transport packets in a to-be-remultiplexed TS is often a constraint that simply must be accepted. It is an object of the present invention to optimize the bandwidth of TSs containing null transport packets.

Sometimes, the TS or ES data is transferred via an asynchronous communication link. It is an object of the present invention to "re-time" such un-timed or asynchronously transferred data. It is also an object of the present invention to minimize jitter in transport packets transmitted from such asynchronous communication links by timing the transmission of such transport packets.

It is also an object of the present invention to enable the user to dynamically change the content remultiplexed into the remultiplexed TS, i.e., in real-time without stopping the flow of transport packets in the outputted remultiplexed TS.

It is a further object of the present invention to distribute the remultiplexing functions over a network. For example, it is an object to place one or more TS or ES sources at arbitrary nodes of an communications network which may be asynchronous (such as an Ethernet LAN) and to place a remultiplexer at another node of such a network.

Summary of the Invention

These and other objects are achieved according to the present invention. An illustrative application of the invention is the remultiplexing one or more MPEG-2 compliant transport streams (TSs). TSs are bit streams that contain the data of one or more compressed/encoded audio-video programs. Each TS is formed as a sequence of fixed length transport packets. Each compressed program includes data for one or more compressed elementary streams (ESs), such as a digital video signal and/or a digital audio

signal. The transport packets also carry program clock references (PCRs) for each program, which are time stamps of an encoder system time clock to which the decoding and presentation of the respective program is synchronized. Each program has a predetermined bit rate and is intended to be decoded at a decoder having a TS buffer and a video decoder buffer of predetermined sizes. Each program is encoded in a fashion so as to prevent overflow and underflow of these buffers. Program specific information (PSI) illustratively is also carried in selected transport packets of the TS for assisting in decoding the TS.

According to one embodiment, a remultiplexer node is provided with one or more adaptors, each adaptor including a cache, a data link control circuit connected to the cache and a direct memory access circuit connected to the cache. The adaptor is a synchronous interface with special features. The data link control circuit has an input port for receiving transport streams and an output port for transmitting transport streams. The direct memory access circuit can be connected to an asynchronous communication link with a varying end-to-end communication delay, such as a bus of the remultiplexer node. Using the asynchronous communication link, the direct memory access circuit can access a memory of the remultiplexer node. The memory can store one or more queues of descriptor storage locations, such as a queue assigned to an input port and a queue assigned to an output port. The memory can also store transport packets in transport packet storage locations to which descriptors stored in such descriptor storage locations of each queue point. Illustratively, the remultiplexer node includes a processor, connected to the bus, for processing transport packets and descriptors.

When an adaptor is used to input transport streams, the data link control circuit allocates to each received transport packet to be retained, an unused descriptor in one of a

sequence of descriptor storage locations, of a queue allocated to the input port. The allocated descriptor is in a descriptor storage location of which the cache has obtained control. The data link control circuit stores each retained transport packet at a transport packet storage location of which the cache has obtained control and which is pointed to by the descriptor allocated thereto. The direct memory access circuit obtains control of one or more unused descriptor storage locations of the queue in the memory following a last descriptor storage location of which the cache has already obtained control. The direct memory access circuit also obtains control of transport packet locations in the memory to which such descriptors in the one or more descriptor storage locations point.

When an adaptor is used to output transport packets, the data link control circuit retrieves from the cache each descriptor of a sequence of descriptor storage locations of a queue assigned to the output port. The descriptors are retrieved from the beginning of the sequence in order. The data link control circuit also retrieves from the cache the transport packets stored in transport packet storage locations to which the retrieved descriptors point. The data link control circuit outputs each retrieved transport packet in a unique time slot (i.e., one transport packet per time slot) of a transport stream outputted from the output port. The direct memory access circuit obtains from the memory for storage in the cache, descriptors of the queue assigned to the output port in storage locations following the descriptor storage locations in which a last cached descriptor of the sequence is stored. The direct memory access circuit also obtains each transport packet stored in a transport packet location to which the obtained descriptors point.

According to another embodiment, each descriptor is (also) used to record a receipt time stamp, indicating when a transport packet is received at an input port, or a dispatch

time stamp, indicating the time at which a transport packet is to be transmitted from an output port. In the case of transport packets received at an input port, the data link control circuit records a receipt time stamp in the descriptor allocated to each received and retained transport packet indicating a time at which the transport packet was received. The descriptors are maintained in order of receipt in the receipt queue. In the case of outputting transport packets from an output port, the data link control circuit sequentially retrieves each descriptor from the transmit queue, and the transport packet to which each retrieved descriptor points. At a time corresponding to a dispatch time recorded in each retrieved descriptor, the data link control circuit transmits the retrieved transport packet to which each retrieved descriptor points in a time slot of the outputted transport stream corresponding to the dispatch time recorded in the retrieved descriptor.

Illustratively, the remultiplexer node processor examines each descriptor in the receipt queue, as well as other queues containing descriptors pointing to to-be-outputted transport packets. The processor allocates a descriptor of the transmit queue associated with an output port from which a transport packet pointed to by each examined descriptor is to be transmitted (if any). The processor assigns a dispatch time to the allocated descriptor of the transmit queue, depending on, for example, a receipt time of the transport packet to which the descriptor points and an internal buffer delay between receipt and output of the transport packet. The processor furthermore orders the descriptors of the transmit queue in order of increasing dispatch time.

A unique PCR normalization process is also provided. The processor schedules each transport packet to be outputted in a time slot at a particular dispatch time, corresponding to a predetermined delay in the remultiplexer node. If the scheduled

transport packet contains a PCR, the PCR is adjusted based on a drift of the local reference clock(s) relative to the program of the system time clock from which the PCR was generated, if any drift exists. The data link control circuit, that transmits such adjusted PCR bearing transport packets, further adjust each adjusted PCR time stamp based on a difference between the scheduled dispatch time of the transport packet and an actual time at which the time slot occurs relative to an external clock.

Illustratively, if more than one transport packet is to be outputted in the same time slot, each such transport packet is outputted in a separate consecutive time slot. The processor calculates an estimated adjustment for each PCR in a transport packet scheduled to be outputted in a time slot other than the time slot as would be determined using the predetermined delay. The estimated adjustment is based on a difference in output time between the time slot in which the processor has actually scheduled the transport packet bearing the PCR to be outputted and the time slot as determined by the predetermined delay. The processor adjusts the PCRs according to this estimated adjustment.

According to one embodiment, the descriptors are also used for controlling scrambling or descrambling of transport packets. In the case of descrambling, the processor defines a sequence of one or more processing steps to be performed on each transport packet and orders descrambling processing within the sequence. The processor stores control word information associated with contents of the transport packet in the control word information storage location of the allocated descriptors. The data link control circuit allocates descriptors to each received, retained transport packet, which descriptors each include one or more processing indications and a storage location for control word information. The data link control circuit sets one or more of the processing indications of

the allocated descriptor to indicate that the next step of processing of the sequence may be performed on each of the allocated descriptors. A descrambler is provided for sequentially accessing each allocated descriptor. If the processing indications of the accessed descriptor are set to indicate that descrambling processing may be performed on the accessed descriptor (and transport packet to which the accessed descriptor points), then the descrambler processes the descriptor and transport packet to which it points. Specifically, if the descriptor points to a to-be-descrambled transport packet, the descrambler descrambles the transport packet using the control word information in the accessed descriptor.

The descrambler may be located on the (receipt) adaptor, in which case the descrambler processing occurs after processing by the data link control circuit (e.g., descriptor allocation, receipt time recording, etc.) but before processing by the direct memory access circuit (e.g., transfer to the memory). Alternatively, the descrambler may be a separate device connected to the asynchronous communication interface, in which case descrambler processing occurs after processing by the direct memory access circuit but before processing by the processor (e.g., estimated departure time calculation, PID remapping, etc.). In either case, the control word information is a base address of a PID index-able control word table maintained by the processor.

In the case of scrambling, the processor defines a sequence of one or more processing steps to be performed on each transport packet and orders scrambling processing within the sequence. The processor allocates a transmit descriptor of a transmit queue to each to-be-transmitted transport packet and stores control word information associated with contents of the transport packet in the control word information storage location of selected

ones of the allocated descriptors. The processor then sets one or more processing indications of the descriptor to indicate that the next step of processing of the sequence may be performed on each of the allocated descriptors. A scrambler is provided for sequentially accessing each allocated descriptor. The scrambler processes each accessed descriptor and transport packet to which the accessed descriptor points, but only if the processing indications of the accessed descriptors are set to indicate that scrambling processing may be performed on the accessed descriptor (and transport packet to which the accessed descriptor points). Specifically, if the accessed descriptor points to a to-be-scrambled transport packet, the scrambler scrambles the transport packet pointed to by the accessed descriptor using the control word information in the accessed descriptor.

The scrambler may be located on the (transmit) adaptor, in which case the scrambler processing occurs after processing by the direct memory access circuit (e.g., transfer from the memory to the cache, etc.) but before processing by the data link control circuit (e.g., output at the correct time slot, final PCR correction, etc.). Alternatively, the scrambler may be a separate device connected to the asynchronous communication interface, in which case descrambler processing occurs after processing by the processor (e.g., transmit queue descriptor allocation, dispatch time assignment, PCR correction, etc.) but before processing by the direct memory access circuit. The control word information may be a base address of a PID index-able control word table maintained by the processor, as with descrambling. Preferably, however, the control word information is the control word itself, used to scramble the transport packet.

In addition, according to an embodiment, a method is provided for re-timing video program bearing data received via an asynchronous communication link. An asynchronous

interface (e.g., an Ethernet interface, ATM interface, etc.) is connected to the remultiplexer node processor (e.g., via a bus) for receiving a video program bearing bit stream from a communication link having a varying end-to-end transmission delay. The processor determines a time at which each of one or more received packets carrying data of the same program of the received bit stream should appear in an outputted TS based on a plurality of time stamps of the program carried in the received bit stream. A synchronous interface, such as a transmit adaptor, selectively transmits selected transport packets carrying received data in an outputted TS with a constant end-to-end delay at times that depend on the determined times.

Illustratively, the remultiplexer node memory stores packets containing data received from the received bit stream in a receipt queue. The processor identifies each packet containing data of a program stored in the receipt queue between first and second particular packets containing consecutive time stamps of that program. The processor determines a (transport) packet rate of the program based on a difference between the first and second time stamps. The processor assigns as a transmit time to each of the identified packets, the sum of a transmit time assigned to the first particular packet and a product of the (transport) packet rate and an offset of the identified packet from the first packet.

According to yet another embodiment, a method is provided for dynamically and seamlessly varying remultiplexing according to a changed user specification. An interface, such as a first adaptor, selectively extracts only particular ones of the transport packets from a TS according to an initial user specification for remultiplexed TS content. A second interface, such as a second adaptor, reassembles selected ones of the extracted transport packets, and, transport packets containing PSI, if any, into an outputted remultiplexed TS,

according to the initial user specification for remultiplexed TS content. The second adaptor furthermore outputs the reassembled remultiplexed TS as ~~a continuous~~ bitstream. The processor dynamically receives one or more new user specifications for remultiplexed TS content which specifies one or more of: (I) different transport packets to be extracted and/or (II) different transport packets to be reassembled, while the first and second adaptors extract transport packets and reassemble and output the remultiplexed TS. In response, the processor causes the first and second adaptors to dynamically cease to extract or reassemble transport packets according to the initial user specification and to dynamically begin to extract or reassemble transport packets according to the new user specification without introducing a discontinuity in the outputted remultiplexed transport stream. For example, the processor may generate substitute PSI that references different transport packets as per the new user specification, for reassembly by the second adaptor.

Illustratively, this seamless remultiplexing variation technique can be used to automatically ensure that the correct ES information of each selected program is always outputted in the remultiplexed outputted TS, despite any changes in the ES make up of that program. A controller may be provided for generating a user specification indicating one or more programs of the inputted TSs to be outputted in the output TS. The first adaptor continuously captures program definitions of an inputted TS. The processor continuously determines from the captured program definitions which elementary streams make up each program. The second adaptor outputs in the outputted TS each transport packet containing ES data of each ES determined to make up each program indicated to be outputted by the user specification without introducing a discontinuity into the outputted TS. Thus, even if

the PIDs of the ESs that make up each program change (in number or value) the correct and complete ES data for each program is nevertheless always outputted in the outputted TS.

According to yet another embodiment, a method is provided for optimizing the bandwidth of a TS which has null transport packets inserted therein. The first interface (adaptor) receives a TS at a predetermined bit rate, which TS includes variably compressed program data bearing transport packets and one or more null transport packets. Each of the null transport packets is inserted into a time slot of the received TS to maintain the predetermined bit rate of the TS when none of the compressed program data bearing transport packets are available for insertion into the received TS at the respective transport packet time slot. The processor selectively replaces one or more of the null transport packets with another to-be-remultiplexed data bearing transport packet. Such replacement data bearing transport packets may contain PSI data or even bursty transactional data, which bursty transactional data has no bit rate or transmission latency requirement for presenting information in a continuous fashion.

Illustratively, the processor extracts selected ones of the transport packets of the received TS and discards each non-selected transport packet including each null transport packet. The selected transport packets are stored in the memory by the processor and first adaptor. As described above, the processor schedules each of the stored transport packets for output in an outputted transport stream at a time that depends on a time at which each of the stored transport packets are received. A second interface (adaptor) outputs each of the stored transport packets in a time slot that corresponds to the schedule. If no transport packet is scheduled for output at one of the time slots of the outputted TS, the second

adaptor outputs a null transport packet. Nevertheless, null transport packets occupy less bandwidth in the outputted TS than in each inputted TS.

According to an additional embodiment, a method is provided for timely outputting compressed program data bearing bit streams on an asynchronous communication link. A synchronous interface (adaptor) provides a bit stream containing transport packets. The processor assigns dispatch times to each of one or more selected ones of the transport packets to maintain a predetermined bit rate of a program for which each selected transport packet carries data and to incur an average latency for each selected transport packet. At times that depend on each of the dispatch times, the asynchronous communication interface receives one or more commands and responds thereto by transmitting the corresponding selected transport packets at approximately the dispatch times so as to minimize a jitter of selected transport packets.

Illustratively, the commands are generated as follows. The processor enqueues transmit descriptors containing the above dispatch times, into a transmit queue. The processor assigns an adaptor of the remultiplexer node to servicing the transmit queue on behalf of the asynchronous interface. The data link control circuit of the assigned adaptor causes each command to issue when the dispatch times of the descriptors equal the time of the reference clock at the adaptor.

Various ones of these techniques may be used to enable network distributed remultiplexing. A network is provided with one or more communication links, and a plurality of nodes, interconnected by the communication links into a communications network. A destination node receives a first bit stream containing data of one or more programs via one of the communications links, the first bit stream having one or more

predetermined bit rates for portions thereof. The destination node can be a remultiplexer node as described above and in any event includes a processor. The processor chooses at least part of the received first bit stream for transmission, and schedules transmission of the chosen part of the first bit stream so as to output the chosen part of the first bit stream in a TS at a rate depending on a predetermined rate of the chosen part of said first bit stream.

In the alternative, the communication links collectively form a shared communications medium. The nodes are divided into a first set of one or more nodes for transmitting one or more bit streams onto the shared communications medium, and a second set of one or more nodes for receiving the transmitted bit streams from the shared communications medium. The nodes of the second set select portions of the transmitted bit streams and transmit one or more remultiplexed TSs as a bit stream containing the selected portions. Each of the transmitted remultiplexed TSs are different than the received ones of the transmitted bit streams. A controller node is provided for selecting the first and second sets of nodes and for causing the selected nodes to communicate the bit streams via the shared communication medium according to one of plural different signal flow patterns, including at least one signal flow pattern that is different from a topological connection of the nodes to the shared communication medium.

Finally, a method is provided for synchronizing the reference clock at each of multiple circuits that receive or transmit transport packets in a remultiplexing system. The reference clock at each circuit that receives transport packets is for indicating a time at which each transport packet is received thereat. The reference clock at each circuit that transmits transport packets is for indicating when to transmit each transport packet therefrom. A master reference clock, to which each other one of the reference clocks is to

be synchronized, is designated. The current time of the master reference clock is periodically obtained. Each other reference clock is adjusted according to a difference between the respective time at the other reference clocks and the current time of the master reference clock so as to match a time of the respective reference clock to a corresponding time of the master reference clock.

Thus, according to the invention, a more flexible remultiplexing system is provided. The increased flexibility enhances multiplexing yet decreases overall system cost.

Brief Description of the Drawing

FIG 1 shows a remultiplexing environment according to another embodiment of the present invention.

FIG 2 shows a remultiplexer node using an asynchronous platform according to an embodiment of the present invention.

FIG 3 shows a flow chart which schematically illustrates how transport packets are processed depending on their PIDs in a remultiplexing node according to an embodiment of the present invention.

FIG 4 shows a network distributed remultiplexer according to an embodiment of the present invention.

Detailed Description of the Invention

For sake of clarity, the description of the invention is divided into sections.

Remultiplexer Environment and Overview

FIG 1 shows a basic remultiplexing environment 10 according to an embodiment of the present invention. A controller 20 provides instructions to a remultiplexer 30 using, for example, any remote procedure call (RPC) protocol. Examples of RPCs that can be used include the digital distributed computing environment protocol (DCE) and the open network computing protocol (ONC). DCE and ONC are network protocols employing protocol stacks that allow a client process to execute a subroutine either locally on the same platform (e.g., controller 20) or on a remote, different platform (e.g., in remultiplexer 30). In other words, the client process can issue control instructions by simple subroutine calls. The DCE or ONC processes issue the appropriate signals and commands to the remultiplexer 30 for effecting the desired control.

The controller 20 may be in the form of a computer, such as a PC compatible computer. The controller 20 includes a processor 21, such as one or more Intel™ Pentium II™ integrated circuits, a main memory 23, a disk memory 25, a monitor and keyboard/mouse 27 and one or more I/O devices 29 connected to a bus 24. The I/O device 29 is any suitable I/O device 29 for communicating with the remultiplexer 30, depending on how the remultiplexer 30 is implemented. Examples of such an I/O device 29 include an RS-422 interface, an Ethernet interface, a modem, and a USB interface.

The remultiplexer 30 is implemented with one or more networked "black boxes". In the example remultiplexer architecture described below, the remultiplexer 30 black boxes may be stand alone PC compatible computers that are interconnected by communications links such as Ethernet, ATM or DS3 communications links. For example, remultiplexer 30 includes one or more black boxes which each are stand alone PC

compatible computers interconnected by an Ethernet network (10 BASE-T, 100 BASE-T or 1000 BASE-T, etc.).

As shown, one or more to-be-remultiplexed TSs, namely, TS1, TS2 and TS3, are received at the remultiplexer 30. As a result of the remultiplexing operation of the remultiplexer 30, one or more TSs, namely, TS4 and TS5, are outputted from the remultiplexer 30. The remultiplexed TSs TS4 and TS5 illustratively, include at least some information (at least one transport packet) from the inputted TSs TS1, TS2 and TS3. At least one storage device 40, e.g., a disk memory or server, is also provided. The storage device 40 can produce TSs or data as inputted, to-be-remultiplexed information for remultiplexing into the outputted TSs TS4 or TS5 by the remultiplexer 30. Likewise, the storage device 40 can store TSs information or data produced by the remultiplexer 30, such as transport packets extracted or copied from the inputted TSs TS1, TS2 or TS3, other information received at the remultiplexer 30 or information generated by the remultiplexer 30.

Also shown are one or more data injection sources 50 and one or more data extraction destinations 60. These sources 50 and destinations 60 may themselves be implemented as PC compatible computers. However, the sources 50 may also be devices such as cameras, video tape players, communication demodulators/receivers and the destinations may be display monitors, video tape recorders, communications modulators/transmitters, etc. The data injection sources 50 supply TS, ES or other data to the remultiplexer 30, e.g., for remultiplexing into the outputted TSs TS4 and/or TS5. Likewise, the data extraction destinations 60 receive TS, ES or other data from the remultiplexer 30, e.g., that is extracted from the inputted TSs TS1, TS2 and/or TS3. For

example, one data injection source 50 may be provided for producing each of the inputted, to-be-remultiplexed TSs, TS1, TS2 and TS3 and one data extraction destination 60 may be provided for receiving each outputted remultiplexed TS TS4 and TS5.

The environment 10 may be viewed as a network. In such a case, the controller 20, each data injection source 50, storage device 40, data extraction destination 60 and each "networked black box" of the remultiplexer 30 in the environment 10 may be viewed as a node of the communications network. Each node may be connected by a synchronous or asynchronous communication link. In addition, the separation of the devices 20, 40, 50 and 60 from the remultiplexer 30 is merely for sake of convenience. In an alternative embodiment, the devices 20, 40, 50 and 60 are part of the remultiplexer 30.

Remultiplexer Architecture

FIG 2 shows a basic architecture for one of the network black boxes or nodes 100 of the remultiplexer 30, referred to herein as a "remultiplexer node" 100. The particular remultiplexer node 100 shown in FIG 2 can serve as the entire remultiplexer 30. Alternatively, as will be appreciated from the discussion below, different portions of the remultiplexer node 100 can be distributed in separate nodes that are interconnected to each other by synchronous or asynchronous communication links. In yet another embodiment, multiple remultiplexer nodes 100, having the same architecture as shown in FIG 2, are interconnected to each other via synchronous or asynchronous communication links and can be programmed to act in concert. These latter two embodiments are referred to herein as network distributed remultiplexers.

Illustratively, the remultiplexer node 100 is a Windows NT™ compatible PC computer platform. The remultiplexer node 100 includes one or more adaptors 110. Each adaptor 110 is connected to a bus 130, which illustratively is a PCI compatible bus. A host memory 120 is also connected to the bus 130. A processor 160, such as an Intel™ Pentium II™ integrated circuit is also connected to the bus 130. It should be noted that the single bus architecture shown in FIG 2 may be a simplified representation of a more complex multiple bus structure. Furthermore, more than one processor 160 may be present which cooperate in performing the processing functions described below.

Illustratively, two interfaces 140 and 150 are provided. These interfaces 140 and 150 are connected to the bus 130, although they may in fact be directly connected to an I/O expansion bus (not shown) which in turn is connected to the bus 130 via an I/O bridge (not shown). The interface 140 illustratively is an asynchronous interface, such as an Ethernet interface. This means that data transmitted via the interface 140 is not guaranteed to occur at precisely any time and may experience a variable end-to-end delay. On the other hand, the interface 150 is a synchronous interface, such as a T1 interface. Communication on the communication link connected to the interface 150 is synchronized to a clock signal maintained at the interface 150. Data is transmitted via the interface 150 at a particular time and experiences a constant end-to-end delay.

FIG 2 also shows that the remultiplexer node 100 can have an optional scrambler/descrambler (which may be implemented as an encryptor/decryptor) 170 and/or a global positioning satellite (GPS) receiver 180. The scrambler/descrambler 170 is for scrambling or descrambling data in transport packets. The GPS receiver 180 is for

receiving a uniform clock signal for purposes of synchronizing the remultiplexer node 100. The purpose and operation of these devices is described in greater detail below.

Each adaptor 110 is a specialized type of synchronous interface. Each adaptor 110 has one or more data link control circuits 112, a reference clock generator 113, one or more descriptor and transport packet caches 114, an optional scrambler/descrambler 115 and one or more DMA control circuits 116. These circuits may be part of one or more processors. Preferably, they are implemented using finite state automata, i.e., as in one or more ASICs or gate arrays (PGAs, FPGAs, etc.). The purpose of each of these circuits is described below.

The reference clock generator 113 illustratively is a 32 bit roll-over counter that counts at 27 MHZ. The system time produced by the reference clock generator 113 can be received at the data link control circuit 112. Furthermore, the processor 160 can directly access the reference clock generator 113 as follows. The processor 160 can read the current system time from an I/O register of the reference clock generator 113. The processor 160 can load a particular value into this same I/O register of the reference clock generator 113. Finally, the processor 160 can set the count frequency of the reference clock generator in an adjustment register so that the reference clock generator 113 counts at a frequency within a particular range.

The purpose of the cache 114 is to temporarily store the next one or more to-be-outputted transport packets pending output from the adaptor 110 or the last one or more transport packets recently received at the adaptor 110. The use of the cache 114 enables transport packets to be received and stored or to be retrieved and outputted with minimal latency (most notably without incurring transfer latency across the bus 130). The cache 114

also stores descriptor data for each transport packet. The purpose and structure of such descriptors is described in greater detail below. In addition, the cache 114 stores a filter map that can be downloaded and modified by the processor 160 in normal operation. Illustratively, the cache 114 may also store control word information for use in scrambling or descrambling, as described in greater detail below. In addition to the processor 160, the cache 114 is accessed by the data link control circuit 112, the DMA control circuit 116 and the optional scrambler/descrambler 115.

As is well known, the cache memory 114 may posses a facsimile or modified copy of data in the host memory 120. Likewise, when needed, the cache 114 should obtain the modified copy of any data in the host memory and not a stale copy in its possession. The same is true for the host memory 120. An "ownership protocol" is employed whereby only a single device, such as the cache memory 114 or host memory 120, has permission to modify the contents of a data storage location at any one time. Herein, the cache memory 114 is said to obtain control of a data storage location when the cache memory has exclusive control to modify the contents of such storage locations. Typically, the cache memory 114 obtains control of the storage location and a facsimile copy of the data stored therein, modifies its copy but defers writing the modifications of the data to the host memory until a later time. By implication, when the cache memory writes data to a storage location in the host memory, the cache memory 114 relinquishes control to the host memory 120.

The DMA control circuit 116 is for transferring transport packet data and descriptor data between the host memory 120 and the cache 114. The DMA control circuit 116 can maintain a sufficient number of transport packets (and descriptors therefor) in the cache 114

to enable the data link control circuit 112 to output transport packets in the output TS continuously (i.e., in successive time slots). The DMA control circuit 116 can also obtain control of a sufficient number of descriptor storage locations, and the packet storage locations to which they point, in the cache 114. The DMA control circuit 116 obtains control of such descriptor and transport packet storage locations for the cache 114. This enables continuous allocation of descriptors and transport packet storage locations to incoming transport packets as they are received (i.e., from successive time slots).

The data link control circuit 112 is for receiving transport packets from an incoming TS or for transmitting transport packets on an outgoing TS. When receiving transport packets, the data link control circuit 112 filters out and retains only selected transport packets received from the incoming TS as specified in a downloadable filter map (provided by the processor 160). The data link control circuit 112 discards each other transport packet. The data link control circuit 112 allocates the next unused descriptor to the received transport packet and stores the received transport packet in the cache 114 for transfer to the transport packet storage location to which the allocated descriptor points. The data link control circuit 112 furthermore obtains the reference time from the reference clock generator 113 corresponding to the receipt time of the transport packet. The data link control circuit 112 records this time as the receipt time stamp in the descriptor that points to the transport packet storage location in which the transport packet is stored.

When transmitting packets, the data link control circuit 112 retrieves descriptors for outgoing transport packets from the cache 114 and transmits the corresponding transport packets in time slots of the outgoing TS that occur when the time of the reference clock generator 113 approximately equals the dispatch times indicated in the respective

descriptors. The data link control circuit 112 furthermore performs any final PCR correction in outputted transport packets as necessary so that the PCR indicated in the transport packets is synchronized with the precise alignment of the transport packet in the outgoing TS.

The processor 160 is for receiving control instructions from the external controller 20 (FIG 1) and for transmitting commands to the adaptor 110, and the interfaces, 140 and 150 for purposes of controlling them. In response, to such instructions, the processor 160 generates a PID filter map and downloads it to the cache 114, or modifies the PID filter map already resident in the cache 114, for use by the data link control circuit 112 in selectively extracting desired transport packets. In addition, the processor 160 generates interrupt receive handlers for processing each received transport packet based on its PID. Receipt interrupt handlers may cause the processor 160 to remap the PID of a transport packet, estimate the departure time of a transport packet, extract the information in a transport packet for further processing, etc. In addition, the processor 160 formulates and executes transmit interrupt handlers which cause the processor to properly sequence transport packets for output, to generate dispatch times for each transport packet, to coarsely correct PCRs in transport packets and to insert PSI into an outputted TS. The processor 160 may also assist in scrambling and descrambling as described in greater detail below.

The host memory 120 is for storing transport packets and descriptors associated therewith. The host memory 120 storage locations are organized as follows. A buffer 122 is provided containing multiple reusable transport packet storage locations for use as a transport packet pool. Descriptor storage locations 129 are organized into multiple rings

21

124. Each ring 124 is a sequence of descriptor storage locations 129 from a starting memory address or top of ring 124-1 to an ending memory address or bottom of ring 124-2. One ring 124 is provided for each outgoing TS transmitted from the remultiplexer node 100 and one ring 124 is provided for each incoming TS received at the remultiplexer node 100. Other rings 124 may be provided as described in greater detail below.

A queue is implemented in each ring 124 by designating a pointer 124-3 to a head of the queue or first used/allocated descriptor storage location 129 in the queue and a pointer 124-4 to a tail of the queue or last used/allocated descriptor storage location 129 in the queue. Descriptor storage locations 129 are allocated for incoming transport packets starting with the unused/non-allocated descriptor storage location 129 immediately following the tail 124-4. Descriptor storage locations 129 for outgoing transport packets are retrieved from the queue starting from the descriptor storage location 129 pointed to by the head 124-3 and proceeding in sequence to the tail 124-4. Whenever the descriptor of the descriptor storage location 129 at the end of the ring 124-2 is reached, allocation or retrieval of descriptors from descriptor storage locations 129 continues with the descriptor of the descriptor storage location 129 at the top of the ring 124-1.

As shown, each descriptor stored in each descriptor storage location 129 includes a number of fields 129-1, 129-2, 129-3, 129-4, 129-5, 129-6, 129-7, 129-8, 129-9 and 129-10. Briefly stated, the purpose of each of these fields is as follows. The field 129-1 is for storing command attributes. The processor 160 can use individual bits of the command attribute field to control the transport packet transmission and descriptor data retrieval of the adaptor 110. For instance, the processor 160 can preset a bit in the field 129-1 of a descriptor in the descriptor storage location 129 pointed to by the bottom 124-2 of the ring

124 to indicate that the descriptor storage location 129 pointed to by the top pointer 124-1 follows the descriptor storage location 129 pointed to by the ~~bottom~~ pointer 124-2.

The field 129-2 is for storing software status bits. These bits are neither accessed nor modified by the adaptor 110 and can be used by the processor 160 for any purposes not involving the adaptor 110.

The field 129-3 is for storing the number of bytes of a to-be-outputted, outgoing transport packet (typically 188 bytes for MPEG-2 transport packets but can be set to a larger or smaller number when the descriptor points to packets according to a different transport protocol or for "gather" and "scatter" support, where packets are fragmented into multiple storage locations or assembled from fragments stored in multiple packet storage locations).

The field 129-4 is for storing a pointer to the transport packet storage location to which the descriptor corresponds. This is illustrated in FIG 2 by use of arrows from the descriptors in descriptor storage locations 129 in the ring 124 to specific storage locations of the transport packet pool 122.

The field 129-5 is for storing the receipt time for an incoming received transport packet or for storing the dispatch time of an outgoing to-be-transmitted transport packet.

The field 129-6 is for storing various exceptions/errors which may have occurred. The bits of this field may be used to indicate a bus 130 error, a data link error on the communication link to which the data link control circuit 112 is connected, receipt of a short or long packet (having less than or more than 188 bytes), etc.

The field 129-7 is for storing status bits that indicate different status aspects of a descriptor such as whether or not the descriptor is valid, invalid pointing to an errored

packet, etc. For example, suppose that multiple devices must process the descriptor and/or packet to which it points in succession. In such a case, four status bits are preferably provided. The first two of these bits can be set to the values 0,1,2 or 3. The value 0 indicates that the descriptor is invalid. The value 1 indicates that the descriptor is valid and may be processed by the last device that must process the descriptor and/or packet to which it points. The value 2 indicates that the descriptor is valid and may be processed by the second to last device that must process the descriptor and/or packet to which it points. The value 3 indicates that the descriptor is valid and may be processed by the third to last device that must process the descriptor and/or packet to which it points. The latter two bits indicate whether or not the descriptor has been fetched from the host memory 120 to the cache 114 and whether or not the descriptor has completed processing at the adaptor 110 and may be stored in the host memory 120. Other status bits may be provided as described in greater detail below.

The field 129-8 contains a transfer count indicating the number of bytes in a received incoming transport packet.

The field 129-9 is for storing a scrambling/descrambling control word or other information for use in scrambling or descrambling. For example, the processor 160 can store a control word (encryption/decryption key) or base address to a table of control words stored in the cache 114 in this field 129-9.

Field 129-10 is for storing a scheduled estimated departure time, actual departure time or actual receipt time. As described in greater detail below, this field is used by the processor 160 for ordering received incoming transport packets for output or for noting the receipt time of incoming transport packets.

Illustratively, one data link control circuit 112, one DMA control circuit 116 and one ring 124 is needed for receiving transport packets at a single input port, and one data link control circuit 112, one DMA control circuit 116 and one ring 124 is needed for transmitting transport packets from a single output port. Descriptors stored in queues associated with input ports are referred to herein as receipt descriptors and descriptors stored in queues associated with output ports are referred to herein as transmit descriptors. As noted below, the input and output ports referred to above may be the input or output port of the communication link to which the data link control circuit 112 is connected or the input or output port of the communication link of another interface 140 or 150 in the remultiplexer node 100. The adaptor 110 is shown as having only a single data link control circuit 112 and a single DMA control circuit 116. This is merely for sake of illustration--multiple data link control circuits 112 and DMA control circuits 116 can be provided on the same adaptor 110. Alternatively, or additionally, multiple adaptors 110 are provided in the remultiplexer node 100.

Basic Transport Packet Receipt, Remultiplexing and Transmission

Consider now the basic operation of the remultiplexer node 100. The operator is provided with a number of choices in how to operate the remultiplexer node 100. In a first manner of operating the remultiplexer node 100, assume that the operator wishes to selectively combine program information of two TSs, namely, TS1 and TS2, into a third TS, namely, TS3. In this scenario, assume that the operator does not initially know what programs, ESs or PIDs are contained in the two to-be-remultiplexed TSs TS1 and TS2. In addition, TS1 illustratively is received at a first adaptor 110, TS2 illustratively is received

2 b

at a second adaptor 110 and TS3 illustratively is transmitted from a third adaptor 110 of the same remultiplexer node 100. As will be appreciated from the description below, each of TS1 and TS2 may instead be received via synchronous or asynchronous interfaces at the same node or at different nodes, and selected portions of TS1 and TS2 may be communicated to a third node via a network of arbitrary configuration for selective combination to form TS3 at the third node.

The operation according to this manner may be summarized as (1) acquiring the content information (program, ES, PAT, PMT, CAT, NIT, etc., and PIDs thereof) of the inputted, to-be-remultiplexed TSs TS1 and TS2; (2) reporting the content information to the operator so that the operator can formulate a user specification; and (3) receiving a user specification for constructing the outputted remultiplexed TS TS3 and dynamically constructing the remultiplexed TS TS3 from the content of the inputted to-be-remultiplexed TSs TS1 and TS2 according to the user specification.

To enable acquisition of the content information, the transport processor 160 allocates one receipt queue to each of the first and second adaptors 110 that receive the TSs TS1 and TS2, respectively. To acquire the content of the TSs TS1 and TS2, no transport packets are discarded at the adaptors 110 for TS1 or TS2 initially. Thus, the processor 160 loads a filter map into the caches 114 of each of the first and second adaptors 110 receiving the TSs TS1 and TS2 causing each transport packet to be retained and transferred to the host memory 120. As each transport packet of a TS (e.g., the TS1) is received at its respective adaptor 110, the data link control circuit 112 allocates the next unused descriptor (following the descriptor stored in the descriptor storage location at the tail 124-4 of the receipt queue), to the received, incoming transport packet. The data link control circuit 112

stores each received transport packet in a transport packet storage location of the cache 114 to which the allocated descriptor points.

The DMA control circuit 116 writes each transport packet to its corresponding storage location of the pool 122 in the host memory 120 and writes descriptor data of the descriptors allocated to the transport packets to their respective descriptor storage locations of the receipt queue. The DMA control circuit 116 may furthermore obtain control of the next few non-allocated descriptor storage locations 129 of the receipt queue (following the storage locations of the sequence of descriptors 129 for which the DMA control circuit 116 had obtained control previously), copies of the descriptors stored therein and the transport packet storage locations to which the descriptors point. Control of such unused, non allocated descriptors and transport packet storage locations is provided to the cache 114 for used by the data link control circuit 112 (i.e., allocation to future transport packets received from TS1).

After the DMA control circuit 116 writes $i \geq 1$ transport packets and data of descriptors allocated thereto to the pool 122 and the receipt queue, the DMA control circuit 116 generates an interrupt. Illustratively, the number i may be selected by the operator using controller 20 and set by the processor 160. The interrupt causes the processor 160 to execute an appropriate receipt "PID" handler subroutine for each received transport packet. Alternatively, another technique such as polling or a timer based process can be used to initiate the processor 160 to execute a receipt PID handler subroutine for each received transport packet. For sake of clarity, an interrupt paradigm is used to illustrate the invention herein. Referring to FIG 3, the processor 160 illustratively has a set of PID handler subroutines for each adaptor 110 (or other device) that receives or transmits a TS

during a remultiplexing session. FIG 3 illustrates two types of PID handler subroutine sets, namely, a receipt PID handler subroutine set and a transmit PID handler subroutine set. Each DMA control circuit 116 generates a recognizably different interrupt thereby enabling the processor 160 to determine which set of PID handler subroutines to use. In response to the interrupt by the DMA control circuit 116, the processor 160 executes step S2 according to which the processor 160 examines the PID of each transport packet pointed to by a recently stored descriptor in the receipt queue of the interrupting adaptor 110. For each PID, the processor 160 consults a table of pointers to receipt PID handler subroutines 402 specific to the adaptor 110 (or other device) that interrupted the processor 160.

Assume that the first adaptor 110 receiving TS1 interrupts the processor 160, in which case the processor 160 determines to consult a table of pointers to receipt PID handler subroutines 402 specific to the adaptor 110 that received the TS TS1. The table of pointers to receipt PID handler subroutines includes 8192 entries, including one entry indexed by each permissible PID (which PIDs have 13 bits according to MPEG-2). Each indexed entry contains a pointer to, or address of, RIV0, RIV1,...,RIV8191, a subroutine to be executed by the processor 160. Using the PID of each transport packet, the processor 160 indexes the entry of the table of pointers to receipt PID handler subroutines 402 in order to identify the pointer to the subroutine to be executed for that particular transport packet.

Each subroutine pointed to by the respective pointer, and executed by the processor 160, is specifically mapped to each PID by virtue of the pointer table 402 to achieve the user's specification. Each subroutine is advantageously predefined and simply mapped by the pointer table 402 according to the user specification. Each subroutine is composed of

a collection of one or more basic building block processes. Some examples of basic building block processes include:

(1) PAT acquisition: Initially, this process is included in the subroutine pointed to by RIV0, the receive PID handler subroutine for PID 0x0000. In executing this process, the processor 160 illustratively extracts the section of the PAT carried in the currently processed transport packet and loads the PAT section into the PAT maintained in memory. Note that multiple versions of the PAT may be used as the programs carried in the TS can change from time to time. The processor 160 is capable of identifying different versions of the PAT and separately aggregating and maintaining a copy of each version of the PAT in the host memory 120. The processor 160 is also capable of identifying which version of the PAT is currently in use at any time based on information contained in various sections of the PAT. The processor 160 also uses information carried in each updated PAT section to identify program numbers of programs carried in the TS at that moment and the PIDs of PMT sections or program definitions for such program numbers. Using such program numbers, the processor 160 can modify the pointer table 402 for the receipt PID handler subroutine to insert pointer for appropriate PIDs (labeling transport packets bearing PMT sections) for executing a subroutine containing a process for acquiring PMT sections/program definitions.

(2) PMT section/program definition acquisition: In this process, the processor 160 extracts the PMT section or program definition contained in the currently processed transport packet and updates the respective portion of the PMT with the extracted program definition or PMT section data. Like the PAT, multiple versions of the PMT may be utilized and the processor 160 can determine in which PMT to store the extracted PMT

section or program definition data. The processor 160 may use PMT information to update a PID filter map used to discard transport packets of programs not to be included in the remultiplexed TS, to identify control words for descrambling ESs and to select subroutines for processing PCRs contained in transport packets having PIDs as identified in the PMT.

(3) PID remapping: This causes the processor 160 to overwrite the PID of the corresponding packet with a different PID. This is desirable to ensure uniqueness of PID assignment. That is, MPEG-2 requires that transport packets carrying different contents, e.g., data of different ESs, data of different PSI streams, etc., be labeled with mutually different PIDs, if such different content carrying transport packets are to be multiplexed into, and carried in, the same outputted remultiplexed TS. Otherwise, a decoder or other device would not be able to distinguish transport packets carrying different kinds of data for extraction, decoding, etc. It is possible that a certain PID is used in TS1 to label transport packets bearing a first type of data and the same PID is used in TS2 to label transport packets bearing a second type of data. If the transport packets of the first and second types are to be included in the outputted remultiplexed TS TS3, then at least one of the two types of transport packets should be re-labeled with a new PID to ensure uniqueness.

(4) Transport packet discarding: As the name suggests, the processor 160 simply discards the transport packet. To this end, the processor 160 deallocates the descriptor pointing to the discarded transport packet. Descriptor deallocation can be achieved by the processor 160 adjusting the sequence of descriptors resident in the descriptor storage locations 129 of the queue to remove the descriptor for the deleted transport packet (e.g., the processor identifies all of the allocated descriptors that follow the descriptor of the to-

be-deleted transport packet in the ring 124 and moves each to the descriptor storage space of the immediately preceding descriptor). The deallocation of the descriptor creates a descriptor storage space 129 in the receipt queue for reallocation.

(5) PCR flag setting: The PMT indicates, for each program, the PIDs of the transport packets that carry the PCRs. However, only some of such transport packets carry PCRs. This can be easily determined by the processor 160 determining if the appropriate indicators in the transport packet are set (the adaption_field_control bits in the transport packet header and PCR_flag bit in the adaption field). If the processor 160 determines that a PCR is present, the processor 160 sets a PCR flag bit in the attribute field 129-1 of the descriptor 129 associated with the respective packet. The purpose of this attribute flag bit is described in greater detail below.

In addition, the processor 160 illustratively calculates the current drift of the reference clock generators 113 relative to the encoder system time clock of the program of which the PCR is a sample. Drift may be determined by the following formula:

$$\text{drift} = \Delta RTS12 - \Delta PCR12;$$

$$\Delta RTS12 = RTS2 - RTS1; \text{ and}$$

$$\Delta PCR12 = PCR1 - PCR2$$

where: $\Delta PCR12$ is a difference in successive PCRs for this program,

PCR2 is the PCR in the currently processed transport packet,

PCR1 is the previously received PCR for this program,

$\Delta RTS12$ is a difference in successive receipt time stamps,

RTS2 is the receipt time stamp recorded for the currently processed transport packet containing PCR2, and

RTS1 is a previous receipt time stamp for the transport packet containing PCR1. After calculating the drift, PCR1 and RTS1 are set equal to PCR2 and RTS2, respectively. The drift is used for adjusting the PCR (if necessary) as described below.

(6) Estimated departure time calculation: According to this process, the processor 160 estimates the (ideal) departure time of the transport packet. Illustratively, this process is included in the receive interrupt handler for each received incoming transport packet to be remultiplexed into an outgoing TS. The estimated departure time can be estimated from the receipt time of the transport packet (in the field 129-5) and the known internal buffering delay at the remultiplexing node 100. The processor 160 writes the expected departure time in the field 129-10.

(7) Scrambling/descrambling control word information insertion: Typically, in either a scrambling or descrambling technique, a dynamically varying control word, such as an encryption or decryption key, is needed to actually scramble or descramble data in the transport packet. Common scrambling and descrambling techniques use odd and even keys, according to which, one key is used for decrypting ES data and the next key to be used subsequently is transferred contemporaneously in the TS. A signal is then transmitted indicating that the most recently transferred key should now be used. Scrambling/descrambling control words can be ES specific or used for a group of ESs (over an entire "conditional access system"). Descrambling or scrambling control words may be maintained in a PID index-able table at the remultiplexer node 100. As described in greater detail below, the processor 160 in executing this process may insert the base address for the control word table, or the control word itself, into the field 129-9 of a descriptor.

Initially, the processor 160 selects a PID handler for acquiring the PAT of each received TS TS1 and TS2 and thereafter discarding each processed transport packet. In the course of receiving the PAT, PIDs of other PSI bearing transport packets, such as program definitions/PMT sections, the NIT, and the CAT, and PIDs of other streams such as ES streams, ECM streams, EMM streams, etc. are obtained. The receipt PID handler subroutine for the PID of the PAT illustratively selects receipt PID handler subroutines for acquiring the PMT, NIT, CAT, etc. This can be achieved easily by having such subroutines available and simply changing the pointers of the entries (indexed by appropriate identified PIDs) in the table 402 to point to such PID handler subroutines. Note that such a simple PID handler subroutine selection process can be dynamically effected even while transport packets are received and processed for TS1 and TS2. The advantages of this are described in greater detail below.

Eventually, a sufficient amount of PSI regarding each TS TS1 and TS2 is acquired to enable the operator to create a user specification of the information to be outputted in the remultiplexed TS TS3. The processor 160 illustratively transmits to the controller 20 the acquired PSI information, e.g., using the asynchronous interface 140. Sufficient information for selecting a user specification is transmitted to the controller 20. This information may be selective, e.g., just a channel map of each TS showing the program numbers contained therein and the different kinds of ESs (described with descriptive service designations such as video, audio1, second audio presentation, closed caption text, etc.) Alternatively, the information may be exhaustive e.g., including the PIDs of each program, ECMs of ESs thereof, etc., and the controller 20 simply displays the information to the operator in a coherent and useful fashion.

Using the information provided, the operator generates a user specification for the outputted to-be-remultiplexed TS TS3. This user specification may specify:

(1) The program numbers in each TS TS1 and TS2 to be retained and outputted in the remultiplexed TS, TS3,

(2) ESs of retained programs to be retained or discarded,

(3) ESs, groups of ESs, programs or groups of programs to be descrambled and/or scrambled, and the source of the control words to be used in scrambling each ES, group of ESs, program or groups of programs,

(4) Any new ECMs or EMMs to be injected or included in the outputted remultiplexed TS TS3, and

(5) Any new PSI information not automatically implicated from the above selections such as an NIT or CAT to be placed in the outputted TS TS3, specific PIDs that are to be remapped and the new PIDs to which they should be remapped, PIDs assigned to other information (e.g., bursty data, as described below) generated at the remultiplexer node and carried in the TS TS3, etc.

The user specification is then transmitted from the controller 20 to the remultiplexer node 100, e.g., via the asynchronous interface 140.

The processor 160 receives the user specification and responds by selecting the appropriate receive PID handler subroutines for appropriate PIDs of each received, to-be-remultiplexed TS, TS1 and TS2. For example, for each PID labeling a transport packet containing data that is to be retained, the processor 160 selects a subroutine in which the processor inserts the process for estimating the departure time. For each PID labeling a

transport packet containing scrambled data, the processor 160 selects a subroutine containing a process for selecting the appropriate control ~~word and~~ inserting it into the descriptor associated with such a transport packet. For each PID labeling a transport packet containing a PCR, the processor 160 can select a subroutine containing the process for setting the PCR flag and for calculating the drift, and so on. The dynamic adjustment of user specification and/or PSI data is described in greater detail below.

The processor 160 allocates a transmit queue to each device that transmits a remultiplexed TS, i.e., the third adaptor 110 that outputs the TS TS3. The processor 160 furthermore loads the PID filter maps in each cache 114 of the first and second adaptors 110 that receive the TSs TS1 and TS2 with the appropriate values for retaining those transport packets to be outputted in remultiplexed TS TS3, for retaining other transport packets containing PSI, for keeping track of the contents of TS1, and TS2 and for discarding each other transport packet.

In addition to selecting receive PID handler subroutines, allocating transmit queues and loading the appropriate PID filter map modifications, the processor 160 illustratively selects a set of transmit PID handler subroutines for each adaptor (or other device) that outputs a remultiplexed TS. This is shown in FIG 3. The transmit PID handler subroutines are selected on a PID and transmit TS basis. As above, in response to receiving an identifiable interrupt (e.g., from a data link control circuit 112 of an adaptor 110 that transmits an outputted TS, such as TS3) the processor 160 executes step S4. In step S4, the processor 160 examines descriptors from the receipt queues (and/or possibly other queues containing descriptors of transport packets not yet scheduled for output) and identifies up to $j \geq 1$ descriptors pointing to transport packets to be outputted from the interrupting adaptor

110. The number j may illustratively be programmable and advantageously is set equal to the number k of transport packets transmitted from a specific adaptor 110 from which an output TS is transmitted between each time the specific adaptor 110 interrupts the processor 160.

In executing step S4, the processor 160 examines each receive queue for descriptors pointing to transport packets that are destined to the specific output TS. The processor 160 determines which transport packets are destined to the output TS by consulting a table of pointers to transmit PID handler subroutines 404. As with the table 402, the table 404 includes one entry for, and indexed by, each PID 0x0000 to 0x1FFF. Each indexed entry contains a pointer to, or address of, TIV0, TIV1,..., TIV8191, a subroutine to be executed in response to a respective PID. The table of pointers to transmit PID handler subroutines 404 is formulated by the processor 160 according to the user specification received from the controller 20, and modified as described below.

The following are illustrative processes that can be combined into a transmit PID handler subroutine:

(1) Nothing: If the current transport packet is not to be outputted in the remultiplexed TS (or other stream) of the device that issued the transmit interrupt to the processor 160, the PID of such a transport packet maps to a subroutine containing only this process. According to this process, the processor 160 simply skips the transport packet and descriptor therefor. The examined descriptor is not counted as one of the j transport packets to be outputted from the specific adaptor 110 that interrupted the processor 160.

(2) Order descriptor for transmission: If the current transport packet is to be outputted in the remultiplexed TS (or other stream) of the device that issued the transmit

interrupt to the processor, the PID of such a transport packet maps to a subroutine containing this process (as well as possibly others). According to this process, the processor 160 allocates a transmit descriptor for this transport packet. The processor 160 then copies pertinent information in the receipt descriptor that points to the transport packet to the newly allocated transmit descriptor. The allocated transmit descriptor is then ordered in the proper sequence within a transmit queue, associated with the device that requested the interrupt, for transmission. In particular, the processor 160 compares the estimated departure time of the packet, to which the newly allocated descriptor points, to the actual dispatch time (the actual time that the transport packet will be transmitted) recorded in the other descriptors in the transmit queue. If possible, the descriptor is placed in the transmit queue before each descriptor with a later actual dispatch time than the estimated departure time of the descriptor and after each descriptor with an earlier actual dispatch time than the estimated departure time of the descriptor. Such an insertion can be achieved by copying each transmit descriptor, of the sequence of transmit descriptors with later actual dispatch times than the estimated dispatch time of the to-be-inserted descriptor, to the respective sequentially next descriptor storage location 129 of the queue. The data of the allocated transmit descriptor can then be stored in the descriptor storage location 129 made available by copying the sequence.

(3) Actual dispatch time determination: The processor 160 can determine the actual dispatch time of the transport packet to which the allocated descriptor points based on the estimated departure time of the transport packet. The actual dispatch time is set by determining in which transport packet time slot of the outputted remultiplexed TS T3 to transmit the transport packet (to which the newly allocated and inserted transmit descriptor

points). That is, the transport packet time slot of the outputted TS T3 nearest in time to the estimated departure time is selected. The transport packet is presumed to be outputted at the time of the selected transport packet time slot, relative to the internal reference time as established by the reference clock generator(s) 113 of the adaptor(s) 110 (which are mutually synchronized as described below). The time associated with the respective transport packet slot time is assigned as the actual dispatch time. The actual dispatch time is then stored in field 129-5 of the transmit descriptor. As described below, the actual dispatch time is really an approximate time at which the data link control circuit 112 of the third adaptor 110 (which outputs the remultiplexed TS TS3) submits the corresponding transport packet for output. The actual output time of the transport packet depends on the alignment of the transport packet time slots, as established by an external clock not known to the processor 160. Additional steps may be carried out, as described below, to dejitter PCRs as a result of this misalignment.

Consider that the bit rates of the TS from which the packet was received (i.e., TS1 or TS2) may be different from the bit rate of the outputted TS, namely TS3. In addition, the transport packets will be internally buffered for a predetermined delay (that depends on the length of the receipt and transmit queues). Nevertheless, assuming that there is no contention between transport packets of different received TSs for the same transport packet slot of the outputted remultiplexed TS TS3, all transport packets will incur approximately the same latency in the remultiplexer node 100. Since the average latency is the same, no jitter is introduced into the transport packets.

Consider now the case that two transport packets are received at nearly the same time from different TSs, i.e., TS1 and TS2, and both are to be outputted in the

remultiplexed TS TS3. Both transport packets may have different estimated departure times that nevertheless correspond to (are nearest in time to) the same transport packet time slot of the outputted remultiplexed TS TS3. The transport packet having the earliest estimated departure time (or receipt time) is assigned to the time slot and the actual dispatch time of this time slot. The other transport packet is assigned the next transport packet time slot of the outputted remultiplexed TS TS3 and the actual dispatch time thereof. Note that the latency incurred by the transport packet assigned to the next time slot is different from the average latency incurred by other transport packets of that program. Thus, the processor 160 illustratively takes steps to remove the latency incurred by this transport packet, including adjusting a PCR of the transport packet (if a PCR is contained therein).

(4) PCR drift and latency adjustment: This process illustratively is contained in the subroutine pointed to by the pointer of the table 404 indexed by the PIDs of transport packets containing PCRs. The processor 160 determines that PCR latency adjustment is only necessary if a transport packet is not assigned to the transport packet time slot of the outputted remultiplexed TS TS3 nearest in time to the estimated departure time of the transport packet (as is done for other transport packets of that program) and if the PCR flag is set in the respective receipt descriptor. PCRs are corrected for the displacement in time incurred by the assignment to the non-ideal slot. This adjustment equals the number of slots from the ideal slot by which the transport packet is displaced times the slot time.

All PCR's are adjusted for drift as described below unless the input and output TSs are exactly aligned in time or the PCR is received from an asynchronous communication link. In the former case, the drift of the internal clock does not affect the timing at which PCR's are outputted. In the latter case, a different drift adjustment is used as described

below. In all other cases, the time at which received PCR's are outputted is affected by drift of the reference clock generator 113 of the adaptors 110 which received the transport packet and the adaptor 110 that transmits the transport packet, relative to the program clock of the PCR. That is, the transport packet containing the PCR is stamped with a receipt time stamp obtained from the reference clock generator 113. This receipt time stamp is used to determine the estimated departure time and the actual dispatch time. As described in detail below, transport packets are dispatched according to their actual dispatch time relative to the reference clock generator 113 on the adaptor 110 that transmits the TS TS3, and all reference clock generators 113 of all adaptors 110 are maintained in synchronicity. However, the reference clock generators 113, while all synchronized to each other, are subject to drift relative to the encoder system time clock that generated the transport packet and PCR thereof. This drift can impact the time at which each PCR is outputted from the remultiplexer node 100 in the outputted remultiplexed TS such as TS3.

According to the invention, the remultiplexer node 100 corrects for such drift. As noted above, part of the receipt handler subroutine for PCRs of each program is to maintain a current measure of drift. A measure of drift of the reference clock generators 113 relative to the encoder system time clock of each program is maintained. For each PCR, the current drift for the program of the PCR (i.e., between the reference clock generators 113 and the encoder system time clock of that program) is subtracted from the PCR.

With the above-noted allocation of queues, selection of PID handler subroutines, and modification of PID filter maps, remultiplexing is performed as follows. The transport packets of TS1 are received at the data link control circuit 112 of the first adaptor 110. Likewise, the transport packets of TS2 are received at the data link control circuit 112 of

the second adaptor 110. The data link control circuit 112 in each of the first and second adaptors 110 consults the local PID filter map stored in the cache 114 thereat and selectively discards each transport packet having a PID indicating that the transport packet is not to be retained. Each data link control circuit 112 retrieves the next unused/non-allocated descriptor from the cache 114 and determines the transport packet storage location associated with the descriptor. (As noted above and below, the DMA control circuit 116 continuously obtains control of a sequence of one or more of the next unused, non-allocated descriptors of the receipt queue assigned to the input port of the data link control circuit 112 and the transport packet storage locations to which these descriptors point.) The next unused, non-allocated descriptor follows the descriptor stored in the descriptor storage location 129 pointed to by the tail pointer 129-4, which tail pointer 129-4 is available to the data link control circuit 112. (As noted above, if the tail pointer 129-4 equals the bottom of the ring address 129-2, the descriptor pointed to by the tail pointer 129-4 will have the end of descriptor ring command bit set in field 129-7 by the processor 160. This will cause the data link control circuit 112 to allocate the descriptor stored in the descriptor storage location 129 at the top of the ring address 129-1, using a wrap-around addressing technique.) The data link control circuit 112 obtains the time of the reference clock generator 113 corresponding to the time the first byte of the transport packet is received and stores this value as the receipt time stamp in the field 129-5 of the allocated descriptor. The data link control circuit 112 stores the number of bytes of the received transport packet in the field 129-8. Also, if any errors occurred in receiving the transport packet (e.g., loss of data link carrier of TS1, short packet, long packet, errored packet, etc.), the data link control circuit 112 indicates such errors by setting appropriate exception bits of 129-6. The data

link control circuit 112 then sets a bit in the status field 129-7 indicating that the descriptor 129 has been processed or processed with exceptions and stores the transport packet at the transport packet storage location of cache 114 pointed to by the pointer in field 129-4. (Note that in the case of a long packet, a sequence of more than one of the next, unused non-allocated descriptors may be allocated to the received transport packet and the excess data stored in the packet storage locations associated with such descriptors. An appropriate gather/scatter bit is set in the attribute field 129-1 of the first of the descriptors to indicate that the packet has more data than in the single transport packet storage space associated with the first of the descriptors. A corresponding bit may also be set in the attribute field 129-1 of the last of the descriptors to indicate that it is the last descriptor of a multi-descriptor transfer. Such a long packet typically occurs when the adaptor receives packets from a stream other than a TS.)

The DMA control circuit 116 writes the transport packet to its corresponding transport packet storage location of transport packet pool 122 in the host memory 120. The DMA control circuit 116 also writes data of the descriptor that points to the written transport packet to the respective descriptor storage location 129 of the receipt queue assigned to the respective adaptor 110. Note that the DMA control circuit 116 can identify which transport packets to write to the host memory 120 by determining which descriptors have the processing completed status bits in the field 129-7 set, and the transport packet storage locations to which such descriptors point. Note that the DMA control circuit 116 may write data of descriptors and transport packets one by one as each is completed. Alternatively, the DMA control circuit 116 may allow a certain threshold number of

transport packets and descriptors to accumulate. The DMA control circuit 116 then writes data of a sequence of $i \geq 1$ multiple completed descriptors and transport packets.

In one embodiment, a scrambler/descrambler circuit 115 is placed on the adaptor 110. In such a case, prior to the DMA control circuit 116 writing data of a transport packet to the host memory 120, the scrambler/descrambler circuit 115 descrambles each transport packet for which descrambling must be performed. This is described in greater detail below.

When the DMA control circuit 116 writes descriptor data and transport packets to the host memory 130, the DMA control circuit 116 interrupts the processor 160. Such interrupts may be initiated by the DMA control circuit 116 every $i \geq 1$ descriptors for which data is written to the host memory 130. The interrupt causes the processor 160 to execute one of the receipt PID handler subroutines for each transport packet which is both PID and input TS specific. As noted above, the receipt PID handler subroutines are selected by appropriate alteration of the pointers in the table 402 so that the processor 160, amongst other things, discards transport packets not to be outputted in the remultiplexed TS, writes an estimated departure time in the descriptors pointing to transport packets that are to be outputted and sets the PCR flag bit in the descriptors pointing to transport packets containing PCRs. In addition, the selected receipt PID handler subroutines preferably cause the processor 160 to continuously acquire and update the PSI tables, adjust the PID filter map and select additional receipt PID handler subroutines as necessary to effect a certain user specification. For example, a user specification can specify that a particular program number is to be continuously outputted in the remultiplexed TS TS3. However, the ESs that make up this program are subject to change due to, amongst other things, reaching an

event boundary. Preferably, the processor 160 will detect such changes in ES make up by monitoring changes to the PAT and PMT and will change the PID filter map and select receipt PID handler subroutines as necessary to continuously cause the ESs of the selected program to be outputted in the remultiplexed TS TS3, whatever the make up of that program is from moment to moment.

Contemporaneously while performing the above functions associated with receiving transport packets, a DMA control circuit 116 and data control link circuit 112 on the third adaptor 110 also perform certain functions associated with transmitting transport packets in TS3. Each time the data link control circuit 112 of this third adaptor 110 outputs $k \geq 1$ transport packets, the data link control circuit 112 generates a transmit interrupt. Illustratively k may be selected by the processor 160. This transmit interrupt is received at the processor 160 which executes an appropriate transmit PID handler subroutine for the outputted remultiplexed TS TS3. In particular, the processor 160 examines the descriptors at the head of each queue that contains descriptors pointing to transport packets to be outputted in TS3. As noted above, two receipt queues contain descriptors pointing to transport packets to be outputted in TS3, including one receipt queue associated with the first adaptor 110 (that receives TS1) and one receipt queue associated with the second adaptor 110 (that receives TS2). As described below, the processor 160 may allocate additional queues containing descriptors pointing to transport packets to be outputted in TS3. The processor 160 identifies the descriptors pointing to the next j transport packets to be outputted in TS3. This is achieved by executing the transmit PID handler subroutines of the set associated with the third adaptor 110 and indexed by the PIDs of the transport packets in the head of the receipt queues. As noted above, if the transport packet

corresponding to a descriptor in a queue examined by the processor 160 is not to be outputted from the third adaptor 110 (that generated the interrupt), the PID of this transport packet will index a transmit PID handler subroutine for the third adaptor 110 that does nothing. If the transport packet corresponding to the descriptor in the queue examined by the processor 160 is to be outputted from the third adaptor 110 (that generated the interrupt), the PID of the transport packet will index a pointer to a transmit PID handler subroutine that will: (1) allocate a transmit descriptor for the transport packet, (2) order the transmit descriptor in the transmit queue associated with the third adaptor 110 in the correct order for transmission, (3) assign an actual dispatch time to the allocated descriptor and transport packet and (4) perform a coarse PCR correction on the transport packet for drift and latency, if necessary. Illustratively, the processor 160 examines descriptors in (receipt) queues until j descriptors pointing to transport packets to be outputted in TS3 or from the third adaptor 110 are identified. The descriptors are examined in order from head 124-3 to tail 124-4. If multiple queues with candidate descriptors are available for examination, the processor 160 may examine the queues in a round-robin fashion, in order of estimated departure time or some other order that may be appropriate considering the content of the transport packets to which the descriptors point (as described below).

The DMA control circuit 116 retrieves from the host memory 120 data of a sequence of $j \geq 1$ descriptors of the queue associated with TS3 or the third adaptor 110. The descriptors are retrieved from the descriptor storage locations 129 of the queue in order from head pointer 124-3 to tail pointer 124-4. The DMA control circuit 116 also retrieves from the host memory 120 the transport packets from the transport packet storage locations

of the pool 122 to which each such retrieved descriptor points. The DMA control circuit 116 stores such retrieved descriptors and transport packets in the cache 114.

The data link control circuit 112 sequentially retrieves from the cache 114 each descriptor in the transmit queue, in order from the head pointer 124-3, and the transport packet in the transport packet storage location to which the descriptor points. When the time of the reference clock generator 113 of the third adaptor 110 equals the time indicated in the dispatch time field 129-5 of the retrieved descriptor, the data link control circuit 112 transmits the transport packet, to which the descriptor (in the storage location pointed to by the head pointer 124-3) points, in TS3. The dispatch time is only the approximate transmit time because each transport packet must be transmitted in alignment with the transport packet time slot boundaries of TS3. Such boundaries are set with reference to an external clock not known to the processor 160. Note also, that the PCRs of each transport packet may be slightly jittered for the same reason. Accordingly, the data link control circuit 112 furthermore finally corrects the PCRs according to the precise transmit time of the transport packet that contains it. Specifically, the precise transmit time is less than a transport packet time off from the estimate. The data link control circuit 112 uses a transport time slot boundary clock, which is previously locked to the time slot boundaries of TS3, to make the final adjustment to the estimated PCRs (namely, by adding the difference between the dispatch time and the actual transmission time to the PCR of the transport packet). Note that the data link control circuit 112 can use the PCR flag bit of the descriptor to determine whether or not a PCR is present in the transport packet (and thus whether or not to correct it).

After transmitting a transport packet, the data link control circuit 112 sets the appropriate status information in field 129-7 of the descriptor that points to the transmitted transport packet and deallocates the descriptor. The DMA control circuit 116 then writes this status information into the appropriate descriptor storage location of the transmit queue.

In another manner of operation, the operator already has full knowledge of the contents of the inputted TSs to be remultiplexed. In this case, the operator simply prepares the user specification and transmits the user specification from the controller 20 to the remultiplexer node 100 (or remultiplexer nodes 100 when multiple nodes operate in concert in a network distributed remultiplexer 100). Preferably, different kinds of information regarding the content of the inputted to-be-remultiplexed TSs (such as the PAT, PMT, etc.) is nevertheless continuously acquired. This enables instantaneous reporting of the content to the operator (via the processor 160 and the controller 20), for example, to enable creation of a modified user specification and to dynamically adjust the remultiplexing according to the modified user specification without ceasing the input of to-be-remultiplexed TSs, the output of the remultiplexed TS or the remultiplexing processing of the remultiplexer 100 noted above.

In addition to the above basic remultiplexing functions, the remultiplexer node 100 can perform more advanced functions. These functions are described individually below.

Dynamic Remultiplexing and Program Specific Information Insertion

As noted above, the operator can use the controller 20 for generating a user specification specifying programs and ESs to retain or discard, programs or ESs to scramble or unscramble (or both), remapping of PIDs, etc. In addition, the processor 160 preferably

continuously acquires content information (e.g., data of the PAT, PMT, CAT, NIT, ECM tables, etc.) This enables simply dynamic, real-time or "on the fly" modification of the user specification and seamless alteration of the remultiplexing according to the new user specification. Specifically, the operator can alter the user specification and cause the remultiplexer 30 to seamlessly switch to remultiplexing according to the new user specification. Nevertheless, the remultiplexer 30 ensures that each outputted remultiplexed TS is always a continuous bitstream containing an unbroken sequence or train of transport packets. Thus, the content of the outputted remultiplexed TS(s) are modified without introducing discontinuities into the outputted remultiplexed TS(s), i.e., no breaks in the train of outputted transport packets, or stoppages in the outputted bit stream, occur.

The above seamless modifications can be affected due to the use of a programmable processor 160 which controls the flow of transport packets between input and output adaptors 110 or interfaces 140 and 150 and other circuits such as the descrambler/scrambler 170. Consider that choosing to retain or discard a different set of ESs can be effected simply by the processor 160 adjusting the appropriate PID filter maps and PID handler subroutines selected by the processor 160 for each PID. Choosing whether to descramble or scramble certain ESs or programs can be achieved by the processor 160 altering the PID handler subroutines executed in response to the PIDs assigned to such ESs or programs to include the appropriate scrambling or descrambling processes (described above and below). A different selection of output ports for outputting a different combination of outputted remultiplexed TSs can be achieved by the processor 160 allocating transmit descriptor queues for the new output ports, deallocating transmit descriptor queues for unneeded output ports, generating tables 404 of pointers to transmit PID handler subroutines for each

new output port and discarding each table 404 of pointers to transmit PID handler subroutines for each deallocated transmit queue. In a like fashion, a different selection of input ports may be achieved by the processor 160 allocating and deallocating receipt queues and generating and discarding tables 402 of pointers to receipt PID handlers for such allocated and deallocated receipt queues, respectively.

In addition to selecting the correct transport packets for output, the remultiplexer node 100 illustratively also provides the correct PSI for each outputted remultiplexed TS. This is achieved as follows. The controller 20 (FIG 2) generates a user specification for the output TS. Consider the above example where the remultiplexer node 100 remultiplexes two TSs, namely, TS1 and TS2 to produce a third TS, namely, TS3. Illustratively, Table 1 sets forth the contents of each of TS1 and TS2.

Table 1

TS1			TS2		
Program	**ES**	**PID**	**Program**	**ES**	**PID**
A	Video A	PID(VA)	E	Video E	PID(VE)
A	Audio A	PID(AA)	E	Audio E	PID(AE)
A	Data A	PID(DA)	PMT	Prog. Def. E	PID(e)
PMT	Prog. Def. A	PID(a)	F	Video F	PID(VF)
B	Video B	PID(VB)	F	Audio F	PID(AF)
B	Audio B	PID(AB)	F	Data F	PID(DF)
PMT	Prog. Def. B	PID(b)	PMT	Prog. Def. F	PID(f)
C	Video C	PID(VC)	G	Video G	PID(VG)
C	Audio C	PID(AC)	G	Audio 1 G	PID(A1G)
C	Decrypt C	PID(ECMC)	G	Audio 2 G	PID(A2G)
PMT	Prog. Def. C	PID(c)	G	Data G	PID(DG)
D	Video D	PID(VD)	G	Decrypt G	PID(ECMG)

Program	ES	PID	Program	ES	PID
D	Audio 1 D	PID(A1D)	PMT	Prog. Def. G	PID(g)
D	Audio 2 D	PID(A2D)	PAT	PAT2	0x0000
D	Data D	PID(DD)			
PMT	Prog. Def. D	PID(d)			
PAT	PAT 1	0x0000			

Preferably, the controller 20 programs the processor 160 to extract the information shown in Table 1 using the acquisition process of receive PID handler subroutines described above.

Suppose the user specification specifies that only programs A, B, F and G should be retained and outputted into a remultiplexed outputted TS TS3. The user indicates this specification at the controller 20 (FIG 1), e.g., using the keyboard/mouse 27 (FIG 1). The controller 20 determines whether or not the user specification is valid. In particular, the controller 20 determines whether or not each output remultiplexed TS, such as TS3, has sufficient bandwidth to output all of the specified programs A, B, F and G and associated PSI (i.e., program definitions a, b, f, g and new, substitute PAT3 to be described below). Such bit rate information can be obtained from the processor 160 if not already known. For example, the processor can execute a PID handler subroutine that determines the bit rate (or transport packet rate) of each program from receipt time stamps assigned to each transport packet of each program bearing a PCR. As described above, such information is obtained anyway by the processor 160 for purposes of performing PCR adjustment. If the user specification is not valid, the controller 20 does not download the user specification.

If the specification is valid, the controller 20 downloads the user specification to the processor 160.

Assume that the user specification can be satisfied by the output bandwidth of TS3. If not already acquired, the processor 160 acquires the PAT and PMT of the inputted TSs TS1 and TS2. Based on the information in PAT1 and PAT2, the processor 160 constructs a substitute PAT3 including only the entries of PAT1 and PAT2 indicating the PIDs of program definitions a, b, f and g associated with programs A, B, F and G. Again, this may be achieved using an appropriate PID handler subroutine for the PIDs of PAT1 and PAT2 and is preferably executed continuously to ensure that any changes to the programs, as reflected in PAT1 and PAT2, are incorporated into the substitute PAT3. The processor 160 generates a sequence of transport packets containing this new substitute PAT3 and stores them in the packet buffer 122. The processor 160 also generates a PAT queue of descriptors pointing to the transport packets bearing PAT3, which queue preferably is implemented as a ring 124. The PAT descriptor queue for the PAT3 transport packets advantageously is dedicated to storing only substitute PAT information. The processor 160 furthermore generates estimated departure times and stores them in the descriptors of the PAT queue that point to the PAT3 transport packets.

The processor 160 can now service the PAT3 descriptor queue in the same way as any of the receipt queues in response to a transmit interrupt. That is, when the data link control circuit 112 transmits $k \geq 1$ packets and interrupts the processor 160, the processor 160 will extract descriptors from the PAT3 queue as well as the receipt queues. Collectively, all queues containing descriptors pointing to to-be-outputted transport packets,

for which transmit descriptors in a transmit queue have not yet been allocated are referred to herein as "connection queues."

The processor 160 then constructs appropriate filter maps and transfers one filter map to a first adaptor 110 that receives TS1 and a second filter map to a second adaptor 110 that receives TS2, respectively. For example, the first filter map may indicate to extract and retain transport packets with the PIDs: PID(VA), PID(AA), PID(DA), PID(a), PID(VB), PID(AB) and PID(b) (as well as possibly other PIDs corresponding to PSI in TS1). Likewise, the second filter map may indicate to extract and retain transport packets with the PIDs: PID(VF), PID(AF), PID(DF), PID(f), PID(VG), PID(A1G), PID(A2G), PID(DG), PID(ECMG) and PID(g) (as well as possibly other PIDs corresponding to PSI in TS2). In response, the first and second data link control circuits 112 receiving TS1 and TS2, extract only those transport packets from TS1 and TS2 according to the filter maps provided by the processor 160. As noted above, the first and second data link control circuits 112 store such extracted packets in a cache 114 and allocate descriptors therefor. First and second DMA control circuits 116 periodically write the extracted transport packets and data of descriptors therefor to the host memory 120. The data of the descriptors written by the first DMA control circuit 116 is stored in respective descriptor storage locations 129 of a first receive queue for the first data link control circuit 112 and the data of the descriptors written by the second DMA control circuit 116 is stored in descriptor storage locations of a second receive queue for the second data link control circuit 112.

In addition, a third DMA control circuit 116 retrieves descriptors from a transmit queue associated with TS3, and transport packets corresponding thereto, and stores them in a cache 114. A third data link control circuit 112 retrieves each descriptor from the

cache 114 and transmits them in TS3. The third data link control circuit 112 generates an interrupt after transmitting k≥1 transport packets. This causes the processor 160 to access the table of pointers to transmit PID handler subroutines for the transmit queue associated with the third data link control circuit 112. In executing the appropriate transmit PID handler subroutine, the processor 160 allocates unused transmit descriptors of the TS3 transmit queue for, and copies pertinent information in such allocated descriptors from, descriptors in available connection queues, namely, the first receive queue, the second receive queue and the PAT3 queue. The transmit descriptors are allocated in the TS3 transmit queue in an order that depends on the estimated dispatch time of the receipt descriptors.

Note also that any kind of PSI may be dynamically inserted, including new program definitions, EMMs, ECMs, a CAT or an NIT.

Consider now a situation where a new user specification is generated while remultiplexing occurs according to a previous user specification. As before, the controller 20 initially verifies that there is sufficient bandwidth to meet the new user specification. If there is, the new user specification is down loaded to the processor 160. The new user specification may require that the processor 160 extract different programs and ESs, map PIDs differently, or generate: (a) new PSI, (b) transport packets bearing the new PSI, and (c) descriptors pointing to the transport packets bearing the new PSI. In the case of modifying the programs or ESs contained in TS3, the processor 160 modifies the PID filter maps to retain the to-be-retained transport packets and to discard the to-be-discarded transport packets according to the new user specification. The new filter maps are transferred to the respective caches 114 which dynamically and immediately switch to

extracting transport packets according to the new user specification. The processor 160 also selects appropriate receipt PID handler subroutines for the new to-be-retained transport packets by modifying the pointers of the receipt PID handler subroutine pointer tables 402 associated with the PIDs of the new, to-be-retained transport packets. Modifications may also be made to pointers of the receipt PID handler subroutine pointer tables 402 indexed by PIDs of transport packets now to-be-discarded. In the case of a new PID remapping, the processor 160 selects appropriate subroutines to perform the new PID remapping.

Such changes may require the generation of new PSI, e.g., a new PAT. The processor 160 selects an appropriate PID handler subroutine for generating the new PSI. For example, in the case of a new PAT, the PID handler subroutines may be triggered by the PIDs of the PATs of TS1 and TS2. The processor 160 generates new PSI and inserts the new PSI into transport packets. Descriptors in a respective PSI queue are allocated for such new PSI transport packets. The processor 160 stops servicing (i.e., refreshing and transferring transport packets from) any PSI descriptor queues pointing to transport packets containing stale PSI and instead services the new PSI descriptor queues.

As each change, i.e., each newly selected PID handler subroutine, each PSI insertion modification or each new PID filter map, is available, the appropriate data link control circuit 112 or processor 160 seamlessly changes its operation. Until such change is effected, the data link control circuit 112 or processor 160 continues to operate under the previous user specification. Some care must be taken in ordering when each change occurs so that the outputted remultiplexed TS is always MPEG-2 compliant. For example, any changes to PID mapping, PID filtering, programs, ESs, ECMs, etc., in the TS, which impact the PMT or PAT are preferably delayed until a new version of the PMT (or specific

program definitions thereof) and/or PAT can be outputted in the TS and an indication for switching over to the new PMT, program definition or PAT is indicated in the TS. Likewise, if EMMs are included or dropped for a conditional access system, the introduction of such EMMs is delayed until a new version of the CAT can be transmitted in the TS. Additional judicious ordering of changes may be desirable for internal processing management of resources, such as storing a pointer to a receipt PID handler subroutine in the appropriate receipt PID handler subroutine pointer table entry indexed by a PID of a transport packet to-be retained (that was previously discarded) prior to altering the PID filter map of the respective adaptor 110 for retaining transport packets with this PID, etc.

The following is an example of modifying the remultiplexing according to a new user specification. Suppose the user provides a new user specification indicating that programs B and F should be dropped and instead, programs C and D should be retained. In response, the controller 20 first determines if there is sufficient bandwidth in the outputted remultiplexed TS TS3 to accommodate all of the new program data, and new PSI that must be generated therefor, in modifying the remultiplexed TS TS3 according to the new user specification. Assuming that there is, the new user specification is downloaded to the remultiplexer node 100. The processor 160 modifies the PID filter map in the first adaptor 110 so as to discard transport packets with PIDs PID(VB), PID(AB), PID(b) and retain transport packets with PIDs PID(VC), PID(AC), PID(ECMC), PID(c), PID(VD), PID(A1D), PID(A2D), PID(DD) and PID(d). Likewise, the processor 160 modifies the PID filter map in the second adaptor 110 so as to discard transport packets with PIDs PID(VF), PID(AF), PID(DF), and PID(f). The processor 160 selects PID handler

subroutines for the PIDs PID(VC), PID(AC), PID(ECMC), PID(c), PID(VD), PID(A1D), PID(A2D), PID(DD) and PID(d), including program definition update processes for each of PIDs PID(c) and PID(d), a control word update process for PID(ECMC), a descrambling control word information insertion process for each of the scrambled ESs of program C, e.g., PID(VC). The processor 160 also generates a different substitute PAT3 including the program definitions a, b, c, d, and g, e.g., in the course of executing PID handler subroutines for PID(0) for each of the first and second adaptors 110.

Now consider the case where another new user specification is provided indicating that the VA video ES of program A should be scrambled. Again, the controller 20 first determines if there is sufficient bandwidth in TS3 to accommodate ECM bearing transport packets for VA and new program definitions for program A. Assuming that there is, the new user specification is downloaded to the remultiplexer node 100. The processor 160 allocates a queue for storing descriptors pointing to transport packets containing the ECMs of VA. The processor 160 selects an appropriate PID handler subroutine for PID(VA) including inserting a scrambling control word into the descriptors pointing to transport packets containing VA. The processor 160 also generates transport packets containing the control words as ECMs for VA and allocates descriptors pointing to these transport packets. This may be achieved using a timer driven interrupt handler subroutine. Alternatively, additional hardware (nor shown) or software executed by the processor 160 generates control words periodically and interrupts the processor 160 when such control words are ready. The processor 160 responds to such interrupts by placing an available control word in one or more transport packets, allocating ECM descriptors of an ECM queue for such transport packets, and loading the new control word into the appropriate control word table.

The processor 160 furthermore selects a receive PID handler subroutine for PID(a) including a process that extracts the information in the program definition a and adds information regarding ECMA (e.g., PID(ECMA), the ES that it encrypts, etc.).

Scrambling/Descrambling Control

One problem associated with scrambling and descrambling is the selection of the correct control word or key for each transport packet. That is, scrambled transport packet data may be scrambled with a PID specific control word or a control word specific to a group of PIDs. A rotating control word scheme may be used where the control word changes from time to time. In short, there may be a large number of control words (e.g., keys) associated with each TS and control words are periodically changed. In the case of descrambling, a mechanism must be provided for continuously receiving control words for each to-be-descrambled ES or group of ESs and for selecting the appropriate control word at each moment of time. In the case of scrambling, a mechanism must be provided for selecting the correct control word for scrambling an ES or group of ESs and for inserting the control word used for scrambling the ESs into the outputted remultiplexed TS sufficiently in advance of any scrambled ES data thereby formed.

The descriptors and their ordering within the receipt and transmit queues can be used to simplify the scrambling and descrambling of TSs. In particular, each receipt descriptor has a field 129-9 in which information pertinent to scrambling or descrambling can be stored, such as the control word to be used in scrambling the transport packet or a pointer to the appropriate control word table containing control words for use in scrambling or descrambling the transport packet.

20

Consider first the steps performed in descrambling a transport packet. The TS containing transport packets to be descrambled contains ECM (ES specific conditional access) and EMM (conditional access specific to a whole group of ESs) bearing transport packets. EMMs are carried in transport packets labeled with PIDs unique to the group of ESs to which they correspond and ECMs are carried in transport packets labeled with PIDs unique to the specific ES to which each ECM corresponds. The PIDs of the EMMs can be correlated to the specific groups of ESs to which they correspond by reference to the CAT. The PIDs of the ECMs can be correlated to each specific ES to which they correspond by reference to the PMT. The processor 160 selects PID handler subroutines for:

(1) recovering each CAT and PMT transmitted in the TS and for identifying which version of the CAT or PMT is currently being used,

(2) by reference to the PMT, recovering a table of ECMs indexed by the PIDs of the transport packets carrying the ESs to which they correspond.

Next, the processor 160 defines a sequence of processing steps to be performed on each transport packet and descriptor. That is, the processor 160 defines the specific order in which the receipt adaptor 110 data link control circuit 112, the (optional) receipt adaptor 110 descrambler 115, the receipt adaptor 110 DMA control circuit 116, the (optional) descrambler 170 and the processor 160 can process a receipt descriptor or packet to which a receipt descriptor points. To this end, the processor 160 may transfer appropriate control information to each of the devices 112, 115 and 116 for causing them to process the transport packet and descriptor that points thereto in the specific order of the defined sequence of processing steps as described below.

If the on adaptor 110 descrambler 115 is used, the order of processing in the sequence is defined as follows. The data link control circuit ~~112 of an~~ adaptor 110 receives transport packets and allocates receipt descriptors for selected ones of those transport packets not discarded as per the PID filter map described above. After storing each retained transport packet in the cache 114, the data link control circuit 112 illustratively sets the status bit(s) 129-7 in the descriptor pointing to the transport packet to indicate that the transport packet may now be processed by the next device according to the order of the defined sequence of processing steps.

The descrambler 115 periodically examines the cache 114 for the next one or more descriptors for which the status bit(s) 129-7 are set to indicate that the descrambler 115 has permission to modify the transport packet. Illustratively, the descrambler 115 accesses the cache 114 after the descrambler 115 has processed $m \geq 1$ descriptors. The descrambler 115 accesses each descriptor of the cache 114 sequentially from the descriptor previously accessed by the descrambler 115 until $m \geq 1$ descriptors are accessed or until a descriptor is reached having the status bit(s) 129-7 set to indicate that processing of a previous step is being performed on the descriptor and transport packet to which it points according to the order of the defined sequence of processing steps.

In processing descriptors and transport packets, the descrambler 115 uses the PID of the transport packet, to which the currently examined descriptor points, to index a descrambling map located in the cache 114. Illustratively, the processor 160 periodically updates the descrambling map in the cache 114 as described below. The location of the descrambling map is provided by a base address located in the descriptor field 129-9. Illustratively, the processor 160 loads the base address of the descrambling map into the

fields 129-9 of each descriptor when allocating the receipt descriptor queues. The indexed entry of the descrambling map indicates whether or not the transport packet is scrambled and, if scrambled, one or more control words that can be used to descramble the transport packet. The indexed entry of the descrambling map can contain the control words corresponding to the PID of the transport packet or a pointer to a memory location in which the respective control word is stored. If the indexed entry of the descrambling map indicates that the transport packet to which the accessed descriptor points is not to be descrambled, the descrambler 115 simply sets the status bit(s) 129-7 of the descriptor to indicate that the next processing step, according to the order of the defined sequence of processing steps, may be performed on the descriptor and transport packet to which it points.

If the indexed entry of the descrambling map indicates that the transport packet is to be descrambled, the descrambler 115 obtains the control word corresponding to the PID of the transport packet and descrambles the transport packet data using the control word. Note that a typical descrambling scheme uses rotating (i.e., odd and even) control words as described above. The correct odd or even control word to use in descrambling a transport packet is indicated by control bits in the transport packet, such as the transport_scrambling_control bits. The descrambler 115 uses these bits, as well as the PID of the transport packet, in indexing the correct control word. That is, the map constructed and maintained by the processor 160 is indexed by both the PID and the odd/even indicator(s). The descrambler 115 then stores the descrambled transport packet data in the transport packet storage location pointed to by the currently examined descriptor thereby overwriting the pre-descrambling data of the transport packet. The descrambler 115 then

sets the status bit(s) 129-7 of the descriptor to indicate that the next processing step according to the order of the defined sequence of processing steps may be performed on the descriptor and transport packet to which it points.

The DMA control circuit 116 periodically writes transport packet data and data of descriptors that point thereto from the cache 114 to respective storage locations 122 and 129 of the host memory 130. In so doing, the DMA control circuit 116 periodically examines a sequence of one or more descriptors in the cache 114 that follow (in receipt queue order) the last descriptor processed by the DMA control circuit 116. If the status bit(s) 129-7 of an examined descriptor indicates that processing by the DMA control circuit 116 may be performed on the examined descriptor, the DMA control circuit 116 sets an appropriate status bit(s) 129-7 in the descriptor indicating that the next step of processing, according to the order of the defined sequence of processing steps, may be performed on the descriptor and the transport packet to which it points. The DMA control circuit 116 then writes the data of the descriptor, and of the transport packet to which it points, to the host memory 130. However, if the status bit(s) 129-7 are set to indicate that a processing step that precedes the processing performed by the DMA control circuit 116 is still being performed on the descriptor, the DMA control circuit 116 refrains from processing the descriptor and transport packet to which it points. Illustratively, when enabled, the DMA control circuit 116 examines descriptors until the DMA control circuit 116 writes data of a sequence of $i \geq 1$ descriptors, and transport packets to which such descriptors point, or a descriptor is encountered having status bit(s) 129-7 indicating that a prior processing step, according to the order of the defined sequence of processing steps, is still being performed

on the descriptor. Each time the DMA control circuit 116 transfers $i \geq 1$ transport packets, the DMA control circuit issues an interrupt.

The processor 160 responds to the interrupt issued by, for example, the DMA control circuit 116, by executing the appropriate receipt PID handler subroutine. The processor 160 examines one or more descriptors of the receipt queue, corresponding to the adaptor 110 from which the interrupt was received, starting from the last descriptor processed by the processor 160. Illustratively, the processor 160 only executes the appropriate receipt PID handler subroutine for those descriptors having a status bit(s) 129-7 set indicating that processing by the processor 160 may be performed on the descriptor. Each time the processor 160 is interrupted, the processor 160 illustratively processes descriptors, and transport packets to which they point, until PID handler subroutines are executed for $i \geq 1$ transport packets or until a descriptor is encountered for which the appropriate status bit(s) 129-7 is set to indicate that processing of a prior processing step (according to the order of the defined sequence of processing steps) is still being performed on the descriptor.

In the course of executing the appropriate receipt PID handler subroutines, the processor 160 recovers all control words for all ESs and updates the descrambling and control word tables or maps used by the descrambler 115 (or 170 as described below). In a rotating control word scheme, the processor 160 maintains multiple (i.e., odd and even) keys for each PID in the control word table or map. The processor 160 may also perform processing for enabling subsequent scrambling of descrambled transport packets (described below). After processing the receipt descriptors, the processor 160 deallocates them by setting their status bit(s) 129-7 to indicate that the descriptor is invalid (and thus the data

link control circuit 112 is the next device to process the descriptors), erasing or resetting selected fields of the descriptor and advancing the head pointer 124-3 to the next descriptor storage location 129.

Consider now the case where the descrambler 115 is not provided on the adaptor 110 or not used. Instead, a descrambler 170 resident on the bus 130 is used. A very similar procedure is carried out as before. However, in this scenario, the order of processing steps of the defined sequence is changed so that the DMA control circuit 116 processes the descriptors (and their corresponding transport packets) after the data link control circuit and before the descrambler and the descrambler 170 processes the descriptors (and their corresponding transport packets) after the DMA control circuit 116 but before the processor 160. Thus, after the data link control circuit 112 allocates a descriptor for a transport packet and sets the appropriate status bit(s) 129-7 to enable the next step of processing to be performed thereon, the DMA control circuit 116 processes the descriptor and transport packet to which it points. As noted above, the DMA control circuit 116, sets the status bit(s) 129-7 to indicate that the next step of processing may be performed on the descriptor and writes the transport packet and descriptor to the host memory 130.

The descrambler 170 periodically examines the descriptors in the receipt queue to identify descriptors that have the status bit(s) 129-7 set to indicate that descrambling processing may be performed on descriptors and transport packets to which they point (according to the order of the defined sequence of processing steps). The descrambler 170 processes such identified transport packets in a similar fashion as discussed above for the descrambler 115. After processing the transport packets, the descrambler 170 sets one or more status bit(s) 129-7 to indicate that the next processing step (according to the order of

the defined sequence of processing steps) can now be performed on the descriptor and transport packet to which it points.

The processor 160 performs the above noted processing in response to the interrupt issued by the DMA control circuit 116, including executing the appropriate receipt PID handler subroutine. Preferably, the queue length of the receipt queue associated with the adaptor 110 that interrupted the processor 160 is sufficiently long relative to the processing time of the descrambler 170 such that the processor 160 examines and processes descriptors that the descrambler 170 had already completed processing. In other words, the processor 160 and descrambler 170 preferably do not attempt to access the same descriptors simultaneously. Rather, the processor 160 begins to process descriptors at a different point in the receipt queue as the descrambler 170.

Consider now the processing associated with scrambling. As with descrambling processing, status bit(s) 129-7 in the descriptor are used to order the processing steps performed on each descriptor and transport packet to which such descriptors point according to an order of a defined sequence of processing steps. Unlike descrambling, scrambling is preferably performed after the processor 160 has allocated transmit descriptors to the to-be-scrambled transport packets. As such, the control word field 129-9 can be used in one of two ways. As in descrambling, an address to the base of a scrambling map may be placed in the control word descriptor field 129-9. Preferably, however, because scrambling occurs after the processor 160 processes the descriptors in the transmit queue, the correct control word, itself, is placed into the control word descriptor field 129-9.

Consider first the scrambling processing wherein scrambling is performed by an on transmit adaptor 110 scrambler 115. The processor 160 obtains ECM transport packets

containing control words that are preferably encrypted. These ECM transport packets are enqueued in a respective corresponding connection queue and are scheduled for output at the correct time. That is, the ECM transport packets are scheduled for injection into the outputted TS sufficiently in advance of the transport packets that they descramble to enable a decoder to recover the control word prior to receiving the transport packets that it descrambles.

At an appropriate time after transmitting the ECM transport packets containing a control word, the processor 160 changes the control word table to cause data to be encrypted using a new key corresponding to the recently transmitted control word. As transport packets are transmitted from an output adaptor, the processor 160 executes transmit PID handler subroutines associated with the PIDs of the transport packets pointed to by descriptors in examined connection queues. For each such to-be-scrambled transport packet, the transmit PID handler subroutine includes a process for inserting control word information into the descriptor associated with the transport packet. The control word information may simply be the base address of a scrambling map to be used in identifying the control word for use in scrambling the transport packet. However, the control word information can also be the correct control word to be used in scrambling the transport packet. The processor 160 may also toggle bits in the transport packet, such as the transport_scrambling_control bits, to indicate which of the most recently transmitted control words should be used to decrypt or descramble the transport packet at the decoder. The processor 160 furthermore illustratively sets one or more status bits 129-7 of the newly allocated transmit descriptor to indicate that the next processing step (according to the order

of the defined sequence of processing steps) should be performed on the transmit descriptor and the transport packet to which it points.

The DMA control circuit 116 of the transmit adaptor 110 periodically retrieves descriptor data from the transmit queue and transport packets to which such descriptors point. In so doing, the DMA control circuit 116 examines the descriptors in the transmit queue following the last descriptor for which the DMA control circuit 116 transferred descriptor data to the cache 114. The DMA control circuit 116 only transfers data of transmit descriptors for which the status bit(s) 129-7 are set to indicate that processing by the DMA control circuit 116 may now be performed (according to the order of the defined sequence of processing steps). For example, the DMA control circuit 116 may examine transmit descriptors until a certain number $k \geq 1$ of transmit descriptors are identified which the DMA control circuit 116 has permission to process or until a descriptor is identified having status bits 129-7 set to indicate that a previous processing step is still being performed on the transmit descriptor and transport packet to which it points. After transferring to the cache 114 data of such transmit descriptors, and the transport packets to which such transmit descriptors point, the DMA control circuit 116 sets the status bit(s) 129-7 of such transferred transmit descriptors to indicate that the next processing step (according to the order of the defined sequence of processing steps) may be performed on the transmit descriptors, and the transport packets to which they point.

Next, the scrambler 115 periodically examines the descriptors in the cache 114 for a sequence of one or more descriptors, and transport packets to which they point, to process. The scrambler 115 only processes those accessed descriptors having one or more status bits 129-7 set to indicate that the scrambling processing step may be performed

thereon (according to the order of the defined sequence of processing steps). The scrambler 115 accesses the control word information field 129-9 and uses the information therein to scramble each to-be-scrambled transport packet. As noted above, the control word information can be used one of two ways. If the control word information is a base address to a scrambling map, the scrambler 115 uses the base address and PID information of the transport packet to index the scrambling map. The indexed entry of the scrambling map indicates whether or not the transport packet is to be scrambled, and if so, a control word to use in scrambling the transport packet. Alternatively, the control word information in the field 129-9, itself, indicates whether or not the transport packet is to be scrambled, and if so, the control word to use in scrambling the transport packet. If the transport packet of the processed descriptor is not to be scrambled, the scrambler 115 simply sets the appropriate status bit(s) 129-7 to indicate that the next processing step (according to the order of the defined sequence of processing steps) may now be performed on the transmit descriptor and the transport packet to which it points. If the transport packet of the processed descriptor is to be scrambled, the scrambler scrambles the transport packet data first, stores the transport packet in the cache in place of the unscrambled transport packet and then sets the appropriate status bit(s) 129-7.

The data link control circuit 112 periodically examines the transmit descriptors in the cache 114 for transmit descriptors having one or more status bits 129-7 set to indicate that processing by the data link control circuit 112 may be performed thereon. For such transmit descriptors, the data link control circuit 112 transmits the transport packets to which such descriptors point, at approximately the actual dispatch time indicated therein. The data link control circuit 112 then deallocates the descriptors (and sets the status bits

129-7 to invalid). Illustratively, each time the data link control circuit 112 transmits a sequence of $k \geq 1$ descriptors, the data link control circuit 112 generates a transmit interrupt for receipt by the processor 160.

In the case that the scrambler 115 is not present or is not used, the scrambler 170 illustratively is used instead. The sequence of processing steps set forth above is changed so that the scrambler 170 processes each transmit descriptor and transport packet to which it points after the processor 160 and before the DMA control circuit 116 and the DMA control circuit 116 processes each transmit descriptor the transport packet to which it points after the scrambler 170 but before the data link control circuit 110.

Bandwidth Optimization

As noted above, often a program bearing TS has null transport packets inserted therein. Such null transport packets are present because excess bandwidth typically must be allocated for each program by the program encoder. This is because the amount of encoded data produced for each ES produced from moment to moment can only be controlled so much. Absent this "overhead bandwidth" encoded ES data would frequently exceed the amount of bandwidth allocated thereto causing encoded ES data to be omitted from the TS. Alternatively, an ES encoder, especially a video ES encoder, might not always have data available to output when a transport packet time slot occurs. For example, a particular picture may take an unexpectedly longer time to encode than previously anticipated, thereby causing a delay in production of encoded video ES data. Such time slots are filled with null transport packets.

Although the presence of null transport packets must be tolerated in the remultiplexer node 100, it is desirable to reduce the number of such bandwidth wasting null transport packets. However, in so doing, the bit rate of each program should not be varied and the end-to-end delay should remain constant for such programs. According to one embodiment, a technique is employed whereby null transport packets are replaced with other to-be-remultiplexed transport packet data, if such other transport packet data is available. This is achieved as follows.

First consider that the processor 160 can have multiple connection queues on hand containing descriptors of to-be-scheduled transport packets, i.e., descriptors in receipt queues, PSI queues, other data queues, etc., not yet transferred to a transmit queue. As noted above, these descriptors may point to transport packets associated with a received incoming TS or to other program related streams generated by the processor 160, such as a PAT stream, a PMT stream, an EMM stream, an ECM stream, a NIT stream, a CAT stream, etc. However, other kinds of to-be-scheduled transport packets and descriptors 129 therefor may be on hand such as non-time sensitive, "bursty" or "best effort" private data bearing transport packets. For example, such extra transport packets may contain transactional computer data, e.g., such as data communicated between a web browser and a web server. (The remultiplexer node 100 may be a server, a terminal or simply an intermediate node in a communication system connected to the "internet." Such a connection to the internet can be achieved using a modem, the adaptor 140 or 150, etc.) Such data does not have a constant end-to-end delay requirement. Rather, such data may be transmitted in bursts whenever there is bandwidth available.

The processor 160 first causes each null transport packet to be discarded. This can be achieved by the processor 160 using a receive PID handler subroutine which discards all null transport packets. This technique illustratively is used when the null transport packets are received from a device other than the adaptor 110, such as the interface 140 or 150. Alternatively, if the null transport packets are received from the adaptor 110, the processor 160 may provide a PID filter map to the data link control circuit 112 which causes each null transport packet to be discarded. Next, according to the receive PID handler subroutine, each incoming transport packet that is to be outputted in the TS is assigned an estimated departure time as a function of the receipt time of the transport packet (recorded in the descriptor therefor) and an internal buffering delay within the remultiplexer node 100. In each respective connection queue containing to-be-scheduled transport packets, the assigned departure times might not be successive transport packet transmission times (corresponding to adjacent time slots) of the outputted TS. Rather, two successive descriptors for transport packets to be outputted in the same output TS may have estimated departure times that are separated by one or more transport packet transmission times (or time slots) of the outputted remultiplexed TS in which the transport packets are to be transmitted.

Preferably, descriptors pointing to program data bearing transport packets, descriptors pointing to PSI, ECM or EMM bearing transport packets and descriptors pointing to bursty data are each maintained in mutually separate connection queues. In implementation, connection queues are each assigned a servicing priority depending on the type of data in the transport packets to which the descriptors enqueued therein point. Preferably, program data received from outside the remultiplexer node (e.g., via a receipt

adaptor 110 or an interface 140 or 150) is assigned the highest priority. Connection queues storing PSI, ECM or EMM streams generated by the remultiplexer node 100 may also be assigned the same priority. Finally, connection queues with descriptors pointing to transport packets containing bursty data with no specific continuity, propagation delay or bit rate requirement, are assigned the lowest priority. In addition, unlike program, PSI, ECM and EMM data, no estimated departure time is assigned to, or recorded in the descriptor of, transport packets bearing bursty data.

In executing transmit PID handler subroutines, the processor 160 transfers descriptors associated with to-be-scheduled transport packets from their respective connection queues to a transmit queue. In so doing, the processor 160 preferably services (i.e., examines the descriptors in) each connection queue of a given priority before resorting to servicing connection queues of a lower priority. In examining descriptors, the processor 160 determines whether or not any examined descriptors of the high priority connection queues (i.e., containing descriptors of transport packets bearing program PSI, ECM or EMM data) point to transport packets that must be transmitted at the next actual dispatch time, based on the estimated departure time assigned to such transport packets. If so, the processor 160 allocates a transmit descriptor for each such transport packet, copies pertinent information from the connection queue descriptor into the allocated transmit queue descriptor and assigns the appropriate dispatch times to each transport packet for which a transmit descriptor is allocated. As noted above, occasionally two or more transport packets contend for the same actual departure time (i.e., the same transport packet time slot of the outputted remultiplexed TS) in which case, a sequence of transport packets are

assigned to consecutive time slots and actual departure times. PCR adjustment for such transport packets is performed, if necessary.

At other times, when the processor 160 services the connection queues, no transport packet of the higher priority connection queues has an estimated departure time that would cause the processor 160 to assign that transport packet to the next available time slot and actual dispatch time of the outputted remultiplexed TS. Ordinarily, this would create a vacant time slot of the outputted remultiplexed TS. Preferably, however, in this situation, the processor 160 services the lower priority connection queues. The processor 160 examines the lower priority connection queues (in order from the head pointer 124-3), selectively assigns a transmit descriptor to each of a sequence of one or more transport packets, to which such examined descriptors point, and copies pertinent information of the examined descriptors to the allocated transmit descriptors. The processor 160 selectively assigns one of the (otherwise) vacant time slots to each transport packet to which such examined descriptors point and stores the actual dispatch time associated with the assigned time slots in the corresponding allocated transmit descriptors.

Occasionally, no transport packets, pointed to by descriptors in a high or low priority connection queue, can be assigned to a time slot of the outputted remultiplexed TS. This can occur because no high priority transport packets have estimated departure times corresponding to the actual dispatch time of the time slot and no bursty data bearing transport packets are buffered pending transmission at the remultiplexer node 100. Alternatively, bursty data bearing transport packets are buffered, but the processor 160 chooses not to assign transmit descriptors therefor at this particular moment of time for reasons discussed below. In such a case, the descriptors in the transmit queue will have

actual transmit times corresponding to a non-continuous sequence of transport packet time slots of the outputted remultiplexed TS. When the data link control circuit 112 of the transmit adaptor 110 encounters such a discontinuity, the data link control circuit 112 transmits a null transport packet at each vacant time slot to which no transport packet is assigned (by virtue of the transmit descriptor actual dispatch time). For example, assume that the dispatch times of two successive descriptors in the transmit queue associated with first and second transport packets indicate that the first transport packet is to be transmitted at a first transport packet time slot and that the second transport packet is to be transmitted at a sixth transport packet time slot. The data link control circuit 112 transmits the first transport packet at the first transport packet time slot. At each of the second, third, fourth, and fifth transport packet time slots, the data link control circuit 112 automatically transmits a null transport packet. At the sixth transport packet time slot, the data link control circuit 112 transmits the second transport packet.

Note that bursty or best effort data typically does not have a rigorous receive buffer constraint. That is, most bursty or best effort data receivers and receiver applications specify no maximum buffer size, data fill rate, etc. Instead, a transport protocol, such as transmit control protocol (TCP) may be employed whereby when a receiver buffer fills, the receiver simply discards subsequently received data. The receiver does not acknowledge receiving the discarded packets and the source retransmits the packets bearing the data not acknowledged as received. This effectively throttles the effective data transmission rate to the receiver. While such a throttling technique might effectively achieve the correct data transmission rate to the receiver it has two problems. First, the network must support two-way communication. Only a fraction of all cable television networks and no direct

broadcast satellite networks support two-way communication between the transmitter and receiver (absent a telephone return path). In any event, where ~~two-~~way communication is supported, the return path from the receiver to the transmitter has substantially less bandwidth than the forward path from the transmitter to the receiver and often must be shared amongst multiple receivers. Thus, an aggressive use of TCP as a throttling mechanism utilizes a large fraction of the return path which must also be used for other receiver to transmitter communications. Moreover, it is undesirable to waste bandwidth of the forward path for transmitting transport packets that are discarded.

Preferably, the insertion of bursty or best effort data should not cause such buffers to overflow. Illustratively, the PID handler subroutine(s) can control the rate of inserting bursty data to achieve some average rate, so as not to exceed some peak rate or even to simply to prevent receiver buffer overflow assuming a certain (or typical) receiver buffer occupancy and pendency of data therein. Thus, even at times when the processor 160 has bursty or best effort data available for insertion into one or more vacant transport packet time slots (and no other data is available for insertion therein), the processor 160 may choose to insert bursty data into only some vacant transport packet time slots, choose to insert bursty data into alternate or spaced apart transport packet time slots or choose not to insert bursty data into any vacant transport packet time slots, so as to regulate the transmission of data to, or to prevent overflow of, an assumed receiver bursty data buffer. In addition, transport packets destined to multiple different receivers may themselves be interleaved, regardless of when they were generated, to maintain some data transmission rate to the receiver.

In any event, the remultiplexer node 100 provides a simple method for optimizing the bandwidth of TSs. All null transport packets in incoming TSs are discarded. If transport packets are available, they are inserted into the time slots that normally would have been allocated to the discarded null transport packets. If transport packets are not available, gaps are left for such time slots by the normal dispatch time assignment process. If no transport packet has a dispatch time indicating that it should be transmitted at the next available time slot of the outputted remultiplexed TS, the data link control circuit 112 automatically inserts a null transport packet into such a time slot.

The benefit of such a bandwidth optimization scheme is two-fold. First, a bandwidth gain is achieved in terms of the outputted remultiplexed TS. Bandwidth normally wasted on null transport packets is now used for transmitting information. Second, best effort or bursty data can be outputted in the TS without specifically allocating bandwidth (or by allocating much less bandwidth) therefor. For example, suppose an outputted remultiplexed TS has a bandwidth of 20 Mbits/sec. Four program bearing TSs of 5 Mbits/sec each are to be remultiplexed and outputted onto the 20 Mbits/sec remultiplexed TS. However, as much as 5% of the bandwidth of each of the four program bearing TSs may be allocated to null packets. As such, it is possible that up to 1 Mbit/sec may be (nominally) available for communicating best effort or bursty data bearing transport packets, albeit without any, or with limited, guarantees of constancy of end-to-end delay.

Re-timing Un-timed Data

As noted above, to-be-remultiplexed program data may be received via the asynchronous interface 140. This presents a problem because the interface 140, and the

communication link to which it attaches, are not designed to transmit data at any specific time and tend to introduce a variable end-to-end delay into communicated data. In comparison, an assumption can be made for program data received at the remultiplexer node 100 via a synchronous communication link (such as is attached to a receiving adaptor 110) that all received transport packets thereof will be outputted without jitter. This is because all such packets incur the same delay at the remultiplexer node 100 (namely, the internal buffering delay), or, if they do not (as a result of time slot contention, as described above), the additional delay is known and the PCRs are adjusted to remove any jitter introduced by such additional delays. In addition, the PCRs are furthermore corrected for drift of the internal clock mechanism relative to the system time clock of each program and for the misalignment between scheduled output time of PCRs and actual output time relative to the slot boundaries of the outputted TS. However, in the case of transport packets received from the interface 140, the transport packets are received at the remultiplexer node 100 at a variable bit rate and at non-constant, jittered times. Thus, if the actual receipt times of the transport packet is used as a basis for estimating the departure of the transport packet, the jitter will remain. Jittered PCRs not only cause decoding and presentation discontinuities at the decoder, they cause buffer overflow and underflow. This is because the bit rate of each program is carefully regulated assuming that the data will be removed from the decoder buffer for decoding and presentation relative to the system time clock of the program.

According to an embodiment, these problems are overcome as follows. The processor 160 identifies the PCRs of each program of the received TS. Using the PCRs, the processor 160 determines the piece-wise transport packet rate of transport packets of

each program between pairs of PCRs. Given the transport packet rate of each (interleaved) sequence of transport packets of each program, the processor 160 can assign estimated departure times based on the times at which each transport packet should have been received.

Illustratively, as the interface 140 receives program data, the received program data is transferred from the interface 140 to the packet buffers 122 of the host memory 120. Specifically, the interface 140 stores received program data in some form of a receipt queue. Preferably, the received program data is in transport packets.

The interface 140 periodically interrupts the processor 160 when it receives data. The interface 140 may interrupt the processor 160 each time it receives any amount of data or may interrupt the processor 160 after receiving a certain amount of data. As with the adaptor 110, a receipt PID handler subroutine pointer table 402 is specially devised for the interface 140. The subroutines pointed to by the pointers may be similar in many ways to the subroutines pointed to by the pointers in the receipt PID handler subroutine pointer table associated with a receive adaptor 110. However, the subroutines are different in at least the following ways. First, the asynchronous interface 140 might not allocate descriptors having the format shown in FIG 2 to received program data and might not receive program data in transport packets. For example, the program data may be PES packet data or PS pack data. In such a case, the subroutines executed by the processor 160 for PIDs of retained transport packets illustratively include a process for inserting program data into transport packets. In addition, a process may be provided for allocating a receipt descriptor of a queue assigned to the adaptor 140 to each received transport packet. The processor 160 stores in the pointer field 129-4 of each allocated descriptor a pointer to the

storage location of the corresponding transport packet. Illustratively, the actual receipt time field 129-5 is initially left blank.

Each transport packet containing a PCR furthermore includes the following process. The first time a PCR bearing transport packet is received for any program, the processor 160 obtains a time stamp from the reference clock generator 113 of any adaptor 110 (or any other reference clock generator 113 that is synchronously locked to the reference clock generators 113 of the adaptors 110). As described below, the reference clocks 113 are synchronously locked. The obtained time stamp is assigned to the first ever received PCR bearing transport packet of a program as the receipt time of this transport packet. Note that other to-be-remultiplexed transport packets may have been received prior to this first received PCR bearing transport packet. The known internal buffering delay at the remultiplexer node 100 may be added to the receipt time stamp to generate an estimated departure time which is assigned to the transport packet (containing the first ever received PCR of a particular program).

After the second successive transport packet bearing a PCR for a particular program is received, the processor 160 can estimate the transport packet rate between PCRs of that program received via the asynchronous interface 140. This is achieved as follows. The processor 160 forms the difference between the two successive PCRs of the program. The processor then divides this difference by the number of transport packets of the same program between the transport packet containing the first PCR and the transport packet containing the second PCR of the program. This produces the transport packet rate for the program. The processor 160 estimates the departure time of each transport packet of a program between the PCRs of that program by multiplying the transport packet rate for the

program with the offset or displacement of each such transport packet from the transport packet containing the first PCR. The offset is determined by subtracting the transport packet queue position of the transport packet bearing the first PCR from the transport packet queue position for which an estimated departure time is being calculated. (Note that the queue position of a transport packet is relative to all received transport packets of all received streams.) The processor 160 then adds the estimated departure time assigned to the transport packet containing the first PCR to the product thus produced. The processor 160 illustratively stores the estimated departure time of each such transport packet in the field 129-10 of the descriptor that points thereto.

After assigning an estimated departure time stamp to the transport packets of a program, the processor 160 may discard transport packets (according to a user specification) that will not be outputted in a TS. The above process is then continuously repeated for each successive pair of PCRs of each program carried in the TS. The data of the descriptors with the estimated departure times may then be transferred to the appropriate transmit queue(s) in the course of the processor 160 executing transmit PID handler subroutines. Note also that initially some transport packets may be received for a program prior to receiving the first PCR of that program. For these transport packets only, the transport packet rate is estimated as the transport packet rate between the first and second PCR of that program (even though these packets are not between the first and second PCR's). The estimated departure time is then determined as above.

As with PCRs received from a synchronous interface such as an adaptor 110, PCRs received via the asynchronous interface 140 are corrected for drift between each program clock and the local reference clocks 113 used to assign estimated receipt time stamps and

to output transport packets. Unlike transport packets received from an adaptor 110, the transport packets received from the interface 140 do not have actual receipt time stamps recorded therefor. As such, there is no reference clock associated with each transport packet from which drift can accurately be measured. Instead, the processor 160 uses a measure of the transmit queue length or current delay therein in the remultiplexer node 100 to estimate drift. Ideally, the transmit queue length should not vary from a predetermined known delay in the remultiplexer node 100. Any variation in transmit queue length is an indication of drift of the reference clock generator(s) 113 of the adaptor(s) 110 relative to the program clocks of the programs. As such, the processor 160 adjusts a measure of drift upwards or downwards depending on the difference between the current transmit queue length and the expected, ideal transmit queue length. For example, each time a transmit descriptor is allocated to a transport packet, the processor 160 measures the current transmit queue length and subtracts it from the ideal transmit queue length in the remultiplexer node 100. The difference is the drift. The drift thus calculated is used to adjust the PCRs and estimated departure times of the transport packets that carry such PCRs. That is, the drift thus calculated is subtracted from the PCR of a transport packet received via the asynchronous interface which is placed into the later time slot than the time slot corresponding to the estimated departure time of the transport packet. Likewise, the drift may be subtracted from the estimated departure time of the PCR bearing transport packet prior to assignment of an actual dispatch time. Note that this estimated drift is only used for transport packets received from the asynchronous interface 140 and not other transport packets received via a synchronous interface such as the adaptor 110.

Now consider the problem of contention. When two (or more) received transport packets contend for assignment to the same transport packet time slot (and actual dispatch time) of the outputted remultiplexed TS, one transport packet is assigned to the time slot and the other is assigned to the next time slot. If the other transport packet contains a PCR, the PCR is adjusted by the number of time slots it is displaced from its ideal time slot to reflect the assignment to a later time slot.

Assisted Output Timing

As noted above, the interface 140 does not receive transport packets at any particular time. Likewise, the interface 140 does not transmit transport packets at any particular time. However, even though the interface 140, and the communication link to which it is attached, do not provide a constant end-to-end delay, it is desirable to reduce the variation in end-to-end delay as much as possible. The remultiplexer node 100 provides a manner for minimizing such variations.

According to an embodiment, the processor 160 allocates a transmit descriptor of a transmit queue assigned to the interface 140 for each transport packet to be outputted via the interface 140. This may be achieved using an appropriate set of transmit PID handler subroutines for the transmit queue assigned to the output port of the interface 140. The processor 160 furthermore assigns an adaptor 110 for managing the outputting of data from this interface 140. Although the transmit queue is technically "assigned" to the interface 140, the DMA control circuit 116 of the adaptor 110 assigned to managing the output from the interface 140 actually obtains control of the descriptors of the descriptor queue assigned to the interface 140. The data link control circuit 112 accesses such descriptors, as

described below, which may be maintained in the cache 114. Thus, the set of transmit PID handler subroutines assigned to this queue, and executed by the processor 160, is actually triggered by an interrupt generated by the data link control circuit 112 which examines the queue.

As above, in response to the interrupt, the processor 160 examines the to-be-scheduled descriptors, i.e., in connection queues, selects one or more descriptors of these connection queues to be outputted from the output port of interface 140 and allocates transmit descriptors for the selected descriptors of the connection queues at the tail of the transmit queue associated with the output port of the interface 140. Unlike the outputting of transport packets described above, the processor 160 may also gather the transport packets associated with the selected descriptors of the connection queues and actually physically organize them into a queue-like buffer, if such buffering is necessary for the interface 140.

As above, the DMA control circuit 116 obtains control of a sequence of one or more descriptors, associated with the output port of the interface 140, following the last descriptor of which the DMA control circuit 116 obtained control. (Note that it is irrelevant whether or not the transport packets corresponding to the descriptors are retrieved. Because the data link control circuit 112 controls the outputting of transport packets at the interface 114, no transport packets are outputted from the output port connected to that data link interface 112. Alternatively, the data link control circuit 112 can operate exactly as described above, thereby producing a mirror copy of the outputted TS. In such a case, a second copy of each transport packet, accessible by the adaptor 110, must also be provided.) As above, the data link control circuit 112 retrieves each descriptor from the

cache and determines, based on the indicated dispatch time recorded in field 129-5, when the corresponding transport packet is to be transmitted relative to the time indicated by the reference clock generator 113. Approximately when the time of the reference clock generator 113 equals the dispatch time, the data link control circuit 112 generates an interrupt to the processor 160 indicating that the transport packet should be transmitted now. This can be the same interrupt as generated by the data link control circuit 112 when it transmits $k \geq 1$ transport packets. However, the interrupt is preferably generated every $k=1$ transport packets. In response, the processor 160 examines the appropriate table of pointers to transmit PID handler subroutines and execute the correct transmit PID handler subroutine. In executing the transmit PID handle subroutine, the processor 160 issues a command or interrupt for causing the interface 140 to transmit a transport packet. This causes the very next transport packet to be transmitted from the output port of the interface 140 approximately when the current time of the reference clock generator 113 matches the dispatch time written in the descriptor corresponding to the transport packet. Note that some bus and interrupt latency will occur between the data link control circuit 112 issuing the interrupt and the interface 140 outputting the transport packet. In addition, some latency may occur on the communication link to which the interface 140 is attached (because it is busy, because of a collision, etc.) To a certain extent, an average amount of such latency can be accommodated through judicious selection of dispatch times of the transport packets by the processor 160. Nevertheless, the outputting of transport packets can be fairly close to the correct time, albeit less close than as can be achieved using the adaptor 110 or interface 150. The processor 160 furthermore transfers one or more

descriptors to the transmit queue assigned to the output port of the interface 140 as described above.

Inter-Adaptor Reference Clock Locking

A particular problem in any synchronous system employing multiple clock generators is that the time or count of each generator is not exactly the same as each other clock generator. Rather, the count of each clock generator is subject to drift (e.g., as a result of manufacturing tolerance, temperature, power variations, etc.). Such a concern is also present in the environment 10. Each remultiplexer node 100, data injector 50, data extractor 60, controller 20, etc. may have a reference clock generator, such as the reference clock generator 113 of the adaptor(s) 110 in the remultiplexer node 100. It is desirable to lock the reference clock generators of at least each node 50, 60 or 100 in the same TS signal flow path so that they have the same time.

In a broadcast environment, it is useful to synchronize all equipment that generates, edits or transmits program information. In analog broadcasting, this may be achieved using a black burst generator or a SMPTE time code generator. Such synchronization enables seamless splicing of real-time video feeds and reduces noise associated with coupling asynchronous video feeds together.

In the remultiplexer node 100, the need for synchronization is even more important. This is because received transport packets are scheduled for departure based on one reference clock and actually retrieved for dispatch based on a second reference clock. It is assumed that any latency incurred by transport packets in the remultiplexer node 100 is identical. However, this assumption is only valid if there is only negligible drift between

the reference clock according to which packet departure is estimated and the reference clock according to which transport packets are actually dispatched.

According to an embodiment, multiple techniques are provided for locking, i.e., synchronizing, reference clock generators 113. In each technique, the time of each "slave" reference clock generator is periodically adjusted in relation to a "master" reference clock generator.

According to a first technique, one reference clock generator 113 of one adaptor 110 is designated as a master reference clock generator. Each other reference clock generator 113 of each other adaptor 110 is designated as a slave reference clock generator. The processor 160 periodically obtains the current system time of each reference clock generator 113, including the master reference clock generator and the slave reference clock generators. Illustratively, this is achieved using a process that "sleeps" i.e., is idle for a particular period of time, wakes up and causes the processor 160 to obtain the current time of each reference clock generator 113. The processor 160 compares the current time of each slave reference clock generator 113 to the current time of the master reference clock generator 113. Based on these comparisons, the processor 160 adjust each slave reference clock generator 113 to synchronize them in relation to the master reference clock generator 113. The adjustment can be achieved simply by reloading the reference clock generators 113, adding an adjusted time value to the system time of the reference clock generator 113 or (filtering and) speeding-up or slowing-down the pulses of the voltage controlled oscillator that supplies the clock pulses to the counter of the reference clock generator 113. The last form of adjustment is analogous to a phase-locked loop feedback adjustment described in the MPEG-2 Systems specification.

Consider now the case where the master reference clock generator and the slave reference clock generator are not located in the same node, but rather are connected to each other by a communication link. For example, the master reference clock generator may be in a first remultiplexer node 100 and the slave reference clock generator may be in a second remultiplexer node 100, where the first and second remultiplexer nodes are connected to each other by a communication link extending between respective adaptors 110 of the first and second remultiplexer nodes 100. Periodically, in response to a timer process, the processor 160 issues a command for obtaining the current time of the master reference clock generator 113. The adaptor 110 responds by providing the current time to the processor 160. The processor 160 then transmits the current time to each other slave reference clock via the communication link. The slave reference clocks are then adjusted, e.g., as described above.

It should be noted that any time source or time server can be used as the master reference clock generator. The time of this master reference clock generator is transmitted via the dedicated communication link with a constant end-to-end delay to each other node containing a slave reference clock.

If two or more nodes 20, 40, 50, 60 or 100 of a remultiplexer 30 are separated by a large geographical distance, it might not be desirable to synchronize the reference clock generators of each node to the reference clock generator of any other node. This is because any signal transmitted on a communication link is subject to some finite propagation delay. Such a delay causes a latency in the transmission of transport packets, especially transport packets bearing synchronizing time stamps. Instead, it might be desirable to use a reference clock source more equally distant from each node of the remultiplexer 30. As is well

known, the U.S. government maintains both terrestrial and satellite reference clock generators. These sources reliably transmit the time on well known carrier signals. Each node, such as the remultiplexer node 100, may be provided with a receiver, such as a GPS receiver 180, that is capable of receiving the broadcasted reference clock. Periodically, the processor 160 (or other circuitry) at each node 20, 40, 50, 60 or 100 obtains the reference clock from the receiver 180. The processor 160 may transfer the obtained time to the adaptor 110 for loading into the reference clock generator 113. Preferably, however, the processor 160 issues a command to the adaptor 110 for obtaining the current time of the reference clock generator 113. The processor 160 then issues a command for adjusting, e.g., speeding up or slowing down, the voltage controlled oscillator of the reference clock generator 113, based on the disparity between the time obtained from the receiver 180 and the current time of the reference clock generator 113.

Networked Remultiplexing

Given the above described operation, the various functions of remultiplexing may be distributed over a network. For example, multiple remultiplexer nodes 100 may be interconnected to each other by various communication links, the adaptor 110, and interfaces 140 and 150. Each of these remultiplexer nodes 100 may be controlled by the controller 20 (FIG 1) to act in concert as a single remultiplexer 30.

Such a network distributed remultiplexer 30 may be desirable as a matter of convenience or flexibility. For example, one remultiplexer node 100 may be connected to multiple file servers or storage devices 40 (FIG 1). A second remultiplexer node 100 may be connected to multiple other input sources, such as cameras, or demodulators/receivers.

Other remultiplexer nodes 100 may each be connected to one or more transmitters/modulators or recorders. Alternatively, remultiplexer nodes 100 may be connected to provide redundant functionality and therefore fault tolerance in the event one remultiplexer node 100 fails or is purposely taken out of service.

Consider a first network remultiplexer 30' shown in FIG 3. In this scenario, multiple remultiplexer nodes 100', 100", 100"' are connected to each other via an asynchronous network, such as a 100 BASE-TX Ethernet network. Each of the first two remultiplexer nodes 100', 100" receives four TSs TS10-TS13 or TS14-TS17 and produces a single remultiplexed output TS TS18 or TS19. The third remultiplexer 100"' receives the TSs TS18 and TS19 and produces the output remultiplexed TS TS20. In the example shown in FIG 3, the remultiplexer node 100' receives real-time transmitted TSs TS10-TS13 from a demodulator/receiver via its adaptor 110 (FIG 2). On the other hand, the remultiplexer 100" receives previously stored TSs TS14-TS17 from a storage device via a synchronous interface 150 (FIG 2). Each of the remultiplexer nodes 100' and 100" transmits its respective outputted remultiplexed TS, i.e., TS18 or TS19, to the remultiplexer node 100"' via an asynchronous (100 BASE-TX Ethernet) interface 140 (FIG 2) to an asynchronous (100 BASE-TX Ethernet) interface 140 (FIG 2) of the remultiplexer node 100"'. Advantageously, each of the remultiplexer nodes 100' and 100" use the above-described assisted output timing technique to minimize the variations in the end-to-end delays caused by such communication. In any event, the remultiplexer node 100"' uses the Re-timing of un-timed data technique described above to estimate the bit rate of each program in TS18 and TS19 and to dejitter TS18 and TS19.

Optionally, a bursty device 200 may also be included on at least one communication link of the system 30'. For example, the communication medium may be shared with other terminals that perform ordinary data processing, as in a LAN. However, bursty devices 200 may also be provided for purposes of injecting and/or extracting data into the TSs, e.g., the TS20. For example, the bursty device 200 may be a server that provides internet access, a web server a web terminal, etc.

Of course, this is simply one example of a network distributed remultiplexer. Other configurations are possible. For example, the communication protocol of the network in which the nodes are connected may be ATM, DS3, etc.

Two important properties of the network distributed remultiplexer 30' should be noted. First, in the particular network shown, any input port can receive data, such as bursty data or TS data, from any output port. That is, the remultiplexer node 100' can receive data from the remultiplexer nodes 100" or 100"' or the bursty device 200, the remultiplexer node 100" can receive data from the remultiplexer nodes 100' or 100"' or the bursty device 200, the remultiplexer node 100"' can receive data from any of the remultiplexer nodes 100' or 100" or the bursty device 200 and the bursty device 200 can receive data from any of the remultiplexer nodes 100', 100" or 100"'. Second, a remultiplexer node that performs data extraction and discarding, i.e., the remultiplexer node 100"' can receive data from more than one source, namely, the remultiplexer nodes 100' or 100" or the bursty device 200, on the same communication link.

As a consequence of these two properties, the "signal flow pattern" of the transport packets from source nodes to destination nodes within the remultiplexer is independent of the network topology in which the nodes are connected. In other words, the node and

communication link path traversed by transport packets in the network distributed remultiplexer 30' does not depend on the precise physical connection of the nodes by communication links. Thus, a very general network topology may be used--remultiplexer nodes 100 may be connected in a somewhat arbitrary topology (bus, ring, chain, tree, star, etc.) yet still be able to remultiplex TSs to achieve virtually any kind of node to node signal flow pattern. For example, the nodes 100', 100", 100"' and 200 are connected in a bus topology. Yet any of the following signal flow patterns for transmitted data (e.g., TSs) can be achieved: from node 100' to node 100" and then to node 100"'; from each of node 100' and 100"' in parallel to node 200; from nodes 200 and 100', in parallel to node 100" and then from node 100" to node 100"', etc. In this kind of transmission, time division multiplexing may be necessary to interleave signal flows between different sets of communicating nodes. For example, in the signal flow illustrated in FIG 3, TS18 and TS19 are time division multiplexed on the shared communications medium.

The above discussion is intended to be merely illustrative of the invention. Those having ordinary skill in the art may devise numerous alternative embodiments without departing from the spirit and scope of the following claims.

Claims

The claimed invention is:

1. A method for optimizing the bandwidth of a transport stream comprising the steps of:

(a) receiving a transport stream at a predetermined bit rate, said transport stream including variably compressed program data bearing transport packets and one or more null transport packets, each of said null transport packets being inserted into a time slot of said received transport stream to maintain said predetermined bit rate of said transport stream when none of said compressed program data bearing transport packets are available for insertion into said received transport stream at said transport packet time slot, and

(b) selectively replacing one or more of said null transport packets with another to-be-remultiplexed data bearing transport packet.

2. The method of claim 1 wherein said another to-be-remultiplexed data bearing transport packet contains program specific information.

3. The method of claim 1 wherein said another to-be-remultiplexed data bearing transport packet contains transactional data having no bit rate or transmission latency requirement for presenting information in a continuous fashion.

4. The method of claim 1 further comprising the steps of:

(c) extracting selected ones of said transport packets of said received transport stream and discarding each non-selected transport packet, each of said null transport packets being discarded,

(d) storing said selected transport packets,

(e) storing at least one other data bearing transport packet,

(f) scheduling each of said stored transport packets for output in an outputted transport stream, and

(g) outputting each of said stored transport packets in a time slot corresponding to said schedule.

5. The method of claim 4 further comprising the steps of:

(h) at each time slot of said outputted transport stream for which a corresponding one of said stored transport packets is scheduled, outputting said corresponding stored transport packet scheduled for said time slot, and

(i) if no transport packet is scheduled for output at one of said time slots, outputting a null transport packet,

wherein said null transport packets of said outputted transport stream occupy less bandwidth of said outputted transport stream than said null transport packets occupy in each transport stream received in step (a).

6. The method of claim 3 wherein said step (b) further comprises selectively assigning data bearing transport packets to time slots of said outputted transport stream so as to regulate a transmission bit rate of said data bearing transport packets to a receiver buffer.

7. A remultiplexer for optimizing the bandwidth of a transport stream comprising:

a first interface for receiving a transport stream at a predetermined bit rate, said transport stream including variably compressed program data bearing transport packets and one or more null transport packets, each of said null transport packets being inserted into a time slot of said received transport stream to maintain said predetermined bit rate of said transport stream when none of said compressed program data bearing transport packets are available for insertion into said received transport stream at said transport packet time slot, and

a processor for selectively replacing one or more of said null transport packets with another to-be-remultiplexed data bearing transport packet.

8. The remultiplexer of claim 7 wherein said another to-be-remultiplexed data bearing transport packet contains program specific information.

9. The remultiplexer of claim 7 wherein said another to-be-remultiplexed data bearing transport packet contains transactional data having no bit rate or transmission latency requirement for presenting information in a continuous fashion.

10. The remultiplexer of claim 7 wherein said first interface and said processor extract selected ones of said transport packets of said received transport stream and discard each non-selected transport packet, each of said null transport packets being discarded, said remultiplexer further comprising:

a memory in which said first interface and said processor store said selected transport packets, and in which said processor stores at least one other data bearing transport packet, said processor scheduling each of said stored transport packets for output in an outputted transport stream, and

a second interface for outputting each of said stored transport packets in a time slot corresponding to said schedule.

11. The remultiplexer of claim 10 wherein, at each time slot of said outputted transport stream for which a corresponding one of said stored transport packets is scheduled, said second interface outputs said corresponding stored transport packet scheduled for said time slot, and, if no transport packet is scheduled for output at one of said time slots, said second interface outputs a null transport packet, said null transport packets of said outputted transport stream occupying less bandwidth of said outputted transport stream than said null transport packets occupy in each received transport stream.

12. The remultiplexer of claim 9 wherein said processor selectively assigns data bearing transport packets to time slots of said outputted transport stream so as to regulate a transmission bit rate of said data bearing transport packets to a receiver buffer.

13. A bandwidth optimized transport stream produced by the steps of:

(a) receiving a transport stream at a predetermined bit rate, said transport stream including variably compressed program data bearing transport packets and one or more null transport packets, each of said null transport packets being inserted into a time slot of said received transport stream to maintain said predetermined bit rate of said transport stream when none of said compressed program data bearing transport packets are available for insertion into said received transport stream at said transport packet time slot, and

(b) selectively replacing one or more of said null transport packets with another to-be-remultiplexed data bearing transport packet.

14. The bandwidth optimized bitstream of claim 13 produced by the further steps of:

(c) extracting selected ones of said transport packets of said received transport stream and discarding each non-selected transport packet, each of said null transport packets being discarded,

(d) storing said selected transport packets,

(e) storing at least one other data bearing transport packet,

(f) scheduling each of said stored transport packets for output in an outputted transport stream, and

(g) outputting each of said stored transport packets in a time slot corresponding to said schedule.

15. The bandwidth optimized transport stream of claim 14 produced by the further steps of:

(h) at each time slot of said outputted transport stream for which a corresponding one of said stored transport packets is scheduled, outputting said corresponding stored transport packet scheduled for said time slot, and

(i) if no transport packet is scheduled for output at one of said time slots, outputting a null transport packet,

wherein said null transport packets of said outputted transport stream occupy less bandwidth of said outputted transport stream than said null transport packets occupy in each transport stream received in step (a).

16. The bandwidth optimized transport stream of claim 13 wherein said step (b) further comprises the step of selectively assigning data bearing transport packets to time slots of said outputted transport stream so as to regulate a transmission bit rate of said data bearing transport packets to a receiver buffer.

17. A method for remultiplexing transport packets, including transport packets containing compressed data for one or more video programs, each of said video programs for which said transport packets contain compressed data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said video program is synchronized, said method comprising the steps of:

(a) receiving a transport packet from a particular input port,

(b) allocating an unused descriptor to said received transport packet, and

(c) recording a receipt time stamp in said allocated descriptor indicating a time at which said transport packet was received,

wherein said allocated descriptors are maintained in a receipt queue associated with said input port in order of receipt from said particular input port.

18. The method of claim 17 further comprising the step of scheduling transmission of said received transport packet according to said receipt time stamp and an internal buffering delay between receipt of said transport packet and output of said transport packet.

19. The method of claim 17 further comprising the steps of:

(d) examining each descriptor in said receipt queue,

(e) allocating a descriptor of a transmit queue associated with an output port from which a transport packet pointed to by each examined descriptor is to be transmitted, if any

(g) assigning a dispatch time to said allocated descriptor of said transmit queue, and

(h) ordering said descriptors of said transmit queue in order of increasing dispatch time.

20. The method of claim 19 further comprising the steps of:

(i) transmitting each transport packet, to which a corresponding descriptor in said transmit queue points, from said output port in a time slot of an outputted transport stream corresponding to said dispatch time assigned to said corresponding descriptor.

21. A method for remultiplexing transport packets, including transport packets containing compressed data for one or more video programs, each of said video programs for which said transport packets contain compressed data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said video program is synchronized, said method comprising the steps of:

(a) sequentially retrieving each descriptor from a queue of transmit descriptors, and a transport packet to which each retrieved descriptor points,

(b) at a time corresponding to a dispatch time recorded in each retrieved descriptor, transmitting said retrieved transport packet to which said retrieved descriptor points in a time slot of an outputted transport stream corresponding to said dispatch time recorded in said retrieved descriptor.

22. The method of claim 21 further comprising the steps of:

(c) examining each descriptor in one or more queues of descriptors pointing to to-be-outputted transport packets,

(d) allocating a descriptor of said transmit queue associated with an output port from which a transport packet pointed to by each examined descriptor is to be transmitted, if any

(e) assigning a dispatch time to said allocated descriptor of said transmit queue, and

(f) ordering said descriptors of said transmit queue in order of increasing dispatch time.

23. A remultiplexer for remultiplexing transport packets, including transport packets containing compressed data for one or more video programs, each of said video programs for which said transport packets contain compressed data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said video program is synchronized, said remultiplexer comprising:

a cache,

a data link control circuit connected to said cache for receiving a transport packet from a particular input port, for allocating an unused descriptor of said cache to said received transport packet, and for recording a receipt time stamp in said allocated descriptor indicating a time at which said transport packet was received,

wherein said allocated descriptors are maintained in a receipt queue associated with said input port in order of receipt from said particular input port.

24. The remultiplexer of claim 23 further comprising a processor for scheduling transmission of said received transport packet according to said receipt time stamp and an internal buffering delay between receipt of said transport packet and output of said transport packet.

25. The remultiplexer of claim 23 further comprising a processor for examining each descriptor in said receipt queue, for allocating a descriptor of a transmit queue associated with an output port from which a transport packet pointed to by each examined descriptor is to be transmitted, if any, for assigning a dispatch time to said allocated descriptor of said

transmit queue, and for ordering said descriptors of said transmit queue in order of increasing dispatch time.

26. The method of claim 25 further comprising:

a second data link control circuit for transmitting each transport packet, to which a corresponding descriptor in said transmit queue points, from said output port in a time slot of an outputted transport stream corresponding to said dispatch time assigned to said corresponding descriptor.

27. A remultiplexer for remultiplexing transport packets, including transport packets containing compressed data for one or more video programs, each of said video programs for which said transport packets contain compressed data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said video program is synchronized, said remultiplexer comprising:

a cache and

a data link control circuit connected to said cache for sequentially retrieving from said cache each descriptor from a queue of transmit descriptors, and a transport packet to which each retrieved descriptor points, and, at a time corresponding to a dispatch time recorded in each retrieved descriptor, for transmitting said retrieved transport packet to which said retrieved descriptor points in a time slot of an outputted transport stream corresponding to said dispatch time recorded in said retrieved descriptor.

28. The remultiplexer of claim 27 further comprising:

a processor for examining each descriptor in one or more queues of descriptors pointing to to-be-outputted transport packets, for allocating a descriptor of said transmit queue associated with an output port from which a transport packet pointed to by each examined descriptor is to be transmitted, if any, for assigning a dispatch time to said allocated descriptor of said transmit queue, and for ordering said descriptors of said transmit queue in order of increasing dispatch time.

29. A transport stream containing transport packets, including transport packets containing compressed data for one or more video programs, each of said video programs for which said transport packets contain compressed data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said video program is synchronized, said transport stream being produced by the steps of:

(a) receiving a transport packet from a particular input port,

(b) allocating an unused descriptor to said received transport packet, and

(c) recording a receipt time stamp in said allocated descriptor indicating a time at which said transport packet was received,

wherein said allocated descriptors are maintained in a receipt queue associated with said input port in order of receipt from said particular input port.

30. A transport stream containing transport packets, including transport packets containing compressed data for one or more video programs, each of said video programs for which said transport packets contain compressed data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said video program is synchronized, said transport stream being produced by the steps of:

(a) sequentially retrieving each descriptor from a queue of transmit descriptors, and a transport packet to which each retrieved descriptor points,

(b) at a time corresponding to a dispatch time recorded in each retrieved descriptor, transmitting said retrieved transport packet to which said retrieved descriptor points in a time slot of an outputted transport stream corresponding to said dispatch time recorded in said retrieved descriptor.

31. A method for remultiplexing one or more program bearing transport streams, each program comprising one or more elementary streams, each transport stream comprising transport packets, including transport packets that carry elementary stream data for one or more programs, said method comprising the steps of:

(a) selectively extracting only particular ones of said transport packets from each of said program bearing transport streams according to an initial user specification for remultiplexed transport stream content,

(b) reassembling said selected ones of said extracted transport packets, and, transport packets containing program specific information, if any, into an outputted

remultiplexed transport stream, according to said initial user specification for remultiplexed transport stream content,

(c) outputting said reassembled remultiplexed transport stream as a continuous bit stream,

(d) while performing said steps (a), (b) and (c), dynamically receiving one or more new user specifications for remultiplexed transport stream content which specifies one or more of:

(I) different transport packets to be extracted in said step (a),

(II) different transport packets to be reassembled in said step (b), and

(e) in response to receiving said one or more new user specifications, dynamically ceasing to extract or reassemble transport packets according to said initial user specification and dynamically beginning to extract or reassemble transport packets according to said new user specification without introducing a discontinuity in said outputted remultiplexed transport stream.

32. The method of claim 31 further comprising the step of:

(f) responding to a new user specification for reassembling different transport packets in step (b) by generating substitute program specific information that references said different transport packets of said new user specification.

33. The method of claim 31 further comprising:

(f) receiving said initial user specification and each new user specification,

(g) determining a total bit rate requirement for said remultiplexed transport stream reassembled according to each of said received user specifications,

(h) performing steps (a), (b) and (e) only if said determined bit rate requirement is less than or equal to a bit rate of said outputted remultiplexed transport stream.

34. The method of claim 31 further comprising the steps of:

(f) continuously identifying streams available for assembly into said outputted remultiplexed transport stream, and

(g) prompting a user for a new user specification that specifies a selection of said identified, available streams as said content for said remultiplexed transport stream.

35. The method of claim 31 wherein said new user specification specifies a new mapping of packet identifiers of one or more transport packets reassembled in said step (b), said step (e) comprising mapping packet identifiers of said one or more transport packets according to said new mapping.

36. The method of claim 31 wherein said new user specification specifies scrambling one or more particular elementary streams, said method further comprising the steps of:

(a) scrambling said transport packets of said specified elementary streams using control words,

(b) providing transport packets containing said control words for reassembly into said remultiplexed transport stream, and

(c) generating transport packets containing program specific information identifying which transport packets contain said control words and to which elementary streams said control word bearing transport packets correspond.

37. A method for remultiplexing transport packets of one or more inputted transport streams into an output transport stream, at least one of said inputted transport streams containing one or more programs and program definitions, each of said programs comprising one or more elementary streams, and each of said at least one inputted transport streams comprising program definitions identifying which transport packets contain elementary stream data for each elementary stream contained in said inputted transport stream and which of said elementary streams make up each program contained in said inputted transport stream, said method comprising the steps of:

(a) generating a user specification indicating one or more programs of said inputted transport streams to be outputted in said output transport stream,

(b) continuously capturing said program definitions,

(c) continuously determining from said captured program definitions which elementary streams make up each program, and

(d) outputting in said outputted transport stream each transport packet containing elementary stream data of each elementary stream determined in said step (c) to make up each program indicated to be outputted in said user specification without introducing a discontinuity in said outputted transport stream.

38. A remultiplexer for remultiplexing one or more program bearing transport streams, each program comprising one or more elementary streams, each transport stream comprising transport packets, including transport packets that carry elementary stream data for one or more programs, said method comprising:

a first interface for selectively extracting only particular ones of said transport packets from each of said program bearing transport streams according to an initial user specification for remultiplexed transport stream content,

a second interface for reassembling said selected ones of said extracted transport packets, and, transport packets containing program specific information, if any, into an outputted remultiplexed transport stream, according to said initial user specification for remultiplexed transport stream content, and for outputting said reassembled remultiplexed transport stream as a continuous bitstream, and

a processor for dynamically receiving one or more new user specifications for remultiplexed transport stream content which specifies one or more of:

(I) different transport packets to be extracted by said first interface,

(II) different transport packets to be reassembled by said second interface,

while said first and second interfaces extract transport packets and reassemble and output said remultiplexed transport stream, and for, in response to receiving said one or more new user specifications, causing said first and second interfaces to dynamically cease to extract or reassemble transport packets according to said initial user specification and dynamically begin to extract or reassemble transport packets according to said new user specification, without introducing a discontinuity in said outputted remultiplexed transport stream.

39. The remultiplexer of claim 38 wherein said processor responds to a new user specification for reassembling different transport packets by generating substitute program specific information that references said different transport packets of said new user specification, for reassembly by said second interface.

40. The remultiplexer of claim 38 further comprising:

a controller for receiving said initial user specification and each new user specification, determining a total bit rate requirement for said remultiplexed transport stream reassembled according to each received user specification, and enabling said first and second interfaces to extract and reassemble according to each of said new user interfaces only if said determined bit rate requirement is less than or equal to a bit rate of said outputted remultiplexed transport stream.

41. The remultiplexer of claim 38 wherein said processor continuously identifies streams available for assembly into said outputted remultiplexed transport stream, said remultiplexer further comprising:

a controller for prompting a user for a new user specification that specifies a selection of said identified, available streams as said content for said remultiplexed transport stream.

42. The remultiplexer of claim 38 wherein said new user specification specifies a new mapping of packet identifiers of one or more transport packets reassembled by said second

interface, said processor mapping packet identifiers of said one or more transport packets according to said new mapping.

43. The remultiplexer of claim 38 wherein said new user specification specifies scrambling one or more particular elementary streams, said remultiplexer further compromising:

a scrambler for scrambling said transport packets of said specified elementary streams using control words,

wherein said processor obtains transport packets containing said control words for reassembly into said remultiplexed transport stream and transport packets containing program specific information identifying which transport packets contain said control words and to which elementary streams said control words correspond.

44. A remultiplexer for remultiplexing transport packets of one or more inputted transport streams into an output transport stream, at least one of said inputted transport stream containing one or more programs and program definitions, each of said programs comprising one or more elementary streams, and each of said at least one inputted transport stream comprising program definitions identifying which transport packets of said inputted transport stream contain elementary stream data for each elementary stream contained in said inputted transport stream and which elementary streams make up each program contained in said elementary stream, said remultiplexer comprising:

a controller for generating a user specification indicating one or more programs of said inputted transport streams to be outputted in said output transport stream,

a first adaptor for continuously capturing said program definitions,

a processor for continuously determining ~~from~~ said captured program definitions which elementary streams make up each program, and

a second adaptor for outputting in said outputted transport stream each transport packet containing elementary stream data of each elementary stream determined to make up each program indicated to be outputted in said user specification without introducing a discontinuity into said outputted transport stream.

45. An outputted remultiplexed transport stream, remultiplexed from one or more program bearing transport streams, each program comprising one or more elementary streams, each transport stream comprising transport packets, including transport packets that carry elementary stream data for one or more programs, said outputted remultiplexed transport stream being produced by the steps of:

(a) selectively extracting only particular ones of said transport packets from each of said program bearing transport streams according to an initial user specification for remultiplexed transport stream content,

(b) reassembling said selected ones of said extracted transport packets, and, transport packets containing program specific information, if any, into an outputted remultiplexed transport stream, according to said initial user specification for remultiplexed transport stream content,

(c) outputting said reassembled remultiplexed transport stream as a continuous bit stream,

(d) while performing said steps (a), (b) and (c), dynamically receiving one or more new user specifications for remultiplexed transport stream content which specifies one or more of:

(I) different transport packets to be extracted in said step (a),

(II) different transport packets to be reassembled in said step (b), and

(e) in response to receiving said one or more new user specifications, dynamically ceasing to extract or reassemble transport packets according to said initial user specification and dynamically beginning to extract or reassemble transport packets according to said new user specification without introducing a discontinuity in said outputted remultiplexed transport stream.

46. An outputted transport stream remultiplexed from one or more inputted transport streams, at least one of said inputted transport streams containing one or more programs and program definitions, each of said programs comprising one or more elementary streams, and each of said at least one inputted transport streams comprising program definitions identifying which transport packets contain elementary stream data for each elementary stream contained in said inputted transport stream and which of said elementary streams make up each program contained in said inputted transport stream, said outputted transport stream being produced by the steps of:

(a) generating a user specification indicating one or more programs of said inputted transport streams to be outputted in said output transport stream,

(b) continuously capturing said program definitions,

(c) continuously determining from said captured program definitions which elementary streams make up each program, and

(d) outputting in said outputted transport stream each transport packet containing elementary stream data of each elementary stream determined in said step (c) to make up each program indicated to be outputted in said user specification without introducing a discontinuity in said outputted transport stream.

47. A method for multiplexing a first video program bearing bit stream into a second bit stream, said first video program bearing bit stream containing a set of plural time stamps for each program contained therein indicating a time relative to a system time clock of an encoder at which each packet of said program should appear in said first bit stream, comprising the steps of:

(a) receiving said first video program bearing bit stream from a communication link having a varying end-to-end transmission delay,

(b) determining a time at which each of one or more of packets carrying data of the same program received from said first video program bearing bit stream should appear in said second bit stream based on a plurality of time stamps of said program received from said first video program bearing bit stream, and

(c) selectively transmitting selected ones of said one or more packets in said second bit stream with a constant end-to-end delay at times that depend on said determined times.

48. The method of claim 47 wherein said step (b) further comprises the steps of:

(b1) storing packets containing data received from said received first video program bearing bit stream in a receipt queue,

(b2) identifying each packet containing data of a program stored in said receipt queue between first and second particular packets containing consecutive time stamps of said program,

(b3) determining a packet rate of said program based on a difference between said first and second time stamps, and

(b4) assigning as a transmit time to each of said identified packets, the sum of a transmit time assigned to said first particular packet and a product of said packet rate and an offset of said identified packet from said first packet.

49. The method of claim 48 further comprising the steps of:

(b5) assigning to a first time stamp bearing packet received for each program carried in said first bitstream a receipt time relative to a local clock, and

(b6) assigning as a transmit time to a packet containing data of said first time stamp bearing packet the sum of said assigned receipt time and a known buffering delay.

50. The method of claim 47 wherein said step (c) further comprises the step of:

(c1) preventing buffer overflow and underflow at a receiver of said second bit stream by inserting said identified packets into said second bit stream at said times that depend on said determined times.

51. The method of claim 50 wherein said receiver buffer removes said identified packets from said second bit stream according to time stamps corresponding to variably compressed portions of said program, and a recovered system time clock for said program,

and wherein said variably compressed portions of said first video program bearing bit stream have a number of bits which number depends on a presumed storage capacity of said receiver buffer and a predetermined bit rate of said first video program.

52. The method of claim 51 wherein said step (c) further comprises the steps of:

(c1) determining a packet time slot of said second bitstream nearest in time to said determined transmit time for a packet,

(c2) if more than one packet is nearest in transport time to a single one of said packet time slots, assigning each of said packets nearest in time to said single packet time slots to sequential packet time slots, and

(c3) adjusting a time stamp of each packet bearing a time stamp and which is assigned to one of said packet time slots other than said single packet time slot based on the number of packet time slots said assigned packet time slot is displaced from said single packet time slot.

53. The method of claim 52 wherein each of said selected received packets is inserted into a queue pending transmission, said step (c3) further comprising the steps of:

(c4) estimating a drift between a local clock and each of one or more system time clocks of encoders that produced said received packets as a function of a difference between a current queue length delay of said queue and an ideal queue length delay of said queue, and

(c5) further adjusting each of said adjusted time stamps according to a corresponding one of said drifts between said local clock and said system time clock of said encoder that produced said packet.

54. The method of claim 47 further comprising the step of:

(d) receiving said first video program bearing bit stream from a computer network.

55. The method of claim 47 further comprising the step of:

(d) receiving said first video program bearing bit stream from an Ethernet network.

56. The method of claim 47 further comprising the step of:

(d) receiving said first video program bearing bit stream from an ATM network.

57. A remultiplexer for multiplexing a first video program bearing bit stream into a second bit stream, said first video program bearing bitstream containing a set of plural time stamps for each program contained therein indicating a time relative to a system time clock of an encoder at which each packet of said program should appear in said first bit stream, comprising:

an asynchronous interface for receiving said first video program bearing bit stream from a communication link having a varying end-to-end transmission delay,

a processor connected to said asynchronous interface for determining a time at which each of one or more of packets carrying data of the same program received from said first video program bearing bit stream should appear in said second bitstream based on a plurality of time stamps of said program received from said first video program bearing bit stream, and

a synchronous interface for selectively transmitting selected ones of said one or more packets in said second bitstream with a constant end-to-end delay at times that depend on said determined times.

58. The remultiplexer of claim 57 further comprising:

a memory for storing packets containing data received from said received first video program bearing bit stream in a receipt queue,

wherein said processor identifies each packet containing data of a program stored in said receipt queue between first and second particular packets containing consecutive time stamps of said program, determines a packet rate of said program based on a difference between said first and second time stamps, and assigns as a transmit time to each of said identified packets, the sum of a transmit time assigned to said first particular packet and a product of said packet rate and an offset of said identified packet from said first packet.

59. The remultiplexer of claim 58 further comprising:

a local clock accessible to said processor, wherein said processor assigns to a first time stamp bearing packet received for each program carried in said first bit stream

a receipt time relative to said local clock, and assigns as a transmit time to a packet containing data of said first time stamp bearing packet the sum of said assigned receipt time and a known buffering delay.

60. The remultiplexer of claim 57:

wherein said transmission of said packets at said times that depend on said determined times by said processor prevents buffer overflow and underflow at a receiver of said second bit stream.

61. The remultiplexer of claim 60 wherein said receiver buffer removes said identified packets from said second bit stream according to time stamps corresponding to variably compressed portions of said program, and a recovered system time clock for said program, and wherein said variably compressed portions of said first video program bearing bit stream have a number of bits which number depends on a presumed storage capacity of said receiver buffer and a predetermined bit rate of said first video program.

62. The remultiplexer of claim 57:

wherein said processor determines a packet time slot of said second bitstream nearest in time to said determined transmit time for a packet,

wherein if more than one packet is nearest in transmit time to a single one of said packet time slots, said processor assigns, to sequential packet time slots, each of said packets nearest in transmit time to said single packet time slots, and

wherein said processor adjusts a time stamp of each packet bearing a time stamp and which is assigned to one of said packet time slots other than said single packet time slot based on the number of packet time slots said assigned packet time slot is displaced from said single packet time slot.

63. The remultiplexer of claim 62 further comprising:

a memory, wherein said asynchronous interface inserts each of said selected received packets into a queue in said memory pending transmission,

wherein said processor estimates a drift between a local clock and each of one or more system time clocks of encoders that produced said received packets as a function of a difference between a current queue length delay of said queue and an ideal queue length delay of said queue, and

wherein said processor further adjusts each of said adjusted time stamps according to a corresponding one of said drifts between said local clock and said system time clock of said encoder that produced said packet.

64. A bit stream produced by multiplexing a first video program bearing bit stream into a second bit stream, said first video program bearing bit stream containing a set of plural time stamps for each program contained therein indicating a time relative to a system time clock of an encoder at which each packet of said program should appear in said first bit stream, said process of remultiplexing comprising the steps of:

(a) receiving said first video program bearing bit stream from a communication link having a varying end-to-end transmission delay,

(b) determining a time at which each of one or more of packets carrying data of the same program received from said first video program bearing bit stream should appear in said second bit stream based on a plurality of time stamps of said program received from said first video program bearing bit stream, and

(c) selectively transmitting selected ones of said one or more packets in said second bitstream with a constant end-to-end delay at times that depend on said determined times.

65. A method for timely outputting compressed video program data bearing bit streams comprising the steps of:

(a) providing a bit stream containing transport packets, said transport packets containing compressed program data of one or more video programs, each of said programs having a predetermined bit rate, said transport packets also containing program clock reference time stamps for each of said programs, to which decoding and presentation of each program is synchronized,

(b) assigning dispatch times to each of one or more selected ones of said transport packets to maintain a predetermined bit rate of a program for which said transport packet carries data and to incur an average latency for each of said transport packets, and

(c) at times that depend on each of said dispatch times, issuing one or more commands to an asynchronous communication interface for causing said asynchronous communication interface to transmit said corresponding selected transport packets at

approximately said dispatch times so as to minimize a jitter of said selected transport packets.

66. The method of claim 65 further comprising the steps of:

(d) allocating a transmit descriptor to each of said transport packets, said transmit descriptors residing in order of said dispatch time in a queue assigned to said asynchronous interface,

(e) recording each of said dispatch times of said transport packets in a transmit descriptor allocated to said respective transport packet,

(f) examining a dispatch time of each of said descriptors in order in said queue,

(g) comparing said examined dispatch time to a time generated by a local reference clock, and

(h) issuing each command at a time determined by said comparison.

67. The method of claim 66 further comprising the steps of:

(i) receiving at least some of said transport packets from another interface,

(j) generating said dispatch times as a function of a time at which each of said transport packets is received and a presumed buffering delay between said time of receipt and said time at which said asynchronous interface generates said output.

68. The method of claim 66 further comprising the steps of:

(i) selecting a transmit PID handler subroutine for performing said steps (b), (d) and (e), and,

(j) each time one of said commands is issued, attempting to repeat steps (b), (d) and (e).

69. The method of claim 65 further comprising the steps of:

(d) receiving said transport packets transmitted from said asynchronous interface at another asynchronous interface of a receiving node,

(e) dejittering said received transport packets at said receiving node, and

(f) remultiplexing at least some of said dejittered transport packets into a second bit stream outputted from said receiving node so that said second bit stream has a continuous end-to-end delay for each program carried therein.

70. A remultiplexer for timely outputting compressed video program data bearing bit streams comprising:

a synchronous interface for providing a bit stream containing transport packets, said transport packets containing compressed program data of one or more video programs, each of said programs having a predetermined bit rate, said transport packets also containing program clock reference time stamps for each of said programs, to which decoding and presentation of each program is synchronized,

a processor for assigning dispatch times to each of one or more selected ones of said transport packets to maintain a predetermined bit rate of a program for which said transport packet carries data and to incur an average latency for each of said transport packets, and

an asynchronous communication interface for, at times that depend on each of said dispatch times, receiving one or more commands and responding thereto by transmitting said corresponding selected transport packets at approximately said dispatch times so as to minimize a jitter of said selected transport packets.

71. The remultiplexer of claim 70 further comprising:

a memory for storing a queue of descriptors assigned to said asynchronous interface, said processor allocating a transmit descriptor to each of said transport packets, said transmit descriptors residing in order of said dispatch time in said queue, said processor also recording each of said dispatch times of said transport packets in a transmit descriptor allocated to said respective transport packet, and

an output data link control circuit examining a dispatch time of each of said descriptors in order in said queue, comparing said examined dispatch time to a time generated by a local reference clock, and causing each command to issue at a time determined by said comparison.

72. The remultiplexer of claim 71 wherein said synchronous interface receives at least some of said transport packets outputted by said asynchronous interface, said processor generating said dispatch times as a function of a time at which each of said transport packets is received at said synchronous interface and a presumed buffering delay between said time of receipt and said time at which said asynchronous interface generates said output.

73. The remultiplexer of claim 71 wherein said processor selects a transmit PID handler subroutine for assigning dispatch times to said transport packets, for allocating descriptors and for recording said assigned dispatch times in said allocated descriptors, and wherein each time one of said commands issues, said processor attempts to assign dispatch times to a subsequent group of said transport packets, allocate descriptors to each transport packet of said subsequent group and record said dispatch times assigned to said subsequent group of said transport packets in said descriptors allocated thereto.

74. The remultiplexer of claim 70, wherein said remultiplexer comprises multiple nodes, said remultiplexer further comprising:

a second asynchronous interface at a receiving node receiving said transport packets transmitted from said asynchronous interface,

a second processor at said receiving node for dejittering said received transport packets at said receiving node, and

an output synchronous interface at said receiving node for remultiplexing at least some of said dejittered transport packets into a second bit stream outputted from said receiving node so that said second bit stream has a continuous end-to-end delay for each program carried therein.

75. A bit stream containing compressed video program data produced by the steps of:

(a) providing a bit stream containing transport packets, said transport packets containing compressed program data of one or more video programs, each of said programs having a predetermined bit rate, said transport packets also containing program clock

reference time stamps for each of said programs, to which decoding and presentation of each program is synchronized,

(b) assigning dispatch times to each of one or more selected ones of said transport packets to maintain a predetermined bit rate of a program for which said transport packet carries data and to incur an average latency for each of said transport packets, and

(c) at times that depend on each of said dispatch times, issuing one or more commands to an asynchronous communication interface for causing said asynchronous communication interface to transmit said corresponding selected transport packets at approximately said dispatch times so as to minimize a jitter of said selected transport packets.

76. The bit stream of claim 75 produced by the further steps of:

(d) receiving said transport packets transmitted from said asynchronous interface at another asynchronous interface of a receiving node,

(e) dejittering said received transport packets at said receiving node, and

(f) remultiplexing at least some of said dejittered transport packets into a second bit stream outputted from said receiving node so that said second bit stream has a continuous end-to-end delay for each program carried therein.

77. A method for remultiplexing one or more bit streams containing compressed program data in an asynchronous communications network comprising plural nodes interconnected by one or more communication links comprising the steps of:

(a) receiving, from one of said communication links at a destination node of said asynchronous communications network, a first bit stream containing data of one or more programs, said first bit stream having one or more predetermined bit rates for portions thereof,

(b) choosing at least part of said received first bit stream for transmission, and

(c) scheduling transmission of said chosen part of said first bitstream so as to output said chosen part of said first bit stream in a transport stream at a rate depending on said predetermined rate of said chosen part of said first bit stream.

78. At multiple nodes of a communication network, a method for remultiplexing one or more portions of bit streams into one or more transport streams containing compressed video program data comprising the steps of:

(a) enabling communication amongst a plurality of nodes connected to a shared communication medium by one or more respective communication links,

(b) selecting a first set of one or more of said nodes for transmitting one or more bit streams onto said shared communications medium,

(c) selecting a second set of one or more of said nodes for receiving said transmitted bit streams from said shared communications medium, for selecting portions of said transmitted bit streams and for transmitting one or more remultiplexed transport streams as a bit stream containing said selected portions, each of said remultiplexed transport streams transmitted as a bit stream being different than said received ones of said transmitted bit streams, and

(d) causing said selected nodes to communicate said bit streams via said shared communication medium according one of plural different signal flow patterns, including at least one signal flow pattern that is different from a topological connection of said nodes to said shared communication medium.

79. The method of claim 78 wherein at least one node can receive bit streams from each of plural other ones of said nodes via a single one of said respective communication links, said method further comprising the step of selecting a subset of said plural other nodes and receiving bit streams at said at least one node from only said selected subset of nodes.

80. The method of claim 78 wherein at least one node receives bit streams from plural other ones of said nodes via a single one of said respective communication links.

81. A network distributed remultiplexer for remultiplexing one or more bit streams containing compressed program data comprising:

one or more communication links, and

a plurality of nodes, interconnected by said one or more communication links into a communications network, said plurality of nodes including a destination node receiving a first bit stream containing data of one or more programs via one of said communications links, said first bit stream having one or more predetermined bit rates for portions thereof, said destination node comprising:

a processor for choosing at least part of said received first bit stream for transmission, and for scheduling transmission of said chosen part of said first

bit stream so as to output said chosen part of said first bit stream in a transport stream at a rate depending on said predetermined rate of said chosen part of said first bit stream.

82. A network distributed remultiplexer for remultiplexing one or more portions of bit streams into one or more transport streams containing compressed video program data comprising:

a shared communication medium comprising one or more communication links,

a plurality of nodes, each of said nodes being connected to said shared communication medium by a respective one or more of said communication links, said plurality of nodes including:

a first set of one or more of said nodes for transmitting one or more bit streams onto said shared communications medium,

a second set of one or more of said nodes for receiving said transmitted bit streams from said shared communications medium, for selecting portions of said transmitted bit streams and for transmitting one or more remultiplexed transport streams as a bit stream containing said selected portions, each of said remultiplexed transport streams transmitted as a bit stream being different than said received ones of said transmitted bit streams, and

a controller node for selecting said first and second sets of nodes and for causing said selected nodes to communicate said bit streams via said shared communication medium according one of plural different signal flow patterns,

including at least one signal flow pattern that is different from a topological connection of said nodes to said shared communication medium.

83. The network distributed remultiplexer of claim 82 wherein said plurality of nodes further comprises at least one node that can receive bit streams from each of plural other ones of said nodes via a single one of said respective communication links, said controller node selecting a subset of said plural other nodes and said at least one node receiving bit streams from only said selected subset of nodes.

84. The network distributed remultiplexer of claim 82 wherein said plurality of nodes comprises at least one node that receives bit streams from plural other ones of said nodes via a single one of said respective communication links.

85. A method for locking reference clocks at circuits that transmit and receive a transport stream formed from a sequence of transport packets containing compressed data for one or more programs, each of said programs having an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said method comprising the steps of:

(a) maintaining a reference clock at each first circuit which receives transport packets and each second circuit which transmits transport packets, said reference clock at each first circuit for indicating a time at which each transport packet is received thereat and said reference clock at each second circuit for indicating when to transmit each transport packet therefrom,

(b) designating a master reference clock to which each other one of said reference clocks is to be synchronized,

(c) periodically obtaining a current time of said master reference clock, and

(d) adjusting each other one of said reference clocks according to a difference between said time at each of said other reference clocks and said current time of said master reference clock so as to match a time of said respective reference clock to a corresponding time of said master reference clock.

86. The method of claim 85 wherein a reference clock at one of said first and second circuits is designated as said master reference clock, said method further comprising the steps of:

(e) simultaneously retrieving a current time of said reference clocks at each said first and second circuits,

(f) forming a difference between said current times of said reference clocks at said one circuit and each of said first and second circuits other than said one circuit, and

(g) adjusting said reference clock at each of said first and second circuits other than said one circuit to reduce said difference.

87. The method of claim 85 wherein said first and second circuits are distributed at multiple nodes, said method further comprising the steps of:

(e) receiving said current time of said master reference clock at a first one of said nodes, and

(f) transmitting said received current time from said first node to a second one of said nodes via a communication link.

88. The method of claim 85 wherein said master reference clock is geographically remote from each of said first and second circuits, said method further comprising the step of:

(e) periodically broadcasting said current time of said master reference clock, and

(f) contemporaneously receiving said broadcasted current time at each of plural remote first and second circuits.

89. A remultiplexing apparatus for remultiplexing a transport stream formed from a sequence of transport packets containing compressed data for one or more programs, each of said programs having an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said remultiplexer comprising:

one or more first circuits that receives transport packets, each first circuit comprising a first reference clock for indicating a time at which each transport packet is received,

one or more second circuit that transmits transport packets, each second circuit comprising a second reference clock for indicating when to transmit each transport packet,

a master reference clock to which each of said first and second reference clocks is to be synchronized, for periodically obtaining a current time of said master reference clock, and

a processor for adjusting each of said first and second reference clocks according to a difference between said time at each of said first and second reference clocks and said current time of said master reference clock so as to match a time of said respective first and second reference clock to a corresponding time of said master reference clock.

90. The remultiplexer of claim 89 wherein a reference clock at one of said first and second circuits is designated as said master reference clock, wherein said processor simultaneously retrieves a current time of said first and second reference clocks at each of said first and second circuits, forms a difference between said current times of said first and second reference clocks at said one circuit and each of said first and second circuits other than said one circuit, and adjusts each first and second reference clock at each of said first and second circuits other than said one circuit to reduce said difference.

91. The remultiplexer of claim 89 wherein said first and second circuits are distributed at multiple nodes, said remultiplexer further comprising:

a communication link connecting first and second ones of said nodes, said first node receiving said current time of said master reference clock and transmitting said received current time from said first node to a second one of said nodes via a communication link.

92. The remultiplexer of claim 89 wherein said master reference clock is geographically remote from each of said first and second circuits, said remultiplexer further comprising:

one or more receivers for contemporaneously receiving a periodic broadcast of said current time of said master reference clock.

93. A method for remultiplexing one or more transport streams formed from a sequence of transport packets, including transport packets containing compressed program data for each of one or more programs and, for each program, program clock reference time stamps, to which decoding and presentation of said program is synchronized, said method comprising the steps of:

(a) providing one or more transport streams,

(b) selecting one or more transport packets of said one or more transport streams for output in a remultiplexed transport stream,

(c) scheduling some of said transport packets for output in a time slot of an outputted transport stream depending on a predetermined delay, each of said time slots occurring approximately at a dispatch time as indicated by a local clock,

(d) adjusting each program clock reference time stamp of each scheduled program clock reference bearing transport packet based on a drift between said local clock and a program system time clock from which said program clock reference time stamp was generated, if any, and

(e) further adjusting each adjusted program clock reference time stamp based on a difference between said dispatch time of said time slot in which said program clock

reference time stamp bearing transport packet is scheduled to be outputted and an actual time at which said time slot occurs relative to an external clock.

94. The method of claim 93 further comprising the steps of:

(f) scheduling other transport packets for output in time slots of said outputted transport stream other than a time slot that depends on said predetermined delay,

(g) calculating an estimated adjustment for each program clock reference time stamp in a selected transport packet outputted in one of said other time slots based on a difference in output time between said one other time slot and a time slot corresponding to said predetermined delay, and

(h) adjusting each program clock reference time stamp, in a program clock reference time stamp bearing transport packet scheduled for output in one of said other time slots, by said estimated adjustment.

95. A remultiplexer for remultiplexing one or more transport streams formed from a sequence of transport packets, including transport packets containing compressed program data for each of one or more programs and, for each program, program clock reference time stamps, to which decoding and presentation of said program is synchronized, said method comprising:

a local clock,

a processor responsive to said local clock for selecting one or more transport packets of one or more transport streams for output in a remultiplexed transport stream, for scheduling some of said transport packets for output in a time slot of an outputted transport

stream depending on a predetermined delay, each of said time slots occurring approximately at a dispatch time as indicated by said local clock, for adjusting each of program clock reference time stamp in each scheduled program clock reference time stamp bearing transport packet depending on a drift between said local clock and a program system time clock from which said program clock reference time stamp was generated, if any, and

an output data link control circuit responsive to transport packets scheduled by said processor for further adjusting each adjusted program clock reference time stamp based on a difference between said dispatch time of said time slot in which said program clock reference time stamp bearing transport packet is scheduled to be outputted and an actual time at which said time slot occurs relative to an external clock.

96. The remultiplexer of claim 95 wherein said processor is also for scheduling other transport packets for output in time slots of said outputted transport stream other than a time slot that depends on said predetermined delay, for calculating an estimated adjustment for each program clock reference time stamp, in a program clock reference time stamp bearing transport packet scheduled for output in one of said other time slots, based on a difference in output time between said one other time slot and a time slot corresponding to said predetermined delay, and for adjusting each program clock reference time stamp by said estimated adjustment.

97. A bit stream formed from a sequence of transport packets, including transport packets containing compressed program data for each of one or more programs and, for

each program, program clock reference time stamps, to which decoding and presentation of said program is synchronized, said bit stream being produced by the steps of:

(a) providing one or more transport streams,

(b) selecting one or more transport packets of said one or more transport streams for output in a remultiplexed transport stream,

(c) scheduling some of said transport packets for output in a time slot of an outputted transport stream depending on a predetermined delay, each of said time slots occurring approximately at a dispatch time as indicated by a local clock,

(d) adjusting each program clock reference time stamp of each scheduled program clock reference bearing transport packet based on a drift between said local clock and a program system time clock from which said program clock reference time stamp was generated, if any, and

(e) further adjusting each adjusted program clock reference time stamp based on a difference between said dispatch time of said time slot in which said program clock reference time stamp bearing transport packet is scheduled to be outputted and an actual time at which said time slot occurs relative to an external clock.

98. The bit stream of claim 97 formed by the further steps of:

(f) scheduling other transport packets for output in time slots of said outputted transport stream other than a time slot that depends on said predetermined delay,

(g) calculating an estimated adjustment for each program clock reference time stamp in a selected transport packet outputted in one of said other time slots based on a

difference in output time between said one other time slot and a time slot corresponding to said predetermined delay, and

(h) adjusting each program clock reference time stamp, in a program clock reference time stamp bearing transport packet scheduled for output in one of said other time slots, by said estimated adjustment.

99. A method for remultiplexing transport packets, including transport packets containing compressed program data, each program for which said transport packets contain program data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said method comprising:

(a) allocating to each received transport packet to be retained, an unused descriptor in one of a sequence of descriptor storage locations of which a cache has obtained control, said sequence of descriptor storage locations being part of a queue allocated to a particular input port,

(b) storing each retained transport packet at a transport packet storage location, of which said cache has obtained control, and to which said allocated descriptor points, and

(c) obtaining control of one or more unused descriptor storage locations of said queue following a last descriptor storage location of which said cache has already obtained control, and transport packet locations to which such descriptors in said one or more descriptor storage locations point, said queue of descriptor storage locations and transport packet storage locations being maintained in a memory that is separated from said cache

by an asynchronous communication link having a varying end-to-end communication delay.

100. The method of claim 99 further comprising:

(d) writing data of said allocated descriptors to corresponding descriptor storage locations of said memory, and writing transport packets to transport packet storage locations pointed to by said allocated descriptors for which data is written to said memory, via said communication link.

101. The method of claim 100 further comprising:

(e) periodically examining said descriptor data written to said descriptor storage locations of each queue in said memory associated with an input port,

(f) processing said transport packets in transport packet locations pointed to by said examined descriptors, and

(g) allocating for selected ones of said descriptors of one or more of said queues associated with input ports, a descriptor of a queue associated with an output port,

(h) copying selected information from each selected descriptor of said one or more queues associated with input ports to said descriptor of said queue associated with said output port, and

(g) ordering said descriptors within said queue associated with said output port in a particular order for transmission from said output port.

102. The method of claim 101 further comprising the steps of:

(i) retrieving each descriptor of said queue associated with said output port from a second cache, each descriptor being retrieved from a beginning of a sequence of descriptor storage locations in said second cache, and retrieving from said second cache each transport packet stored in a transport packet storage location to which each retrieved descriptor points,

(j) outputting each retrieved transport packet in a unique time slot of a transport stream outputted from said particular output port, and

(k) obtaining from said memory for storage in said second cache, descriptors of said queue associated with said output port in descriptor storage locations following said descriptor storage locations in which a last cached descriptor of said sequence is stored, and each transport packet stored in a transport packet location to which said obtained descriptors point.

103. A method for remultiplexing transport packets, including transport packets containing compressed program data, each program for which said transport packets contain program data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said method comprising:

(a) retrieving from a cache, each descriptor of a sequence of descriptor storage locations of a queue assigned to an output port, each descriptor being retrieved from a beginning of said sequence, and retrieving from said cache each transport packet stored in a transport packet storage location to which each retrieved descriptor points,

(b) outputting each retrieved transport packet in a unique time slot of a transport stream outputted from said particular output port, and

(c) obtaining from a memory for storage in said cache, via an asynchronous communication link having a varying end-to-end communication delay, one or more descriptors in descriptor storage locations of said queue following a descriptor storage location in which a last cached descriptor of said sequence is stored, and each transport packet stored in a transport packet location to which said obtained descriptors point.

104. The method of claim 103 further comprising:

(d) providing in said memory additional queues of descriptors storage locations containing one or more descriptors pointing to one or more transport packet storage locations, in which to-be-outputted transport packets are stored,

(e) periodically examining descriptor data written to said descriptor storage locations of each of said additional queues in said memory,

(f) processing said transport packets in transport packet locations pointed to by said examined descriptors, and

(g) allocating to selected ones of said descriptors of one or more of said additional queues, a descriptor of said queue assigned to said output port, copying selected information from each selected descriptor of said one or more additional queues to said allocated descriptor of said queue assigned to said output port and ordering said allocated descriptors of said queue assigned to said output port in a particular order for transmission from said output port.

105. A remultiplexer for remultiplexing transport packets, including transport packets containing compressed program data, each program for which said transport packets contain program data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said remultiplexer comprising:

a cache,

a data link control circuit connected to said cache for allocating to each received transport packet to be retained, an unused descriptor in one of a sequence of descriptor storage locations of which said cache has obtained control, said sequence of descriptor storage locations being part of a queue allocated to a particular input port, and for storing each retained transport packet at a transport packet storage location of which said cache has obtained control and to which said allocated descriptor points, and

a direct memory access circuit connected to said cache for obtaining control of one or more unused descriptor storage locations of said queue following a last descriptor storage location of which said cache has already obtained control, and transport packet locations to which such descriptors in said one or more descriptor storage locations point, said queue of descriptor storage locations, and transport packet storage locations being maintained in a memory that is separated from said cache by an asynchronous communication link having a varying end-to-end communication delay.

106. The remultiplexer of claim 105 wherein said direct memory access circuit writes data of said allocated descriptors to corresponding descriptor storage locations of said

memory, and writes transport packets to transport packet storage locations pointed to by said allocated descriptors for which data is written of said memory, via said asynchronous communication link.

107. The remultiplexer of claim 106 further comprising:

a processor for periodically examining said descriptor data written to said descriptor storage locations of each queue in said memory associated with an input port, for processing said transport packets in transport packet locations pointed to by said examined descriptors, for allocating for selected ones of said descriptors of one or more of said queues associated with input ports, a descriptor of a queue associated with an output port, for copying selected information from each selected descriptor of said one or more queues associated with input ports to said descriptor within said queue associated with said output port, and for ordering said descriptors within said queue associated with said output port in a particular order for transmission from said output port.

108. The remultiplexer of claim 107 further comprising:

a second cache,

a second data link control circuit for retrieving from said second cache each descriptor of said queue associated with said output port, each descriptor being retrieved from a beginning of a sequence of descriptor storage locations in said second cache, for retrieving from said second cache each transport packet stored in a transport packet storage location to which each retrieved descriptor points, and for outputting each retrieved

transport packet in a unique time slot of a transport stream outputted from said particular output port, and

a second direct memory access circuit connected to said asynchronous communication link for obtaining from said memory for storage in said second cache, descriptors of said queue associated with said output port in descriptor storage locations following said descriptor storage locations in which a last cached descriptor of said sequence is stored, and each transport packet stored in a transport packet location to which said obtained descriptors point.

109. A remultiplexer for remultiplexing transport packets, including transport packets containing compressed program data, each program for which said transport packets contain program data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said remultiplexer comprising:

a cache,

a data link control circuit connected to said cache for retrieving from said cache each descriptor of a sequence of descriptor storage locations of a queue assigned to an output port, each descriptor being retrieved from a beginning of said sequence, for retrieving from said cache each transport packet stored in a transport packet storage location to which each retrieved descriptor points, and for outputting each retrieved transport packet in a unique time slot of a transport stream outputted from said particular output port, and

a direct memory access circuit connected to said cache for obtaining from said memory for storage in said cache, via an asynchronous communication link having a varying end-to-end communication delay, one or more descriptors in descriptor storage locations of said queue following a descriptor storage location in which a last cached descriptor of said sequence is stored, and each transport packet stored in a transport packet location to which said obtained descriptors point.

110. The remultiplexer of claim 109 further comprising:

a memory connected to said asynchronous communication link for maintaining additional queues of descriptors storage locations containing one or more descriptors pointing to one or more transport packet storage locations, in which to-be-outputted transport packets are stored,

a processor connected to said asynchronous communication link for periodically examining descriptor data written to said descriptor storage locations of each of said additional queues in said memory, for processing said transport packets in transport packet locations pointed to by said examined descriptors, and for allocating to selected ones of said descriptors of one or more of said additional queues, a descriptor of said queue assigned to said output port, copying selected information from each selected descriptor of said one or more additional queues to said allocated descriptor of said queue assigned to said output port and ordering said allocated descriptors of said queue assigned to said output port in a particular order for transmission from said output port.

111. A transport stream containing transport packets, including transport packets containing compressed program data, each program for which said transport packets contain program data comprising a constant end-to-end communication delay requirement, an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said transport stream being produced by the steps of:

(a) allocating to each received transport packet to be retained, an unused descriptor in one of a sequence of descriptor storage locations of which a cache has obtained control, said sequence of descriptor storage locations being part of a queue allocated to a particular input port,

(b) storing each retained transport packet at a transport packet storage location of which said cache has obtained control pointed to by said descriptor allocated thereto, and

(c) obtaining control of one or more unused descriptor storage locations of said queue following a last descriptor storage location of which said cache has already obtained control, and transport packet locations to which such descriptors in said one or more descriptor storage locations point, said queue of descriptor storage locations and transport packet storage locations being maintained in a memory that is separated from said cache by an asynchronous communication link having a varying end-to-end communication delay.

112. A transport stream containing transport packets, including transport packets containing compressed program data, each program for which said transport packets contain program data comprising a constant end-to-end communication delay requirement,

an independent bit rate and program clock reference time stamps of an independent encoder system time clock to which decoding and presentation of said program is synchronized, said transport stream being produced by the steps of:

(a) retrieving from a cache each descriptor of a sequence of descriptor storage locations of a queue assigned to an output port, each descriptor being retrieved from a beginning of said sequence, and retrieving from said cache each transport packet stored in a transport packet storage location to which each retrieved descriptor points,

(b) outputting each retrieved transport packet in a unique time slot of a transport stream outputted from said particular output port, and

(c) obtaining from a memory for storage in said cache, via an asynchronous communication link having a varying end-to-end communication delay, one or more descriptors in descriptor storage locations of said queue following a descriptor storage location in which a last cached descriptor of said sequence is stored, and each transport packet stored in a transport packet location to which said obtained descriptors point.

113. A method for descrambling transport packets of a transport stream, said transport packets containing elementary stream data of one or more video programs, said method comprising the steps of:

(a) defining a sequence of one or more processing steps to be performed on each transport packet and ordering the step of descrambling processing within said sequence,

(b) allocating to each transport packet a descriptor of a queue, each allocated descriptor containing a pointer to said transport packet to which it is allocated, one or more processing indications and a storage location for control word information,

(c) storing control word information associated with contents of said transport packet in said control word information storage location of selected ones of said allocated descriptors,

(d) setting said one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on each of said allocated descriptors,

(e) sequentially accessing each allocated descriptor, and

(f) for each accessed descriptor pointing to a to-be-descrambled transport packet, descrambling said transport packet pointed to by said accessed descriptor using said control word information in said accessed descriptor, only if said one or more processing indications of said accessed descriptor are set to indicate that descrambling processing may be performed on said accessed descriptor and transport packet to which said accessed descriptor points.

114. The method of claim 113 wherein said control word information is a base address of a control word table.

115. The method of claim 114 further comprising the steps of:

(g) during said step of descrambling, locating a control word table using said base address and retrieving a control word from an entry of said control word table indexed by a packet identifier of said transport packet, each packet identifier uniquely indicating the elementary stream data contained in said transport packet.

116. The method of claim 115 wherein said step of locating further comprises using an odd/even control word indication of said transport packet for retrieving said control word.

117. The method of claim 115 further comprising the steps of:

(g) maintaining a control word table containing said control words for descrambling contents of said transport packets.

118. The method of claim 113 further comprising the steps of:

(g) writing descrambled transport packet data into a transport packet storage location pointed to by said pointer of said allocated descriptor, thereby overwriting pre-descrambling data of said transport packet, and

(h) after examining each descriptor containing processing indications that indicate that descrambling processing may be performed, setting one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on said descriptor, and transport packet to which said descriptor points.

119. A method for scrambling transport packets of a transport stream, said transport packets containing elementary stream data of one or more video programs, said method comprising the steps of:

(a) defining a sequence of one or more steps to be performed on each transport packet and ordering scrambling processing within said sequence,

(b) allocating to each transport packet a descriptor of a queue, each allocated descriptor containing a pointer to said transport packet to which it is allocated, one or more processing indications and a storage location for control word information,

(c) storing control word information associated with contents of said transport packet in said control word information storage location of selected ones of said allocated descriptors,

(d) setting said one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on each of said allocated descriptors,

(e) sequentially accessing each allocated descriptor, and

(f) for each accessed descriptor pointing to a to-be-scrambled transport packet, scrambling said transport packet pointed to by said accessed descriptor using said control word information in said accessed descriptor, only if said one or more processing indications of said accessed descriptor are set to indicate that scrambling processing may be performed on said accessed descriptor and transport packet to which said accessed descriptor points.

120. The method of claim 119 wherein said control word information is a control word corresponding to contents of each transport packet.

121. The method of claim 120 further comprising the steps of:

(g) during said step of allocating, retrieving said control word from an entry of a control word table indexed by a packet identifier of said transport packet, each packet

identifier uniquely indicating the elementary stream data contained in said transport packet, and

(h) storing said retrieved control word in said control word storage location of said descriptor.

122. The method of claim 121 further comprising the steps of:

(i) maintaining a control word table containing said control words for scrambling contents of said transport packets.

123. The method of claim 119 further comprising the steps of:

(g) writing scrambled transport packet data into a transport packet storage location pointed to by said pointer of said allocated descriptor, thereby overwriting pre-scrambled data of said transport packet, and

(h) after examining each descriptor containing one or more processing indications that indicate that scrambling processing may be performed, setting one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on said descriptor, and transport packet to which said descriptor points.

124. A remultiplexer for descrambling transport packets of a transport stream, said transport packets containing elementary stream data of one or more video programs, said remultiplexer comprising:

a processor for defining a sequence of one or more processing steps to be performed on each transport packet and for ordering descrambling processing within said sequence,

a data link control circuit for allocating to each transport packet a descriptor of a queue, each allocated descriptor containing a pointer to said transport packet to which it is allocated, one or more processing indications and a storage location for control word information, and for setting said one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on each of said allocated descriptors, and

a descrambler for sequentially accessing each allocated descriptor, and, for each accessed descriptor pointing to a to-be-descrambled transport packet, descrambling said transport packet pointed to by said accessed descriptor using control word information in said accessed descriptor, only if said one or more processing indications of said accessed descriptor are set to indicate that descrambling processing may be performed on said accessed descriptor and transport packet to which said accessed descriptor points,

wherein said processor also stores control word information associated with the contents of received transport packets in said control word storage locations of corresponding ones of said descriptors.

125. The remultiplexer of claim 124 wherein said control word information is a base address of a control word table.

126. The remultiplexer of claim 125 wherein said descrambler locates a control word table using said base address and retrieves a control word from an entry of said control word table indexed by a packet identifier of said transport packet, each packet identifier uniquely indicating the elementary stream data contained in said transport packet.

127. The remultiplexer of claim 126 wherein said descrambler locates said control word using an odd/even indicator of said transport packet to index said control word table.

128. The remultiplexer of claim 126 wherein said processor maintains a control word table containing said control words for descrambling contents of said transport packets.

129. The remultiplexer of claim 124 wherein said descrambler writes descrambled transport packet data into a transport packet storage location pointed to by said pointer of said allocated descriptor, thereby overwriting pre-descrambling data of said transport packet, and, after examining each descriptor containing processing indications that indicate that descrambling processing may be performed, sets one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on said descriptor and transport packet to which said descriptor points.

130. A remultiplexer for scrambling transport packets of a transport stream, said transport packets containing elementary stream data of one or more video programs, said remultiplexer comprising:

a processor for defining a sequence of one or more processing steps to be performed on each transport packet, for ordering scrambling processing within said sequence, for allocating to each transport packet a descriptor of a queue, each allocated descriptor containing a pointer to said transport packet to which it is allocated, one or more processing indications and a storage location for control word information, storing control word information associated with contents of said transport packet in said control word information storage location of selected ones of said allocated descriptors, and for setting one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on each of said allocated descriptors, and

a scrambler for sequentially accessing each allocated descriptor, and, for each accessed descriptor pointing to a to-be-scrambled transport packet, scrambling said transport packet pointed to by said accessed descriptor using said control word information in said accessed descriptor, only if said one or more processing indications of said accessed descriptor are set to indicate that scrambling processing may be performed on said accessed descriptor and transport packet to which said accessed descriptor points.

131. The remultiplexer of claim 130 wherein said control word information is a control word corresponding to contents of each transport packet.

132. The remultiplexer of claim 131 wherein said processor retrieves said control word from an entry of a control word table indexed by a packet identifier of said transport packet, each packet identifier uniquely indicating the elementary stream data contained in said

transport packet, and stores said retrieved control word in said control word storage location of said descriptor.

133. The remultiplexer of claim 132 wherein said processor maintains a control word table containing said control words for scrambling contents of said transport packets.

134. The remultiplexer of claim 130 wherein said scrambler writes scrambled transport packet data into a transport packet storage location pointed to by said pointer of said allocated descriptor, thereby overwriting pre-scrambled data of said transport packet, and, after examining each descriptor, containing one or more processing indications that indicate that scrambling processing may be performed, sets one or more of said processing indications to indicate that the nest step of processing of said sequence may be performed on said descriptor and transport packet to which said descriptor points.

135. A transport stream containing descrambled transport packets, said transport packets containing elementary stream data of one or more video programs, said transport stream being produced by the steps of:

(a) defining a sequence of one or more processing steps to be performed on each transport packet and ordering descrambling processing within said sequence,

(b) allocating to each transport packet a descriptor of a queue, each allocated descriptor containing a pointer to said transport packet to which it is allocated, one or more processing indications and a storage location for control word information,

(c) storing control word information associated with contents of said transport packet in said control word information storage location of selected ones of said allocated descriptors,

(d) setting one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on each of said allocated descriptors,

(e) sequentially accessing each allocated descriptor, and

(f) for each accessed descriptor pointing to a to-be-descrambled transport packet, descrambling said transport packet pointed to by said accessed descriptor using said control word information in said accessed descriptor, only if said one or more processing indications of said accessed descriptor are set to indicate that descrambling processing may be performed on said accessed descriptor and transport packet to which said accessed descriptor points.

136. A transport stream containing scrambled transport packets, said transport packets containing elementary stream data of one or more video programs, said transport stream being produced by the steps of:

(a) defining a sequence of one or more processing steps to be performed on each transport packet and ordering scrambling processing within said sequence,

(b) allocating to each transport packet a descriptor of a queue, each allocated descriptor containing a pointer to said transport packet to which it is allocated, one or more processing indications and a storage location for control word information,

(c) storing control word information associated with contents of said transport packet in said control word information storage location of selected ones of said allocated descriptors,

(d) setting one or more of said processing indications to indicate that the next step of processing of said sequence may be performed on each of said allocated descriptors,

(e) sequentially accessing each allocated descriptor, and

(f) for each accessed descriptor pointing to a to-be-scrambled transport packet, scrambling said transport packet pointed to by said accessed descriptor using said control word information in said accessed descriptor, only if said one or more processing indications of said accessed descriptor are set to indicate that scrambling processing may be performed on said accessed descriptor and transport packet to which said accessed descriptor points.

FIG. 1

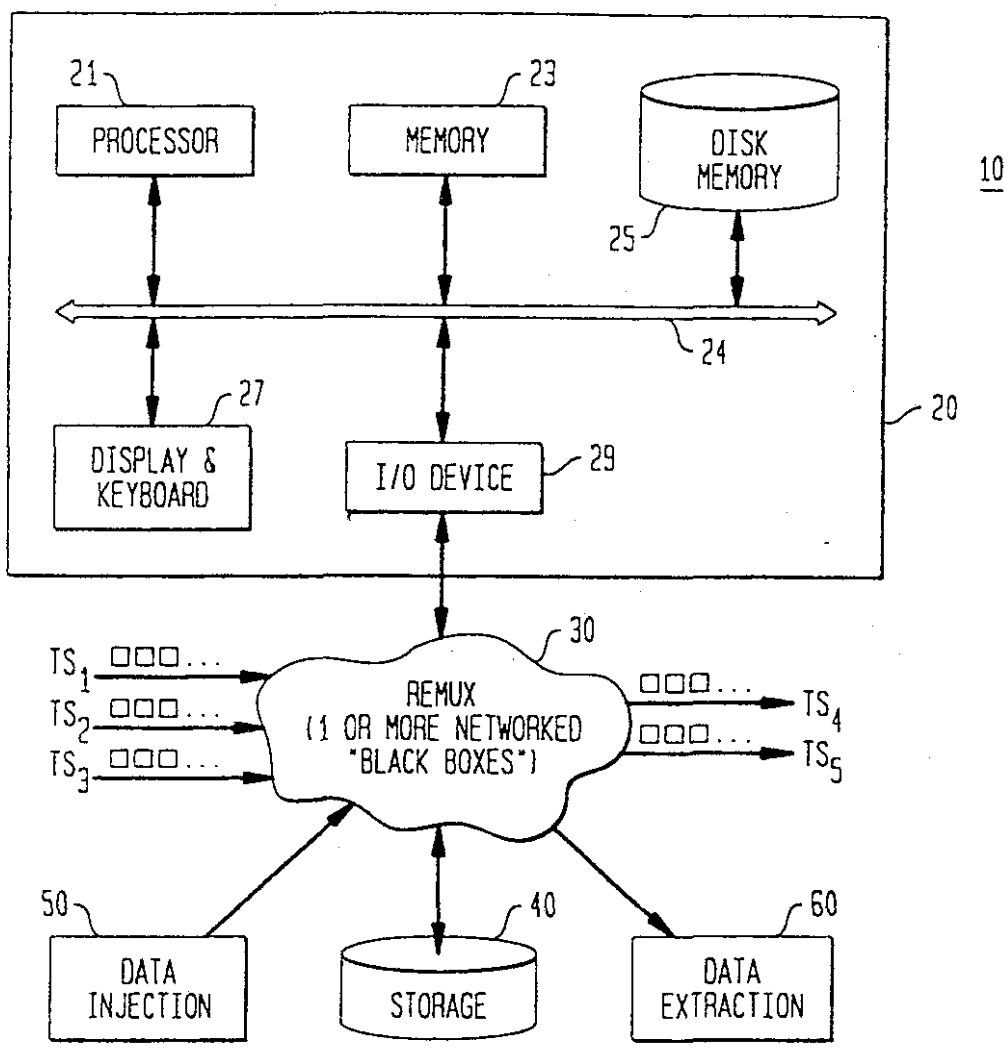

21
PROCESSOR
23
MEMORY
DISK MEMORY
25
10
24
27
DISPLAY & KEYBOARD
I/O DEVICE
29
20
30
TS1
TS2
TS3
REMUX
(1 OR MORE NETWORKED "BLACK BOXES")
TS4
TS5
50
DATA INJECTION
40
STORAGE
60
DATA EXTRACTION

FIG. 4

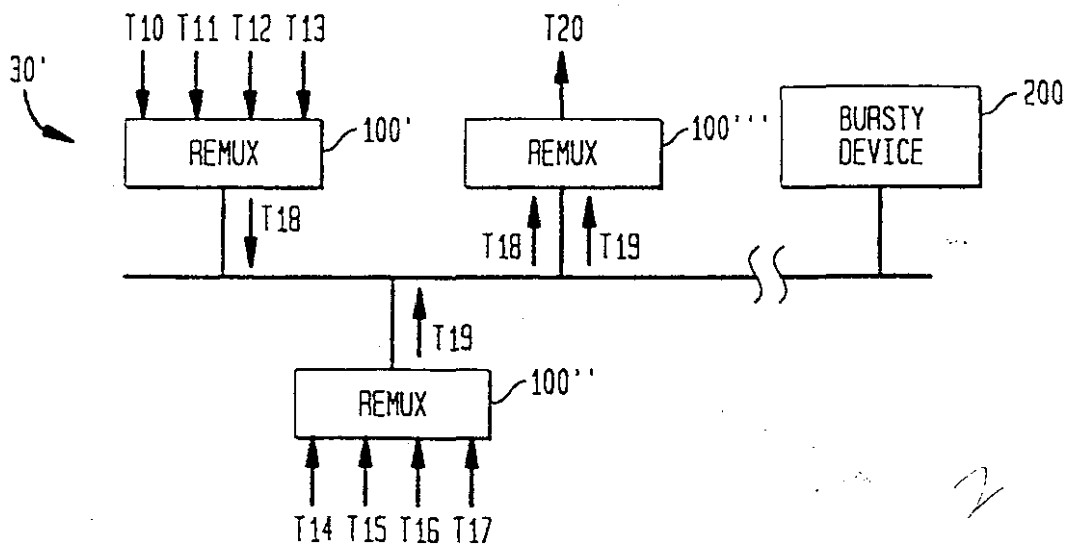

T10 T11 T12 T13
T20
30'
REMUX
100'
REMUX
100'''
BURSTY DEVICE
200
T18
T18
T19
T19
REMUX
100''
T14 T15 T16 T17

FIG. 2

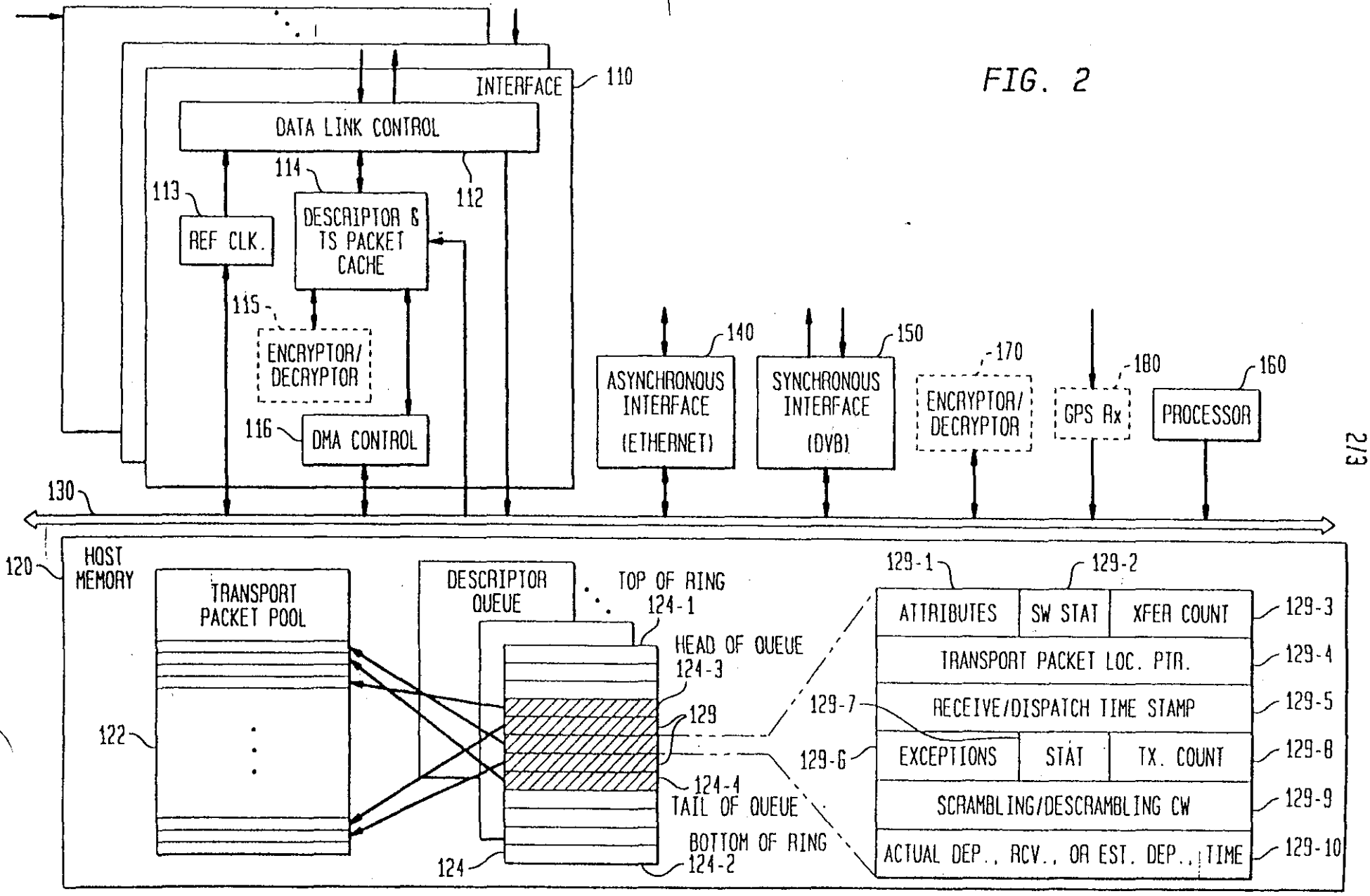
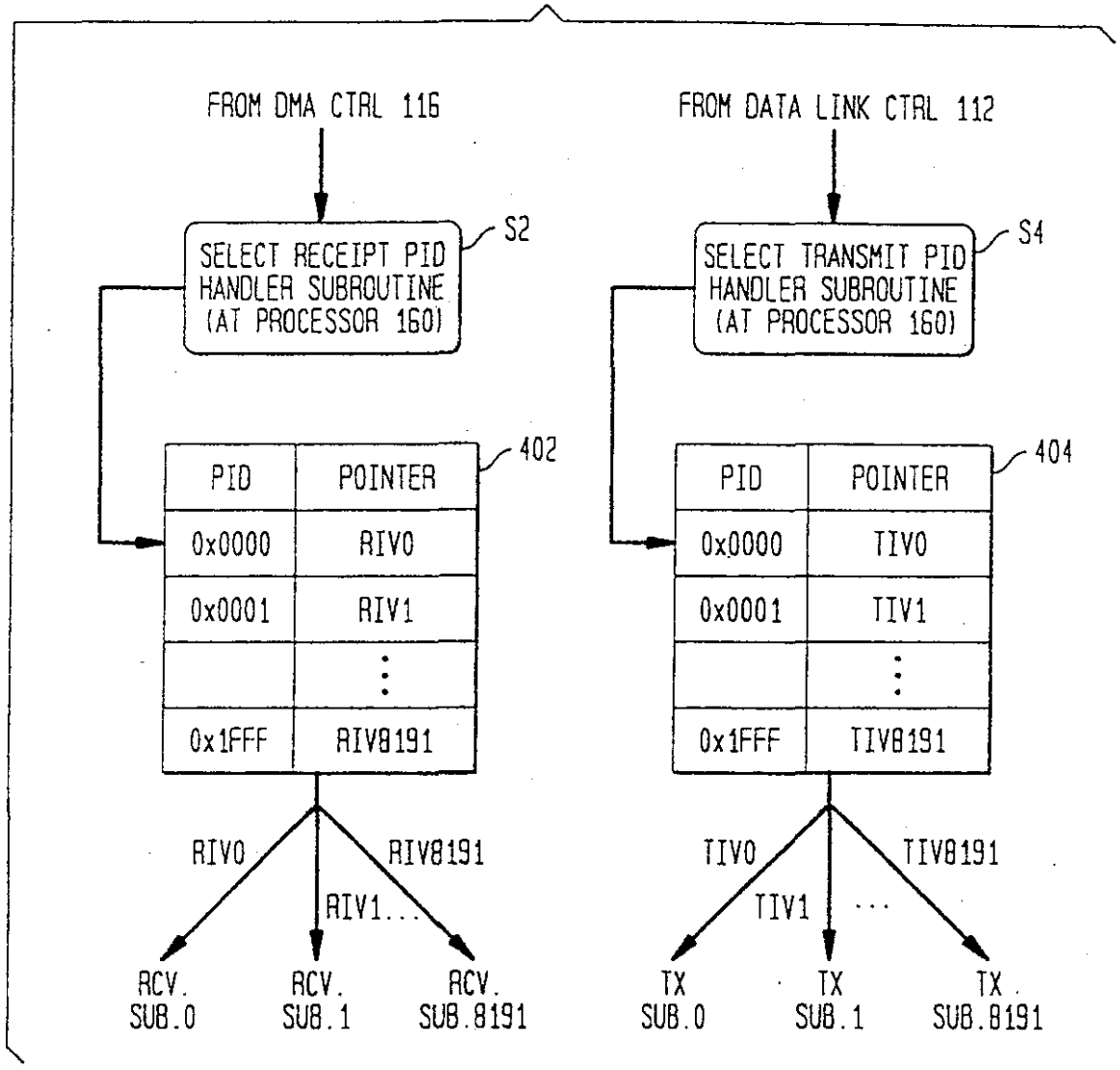

INTERFACE
110
DATA LINK CONTROL
112
113
REF CLK.
114
DESCRIPTOR & TS PACKET CACHE
115
ENCRYPTOR/ DECRYPTOR
116
DMA CONTROL
140
ASYNCHRONOUS INTERFACE (ETHERNET)
150
SYNCHRONOUS INTERFACE (DVB)
170
ENCRYPTOR/ DECRYPTOR
180
GPS Rx
160
PROCESSOR
130
120
HOST MEMORY
TRANSPORT PACKET POOL
122
DESCRIPTOR QUEUE
TOP OF RING
124-1
HEAD OF QUEUE
124-3
129
124-4
TAIL OF QUEUE
BOTTOM OF RING
124-2
124
129-1
129-2
ATTRIBUTES
SW STAT
XFER COUNT
129-3
TRANSPORT PACKET LOC. PTR.
129-4
RECEIVE/DISPATCH TIME STAMP
129-5
129-7
EXCEPTIONS
STAT
TX. COUNT
129-8
129-6
SCRAMBLING/DESCRAMBLING CW
129-9
ACTUAL DEP., RCV., OR EST. DEP., TIME
129-10

FIG. 3


FROM DMA CTRL 116
FROM DATA LINK CTRL 112
SELECT RECEIPT PID HANDLER SUBROUTINE (AT PROCESSOR 160)
S2
SELECT TRANSMIT PID HANDLER SUBROUTINE (AT PROCESSOR 160)
S4
PID
POINTER
0x0000
RIV0
0x0001
RIV1
0x1FFF
RIV8191
402
TIV0
TIV1
TIV8191
404
RCV. SUB.0
RCV. SUB.1
RCV. SUB.8191
TX SUB.0
TX SUB.1
TX SUB.8191

[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
H04J 3/24
H04L 9/18 H04L 12/56
H04K 1/00 H04N 7/10
H04N 7/12 H04N 7/52

[12]发明专利申请公开说明书

[21]申请号 99803994.2

[43]公开日 2001年5月2日

[11]公开号 CN 1293845A

[22]申请日 1999.1.7 [21]申请号 99803994.2
[30]优先权
[32]1998.1.14 [33]US [31]09/007,211
[32]1998.1.14 [33]US [31]09/007,212
[32]1998.1.14 [33]US [31]09/007,334
[32]1998.1.14 [33]US [31]09/007,203
[32]1998.1.14 [33]US [31]09/007,204
[32]1998.1.14 [33]US [31]09/007,210
[32]1998.1.14 [33]US [31]09/006,963
[32]1998.1.14 [33]US [31]09/006,964
[32]1998.1.14 [33]US [31]09/007,198
[32]1998.1.14 [33]US [31]09/007,199
[86]国际申请 PCT/US99/00360 1999.1.7
[87]国际公布 WO99/37048 英 1999.7.22
[85]进入国家阶段日期 2000.9.14

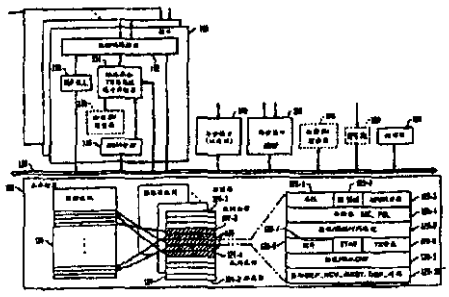
[71]申请人 天溪有限公司
地址 美国加利福尼亚州
[72]发明人 R·格拉塔盖普 W·斯莱特里
R·罗比内特
[74]专利代理机构 上海专利商标事务所
代理人 洪 玲

权利要求书35页 说明书54页 附图页数3页

[54]发明名称 带有视频程序的传输流再分多路复用器

[57]摘要

一种方法和系统(30,30',100,100',100",100"')使用即时地输出 传输包的基于 描述符的系统(122,124,129—4),使用用于对来自任何异步处理 (160,120,130,S2,402,S4,404)的传输包的同步接收和发送去耦合的描 述符和传输 包高速缓存技术(116,122,124,114),使用用于管理加扰和解扰控制字(129 - 9)、通过以传输包数据替换空传输包来优化传输流的带宽的描述符以及使用用于锁 定 多个内部基准时钟发生器(113)的技术(180),对带有视频程序的数据(T S1 - TS5, TS10 - TS20)进行再分多路复用。



ISSN1008-4274

权 利 要 求 书

1. 一种用于优化传输流的带宽的方法，其特征在于包括以下步骤：

(a) 以预定的位速率接收一传输流，所述传输流包括带有可变压缩程序数据的传输包和一个或多个空传输包，所述空传输包中的每一个插入所述接收到的传输流的一时隙中，从而当不能获得在所述传输包时隙处插入所述接收到的传输流中的带有所述压缩程序数据的传输包时保持所述传输流的所述预定位速率，以及

(b) 以带有另一待再分多路复用数据的传输包来选择性地替换一个或多个所述空传输包。

2. 如权利要求 1 所述的方法，其特征在于带有所述另一待再分多路复用数据的传输包包含程序专用信息。

3. 如权利要求 1 所述的方法，其特征在于带有所述另一待再分多路复用数据的传输包包含对以连续方式呈现信息而没有位速率或发送等待时间要求的事务处理数据。

4. 如权利要求 1 所述的方法，其特征在于还包括以下步骤：

(c) 提取所述接收到的传输流中被选中的所述传输包，并丢弃每个未选中的传输包，丢弃每个所述空传输包，

(d) 存储所述选中的传输包，

(e) 存储带有至少一个其它数据的传输包，

(f) 调度每个所述存储的传输包，以在输出的传输流中输出，以及

(g) 以一相应于所述调度的时隙来输出每个所述存储的传输包。

5. 如权利要求 4 所述的方法，其特征在于还包括以下步骤：

(h) 在对所述存储的传输包中相应的一个进行调度的所述输出传输流的每个时隙处，在所述调度的时隙输出所述相应的被存储传输包，以及

(i) 如果在所述时隙中的一个处没有被调度输出的传输包，则输出一空传输包，

其中与在步骤(a)中接收到的每个传输流中占据的所述空传输包相比，所述输出的传输流的所述空传输包占据所述输出传输流的较少带宽。

6. 如权利要求 3 所述的方法，其特征在于所述步骤(b)还包括把带有数据的传输包选择性地指派给所述输出传输流的时隙，从而调整至一接收器缓

冲器的带有所述数据的传输包的发送位速率。

7. 一种用于优化传输流的带宽的再分多路复用器，其特征在于包括：

第一接口，用于以预定的位速率接收一传输流，所述传输流包括带有可变压缩程序数据的传输包和一个或多个空传输包，所述空传输包中的每一个插入所述接收到的传输流的一时隙中，从而当不能获得在所述传输包时隙处插入所述接收到的传输流中的带有所述压缩程序数据的传输包时保持所述传输流的所述预定位速率，以及

处理器，用于以带有另一待再分多路复用数据的传输包来选择性地替换一个或多个所述空传输包。

8. 如权利要求 7 所述的再分多路复用器，其特征在于带有所述另一待再分多路复用数据的传输包包含程序专用信息。

9. 如权利要求 7 所述的再分多路复用器，其特征在于带有所述另一待再分多路复用数据的传输包包含对以连续方式呈现信息而没有位速率或发送等待时间要求的事务处理数据。

10. 如权利要求 7 所述的再分多路复用器，其特征在于所述第一接口和所述处理器提取所述接收到的传输流中被选中的所述传输包，并丢弃每个未选中的传输包，丢弃每个所述空传输包，所述多路复用器还包括：

存储器，在所述存储器中，所述第一接口和所述处理器存储所述选中的传输包以及存储带有至少一个其它数据的传输包，所述处理器调度每个所述存储的传输包，以在输出的传输流中输出，以及

第二接口，用于以一相应于所述调度的时隙来输出每个所述存储的传输包。

11. 如权利要求 10 所述的再分多路复用器，其特征在于在对所述存储的传输包中相应的一个进行调度的所述输出传输流的每个时隙处，所述第二接口输出在所述时隙调度的所述相应的被存储传输包，如果在所述时隙中的一个处没有被调度输出的传输包，则输出一空传输包，与占据在接收到的每个传输流中的所述空传输包相比，所述输出的传输流的所述空传输包占据所述输出传输流的较少带宽。

12. 如权利要求 9 所述的再分多路复用器，其特征在于带有所述处理器把数据的传输包选择性地指派给所述输出传输流的时隙，从而调整至一接收器缓冲器的带有所述数据的传输包的发送位速率。

13. 一种通过以下步骤产生的带宽优化传输流，其特征在于：

(a) 以预定的位速率接收一传输流，所述传输流包括带有可变压缩程序数据的传输包和一个或多个空传输包，所述空传输包中的每一个插入所述接收到的传输流的一时隙中，从而当不能获得在所述传输包时隙处插入所述接收到的传输流中的带有所述压缩程序数据的传输包时保持所述传输流的所述预定位速率，以及

(b) 以带有另一待再分多路复用数据的传输包来选择性地替换一个或多个所述空传输包。

14. 通过以下的进一步步骤产生的如权利要求 13 所述的带宽优化位流，其特征在于：

(c) 提取所述接收到的传输流中被选中的所述传输包，并丢弃每个未选中的传输包，丢弃每个所述空传输包，

(d) 存储所述选中的传输包，

(e) 存储带有至少一个其它数据的传输包，

(f) 调度每个所述存储的传输包，以在输出的传输流中输出，以及

(g) 以一相应于所述调度的时隙来输出每个所述存储的传输包。

15. 通过以下的进一步步骤产生的如权利要求 13 所述的带宽优化传输流，其特征在于：

(h) 在对所述存储的传输包中相应的一个进行调度的所述输出传输流的每个时隙处，输出在所述时隙调度的所述相应的被存储传输包，以及

(i) 如果在所述时隙中的一个处没有被调度输出的传输包，则输出一空传输包，

其中与占据在步骤(a)中接收到的每个传输流中的所述空传输包相比，所述输出的传输流的所述空传输包占据所述输出传输流的较少的带宽。

16. 如权利要求 13 所述的带宽优化传输流，其特征在于所述步骤(b)还包括把带有数据的传输包选择性地指派给所述输出传输流的时隙，从而调整至一接收器缓冲器的带有所述数据的传输包的发送位速率。

17. 一种用于对传输包进行再分多路复用的方法，这些传输包包括包含一个或多个视频程序的压缩数据的传输包，所述传输包所包含的压缩数据的每个所述视频程序包括一恒定端-端通信延迟要求、一独立的位速率以及使所述视频程序的解码和呈现同步的一独立编码器系统定时时钟的程序时钟基准

时间标记，所述方法包括以下步骤：

(a) 接收来自一特定输入端口的传输包，

(b) 把一未使用的描述符分配给所述接收到的传输包，以及

(c) 把一接收时间标记记录在所述分配的描述符中，以指示接收到所述传输包的时间，

其中按照从所述特定输入端口接收到的顺序把所述分配的描述符保持在一有关所述输入端口的接收队列中。

18. 如权利要求 17 所述的方法，其特征在于还包括依据所述接收时间标记和所述传输包的接收与所述传输包的输出之间的内部缓冲延迟来调度所述接收到的传输包的发送的步骤。

19. 如权利要求 17 所述的方法，其特征在于还包括以下步骤：

(d) 检查所述接收队列中的每个描述符，

(e) 分配有关一输出端口的发送队列的描述符，将从所述输出端口中输出每个被检查的描述符指向的传输包，如果有的话

(g) 把一派送时间分配给所述发送队列的所述分配描述符，以及

(h) 按照派送时间增加的顺序对所述发送队列的所述描述符进行排序。

20. 如权利要求 19 所述的方法，其特征在于还包括以下步骤：

(i) 从所述输出端口发送所述发送队列中的一相应描述符所指向的每个传输包，所述发送是在相应于指派给所述相应描述符的所述派送时间的一输出传输流的时隙中进行的。

21. 一种用于对传输包进行再分多路复用的方法，这些传输包包括包含一个或多个视频程序的压缩数据的传输包，所述传输包所包含的压缩数据的每个所述视频程序包括一恒定端-端通信延迟要求、一独立的位速率以及使所述视频程序的解码和呈现同步的一独立编码器系统定时时钟的程序时钟基准时间标记，所述方法包括以下步骤：

(a) 依次检索一发送描述符队列中的每个描述符以及所述被检索的描述符指向的传输包，

(b) 在相应于记录在每个被检索的描述符中的派送时间的时间处，在相应于记录在所述被检索的描述符的所述派送时间的输出传输流的时隙中发送所述被检索的描述符指向的所述被检索的传输包。

22. 如权利要求 21 所述的方法，其特征在于还包括以下步骤：

(c) 检查指向待输出传输包的一个或多个描述符队列中的每个描述符，

(d) 分配有关一输出端口的所述发送队列的描述符，从所述输出端口将发送每个被检查的描述符指向的传输包，如果有的话

(e) 把一派送时间指派给所述发送队列的所述分配的描述符，以及

(f) 按照派送时间增加的顺序对所述发送队列的所述描述符进行排序。

23. 一种用于对传输包进行再分多路复用的再分多路复用器，这些传输包包括包含一个或多个视频程序的压缩数据的传输包，所述传输包所包含的压缩数据的每个所述视频程序包括一恒定端-端通信延迟要求、一独立的位速率以及使所述视频程序的解码和呈现同步的一独立编码器系统定时时钟的程序时钟基准时间标记，所述再分多路复用器包括：

高速缓冲存储器，

连到所述高速缓冲存储器的数据链路控制电路，用于接收来自一特定输入端口的传输包；把所述高速缓冲存储器中一未使用的描述符分配给所述接收到的传输包；以及把一接收时间标记记录在所述分配的描述符中，以指示接收到所述传输包的时间，

其中按照从所述特定输入端口接收到的顺序把所述分配的描述符保持在一与所述输入端口有关的接收队列中。

24. 如权利要求 23 所述的再分多路复用器，其特征在于还包括一处理器，用于依据所述接收时间标记和所述传输包的接收与所述传输包的输出之间的内部缓冲延迟来调度所述接收到的传输包的发送。

25. 如权利要求 23 所述的再分多路复用器，其特征在于还包括一处理器，用于检查所述接收队列中的每个描述符；分配有关一输出端口的发送队列的描述符，从所述输出端口中将输出每个被检查的描述符指向的传输包，如果有的话；把一派送时间分配给所述发送队列的所述分配描述符；以及按照派送时间增加的顺序对所述发送队列的所述描述符进行排序。

26. 如权利要求 25 所述的再分多路复用器，其特征在于还包括：

第二数据链路控制电路，用于从所述输出端口发送所述发送队列中的一相应描述符所指向的每个传输包，所述发送是在相应于指派给所述相应描述符的所述派送时间的一输出传输流的时隙中进行的。

27. 一种用于对传输包进行再分多路复用的再分多路复用器，这些传输包包括包含一个或多个视频程序的压缩数据的传输包，所述传输包所包含的

压缩数据的每个所述视频程序包括一恒定端-端通信延迟要求、一独立的位速率以及使所述视频程序的解码和呈现同步的一独立编码器系统定时时钟的程序时钟基准时间标记，所述再分多路复用器包括：

高速缓冲存储器，

连到所述高速缓冲存储器的数据链路控制电路，用于依次检索一发送描述符队列中的每个描述符以及所述被检索的描述符指向的传输包；以及在相应于记录在每个被检索的描述符中的派送时间的时间处，在相应于记录在所述被检索的描述符的所述派送时间的输出传输流的时隙中发送所述被检索的描述符指向的所述被检索的传输包。

28. 如权利要求 27 所述的再分多路复用器，其特征在于还包括：

一处理器，用于检查指向待输出传输包的一个或多个描述符队列中的每个描述符；分配与一输出端口有关的所述发送队列的描述符，从所述输出端口将发送每个被检查的描述符指向的传输包，如果有的话；把一派送时间指派给所述发送队列的所述分配的描述符；以及按照派送时间增加的顺序对所述发送队列的所述描述符进行排序。

29. 一种包含传输包的传输流，这些传输包包括包含一个或多个视频程序的压缩数据的传输包，所述传输包所包含的压缩数据的每个所述视频程序包括一恒定端-端通信延迟要求、一独立的位速率以及使所述视频程序的解码和呈现同步的一独立编码器系统定时时钟的程序时钟基准时间标记，所述传输流是通过以下步骤产生的：

(a) 接收来自一特定输入端口的传输包，

(b) 把一未使用的描述符分配给所述接收到的传输包，以及

(c) 把一接收时间标记记录在所述分配的描述符中，以指示接收到所述传输包的时间，

其中按照从所述特定输入端口接收到的顺序把所述分配的描述符保持在一与所述输入端口有关的接收队列中。

30. 一种包含传输包的传输流，这些传输包包括包含一个或多个视频程序的压缩数据的传输包，所述传输包所包含的压缩数据的每个所述视频程序包括一恒定端-端通信延迟要求、一独立的位速率以及使所述视频程序的解码和呈现同步的一独立编码器系统定时时钟的程序时钟基准时间标记，所述传输流是通过以下步骤产生的：

(a) 依次检索一发送描述符队列中的每个描述符以及所述被检索的描述符指向的传输包，

(b) 在相应于记录在每个被检索的描述符中的派送时间的时间处，在相应于记录在所述被检索的描述符的所述派送时间的输出传输流的时隙中发送所述被检索的描述符指向的所述被检索的传输包。

31. 一种用于对带有一个或多个程序的传输包进行再分多路复用的方法，每个程序包括一个或多个流元，每个传输流包括传输包，这些传输包包括携带有一个或多个程序的流元数据的传输包，所述方法包括以下步骤：

(a) 依据再分多路复用传输流内容的初始用户规定，仅从带有每个所述程序的传输流中选择性地提取特定的一些所述传输包，

(b) 依据再分多路复用传输流内容的所述初始用户规定，把所述提取的传输包中所述选中的传输包和包含程序专用信息的传输包，如果有的话，重新装配成为输出的再分多路复用传输流，

(c) 把所述重新装配的再分多路复用传输流作为一连续位流输出，

(d) 在执行所述步骤(a)、(b)和(c)的同时，动态地接收指定以下一个或多个的再分多路复用传输流内容的一个或多个新的用户规定：

(I) 在所述步骤(a)中待提取的不同传输包，

(II) 在所述步骤(b)中待重新装配的不同传输包，以及

(e) 响应于接收到所述一个或多个新的用户规定，动态地停止依据所述初始用户规定提取或重新装配传输包，并动态地开始依据所述新的用户规定提取或重新装配传输包，而不在所述输出的再分多路复用传输流中引入不连续。

32. 如权利要求 31 所述的方法，其特征在于还包括以下步骤：

(f) 通过产生引用一新的用户规定的所述不同传输包的替换程序专用信息，来响应于在步骤(b)中重新装配不同传输包的所述新用户规定。

33. 如权利要求 31 所述的方法，其特征在于还包括：

(f) 接收所述初始用户规定和每个新的用户规定，

(g) 确定依据所述接收到的用户规定中的每一个重新装配的所述再分多路复用传输流的总位速率要求，

(h) 仅在所述确定的位速率要求小于或等于所述输出的再分多路复用传输流的位速率时才执行步骤(a)、(b)和(e)。

34. 如权利要求 31 所述的方法，其特征在于还包括以下步骤：

(f) 连续地识别可获得的装配成所述输出的再分多路复用传输流的流，以及

(g) 向用户提示一新的用户规定，所述新的用户规定指定所述被识别的可获得流的选择作为所述再分多路复用传输流的所述内容。

35. 如权利要求 31 所述的方法，其特征在于所述新的用户规定指定在所述步骤(b)中重新装配的一个或多个传输包的包标识符的新映射，所述步骤(e)包括依据所述新映射来映射所述一个或多个传输包的包标识符。

36. 如权利要求 31 所述的方法，其特征在于所述新的用户规定指定对一个或多个特定流元进行加扰，所述方法还包括以下步骤：

(a) 使用控制字对所述被指定的流元的所述传输包进行加扰，

(b) 提供包含所述控制字的传输包，以重新装配成所述再分多路复用的传输流，以及

(c) 产生包含程序专用信息的传输包，所述信息识别哪些传输包包含所述控制字以及带有所述控制字的传输包相应于哪些流元。

37. 一种用于把一个或多个输入的传输流的传输包再分多路复用成为一输出传输流的方法，所述输入传输流中的至少一个包含一个或多个程序和程序定义，每个所述程序包括一个或多个流元，所述至少一个输入传输流中的每一个包括程序定义，所述程序定义识别哪些传输包包含所述输入传输流中所包含的每个流元的流元数据以及所述流元中的哪些构成包含在所述输入传输流中的每个程序，所述方法包括以下步骤：

(a) 产生一用户规定，所述用户规定指示将在所述输出传输流中输出的所述输入传输流的一个或多个程序，

(b) 连续地获取所述程序定义，

(c) 从所述获取的程序定义中连续地确定哪些流元构成每个程序，以及

(d) 在所述输出的传输流中输出每个传输包，而不会在所述输出的传输流中引入不连续，每个所述传输包包含每个流元的流元数据，每个所述流元在所述步骤(c)中被确定为构成被指示在所述用户规定中输出的每个程序。

38. 一种用于对带有一个或多个程序的传输流进行再分多路复用的再分多路复用器，所述程序包括一个或多个流元，每个传输流包括传输包，这些传输包包括携带一个或多个程序的流元数据的传输包，所述方法包括：

第一接口，依据对再分多路复用传输流内容的初始用户规定，仅从带有每个所述程序的传输流中选择性地提取特定的一些所述传输包，

第二接口，依据再分多路复用传输流内容的所述初始用户规定，把所述提取的传输包中所述选中的传输包和包含程序专用信息的传输包，如果有的话，重新装配成为输出的再分多路复用传输流，以及用于把所述重新装配的再分多路复用传输流作为一连续位流输出，以及

处理器，用于动态地接收指定以下一个或多个的再分多路复用传输流内容的一个或多个新的用户规定：

(I) 将由所述第一接口提取的不同传输包，

(II) 将由所述第二接口重新装配的不同传输包，

在所述第一和第二接口提取传输包和重新装配以及输出所述再分多路复用传输流的同时，响应于接收到所述一个或多个新的用户规定，动态地停止依据所述初始用户规定提取或重新装配传输包，并动态地开始依据所述新的用户规定提取或重新装配传输包，而不在所述输出的再分多路复用传输流中引入不连续。

39. 如权利要求 38 所述的再分多路复用器，其特征在于所述处理器通过产生引用使所述第二接口进行重新装配的新用户规定的所述不同传输包的替换程序专用信息，来响应于重新装配不同传输包用的所述新用户规定。

40. 如权利要求 38 所述的再分多路复用器，其特征在于还包括：

控制器，用于接收所述初始用户规定和每个新的用户规定；确定依据所述接收到的用户规定中的每一个重新装配的所述再分多路复用传输流的总位速率要求；以及仅在所述确定的位速率要求小于或等于所述输出的再分多路复用传输流的位速率时才使所述第一和第二接口依据所述新用户接口中的每一个来进行提取和重新装配。

41. 如权利要求 38 所述的再分多路复用器，其特征在于所述处理器连续地识别可获得的装配成所述输出的再分多路复用传输流的流，所述在多路复用器还包括：

控制器，用于向用户提示一新的用户规定，所述新的用户规定指定所述被识别的可获得流的选择作为所述再分多路复用传输流的所述内容。

42. 如权利要求 38 所述的再分多路复用器，其特征在于所述新的用户规定指定由所述第二接口重新装配的一个或多个传输包的包标识符的新映射，

所述处理器依据所述新映射来映射所述一个或多个传输包的包标识符。

43. 如权利要求 38 所述的再分多路复用器，其特征在于所述新的用户规定指定对一个或多个特定流元进行加扰，所述再分多路复用器还包括：

加扰器，用于使用控制字对所述被指定的流元的所述传输包进行加扰，

其中，所述处理器获得包含重新装配成所述再分多路复用传输流用的所述控制字的传输包以及包含程序专用信息的传输包，所述信息识别哪些传输包包含所述控制字以及所述控制字相应于哪些流元。

44. 一种用于把一个或多个输入的传输流的传输包再分多路复用成为一输出传输流的再分多路复用器，所述输入传输流中的至少一个包含一个或多个程序和程序定义，每个所述程序包括一个或多个流元，所述至少一个输入传输流中的每一个包括程序定义，所述程序定义识别哪些传输包包含所述输入传输流中所包含的每个流元的流元数据以及所述流元中的哪些构成包含在所述输入传输流中的每个程序，所述再分多路复用器包括：

控制器，用于产生一用户规定，所述用户规定指示将在所述输出传输流中输出的所述输入传输流的一个或多个程序，

第一适配器，用于连续地获取所述程序定义，

处理器，用于从所述获取的程序定义中连续地确定哪些流元构成每个程序，以及

第二适配器，在所述输出的传输流中输出每个传输包，而不会在所述输出的传输流中引入不连续，每个所述传输包包含每个流元的流元数据，每个所述流元被确定为构成被指示在所述用户规定中输出的每个程序。

45. 一种由带有一个或多个程序的传输流再分多路复用而成的输出再分多路复用传输流，每个程序包括一个或多个流元，每个传输流包括传输包，这些传输包包括携带有一个或多个程序的流元数据的传输包，所述输出多路复用传输流是通过以下步骤产生的：

(a) 依据再分多路复用传输流内容的初始用户规定，仅从带有每个所述程序的传输流中选择性地提取特定的一些所述传输包，

(b) 依据再分多路复用传输流内容的所述初始用户规定，把所述提取的传输包中所述选中的传输包和包含程序专用信息的传输包，如果有的话，重新装配成为输出的再分多路复用传输流，

(c) 把所述重新装配的再分多路复用传输流作为一连续位流输出，

(d) 在执行所述步骤(a)、(b)和(c)的同时，动态地接收指定以下一个或多个的再分多路复用传输流内容的一个或多个新的用户规定：

(I) 在所述步骤(a)中待提取的不同传输包，

(II) 在所述步骤(b)中待重新装配的不同传输包，以及

(e) 响应于接收到所述一个或多个新的用户规定，动态地停止依据所述初始用户规定提取或重新装配传输包，并动态地开始依据所述新的用户规定提取或重新装配传输包，而不在所述输出的再分多路复用传输流中引入不连续。

46. 一种由一个或多个输入传输流再分多路复用而成的输出传输流，所述输入传输流中的至少一个包含一个或多个程序和程序定义，每个所述程序包括一个或多个流元，所述至少一个输入传输流中的每一个包括程序定义，所述程序定义识别哪些传输包包含所述输入传输流中所包含的每个流元的流元数据以及所述流元中的哪些构成包含在所述输入传输流中的每个程序，所述方法包括以下步骤：

(a) 产生一用户规定，所述用户规定指示将在所述输出传输流中输出的所述输入传输流的一个或多个程序，

(b) 连续地获取所述程序定义，

(c) 从所述获取的程序定义中连续地确定哪些流元构成每个程序，以及

(d) 在所述输出的传输流中输出每个传输包，而不会在所述输出的传输流中引入不连续，每个所述传输包包含每个流元的流元数据，每个所述流元在所述步骤(c)中被确定为构成被指示在所述用户规定中输出的每个程序。

47. 一种用于把带有第一视频程序的位流多路复用成为第二位流的方法，带有所述第一视频程序的位流包含用于其中所包含的每个程序的一组多个时间标记，所述时间标记指示与所述程序的每个包应出现在所述第一位流中的一编码器的系统定时时钟相对的时间，包括以下步骤：

(a) 从一具有可变端-端发送延迟的通信链路接收带有所述第一视频程序的位流，

(b) 根据从带有所述第一视频程序的位流接收到的所述程序的多个时间标记，确定携带有从带有所述第一视频程序的位流接收到的同一程序的数据的一个或多个包中每一个包应出现在所述第二位流中的时间，以及

(c) 在与所述确定的时间有关的时间，以恒定的端-端延迟在所述第二位

流中选择性地发送所述一个或多个包中被选中的包。

48. 如权利要求 47 所述的方法，其特征在于所述步骤(b)还包括以下步骤：

(b1) 把包含从带有所述接收到的第一视频程序的位流中接收到的数据的包存储在一接收队列中，

(b2) 在包含存储在所述接收队列中的程序的相继时间标记的第一和第二特定包之间，识别包含所述程序的数据的每个包，

(b3) 根据所述第一和第二时间标记之差来确定所述程序的包速率，以及

(b4) 把指派给所述第一特定包的发送时间同所述包速率与所述被识别包离开所述第一包的偏移之积的总和指派为所述被识别包中每一个的发送时间。

49. 如权利要求 48 所述的方法，其特征在于还包括以下步骤：

(b5) 给相对于所述第一位流中所携带的每个程序接收到的带有第一时间标记的包指派相对于一本地时钟的接收时间，以及

(b6) 把所述指派的接收时间与一已知的缓冲延迟的总和指派为包含带有所述第一时间标记的包的数据的包的发送时间。

50. 如权利要求 47 所述的方法，其特征在于所述步骤(c)还包括以下步骤：

(c1) 通过在与所述确定的时间有关的所述时间处把所述被识别的包插入所述第二位流来防止所述第二位流的接收器处的缓冲器上溢和下溢。

51. 如权利要求 50 所述的方法，其特征在于所述接收器缓冲器依据相应于所述程序的可变压缩部分的时间标记以及所述程序恢复的系统定时时钟，把所述被识别的包从所述第二位流中除去，带有所述第一视频程序的位流的所述可变压缩部分的位数与所述接收器缓冲器的预设存储容量以及所述第一视频程序的预定位速率有关。

52. 如权利要求 51 所述的方法，其特征在于所述步骤(c)还包括以下步骤：

(c1) 确定所述第二位流中在时间上最接近一包的所述预定发送时间的包时隙，

(c2) 如果不止一个包在传输时间上最接近所述包时隙中的单个包时隙，则把在时间上最接近所述单个包时隙的每个所述包指派给连续包时隙，以及

(c3) 调节带有时间标记的每个包的时间标记，根据所述被指派的包时隙偏离所述单个包时隙的包时隙数，把每个所述包指派给所述单个包时隙以外的所述包时隙之一。

53. 如权利要求 52 所述的方法，其特征在于把每个所述被选中的接收到的包插入一未决发送的队列，所述步骤(c3)还包括以下步骤：

(c4) 估计本地时钟与编码器的一个或多个系统定时时钟中每一个时钟的偏移，所述编码器按所述队列的当前队列长度延迟与所述队列的理想队列长度延迟之差的函数来产生所述接收到的包，以及

(c5) 依据所述本地时钟与产生所述包的所述编码器的所述系统定时时钟之间的所述偏移中相应的一个偏移，进一步调节每个所述被调节的时间标记。

54. 如权利要求 47 所述的方法，其特征在于还包括以下步骤：

(d) 从一计算机网络接收带有所述第一视频程序的位流。

55. 如权利要求 47 所述的方法，其特征在于还包括以下步骤：

(d) 从一以太网接收带有所述第一视频程序的位流。

56. 如权利要求 47 所述的方法，其特征在于还包括以下步骤：

(d) 从一 ATM 网络接收带有所述第一视频程序的位流。

57. 一种用于把带有第一视频程序的位流多路复用成为第二位流的再分多路复用器，带有所述第一视频程序的位流包含用于其中所包含的每个程序的一组多个时间标记，所述时间标记指示与所述程序的每个包应出现在所述第一位流中的一编码器的系统定时时钟相对的时间，包括：

异步接口，用于从一具有可变端-端发送延迟的通信链路接收带有所述第一视频程序的位流，

连到所述异步接口的处理器，用于根据从带有所述第一视频程序的位流接收到的所述程序的多个时间标记，确定携带有从带有所述第一视频程序的位流接收到的同一程序的数据的一个或多个包中每一个包应出现在所述第二位流中的时间，以及

同步接口，用于在与所述确定的时间有关的时间，以恒定的端-端延迟在所述第二位流中选择性地发送所述一个或多个包中被选中的包。

58. 如权利要求 57 所述的再分多路复用器，其特征在于还包括：

存储器，用于把包含从带有所述接收到的第一视频程序的位流中接收到的数据的包存储在一接收队列中，

其中，所述处理器在包含存储在所述接收队列中的程序的相继时间标记的第一和第二特定包之间，识别包含所述程序的数据的每个包；根据所述第一和第二时间标记之差来确定所述程序的包速率；以及把指派给所述第一特定包的发送时间同所述包速率与所述被识别包离开所述第一包的偏移之积的总和指派为所述被识别包中每一个的发送时间。

59. 如权利要求 58 所述的再分多路复用器，其特征在于还包括：

所述处理器可访问的时钟，其中所述处理器给相对于所述第一位流中所携带的每个程序接收到的带有第一时间标记的包指派相对于一本地时钟的接收时间，并且把所述指派的接收时间与一已知的缓冲延迟的总和指派为包含带有所述第一时间标记的包的数据的包的发送时间。

60. 如权利要求 67 所述的再分多路复用器：

其特征在于所述处理器在与所述确定的时间有关的所述时间处对所述包的所述发送防止了所述第二位流的接收器处的缓冲器的上溢和下溢。

61. 如权利要求 60 所述的再分多路复用器，其特征在于所述接收器缓冲器依据相应于所述程序的可变压缩部分的时间标记以及所述程序的恢复的系统定时时钟，把所述被识别的包从所述第二位流中除去，带有所述第一视频程序的位流的所述可变压缩部分的位数与所述接收器缓冲器的预设存储容量以及所述第一视频程序的预定位速率有关。

62. 如权利要求 61 所述的再分多路复用器：

其特征在于所述处理器确定所述第二位流中在时间上最接近一包的所述预定发送时间的包时隙，

如果不止一个包在传输时间上最接近所述包时隙中的单个包时隙，则所述处理器把在发送时间上最接近所述单个包时隙的每个所述包指派给连续包时隙，以及

所述处理器调节带有时间标记的每个包的时间标记，根据所述被指派的包时隙偏离所述单个包时隙的包时隙数，把每个所述包指派给所述单个包时隙以外的所述包时隙之一。

63. 如权利要求 62 所述的再分多路复用器，其特征在于还包括：

存储器，其中所述异步接口把每个所述被选中的接收到的包插入所述存储器中一未决发送的队列，

所述处理器估计本地时钟与编码器的一个或多个系统定时时钟中每一个

时钟的偏移，所述编码器按所述队列的当前队列长度延迟与所述队列的理想队列长度延迟之差的函数来产生所述接收到的包，以及

所述处理器依据所述本地时钟与产生所述包的所述编码器的所述系统定时时钟之间的所述偏移中相应的一个偏移，进一步调节每个所述被调节的时间标记。

64. 一种通过把带有第一视频程序的位流多路复用成为第二位流而产生的位流，带有所述第一视频程序的位流包含用于其中所包含的每个程序的一组多个时间标记，所述时间标记指示与所述程序的每个包应出现在所述第一位流中的一编码器的系统定时时钟相对的时间，包括以下步骤：

(a) 从一具有可变端-端发送延迟的通信链路接收带有所述第一视频程序的位流，

(b) 根据从带有所述第一视频程序的位流接收到的所述程序的多个时间标记，确定携带有从带有所述第一视频程序的位流接收到的同一程序的数据的一个或多个包中每一个包应出现在所述第二位流中的时间，以及

(c) 在与所述确定的时间有关的时间，以恒定的端-端延迟在所述第二位流中选择性地发送所述一个或多个包中被选中的包。

65. 一种用于及时地输出带有压缩视频程序数据的位流的方法，包括以下步骤：

(a) 提供一包含传输包的位流，所述传输包包含一个或多个视频程序的压缩程序数据，每个所述程序具有一预定位速率，所述传输包还包含每个所述程序的程序时钟基准时间标记，每个程序的解码和呈现与所述基准时间标记同步，

(b) 给所述传输包中的一个或多个选中传输包中的每一个传输包指派派送时间，以保持其数据被所述传输包携带的程序的预定位速率，以及引起每个所述传输包的平均等待时间，以及

(c) 在与每个所述派送时间有关的时间处，向一异步通信接口发出一个或多个命令以使所述异步通信接口大致在所述派送时间处发送所述相应的选中传输包，从而把所述选中传输包的抖动减到最小。

66. 如权利要求 65 所述的方法，其特征在于还包括以下步骤：

(d) 把一发送描述符分配给每个所述传输包，所述发送描述符按照所述派送时间的顺序驻留在指派给所述异步接口的队列中，

(e) 把所述传输包的每个所述派送时间记录在分配给所述各个传输包的发送描述符中，

(f) 依次检查所述队列中的每个所述描述符的派送时间，

(g) 把所述被检查的派送时间与本地基准时钟所产生的时间相比较，以及

(h) 在所述比较所确定的时间处发出每个命令。

67. 如权利要求 66 所述的方法，其特征在于还包括以下步骤：

(i) 从另一个接口接收所述传输包中的至少一些，

(j) 按照接收到每个所述传输包的时间以及所述接收时间与所述异步接口产生所述输出的所述时间之间的预设缓冲延迟的函数，产生所述派送时间。

68. 如权利要求 66 所述的方法，其特征在于还包括以下步骤：

(i) 选择执行所述步骤(b)、(d)和(e)的发送 PID 处理器子程序，以及

(j) 每当发出所述命令中的一个命令时，尝试重复步骤(b)、(d)和(e)。

69. 如权利要求 65 所述的方法，其特征在于还包括以下步骤：

(d) 接收从所述异步接口发送的所述传输包，所述接收是在一接收节点的另一异步接口处进行的，

(e) 对所述接收节点处的所述接收到的传输包去抖动，以及

(f) 把所述去抖动的传输包中的至少一些再分多路复用成为从所述接收节点输出的第二位流，从而所述第二位流中所携带的每个程序具有连续端-端延迟。

70. 一种用于及时地输出带有压缩视频程序数据的位流的再分多路复用器，包括以下步骤：

同步接口，用于提供一包含传输包的位流，所述传输包包含一个或多个视频程序的压缩程序数据，每个所述程序具有一预定位速率，所述传输包还包含每个所述程序的程序时钟基准时间标记，每个程序的解码和呈现与所述基准时间标记同步，

处理器，用于给所述传输包中的一个或多个选中传输包中的每一个传输包指派派送时间，以保持其数据被所述传输包携带的程序的预定位速率，以及引起每个所述传输包的平均等待时间，以及

异步通信接口，在与每个所述派送时间有关的时间处，向一异步通信接口发出一个或多个命令以使所述异步通信接口大致在所述派送时间处发送所

述相应的选中传输包，从而把所述选中传输包的抖动减到最小。

71. 如权利要求 70 所述的再分多路复用器，其特征在于还包括：

存储器，用于存储一指派给所述异步接口的描述符队列，所述处理器把一发送描述符分配给每个所述传输包，所述发送描述符按照所述派送时间的顺序驻留在所述队列中，所述处理器还把所述传输包的每个所述派送时间记录在分配给所述各个传输包的发送描述符中，以及

输出数据链路控制电路，依次检查所述队列中的每个所述描述符的派送时间；把所述被检查的派送时间与本地基准时钟所产生的时间相比较；以及在所述比较所确定的时间处发出每个命令。

72. 如权利要求 71 所述的再分多路复用器，其特征在于所述同步接口从另一个接口接收所述传输包中的至少一些，所述处理器按照在所述同步接口处接收到每个所述传输包的时间以及所述接收时间与所述异步接口产生所述输出的所述时间之间的预设缓冲延迟的函数，产生所述派送时间。

73. 如权利要求 71 所述的再分多路复用器，其特征在于所述处理器选择用于把派送时间指派给所述传输包、分配描述符以及把所述指派的派送时间记录在所述分配的描述符中的发送 PID 处理器子程序，每当发出所述命令中的一个命令时，所述处理器尝试把派送时间指派给后续的一组所述传输包，把描述符分配给所述后续组中的每个传输包，并把指派给所述传输包的所述后续组的所述派送时间记录在对其所分配的所述描述符中。

74. 如权利要求 70 所述的再分多路复用器，其特征在于所述再分多路复用器包括多个节点，所述再分多路复用器还包括：

位于一接收节点处的第二异步接口，用于接收从所述异步接口发送的所述传输包，

位于所述接收节点处的第二处理器，用于对所述接收节点处的所述接收到的传输包去抖动，以及

位于所述接收节点处的输出异步接口，用于把所述去抖动的传输包中的至少一些再分多路复用成为从所述接收节点输出的第二位流，从而所述第二位流中所携带的每个程序具有连续端-端延迟。

75. 一种包含压缩视频程序数据的位流，所述位流是通过以下步骤产生的：

(a) 提供一包含传输包的位流，所述传输包包含一个或多个视频程序的

压缩程序数据，每个所述程序具有一预定位速率，所述传输包还包含每个所述程序的程序时钟基准时间标记，每个程序的解码和呈现与所述基准时间标记同步，

(b) 给所述传输包中的一个或多个选中传输包中的每一个传输包指派派送时间，以保持其数据被所述传输包携带的程序的预定位速率，以及引起每个所述传输包的平均等待时间，以及

(c) 在与每个所述派送时间有关的时间处，向一异步通信接口发出一个或多个命令以使所述异步通信接口大致在所述派送时间处发送所述相应的选中传输包，从而把所述选中传输包的抖动减到最小。

76. 如权利要求 75 所述的位流，其特征在于还包括以下步骤：

(d) 接收从所述异步接口发送的所述传输包，所述接收是在一接收节点的另一异步接口处进行的，

(e) 对所述接收节点处的所述接收到的传输包去抖动，以及

(f) 把所述去抖动的传输包中的至少一些再分多路复用成为从所述接收节点输出的第二位流，从而所述第二位流具有每个程序中携带的连续端-端延迟。

77. 一种用于在一异步通信网络中对包含压缩程序数据的一个或多个位流进行再分多路复用的方法，所述异步通信网络包括通过一个或多个通信链路互连的多个节点，所述方法包括以下步骤：

(a) 从位于所述异步通信网络的一个目的地节点处的所述通信链路之一接收包含一个或多个程序的数据的第一位流，所述第一位流的一部分具有一个或多个预定位速率，

(b) 选择所述接收到的第一位流中的至少一部分用以发送，以及

(c) 调度所述第一位流中所述选中部分的发送，从而以与所述第一位流的所述选中部分的所述预定速率有关的速率在一传输流中输出所述第一位流的所述选中部分。

78. 一种用于在一通信网络的多个节点处，把位流的一个或多个部分再分多路复用成为包含压缩视频程序数据的一个或多个传输流的方法，所述方法包括以下步骤：

(a) 使能在通过一个或多个各通信链路连到一共享通信媒体的多个节点中进行通信，

(b) 选择所述节点中的一个或多个构成的第一组用以把一个或多个位流发送到所述共享的通信媒体上，

(c) 选择所述节点中的一个或多个构成的第二组用以接收来自所述共享通信媒体的所述发送的位流，用以选择所述发送的位流的部分并发送作为一包含所述选中部分的位流的一个或多个再分多路复用传输流，作为一位流发送的每个所述再分多路复用传输流不同于所述发送位流中所述接收到的位流，以及

(d) 使所述选中的节点依据多个不同的信号流动模式之一经由所述共享通信媒体来传送所述位流，所述信号流动模式包括不同于至所述共享通信媒体的所述节点的拓扑连接的至少一个信号流动模式。

79. 如权利要求 78 所述的方法，其特征在于至少一个节点可经由所述各通信链路中的单个接收来自所述节点中多个其它节点中每一个节点的位流，所述方法还包括选择所述多个其它节点的子组并所述至少一个节点处仅接收来自所述选中的节点子组的位流的步骤。

80. 如权利要求 78 所述的方法，其特征在于至少一个节点经由所述各通信链路中的单个接收来自所述节点中多个其它节点的位流。

81. 一种用于对包含压缩程序数据的一个或多个位流进行再分多路复用的网络分布式再分多路复用器，包括：

一个或多个通信链路，以及

通过所述一个或多个通信链路互连成为一通信网络的多个节点，所述多个节点包括一经由所述通信链路之一接收包含一个或多个程序的数据的第一位流的目的地节点，所述第一位流的一部分具有一个或多个预定位速率，所述目的地节点包括：

处理器，选择所述接收到的第一位流中的至少一部分用以发送，以及调度所述第一位流中所述选中部分的发送，从而以与所述第一位流的所述选中部分的所述预定速率有关的速率在一传输流中输出所述第一位流的所述选中部分。

82. 一种用于把位流的一个或多个部分再分多路复用成为包含压缩视频程序数据的一个或多个传输流的网络分布式再分多路复用器，包括：

一共享通信媒体，包括一个或多个通信链路，

多个节点，每个所述节点通过所述通信链路中的各个或多个连到所述共

享通信媒体，所述多个节点包括：

所述节点中的一个或多个构成的第一组，用于把一个或多个位流发送到所述共享通信媒体上，

所述节点中的一个或多个构成的第二组，用于接收来自所述共享通信媒体的所述发送的位流，选择所述发送的位流的部分并发送作为一包含所述选中部分的位流的一个或多个再分多路复用传输流，作为一位流发送的每个所述再分多路复用传输流不同于所述发送位流中所述接收到的位流，以及

控制器节点，用于选择所述第一和第二组节点，并使所述选中的节点依据多个不同的信号流动模式之一经由所述共享通信媒体来传送所述位流，所述信号流动模式包括不同于至所述共享通信媒体的所述节点的拓扑连接的至少一个信号流动模式。

83. 如权利要求 82 所述的网络分布式再分多路复用器，其特征在于所述多个节点还包括至少一个节点，所述至少一个节点可经由所述各通信链路中的单个接收来自所述节点中多个其它节点中每一个节点的位流，所述控制器节点选择所述多个其它节点的子组，在所述至少一个节点处仅接收来自所述选中的节点子组的位流。

84. 如权利要求 82 所述的网络分布式再分多路复用器，其特征在于所述多个节点包括至少一个节点，所述至少一个节点经由所述各通信链路中的单个接收来自所述节点中多个其它节点的位流。

85. 一种用于锁定电路处的基准时钟的方法，所述电路发送和接收由包含一个或多个程序的压缩数据的传输包序列形成的传输流，每个所述程序具有一独立的位速率和一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述方法包括以下步骤：

(a) 保持接收传输包的每个第一电路以及发送传输包的每个第二电路处的基准时钟，所述第一电路处的所述基准时钟用于指示在该处接收到每个传输包的时间，每个第二电路处的所述基准时钟用于指示何时从中发送每个传输包，

(b) 指定一主基准时钟，每个其它的所述基准时钟将与所述主基准时钟同步，

(c) 周期性地获得所述主基准时钟的当前时间，以及

(d) 依据每个所述其它基准时钟处的所述时间与所述主基准时钟的所述

当前时间之差来调节每个其它的所述基准时钟，从而所述各基准时钟的时间与所述主基准时钟的相应时间匹配。

86. 如权利要求 85 所述的方法，其特征在于把所述第一和第二电路之一处的一个基准时钟指定为所述主基准时钟，所述方法还包括以下步骤：

(e) 同步地检索每个所述第一和第二电路处的所述基准时钟的当前时间，

(f) 形成所述一个电路处与所述一个电路以外的每个所述第一和第二电路处的所述基准时钟的所述各当前时间之差，以及

(g) 调节所述一个电路以外的每个所述第一和第二电路的所述基准时钟，以减少所述差值。

87. 如权利要求 85 所述的方法，其特征在于所述第一和第二电路分布于多个节点处，所述方法还包括以下步骤：

(e) 接收位于所述节点中第一个节点处的所述主基准时钟的所述当前时间，以及

(f) 经由一通信链路把所述接收到的当前时间从所述第一节点发送到所述节点中的第二节点。

88. 如权利要求 85 所述的方法，其特征在于所述主基准时钟在地理上远离每个所述第一和第二电路中，所述方法还包括以下步骤：

(e) 周期性地广播所述主基准时钟的所述当前时间，以及

(f) 在多个远程的第一和第二电路中的每一个处同时接收所述广播的当前时间。

89. 一种用于对由包含一个或多个程序的压缩数据的传输包序列形成的传输流进行再分多路复用的再分多路复用设备，每个所述程序具有一独立的位速率和一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述再分多路复用器包括：

接收传输包的一个或多个第一电路，每个第一电路包括用于指示接收到每个传输包的时间的第一基准时钟，

发送传输包的一个或多个第二电路，每个第二电路包括用于指示何时发送每个传输包的第二基准时钟，

主基准时钟，每个所述第一和第二基准时钟将与所述主基准时钟同步，用于周期性地获得所述主基准时钟的当前时间，以及

处理器，依据每个所述第一和第二基准时钟处的所述时间与所述主基准

时钟的所述当前时间之差来调节每个所述第一和第二基准时钟，从而所述各个第一和第二基准时钟的时间与所述主基准时钟的相应时间匹配。

90. 如权利要求 89 所述的再分多路复用器，其特征在于把所述第一和第二电路之一处的一个基准时钟指定为所述主基准时钟，所述处理器同步地检索每个所述第一和第二电路处的所述第一和第二基准时钟的当前时间；形成所述一个电路处与所述一个电路以外的每个所述第一和第二电路处的所述基准时钟的所述当前时间之差；以及调节所述一个电路以外的每个所述第一和第二电路处的所述基准时钟，以减少所述差值。

91. 如权利要求 89 所述的再分多路复用器，其特征在于所述第一和第二电路分布于多个节点处，所述再分多路复用器还包括：

连接所述节点中的第一和第二节点的通信链路，所述第一节点接收所述主基准时钟的所述当前时间并经由一通信链路把所述接收到的当前时间从所述第一节点发送到所述节点中的第二节点。

92. 如权利要求 89 所述的再分多路复用器，其特征在于所述主基准时钟在地理上远离每个所述第一和第二电路，所述再分多路复用器还包括：

一个或多个接收器，用于同时接收所述主基准时钟的所述当前时间的周期性广播。

93. 一种用于对由传输包序列形成的一个或多个传输流进行再分多路复用的方法，所述传输包序列包括包含一个或多个程序中每一个的压缩程序数据以及每个程序的程序时钟基准时间标记的传输包，所述程序的解码和呈现与所述时间标记同步，所述方法包括以下步骤：

(a) 提供一个或多个传输流，

(b) 选择所述一个或多个传输流中的一个或多个传输包以在一再分多路复用传输流中输出，

(c) 依据一预定延迟调度所述传输包中的一些以在一输出传输流的时隙中输出，每个所述时隙大致发生在由本地时钟所指示的派送时间处，

(d) 根据所述本地时钟与产生所述程序时钟基准时间标记的程序系统定时时钟之间的偏移，如果有的话，调节带有每个被调度的程序时钟基准的传输包的每个程序时钟基准时间标记，以及

(e) 根据调度输出带有所述程序时钟基准时间标记的传输包的所述时隙的所述派送时间与所述时隙相对于一外部时钟发生的实际时间之差，进一步

调节每个被调节的程序时钟基准时间标记。

94. 如权利要求 93 所述的方法，其特征在于还包括以下步骤：

(f) 调度其它传输包，以在与所述预定延迟有关的时隙以外的所述输出传输流的时隙中输出，

(g) 根据所述一个其它时隙与相应于所述预定延迟的时隙之间的输出时间之差，计算对在所述其它时隙之一中输出的选中传输包中每个程序时钟基准时间标记所估计的调节，以及

(h) 通过所述估计的调节对每个程序时钟基准时间标记进行调节，所述时间标记位于被调度在所述其它时隙之一中输出的带有程序时钟基准时间标记的传输包中。

95. 一种用于对由传输包序列形成的一个或多个传输流进行再分多路复用的再分多路复用器，所述传输包序列包括包含一个或多个程序中每一个的压缩程序数据以及每个程序的程序时钟基准时间标记的传输包，所述程序的解码和呈现与所述时间标记同步，所述方法包括：

本地时钟，

响应于所述基准时钟的处理器，用于依据一预定延迟调度所述传输包中的一些以在一输出传输流的时隙中输出；调度所述传输包中的一些以在与一预定延迟有关的输出传输流的时隙中输出，每个所述时隙大致发生在由本地时钟所指示的派送时间处；依据所述本地时钟与产生所述程序时钟基准时间标记的程序系统定时时钟之间的偏移，如果有的话，调节带有每个被调度的程序时钟基准的传输包的每个程序时钟基准时间标记，以及

响应于由所述处理器调度的传输包的输出数据链路控制电路，用于根据调度输出带有所述程序时钟基准时间标记的传输包的所述时隙的所述派送时间与所述时隙相对于一外部时钟发生的实际时间之差，进一步调节每个被调节的程序时钟基准时间标记。

96. 如权利要求 95 所述的再分多路复用器，其特征在于所述处理器还用于调度其它传输包，以在与所述预定延迟有关的时隙以外的所述输出传输流的时隙中输出；根据所述一个其它时隙与相应于所述预定延迟的时隙之间的输出时间之差，计算对在所述其它时隙之一中输出的选中传输包中每个程序时钟基准时间标记所估计的调节；以及通过所述估计的调节对每个程序时钟基准时间标记进行调节。

97. 一种由一传输包序列形成的位流，所述传输包序列包括包含一个或多个程序中每一个的压缩程序数据以及每个程序的程序时钟基准时间标记的传输包，所述程序的解码和呈现与所述时间标记同步，所述是通过以下步骤产生的：

(a) 提供一个或多个传输流，

(b) 选择所述一个或多个传输流中的一个或多个传输包以在一再分多路复用传输流中输出，

(c) 依据一预定延迟调度所述传输包中的一些以在一输出传输流的时隙中输出，每个所述时隙大致发生在由本地时钟所指示的派送时间处，

(d) 根据所述本地时钟与产生所述程序时钟基准时间标记的程序系统定时时钟之间的偏移，如果有的话，调节带有每个被调度的程序时钟基准的传输包的每个程序时钟基准时间标记，以及

(e) 根据调度输出带有所述程序时钟基准时间标记的传输包的所述时隙的所述派送时间与所述时隙相对于一外部时钟发生的实际时间之差，进一步调节每个被调节的程序时钟基准时间标记。

98. 如权利要求 97 所述的位流，其特征在于所述位流还通过以下步骤形成：

(f) 调度其它传输包，以在与所述预定延迟有关的时隙以外的所述输出传输流的时隙中输出，

(g) 根据所述一个其它时隙与相应于所述预定延迟的时隙之间的输出时间之差，计算对在所述其它时隙之一中输出的选中传输包中每个程序时钟基准时间标记所估计的调节，以及

(h) 通过所述估计的调节对每个程序时钟基准时间标记进行调节，所述时间标记位于被调度在所述其它时隙之一中输出的带有程序时钟基准时间标记的传输包中。

99. 一种用于对传输包进行再分多路复用的方法，所述传输包包括包含压缩程序数据的传输包，其程序数据被所述传输包包含的每个程序包括一恒定的端-端通信延迟要求、一独立的位速率以及一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述方法包括：

(a) 给待保留的每个接收到的传输包分配位于一描述符存储单元序列之

一中的一个未使用描述符，一高速缓冲存储器已获得对所述描述符存储单元序列的控制，所述描述符存储单元序列是分配给一特定输入端口的队列的一部分，

(b) 把每个保留的传输包存储在传输包存储单元处，所述高速缓冲存储器已获得对所述传输包存储单元的控制，且所述分配的描述符指向所述传输包存储单元，以及

(c) 获得对所述队列的一个或多个未使用描述符存储单元和所述一个或多个描述符存储单元中的这些描述符指向的传输包单元的控制，所述队列接在所述高速缓冲存储器已获得控制的最后一个描述符存储单元后，由一具有可变端-端通信延迟的异步通信链路把描述符存储单元和传输包存储单元的所述队列保持在与所述高速缓冲存储器分开的存储器中。

100. 如权利要求 99 所述的方法，其特征在于还包括：

(d) 经由所述通信链路把所述分配的描述符的数据写到所述存储器中相应的描述符存储单元并把传输包写到传输包存储单元，所述传输包存储单元被其数据已写到所述存储器的所述分配描述符所指向。

101. 如权利要求 100 所述的方法，其特征在于还包括：

(e) 周期性地检查写到有关一输入端口的所述存储器中的每个队列的所述描述符存储单元中的所述描述符数据，

(f) 处理被所述检查的描述符指向的传输包单元中的所述传输包，以及

(g) 给有关输入端口的一个或多个所述队列的所述描述符中的选中描述符分配有关一输出端口的队列的描述符，

(h) 把选中的信息从有关输入端口的所述一个或多个队列的每个选中描述符拷贝到有关所述输出端口的所述队列的所述描述符，以及

(g) 按照从所述输出端口发送的特定顺序对有关所述输出端口的所述队列内的所述描述符进行排序。

102. 如权利要求 101 所述的方法，其特征在于还包括以下步骤：

(i) 在第二高速缓冲存储器中检索与所述输出端口有关的所述队列的每个描述符，从所述第二高速缓冲存储器中的一个描述符存储单元序列的开始处检索每个描述符，以及在所述第二高速缓冲存储器中检索存储在每个检索到的描述符指向的传输包存储单元中的每个传输包，

(j) 在从所述特定输出端口输出的传输流所独有的时隙中输出每个检索

到的传输包，以及

(k) 从所述存储器中，获得接在其中存储有所述序列的最后一个高速缓存描述符的所述描述符存储单元后的描述符存储单元中的有关所述输出端口的所述队列的描述符以及存储在所述获得的描述符所指向的传输包单元中的每个传输包，以把它们存储在所述第二高速缓冲存储器中。

103. 一种用于对传输包进行再分多路复用的方法，所述传输包包括包含压缩程序数据的传输包，其程序数据被所述传输包包含的每个程序包括一恒定的端-端通信延迟要求、一独立的位速率以及一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述方法包括：

(a) 从一高速缓冲存储器中，检索被指派给一输出端口的一个队列的描述符存储单元序列中的每个描述符，从所述序列的开始检索每个描述符，以及从所述高速缓冲存储器中检索存储在每个检索到的描述符指向的传输包存储单元中的每个传输包，

(b) 在从所述特定输出端口输出的传输包的独有时隙中输出每个检索到的传输包，以及

(c) 经由一具有可变端-端通信延迟的异步通信链路，从一存储器中，获得接在其中存储有所述序列的最后一个高速缓存描述符的描述符存储单元后的所述队列的描述符存储单元中的一个或多个描述符以及存储在所述获得的描述符指向的传输包单元中的每个传输包，以把它们存储在所述高速缓冲存储器中。

104. 如权利要求 103 所述的方法，其特征在于还包括：

(d) 在所述存储器中提供附加的描述符存储单元队列，所述队列包括指向其中存储有待输出的传输包的一个或多个传输包存储单元的一个或多个描述符，

(e) 周期性地检查写到所述存储器中的每个所述附加队列的所述描述符存储单元的描述符数据，

(f) 处理由所述被检查的描述符指向的传输包单元中的所述传输包，以及

(g) 给一个或多个所述附加队列的所述描述符中的选中描述符分配指派给所述输出端口的所述队列的一个描述符，把选中的信息从所述一个或多个

附加队列的每个选中描述符拷贝到指派给所述输出端口的所述队列的所述分配描述符，并按照从所述输出端口发送的特定顺序对指派给所述输出端口的所述队列的所述分配描述符进行排序。

105. 一种用于对传输包进行再分多路复用的再分多路复用器，所述传输包包括包含压缩程序数据的传输包，其程序数据被所述传输包包含的每个程序包括一恒定的端-端通信延迟要求、一独立的位速率以及一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述再分多路复用器包括：

高速缓冲存储器，

连到所述高速缓冲存储器的数据链路控制电路，用于给待保留的每个接收到的传输包分配位于一描述符存储单元序列之一中的一个未使用描述符，所述高速缓冲存储器已获得对所述描述符存储单元序列的控制，所述描述符存储单元序列是分配给一特定输入端口的队列的一部分；以及用于把每个保留的传输包存储在传输包存储单元处，所述高速缓冲存储器已获得对所述传输包存储单元的控制，且所述分配的描述符指向所述传输包存储单元，以及

连到所述高速缓冲存储器的直接存储器访问电路，用于获得对所述队列的一个或多个未使用描述符存储单元和所述一个或多个描述符存储单元中的这些描述符指向的传输包单元的控制，所述队列接在所述高速缓冲存储器已获得控制的最后一个描述符存储单元后，由一具有可变端-端通信延迟的异步通信链路把描述符存储单元和传输包存储单元的所述队列保持在与所述高速缓冲存储器分开的存储器中。

106. 如权利要求 105 所述的再分多路复用器，其特征在于所述直接存储器访问电路经由所述通信链路把所述分配的描述符的数据写到所述存储器中相应的描述符存储单元并把传输包写到传输包存储单元，所述传输包存储单元被其数据已写到所述存储器的所述分配描述符所指向。

107. 如权利要求 106 所述的再分多路复用器，其特征在于还包括：

处理器，用于周期性地检查写到有关一输入端口的所述存储器中的每个队列的所述描述符存储单元中的所述描述符数据；处理被所述检查的描述符指向的传输包单元中的所述传输包；给有关输入端口的一个或多个所述队列的所述描述符中的选中描述符分配有关一输出端口的队列的描述符；把选中的信息从有关输入端口的所述一个或多个队列的每个选中描述符拷贝到有关

所述输出端口的所述队列的所述描述符；以及按照从所述输出端口发送的特定顺序对有关所述输出端口的所述队列内的所述描述符进行排序。

108. 如权利要求 107 所述的再分多路复用器，其特征在于还包括：

第二高速缓冲存储器，

第二数据链路控制电路，用于在第二高速缓冲存储器中检索有关所述输出端口的所述队列的每个描述符，从所述第二高速缓冲存储器中的一个描述符存储单元序列的开始处检索每个描述符，以及在所述第二高速缓冲存储器中检索存储在每个检索到的描述符指向的传输包存储单元中的每个传输包，以及在从所述特定输出端口输出的传输流所独有的时隙中输出每个检索到的传输包，以及

连到所述异步通信链路的第二直接存储器访问电路，用于从所述存储器中，获得接在其中存储有所述序列的最后一个高速缓存描述符的所述描述符存储单元后的描述符存储单元中的有关所述输出端口的所述队列的描述符以及存储在所述获得的描述符所指向的传输包单元中的每个传输包，以把它们存储在所述第二高速缓冲存储器中。

109. 一种用于对传输包进行再分多路复用的再分多路复用器，所述传输包包括包含压缩程序数据的传输包，其程序数据被所述传输包包含的每个程序包括一恒定的端-端通信延迟要求、一独立的位速率以及一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述再分多路复用器包括：

高速缓冲存储器，

连到所述高速缓冲存储器的数据链路控制电路，用于从所述高速缓冲存储器中，检索被指派给一输出端口的一个队列的描述符存储单元序列中的每个描述符，从所述序列的开始检索每个描述符，以及从所述高速缓冲存储器中检索存储在每个检索到的描述符指向的传输包存储单元中的每个传输包，以及在从所述特定输出端口输出的传输包的独有时隙中输出每个检索到的传输包，以及

连到所述高速缓冲存储器的直接存储器访问电路，经由一具有可变端-端通信延迟的异步通信链路，从所述存储器中，获得接在其中存储有所述序列的最后一个高速缓存描述符的描述符存储单元后的所述队列的描述符存储单元中的一个或多个描述符以及存储在所述获得的描述符指向的传输包单元中

的每个传输包，以把它们存储在所述高速缓冲存储器中。

110. 如权利要求 109 所述的再分多路复用器，其特征在于还包括：

连到所述异步通信链路的存储器，用于保持附加的描述符存储单元队列，所述队列包括指向其中存储有待输出的传输包的一个或多个传输包存储单元的一个或多个描述符，

连到所述异步通信链路的处理器，用于周期性地检查写到所述存储器中的每个所述附加队列的所述描述符存储单元的描述符数据；处理由所述被检查的描述符指向的传输包单元中的所述传输包；以及给一个或多个所述附加队列的所述描述符中的选中描述符分配指派给所述输出端口的所述队列的一个描述符，把选中的信息从所述一个或多个附加队列的每个选中描述符拷贝到指派给所述输出端口的所述队列的所述分配描述符，并按照从所述输出端口发送的特定顺序对指派给所述输出端口的所述队列的所述分配描述符进行排序。

111. 一种包含传输包的传输流，所述传输包包括包含压缩程序数据的传输包，其程序数据被所述传输包包含的每个程序包括一恒定的端-端通信延迟要求、一独立的位速率以及一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述传输流是通过以下步骤产生的：

(a) 给待保留的每个接收到的传输包分配位于一描述符存储单元序列之一中的一个未使用描述符，一高速缓冲存储器已获得对所述描述符存储单元序列的控制，所述描述符存储单元序列是分配给一特定输入端口的队列的一部分，

(b) 把每个保留的传输包存储在传输包存储单元处，所述高速缓冲存储器已获得对所述传输包存储单元的控制，且所述分配的描述符指向所述传输包存储单元，以及

(c) 获得对所述队列的一个或多个未使用描述符存储单元和所述一个或多个描述符存储单元中的这些描述符指向的传输包单元的控制，所述队列接在所述高速缓冲存储器已获得控制的最后一个描述符存储单元后，由一具有可变端-端通信延迟的异步通信链路把描述符存储单元和传输包存储单元的所述队列保持在与所述高速缓冲存储器分开的存储器中。

112. 一种包含传输包的传输流，所述传输包包括包含压缩程序数据的传

输包，其程序数据被所述传输包包含的每个程序包括一恒定的端-端通信延迟要求、一独立的位速率以及一独立的编码器系统定时时钟的程序时钟基准时间标记，所述程序的解码和呈现与所述时间标记同步，所述传输流是通过以下步骤产生的：

(a) 从一高速缓冲存储器中，检索被指派给一输出端口的一个队列的描述符存储单元序列中的每个描述符，从所述序列的开始检索每个描述符，以及从所述高速缓冲存储器中检索存储在每个检索到的描述符指向的传输包存储单元中的每个传输包，

(b) 在从所述特定输出端口输出的传输包的独有时隙中输出每个检索到的传输包，以及

(c) 经由一具有可变端-端通信延迟的异步通信链路，从一存储器中，获得接在其中存储有所述序列的最后一个高速缓存描述符的描述符存储单元后的所述队列的描述符存储单元中的一个或多个描述符以及存储在所述获得的描述符指向的传输包单元中的每个传输包，以把它们存储在所述高速缓冲存储器中。

113. 一种用于对一传输流的传输包进行解扰的方法，所述传输包包含一个或多个视频程序的流元数据，所述方法包括以下步骤：

(a) 定义将在每个传输包上执行的一个或多个处理步骤构成的序列，并对所述序列内的解扰处理步骤进行排序，

(b) 给每个传输包分配一队列的一个描述符，所述分配的描述符包含对其所分配的所述传输包的指针、一个或多个处理指示以及控制字信息的存储单元，

(c) 把有关所述传输包的内容的控制字信息存储在所述分配的描述符中的选中描述符的所述控制字信息存储单元中，

(d) 设定所述一个或多个所述处理指示，以指示可对每个所述分配的描述符执行所述序列的下一处理步骤，

(e) 依次访问每个分配的描述符，以及

(f) 对于指向待解扰传输包的每个被访问的描述符，仅在所述被访问的描述符的所述一个或多个处理指示被设定为指示可对所述被访问的描述符和所述被访问的描述符指向的传输包执行解扰处理时，才使用所述被访问的描述符中的所述控制字信息对所述被访问的描述符指向的所述传输包进行解

扰。

114. 如权利要求 113 所述的方法，其特征在于所述控制字信息是一控制字表的基址。

115. 如权利要求 114 所述的方法，其特征在于还包括以下步骤：

(g) 在所述解扰步骤期间，使用所述基址来定位一控制字表，并从被所述传输包的一包标识符索引的所述控制字表的一个表目开始检索控制字，每个包标识符独有地指示包含在所述传输包中的流元数据。

116. 如权利要求 115 所述的方法，其特征在于所述定位步骤还包括使用所述传输包的奇/偶控制字指示来检索所述控制字。

117. 如权利要求 115 所述的方法，其特征在于还包括以下步骤：

(g) 保持包含用于对所述传输包的内容进行解扰的所述控制字的控制字表。

118. 如权利要求 113 所述的方法，其特征在于还包括以下步骤：

(g) 把经解扰的传输包数据写入由所述分配的描述符的所述指针指向的传输包存储单元，从而改写所述传输包的预解扰数据，以及

(h) 在检查包含指示可执行解扰处理的处理指示的每个描述符后，把一个或多个所述处理指示设定为指示可对所述描述符及所述描述符指向的传输包执行所述序列的下一处理步骤。

119. 一种用于对一传输流的传输包进行加扰的方法，所述传输包包含一个或多个视频程序的流元数据，所述方法包括以下步骤：

(a) 定义将在每个传输包上执行的一个或多个处理步骤构成的序列，并对所述序列内的加扰处理步骤进行排序，

(b) 给每个传输包分配一队列的一个描述符，每个所分配的描述符包含对其所分配的所述传输包的指针、一个或多个处理指示以及控制字信息的存储单元，

(c) 把有关所述传输包的内容的控制字信息存储在所述分配的描述符中的选中描述符的所述控制字信息存储单元中，

(d) 设定所述一个或多个所述处理指示，以指示可对每个所述分配的描述符执行所述序列的下一处理步骤，

(e) 依次访问每个分配的描述符，以及

(f) 对于指向待加扰传输包的每个被访问的描述符，仅在所述被访问的

描述符的所述一个或多个处理指示被设定为指示可对所述被访问的描述符和所述被访问的描述符指向的传输包执行加扰处理时，才使用所述被访问的描述符中的所述控制字信息对所述被访问的描述符指向的所述传输包进行加扰。

120. 如权利要求 119 所述的方法，其特征在于所述控制字信息是一相应于每个传输包的内容的控制字。

121. 如权利要求 120 所述的方法，其特征在于还包括以下步骤：

(g) 在所述分配步骤期间，从被所述传输包的一包标识符索引的所述控制字表的一个表目开始检索所述控制字，每个包标识符独有地指示包含在所述传输包中的流元数据，以及

(h) 把所述检索到的控制字存储在所述描述符的所述控制字存储单元中。

122. 如权利要求 121 所述的方法，其特征在于还包括以下步骤：

(g) 保持包含用于对所述传输包的内容进行加扰的所述控制字的控制字表。

123. 如权利要求 119 所述的方法，其特征在于还包括以下步骤：

(g) 把经加扰的传输包数据写入由所述分配的描述符的所述指针指向的传输包存储单元，从而改写所述传输包的预加扰数据，以及

(h) 在检查包含指示可执行加扰处理的处理指示的每个描述符后，把一个或多个所述处理指示设定为指示可对所述描述符及所述描述符指向的传输包执行所述序列的下一处理步骤。

124. 一种用于对一传输流的传输包进行解扰的再分多路复用器，所述传输包包含一个或多个视频程序的流元数据，所述再分多路复用器包括：

处理器，用于定义将在每个传输包上执行的一个或多个处理步骤构成的序列，并对所述序列内的解扰处理步骤进行排序，

数据链路控制电路，用于给每个传输包分配一队列的一个描述符，所述分配的每个描述符包含对其所分配的所述传输包的指针、一个或多个处理指示以及控制字信息的存储单元，以及用于设定所述一个或多个所述处理指示，以指示可对每个所述分配的描述符执行所述序列的下一处理步骤，以及

解扰器，用于依次访问每个分配的描述符，以及对于指向待解扰传输包的每个被访问的描述符，仅在所述被访问的描述符的所述一个或多个处理指示被设定为指示可对所述被访问的描述符和所述被访问的描述符指向的传输

包执行解扰处理时，才使用所述被访问的描述符中的所述控制字信息对所述被访问的描述符指向的所述传输包进行解扰，

其中所述处理器还把有关所述接收的传输包的内容的控制字信息存储在所述描述符中的相应描述符的所述控制字存储单元中。

125. 如权利要求 124 所述的再分多路复用器，其特征在于所述控制字信息是一控制字表的基址。

126. 如权利要求 125 所述的再分多路复用器，其特征在于所述解扰器使用所述基址来定位一控制字表，并从被所述传输包的一包标识符索引的所述控制字表的一个表目开始检索控制字，每个包标识符独有地指示包含在所述传输包中的流元数据。

127. 如权利要求 126 所述的再分多路复用器，其特征在于所述解扰器使用所述传输包的奇/偶控制字指示来定位所述控制字以索引所述控制字表。

128. 如权利要求 126 所述的再分多路复用器，其特征在于所述处理器保持包含用于对所述传输包的内容进行解扰的所述控制字的控制字表。

129. 如权利要求 124 所述的再分多路复用器，其特征在于所述解扰器把经解扰的传输包数据写入由所述分配的描述符的所述指针指向的传输包存储单元，从而改写所述传输包的预解扰数据，以及在检查包含指示可执行解扰处理的处理指示的每个描述符后，把一个或多个所述处理指示设定为指示可对所述描述符及所述描述符指向的传输包执行所述序列的下一处理步骤。

130. 一种用于对一传输流的传输包进行加扰的再分多路复用器，所述传输包包含一个或多个视频程序的流元数据，所述再分多路复用器包括：

处理器，用于定义将在每个传输包上执行的一个或多个处理步骤构成的序列，并对所述序列内的加扰处理步骤进行排序；给每个传输包分配一队列的一个描述符，每个所分配的描述符包含对其所分配的所述传输包的指针、一个或多个处理指示以及控制字信息的存储单元；把有关所述传输包的内容的控制字信息存储在所述分配的描述符中的选中描述符的所述控制字信息存储单元中；以及用于设定所述一个或多个所述处理指示，以指示可对每个所述分配的描述符执行所述序列的下一处理步骤，

加扰器，依次访问每个分配的描述符，以及对于指向待加扰传输包的每个被访问的描述符，仅在所述被访问的描述符的所述一个或多个处理指示被设定为指示可对所述被访问的描述符和所述被访问的描述符指向的传输包执

行加扰处理时，才使用所述被访问的描述符中的所述控制字信息对所述被访问的描述符指向的所述传输包进行加扰。

131. 如权利要求 130 所述的再分多路复用器，其特征在于所述控制字信息是一相应于每个传输包的内容的控制字。

132. 如权利要求 131 所述的再分多路复用器，其特征在于所述处理器从被所述传输包的一包标识符索引的所述控制字表的一个表目开始检索所述控制字，每个包标识符独有地指示包含在所述传输包中的流元数据，以及把所述检索到的控制字存储在所述描述符的所述控制字存储单元中。

133. 如权利要求 132 所述的再分多路复用器，其特征在于所述处理器保持包含用于对所述传输包的内容进行加扰的所述控制字的控制字表。

134. 如权利要求 130 所述的再分多路复用器，其特征在于所述加扰器把经加扰的传输包数据写入由所述分配的描述符的所述指针指向的传输包存储单元，从而改写所述传输包的预加扰数据，以及在检查包含指示可执行加扰处理的处理指示的每个描述符后，把一个或多个所述处理指示设定为指示可对所述描述符及所述描述符指向的传输包执行所述序列的下一处理步骤。

135. 一种包含经解扰的传输包的传输流，所述传输包包含一个或多个视频程序的流元数据，所述传输流是通过以下步骤产生的：

(a) 定义将在每个传输包上执行的一个或多个处理步骤构成的序列，并对所述序列内的解扰处理步骤进行排序，

(b) 给每个传输包分配一队列的一个描述符，所述分配的描述符包含对其所分配的所述传输包的指针、一个或多个处理指示以及控制字信息的存储单元，

(c) 把有关所述传输包的内容的控制字信息存储在所述分配的描述符中的选中描述符的所述控制字信息存储单元中，

(d) 设定所述一个或多个所述处理指示，以指示可对每个所述分配的描述符执行所述序列的下一处理步骤，

(e) 依次访问每个分配的描述符，以及

(f) 对于指向待解扰传输包的每个被访问的描述符，仅在所述被访问的描述符的所述一个或多个处理指示被设定为指示可对所述被访问的描述符和所述被访问的描述符指向的传输包执行解扰处理时，才使用所述被访问的描述符中的所述控制字信息对所述被访问的描述符指向的所述传输包进行解

扰。

136. 一种包含经加扰的传输包的传输流，所述传输包包含一个或多个视频程序的流元数据，所述传输流是通过以下步骤产生的：

(a) 定义将在每个传输包上执行的一个或多个处理步骤构成的序列，并对所述序列内的加扰处理步骤进行排序，

(b) 给每个传输包分配一队列的一个描述符，每个所分配的描述符包含对其所分配的所述传输包的指针、一个或多个处理指示以及控制字信息的存储单元，

(c) 把有关所述传输包的内容的控制字信息存储在所述分配的描述符中的选中描述符的所述控制字信息存储单元中，

(d) 设定所述一个或多个所述处理指示，以指示可对每个所述分配的描述符执行所述序列的下一处理步骤，

(e) 依次访问每个分配的描述符，以及

(f) 对于指向待加扰传输包的每个被访问的描述符，仅在所述被访问的描述符的所述一个或多个处理指示被设定为指示可对所述被访问的描述符和所述被访问的描述符指向的传输包执行加扰处理时，才使用所述被访问的描述符中的所述控制字信息对所述被访问的描述符指向的所述传输包进行加扰。

说　明　书

带有视频程序的传输流再分多路复用器

技术领域

本发明属于通信系统。尤其是，本发明属于选择性地多路复用包含一个或多个程序(诸如实时音频-视频程序)的位流。调节程序专用信息和有关其它程序的信息，从而使得可在位流的接收端对程序进行识别、提取和实时再现。

背景技术

近来，已提出了有效地压缩音频-视频程序以便存储和发送的技术。例如，见 ISO\IEC IS 13818-1,2,3：信息技术-移动图像及相关的音频信息的通用编码：系统、视频和和音频(“MPEG-2”)；ISO\IEC IS 11172-1,2,3：信息技术-用于高达约 1.5 兆位/秒的数字存储媒体的移动画面及相关视频的通用编码：系统、视频和音频(“MPEG-1”)；杜比 AC-3；运动 JPEG 等。这里，按照 MPEG-2 的说法，术语程序指具有公共时间基准且打算同步呈现的有关音频-视频信号的集合。

MPEG-1 和 MPEG-2 为分层流而设。即，音频-视频程序由一个或多个经编码的位流或“流元(elementary stream)”(“ES”)构成，流元诸如经编码的视频 ES 和经编码的音频 ES、经第二语言编码的音频 ES、闭路字幕文本(closed caption text)ES 等。对每个 ES，尤其是对每个音频和视频 ES 分开地进行编码。然后，把经编码的 ES 组合成为诸如程序流“PS”或传输流“TS”等系统层流。PS 或 TS 的目的在于使得可从一程序中提取经编码的 ES，分离每个 ES 并对其进行分开的解码，并同步地呈现经编码的 ES。可把 TS 或 PS 封装在一较高的通道层中或以为前向纠错而设的存储格式来封装。

流元

通常以例如 384kbps 的恒定位速率对音频 ES 进行编码。另一方面，以可变的位速率依据 MPEG-1 或 MPEG-2 对视频 ES 进行编码。这意味着，每一经压缩/编码的图像的位数依图像而不同(这些图像是以恒定的速率呈现或显示的)。视频编码包含从空间上和时间上对视频图像进行编码的步骤。空间编码包括亮度和色度象素数据的离散余弦变换、量化、(曲折)扫描、游程长度编码和可变长度编码块。时间编码涉及估计宏块(例如，由亮度块和覆盖在其上的每个色度块形成的 4x4 阵

列)的运动来识别运动矢量，对宏块进行运动补偿来形成预测误差宏块，对预测误差宏块进行空间编码以及对运动矢量进行可变长度编码。仅对一些图像(叫做 I 图像)进行空间编码，而对诸如 P 和 B 图像等其它图像进行空间和运动补偿编码(即，从其它图像进行时间预测)。经编码的 I 图像所具有的位通常比经编码的 P 图像的位多，经编码的 P 图像所具有的位通常比经编码的 B 图像的位多。在任一种情况下，甚至同一类型的经编码图像可能具有不同的位数。

MPEG-2 限定了对经编码的视频 ES 的缓冲器尺寸约束。尤其是，假定解码器的缓冲器具有预定的最大存储容量。经编码的视频 ES 不得使解码器的缓冲器上溢(在某些情况下，不得使解码器的缓冲器下溢)。为使能对预测图像(来自预测这些预测图像的基准图像)的解码，MPEG-2 具体地限定了相对于视频 ES 的位速率从解码器的缓冲器中除去每个图像的压缩数据的时间、图像显示速率和某些图像重新排序的约束。如果给定这些约束，则可调节(与以宏块为基础对宏块进行调节一样频繁)在压缩图像时所产生的位数，以保证视频 ES 不引起视频 ES 编码器的缓冲器下溢或上溢。

传输流

这里以 TS 示出本发明。为了简洁，省略对 PS 的讨论。然而，本领域内的技术人员应理解本发明的某些方面对 PS 的可应用性。

每个 ES 的数据形成可变长度程序流元或“PES”包(packet)。PES 包包含单个 ES 的数据，但也可包含不止一个解码单元的数据(例如，可包含不止一个压缩图像、不止一个压缩音频帧等)。在 TS 的情况下，首先把 PES 包分成许多有效负载(payload)单元，并把它们插入固定长度(188 字节长)的传输包中。每个传输包可携带单一类型的有效负载数据，例如单一 ES 的 PES 包数据。每个 TS 设有包括包标识符或“PID”的四字节标头。此 PID 类似于唯一地指示传输包的内容的标签(tag)。因而，把一个 PID 指派给特定程序的视频 ES，把另一个不同的 PID 指派给特定程序的音频 ES 等。

相对于单个编码器系统的定时时钟对每个程序的 ES 进行编码。同样，依次使这些 ES 的解码和同步呈现相对于同一编码器系统定时时钟同步。因而，解码器必须能恢复原始的编码器系统定时时钟，以便能以及时且相互同步的方式对每个 ES 进行解码并呈现每个经解码的 ES。为此，把系统定时时钟的时间标记(叫做程序时钟基准或“PCR”)插入选中的传输包的有效负载中(具体来说，在适应(adaption)字段中)。解码器从传输包中提取 PCR 并使用 PCR 来恢复编码器系统定时时钟。PES

包可包含解码时间标记或“DTS”和/或呈现时间标记或“PTS”。DTS 指示相对于所恢复的编码器系统定时时钟应对下一解码单元(即，经压缩的音频帧、经压缩的视频图像等)进行解码的时间。PTS 指示相对于所恢复的编码器系统定时时钟应呈现或显示下一呈现单元(即，经解压缩的音频帧、经解压缩的图像等)的时间。

与 PS 不同，TS 可具有携带不止一个程序的程序数据的传输包。已在相对于不同的编码器系统定时时钟的不同编码器处对每个程序进行了编码。TS 使得解码器可恢复解码器想要解码的程序的专用系统定时时钟。为此，TS 必须携带不同组 PCR 组，即其中的一组 PCR 用于恢复每个程序的编码器系统定时时钟。

TS 还携带了传输包中的程序专用信息或(PSI)。PSI 用于识别想要程序的数据或有助于程序解码的其它信息。设有程序关联表或“PAT”，在具有 PID 0x0000 的传输包中携带有此关联表。PAT 把每个程序号与携带该程序的程序定义传输包的 PID 相关。程序定义：(1)指示哪些 ES 构成程序定义所对应的程序，(2)识别这些 ES 中每一个的 PID，(3)指示携带该程序的 PCR 的传输包的 PID，(4)识别携带 ES 专用命名控制消息(例如，解扰或解密密钥)和其它信息。总的来说，TS 的所有程序定义都叫做程序映射表(PMT)。因而，解码器可从传输包中提取 PAT 数据并使用 PAT 来识别携带想要程序的程序定义的传输包的 PID。然后，解码器可从传输包中提取想要的程序的程序定义数据，并识别携带构成该想要的程序的 ES 数据的传输包的 PID 以及携带 PCR 的传输包的 PID。然后，使用所识别的这些 PID，解码器可从 TS 的传输包中提取想要的程序的 ES 的 ES 数据及该程序的 PCR。解码器从想要程序的 PCR 中恢复编码器系统定时时钟，并在相对于所恢复的编码器系统定时时钟的时间对 ES 数据进行解码并呈现。

任选地设置在 TS 中的其它类型信息包括命名控制消息(ECM)、命名管理消息(EMM)、有条件访问(access)表(CAT)和网络信息表(NIT)(CAT 和 NIT 也是 PSI 型的)。ECM 是用于控制解码器翻译 ECM 所属 ES 的能力的 ES 专用消息。例如，可对 ES 进行加扰，解扰密钥或控制字可以是 ECM。把与特定 ES 相关的 ECM 置于它们自己的传输包中，并标上独有的 PID。另一方面，EMM 是用于控制一组解码器(这组解码器位于一个叫做“有条件访问系统”的系统中)翻译一部分 TS 的能力的全系统消息。把 EMM 置于它们自己的传输包中，并标上对 EMM 所属的有条件访问系统所独有的 PID。为使解码器对有条件访问系统(解码器是该系统的一部分，即该解码器是该组解码器的一个成员)的 EMM 定位，每当存在 EMM 时就设置一 CAT。NIT 保存了各种网络参数。例如，如果在解码器接收器可调谐到的不同载波频率上调

制多个 TS，则 NIT 可指示在哪个载波频率(携带 TS)调制每个程序。

与视频 ES 相类似的，MPEG-2 要求，由具有预定尺寸来存储程序 ES 和 PSI 数据的 TS 缓冲器的解码器对 TS 进行解码。MPEG-2 还限定了数据流入和流出这种缓冲器的速率。最重要的是，MPEG-2 要求，TS 不能上溢或下溢 TS 缓冲器。

为了进一步防止缓冲器上溢或下溢，MPEG-2 要求，从编码器传输到解码器的数据需经历恒定的端-端延迟，并保持适当的程序和 ES 位速率。此外，为了保证对 ES 进行及时地解码和呈现，TS 中 PCR 的相对到达时间不应与此 PCR 所指示的相对时间改变得太多。换句话说，每个 PCR 指示当接收到包含 PCR 一部分的最后一个字节时系统定时时钟(在解码器处所恢复的)应具有的时间。因而，下一个 PCR 的接收时间应与每个 PCR 所指示的时间相关。

再分多路复用

通常，想要对 TS 进行“再分多路复用”。再分多路复用涉及选择性地修改 TS 的内容，诸如增加 TS 的传输包、从 TS 中删除传输包、重新排列 TS 中传输包的顺序和/或修改包含在传输包中的数据。例如，有时想要把包含第一程序的传输包加到包含其它程序的 TS 中。这种操作涉及的步骤比简单地增加第一程序的传输包要多。至少，必须修改诸如 PAT 和 PMT 等 PSI，从而它能正确地引用(reference)TS 的内容。然而，必须进一步修改 TS，以保持其中所携带的每个程序的端-端延迟。具体来说，不可以改变每个程序的位速率，以防止 TS 和视频解码器的缓冲器下溢和上溢。此外，必须除去引入 TS 的 PC 的任何时间的未对准，这些未对准是改变带有(bearing)同一程序的 PCR 的相继传输包的相对接收间隔/速率的结果。

已有技术已提出了一种用于 MPEG-2 TS 的再分多路复用器。所提出的再分多路复用器是一块复杂的专用硬件，它在接收每个输入的待多路复用 TS 的点到输出最后一个经再分多路复用的输出 TS 的点之间提供完全的同步——单个系统定时时钟控制传输包的接收、缓冲、修改、传送、重新装配和输出并使这些操作同步。虽然这种再分多路复用器能对 TS 进行再分多路复用，但再分多路复用器的体系结构是复杂的，而且需要统一同步的专用平台。

本发明的一个目的是提供一种灵活的再分多路复用体系结构，例如它可存在于可能异步的任何平台上。

对单个程序的视频和音频进行压缩并产生带有单个程序的 TS 的程序编码器是公知的。如上所述，MPEG-2 给任何时刻可在视频解码器缓冲器中存在的 TS 的位速率和位数加上了非常强的约束。难于对 ES 尤其是视频 ES 进行编码并保证位

速率在所有的时刻保持完全恒定。再者，必须把某些开销带宽分配给每个程序，以保证不因 ES 编码器产生意外的过量编码信息而省略 ES 数据。另一方面，程序编码器偶尔没有在特定传输包时隙输出任何经编码的程序数据。这可能因为程序编码器为防止解码器缓冲器的上溢而减少在该时刻待输出的位数而发生。或者，这可能因为程序编码器未曾预料到对 ES 进行编码需要较长的时间因而在该瞬时没有可获得的数据而发生。为了保持 TS 的位速率并防止 TS 解码器缓冲器的下溢，把空的传输包插入传输包时隙中。

在待再分多路复用的 TS 中存在空的传输包通常是必须简单接受的一个约束。本发明的一个目的是优化包含空传输包的 TS 的带宽。

有时，经由异步通信链路来传送 TS 或 ES 数据。本发明的一个目的是对此未定时或异步传送的数据进行“重新定时”。本发明的还有一个目的是通过对这些传输包的发送进行定时而把从这些异步通信链路发送的传输包的抖动(jitter)减到最少。

本发明的还有一个目的是使用户可动态地(即，实时地)改变经再分多路复用的内容成为再分多路复用 TS，而不停止输出的再分多路复用 TS 中的传输包的流动。

本发明的另一个目的是在一网络上分布再分多路复用功能。例如，一个目的是把一个或多个 TS 或 ES 源置于可能异步的通信网络(诸如以太网 LAN)的任意节点处以及把再分多路复用器置于这种网络的另一个节点处。

发明内容

依据本发明实现了这些和其它目的。本发明的一个示例应用是对符合 MPEG-2 的一个或多个传输流(TS)进行再分多路复用。TS 是包含一个或多个经压缩/编码的音频-视频程序的数据的位流。每个 TS 形成固定长度的传输包的序列。每个经压缩的传输包括诸如数字视频信号和/或数字音频信号的一个或多个经压缩的流元(ES)的数据。传输包还携带了每个程序的程序时钟基准(PCR)，这些 PCR 是使各个程序的解码和呈现被同步的编码器系统定时时钟的时间标记。每个程序具有预定的位速率，且将在具有预定尺寸的 TS 缓冲器和预定解码器缓冲器的解码器处被解码。以此方式对每个程序进行编码，从而防止这些缓冲器的上溢和下溢。示例地，在 TS 的选中传输包中还携带了有助于 TS 的解码的程序专用信息(PSI)。

依据一个实施例，再分多路复用器节点设有一个或多个适配器，每个适配器

包括一高速缓冲存储器、连到该高速缓冲存储器的数据链路控制电路和连到该高速缓冲存储器的直接存储器访问电路。适配器是具有特殊特性的同步接口。数据链路控制电路具有用于接收传输流的输入端口和用于发送传输流的输出端口。直接存储器访问电路可连到具有可变端-端通信延迟的异步通信链路(诸如再分多路复用器节点的一条总线)。使用异步通信链路，直接存储器访问电路可访问再分多路复用器节点的存储器。存储器可存储一个或多个描述符存储单元(storage location)的队列，诸如指派给输入端口的一个队列和指派给输出端口的一个队列。存储器还可在传输包存储单元中存储传输包，对于这些单元描述符存储在每个队列点的描述符存储单元中。示例地，再分多路复用器节点包括连到总线的处理器，该处理器用于处理传输包和描述符。

当使用适配器来输入传输流时，数据链路控制电路把对输入端口所分配的队列中的描述符存储单元序列之一中未使用的描述符分配给待保留的每个接收到的传输包。所分配的描述符位于高速缓冲存储器已从中获得控制的描述符存储单元。数据链路控制电路把每个保留的传输包存储在高速缓冲存储器已从中获得控制并由所分配的描述符所指向的传输包存储单元处。直接存储器访问电路获得对存储器中该队列的一个或多个未使用描述符存储单元(位于高速缓冲存储器已从中获得控制的最后一个描述符存储单元后)的控制。直接存储器访问电路还获得对一个或多个描述符存储单元中的这些描述符所指向的存储器中的传输包单元的控制。

在使用适配器来输出传输包时，数据链路控制电路从高速缓冲存储器检索(retrieve)指派给输出端口的队列的描述符存储单元序列中的每个描述符。从序列的开始依次检索描述符。数据链路控制电路还从高速缓冲存储器检索存储在检索的描述符所指向的传输包存储单元中的传输包。数据链路控制电路以从输出端口输出的传输流的独有时隙输出每个检索的传输包(即，每时隙一个传输包)。直接存储器访问电路从存储器中获得指派给存储单元(位于其中存储有该序列的最后一个高速缓存描述符的描述符存储单元后)的输出端口的队列的描述符，以存储在高速缓冲存储器中。直接存储器访问电路还获得存储在所获得的描述符指向的传输包单元中的每个传输包。

依据另一个实施例，(还)使用每个描述符来记录指示在输入端口处何时接收到传输包接收时间标记，或指示将从输出端口发送初始包的时间的派送时间标记。在输入端口接收到传输包的情况下，数据链路控制电路把接收时间标记记录在分配给每个接收到并保留的传输包的描述符中，以指示接收到该传输包的时间。按

接收的顺序把描述符保存在接收队列中。在从输出端口输出传输包的情况下，数据链路控制电路从发送队列中依次检索每个描述符及每个检索的描述符指向的传输包。在相应于记录在每个检索的描述符中的派送时间的时刻，数据链路控制电路以相应于记录在检索的描述符中的派送时间的输出传输流的时隙来发送每个检索的描述符指向的被检索的传输包。

示例地，再分多路复用器节点处理器检查接收队列中的每个描述符以及包含指向待输出传输包的描述符的其它队列。处理器分配与输出端口(从中将发送由每个经检查的描述符(如果有的话)所指向的传输包)有关的发送队列的描述符。处理器依据例如描述符指向的传输包的接收时间及传输包的接收和输出之间的内部缓冲器延迟，把派送时间指派给对发送队列所分配的描述符。此外，处理器还按派送时间的升序对发送队列的描述符进行排序。

还提供了独有的 PCR 归一化处理。处理器调度将在相应于再分多路复用器节点中的预定延迟的特定派送时间处以某一时隙输出的每个传输包。如果被调度的传输包包含 PCR，则根据本地基准时钟相对于从中产生 PCR 的系统定时时钟的程序的偏移(drift)(如果存在任何偏移)来调节 PCR。数据链路控制电路(发送带有此被调节 PCR 的传输包)进一步根据传输包的调度派送时间与相对于外部时钟发生该时隙的实际时间之差来调节每个经调节的 PCR 时间标记。

示例地，如果要以同一时隙输出不止一个传输包，则以分开的连续时隙来输出每个这样的传输包。处理器计算传输包中每个 PCR 的估计调节，此传输包被调度到以与使用预定延迟所确定的时隙不同的时隙输出。所估计的调节以处理器实际调度带有待输出 PCR 的传输包的时隙与预定延迟所确定的时隙之间的输出时间差为基础。处理器依据此估计的调节来调节 PCR。

依据一个实施例，描述符还用于控制对传输包的加扰或解扰。在解扰的情况下，处理器定义了将在每个传输包上所进行的一个或多个处理步骤的序列并对该序列内的解扰处理排序。处理器把与传输包的内容有关的控制字信息存入所分配的描述符的控制字信息存储单元中。数据链路控制电路把描述符分配给每个接收到的并保留的传输包，每个这样的描述符包括一个或多个处理指示及控制字信息的存储单元。数据链路控制电路设定所分配的描述符的一个或多个处理指示，以指示可在每个所分配的描述符上执行的序列的下一处理步骤。提供了依次访问每个所分配的描述符的解扰器。如果把访问到的描述符的处理指示设定为指示可对访问到的描述符(和访问到的描述符所指向的传输包)进行解扰处理，则解扰器对

此描述符及其指向的传输包进行处理。具体来说，如果描述符指向一待解扰的传输包，则解扰器使用访问到的描述符中的控制字信息对传输包进行解扰。

解扰器可位于(接收)适配器上，在此情况下，解扰器的处理发生在数据链路控制电路的处理(例如，描述符分配、接收时间记录等)后，但发生在直接存储器访问电路的处理(例如，传送到存储器)前。或者，解扰器可以是连到异步通信接口的分开的器件，在此情况下，解扰器的处理发生在直接存储器访问电路的处理后，但发生在处理器的处理(例如，估计的离开时间计算、PID 重新映射等)前。在任一种情况下，控制字信息都是处理器所保存的 PID 可索引控制字表的基址。

在加扰的情况下，处理器限定了将在每个传输包上执行的一个或多个处理步骤的序列，并对该序列内的加扰处理进行排序。处理器把发送队列的发送描述符分配给每个待发送的传输包，并把与传输包有关的控制字信息存入所分配的描述符中的选中描述符的控制字信息存储单元中。然后，处理器设定描述符的一个或多个处理指示，以指示可对每个所分配的描述符进行的序列中的下一处理步骤。提供了依次访问每个所分配的描述符的加扰器。加扰器处理每个访问到的描述符及访问到的描述符指向的传输包，但此处理仅在把访问到的描述符的处理指示设定为指示可对访问到的描述符(和访问到的描述符所指向的传输包)进行加扰处理时进行。具体来说，如果访问到的描述符指向待加扰的传输包，则加扰器使用访问到的描述符中的控制字信息对访问到的描述符所指向的传输包进行加扰。

加扰器可位于(发送)适配器上，在此情况下，加扰器的处理发生在直接存储器访问电路的处理(例如，从存储器传送到高速缓冲存储器等)后，但发生在数据链路控制电路的处理(例如，以正确的时隙输出、最后一个 PCR 校正等)前。或者，加扰器可以是连到异步通信接口的分开的器件，在此情况下，解扰器的处理发生在处理器的处理(例如，发送队列描述符分配、派送时间指派、PCR 校正等)后，但发生在直接存储器访问电路的处理前。如同解扰，控制字信息可以是处理器所保存的 PID 可索引控制字表的基址。然而，控制字信息最好是用来对传输包进行加扰的控制字本身。

此外，依据一个实施例，提供了对带有经由异步通信链路接收到的数据的视频程序进行重新定时的方法。异步通信接口(例如，以太网接口、ATM 接口等)连到再分多路复用器节点处理器(例如，经由总线)，以接收来自具有变化的端-端发送延迟的通信链路的带有视频程序的位流。该处理器根据接收到的位流中所携带的程序的多个时间标记来确定携带有接收到的位流的同一程

序的数据的一个或多个接收到的包中每个包应在输出的 TS 中出现的时间。诸如发送适配器等同步接口在依据所确定时间的时间处，选择性地发送输出的TS(具有恒定的端-端延迟)中携带有被接收数据的选中传输包。

示例地，再分多路复用器节点存储器存储包含从接收队列中接收到的位流中接收到的数据的包。处理器识别包含存储在接收队列中的程序的数据的每个包，此识别是在包含该程序的连续时间标记的第一和第二特定包之间进行。处理器根据此第一和第二时间标记之差来定该程序的(传输)包速率。处理器把指派给第一特定包的发送时间及(传输)包速率与所识别的包同第一包的偏差之积的总和作为发送时间指派给每个所识别的包。

依据另一个实施例，提供了依据变化的用户指定动态而无缝地改变再分多路复用的方法。诸如第一适配器等接口依据对再分多路复用的 TS 内容的初始用户指定从 TS 中选择性地仅提取一些特定的传输包。诸如第二适配器等第二接口依据对再分多路复用的 TS 内容的初始用户指定，把从所提取的传输包和包含 PSI 的传输包(如果有的话)中选中的那些传输包重新装配成输出的再分多路复用的 TS。此外，第二适配器输出重新装配的经再分多路复用的 TS 作为连续位流。处理器动态地接收对再分多路复用的 TS 内容的一个或多个新的用户指定，它们指定了以下的一个或多个：(I)待提取的不同传输包和/或(II)待重新装配的不同传输包，与此同时第一和第二适配器提取传输包进行重新装配，并输出再分多路复用的 TS。与之响应处理器使第一和第二适配器动态地停止依据初始用户指定对传输包所进行的提取或重新装配，且动态地开始依据新的用户指定对传输包所进行的提取或重新装配，而不在所输出的再分多路复用的传输流中引入不连续。例如，处理器可按每个新的用户指定而产生引用不同传输包的替代 PSI，以由第二适配器进行重新装配。

示例地，可使用此无缝再分多路复用的改变技术来自动地保证在再分多路复用的 TS 总是输出每个选中程序的正确的 ES 信息，尽管构成该程序的 ES 中可能有任何变化。可设置产生用户指定的控制器，该用户指定指示将在输出 TS 中输出的输入 TS 的一个或多个程序。第一适配器连续地捕获-输入 TS 的程序定义。处理器从俘获的程序定义(这些流元构成每个程序)中进行连续地确定。第二适配器在输出的 TS 中输出传输包，每个传输包包含每个 ES(这些 ES 被确定为构成由用户指定要输出的每个程序)的 ES 数据，而不在输出的 TS 中引入不连续。因而，即使构成每个程序的 ES 的 PID 改变(数目或值)，但

在输出的 TS 中总是输出每个程序的正确的完整 ES 数据。

依据再一个实施例，提供了一种优化其中插入空传输包的 TS 的端口的方法。第一接口(适配器)以预定位速率接收 TS，该 TS 包括带有可变压缩程序数据的传输包以及一个或多个空的传输包。当不能获得插入在各个传输包时隙处接收到的 TS 中的带有压缩程序数据的传输包时，把每个空的传输包插入接收到的 TS 的一个时隙中。处理器以其它带有待再分多路复用数据的传输包来替换一个或多个空的传输包。带有这种替换数据的传输包可包含 PSI 数据或者甚至突发(bursty)事务处理(transactional)数据，这些突发事务处理数据对以连续方式呈现信息没有位速率或发送等待时间(latency)的要求。

示例地，处理器从接收到的 TS 中提取选中的一些传输包，并丢弃每个未选中传输包(包括每个空传输包)。由处理器和第一适配器把选中的传输包存储在存储器中。如上所述，处理器调度每个所存储的传输包，从而在以接收到每个所存储的传输包的时间为依据的时间在输出的传输流中输出。第二接口(适配器)以相应于此调度的时隙输出每个所存储的传输包。如果没有要调度在输出 TS 的时隙之一处输出的传输包，则第二适配器输出空传输包。然而，与每个输入的 TS 相比，空的传输包在输出的 TS 中占据较少的带宽。

依据附加的实施例，提供了一种在异步通信链路上及时地输出带有压缩程序数据的位流的方法。同步接口(适配器)提供了包含传输包的位流。处理器把派送时间指派给一个或多个选中传输包中的每一个，以保持一程序(每个被选中的传输包携带有该程序的数据)的预定位流，并引起每个选中传输包的平均等待时间。在以每个派送时间为依据的时间，异步通信接口接收一个或多个命令，并通过在近似于这些派送时间的时候发送相应的选中传输包作为响应，从而把选中传输包的抖动减到最少。

示例地，如下产生命令。处理器对包含以上派送时间的发送描述符进行排队而成为发送队列。处理器指派再分多路复用器节点的适配器代表异步接口服务于发送队列。当描述符的派送时间等于适配器处的基准时钟的时间时，所指派的适配器的数据链路控制电路使得每个命令发出。

可使用这些技术中的各种技术来使能网络分布式再分多路复用。网络设有一个或多个通信链路和通过通信链路互连成一通信网络的多个节点。目的地节点经由通信链路之一接收包含一个或多个程序的数据的第一位流，第一位流的各部分具有一个或多个预定的位速率。目的地节点可以是如上所述的

再分多路复用器节点，在任何情况下它都包括处理器。处理器至少选择接收到的第一位流的一部分用以发送，并调度第一位流的选中部分的发送，从而以所述第一位流的选中部分的预定速率为依据的速率在 TS 中输出第一位流的选中部分。

或者，通信链路集中地形成共享通信媒体。把这些节点分割成用于把一个或多个位流发送到此共享通信媒体上的第一组一个或多个节点以及用于接收从共享通信媒体发送的位流的第二组一个或多个节点。第二组节点选择所发送的位流的部分并发送作为包含选中部分的位流的一个或多个再分多路复用的 TS。每个所发送的再分多路复用的 TS 不同于发送位流中接收到的位流。设置了选择第一和第二组节点并使选中的节点依据多个不同的信号流模式之一经由共享通信媒体传送位流，这些信号流模式包括不同于节点至共享通信媒体的拓扑连接的至少一个信号流模式。

最后，还提供了一种使得在再分多路复用系统中接收或发送传输包的多个电路中的每一个处的基准时钟同步的方法。接收传输包的每个电路处的基准时钟指示在该处接收到每个传输包的时间。发送传输包的每个电路处的基准时钟指示何时从该处发送每个传输包。指定使这些基准时钟相互同步的主基准时钟。周期性地获得主基准时钟的当前时间。依据其它基准时钟处的各个时间与主基准时钟的当前时间之差来相互调节基准时钟，从而使各基准时钟的时间与主基准时钟的相应时间匹配。

因而，依据本发明，提供了一种更灵活的再分多路复用系统。此增加的灵活性增强了再分多路复用也降低了整个系统的成本。

附图概述

图 1 示出依据本发明另一个实施例的再分多路复用环境。

图 2 示出依据本发明一个实施例的使用异步平台的再分多路复用器节点。

图 3 示出一流程图，该流程图示意地示出依据本发明的一个实施例，如何根据传输包在再分多路复用节点中的 PID 对传输包进行处理。

图 4 示出依据本发明一个实施例的网络分布式再分多路复用器。

本发明的较佳实施方式

为了清楚，把对本发明的描述分成几部分。

再分多路复用器环境和概述

图 1 示出依据本发明一个实施例的基本再分多路复用环境 10。控制器 20 使用例如任何远程过程调用(RPC)协议向再分多路复用器 30 提供指令。可使用的 RPC 的例子包括数字分布式计算环境协议(DEC)和开放式网络计算协议(ONC)。DEC 和 ONC 是利用协议堆栈的网络协议，这些协议堆栈使客户进程(client process)可执行位于同一平台(例如，控制器 20)或另一远程平台(例如，在再分多路复用器 30 中)的子程序。换句话说，客户进程可通过简单的子程序调用来发出控制指令。DCE 或 ONC 进程向再分多路复用器 30 发出适当的信号和命令以实行想要的控制。

控制器 20 可以是诸如 PC 兼容计算机等计算机形式。控制器 20 包括连到总线 24 的诸如一个或多个 Intel™ Pentium II™ 集成电路等处理器 21、主存储器 23、磁盘存储器 25、监视器和键盘/鼠标器 27 以及一个或多个 I/O 器件 29。I/O 器件 29 是依据如何实现再分多路复用器 30 而与再分多路复用器 30 进行通信的任何适当的 I/O 器件 29。这种 I/O 器件 29 的例子包括 RS-422 接口、以太网接口、调制解调器和 USB 接口。

以一个或多个网络连接的“黑盒”来实现再分多路复用器 30。在以下所述的示例再分多路复用器的体系结构中，再分多路复用器 30 的黑盒可以是通过诸如以太网、ATM 或 DS3 通信链路等通信链路互连的独立 PC 兼容计算机。例如，再分多路复用器 30 包括一个或多个黑盒，每个黑盒是通过以太网网络(10 BASE-T、100 BASE-T 或 1000 BASE-T 等)互连的独立 PC 兼容计算机。

如图所示，在再分多路复用器 30 处接收到一个或多个待再分多路复用的 TS，即 TS1、TS2 和 TS3。由于再分多路复用器 30 的再分多路复用操作的结果，从再分多路复用器 30 输出一个或多个 TS,即 TS4 和 TS5。再分多路复用的 TS(示例的 TS4 和 TS5)包括来自输入 TS(TS1、TS2 和 TS3)的至少一些信息(至少一个传输包)。还设置至少一个存储器件 40，例如磁盘存储器或服务器。存储器件 40 可产生作为输入的待再分多路复用的信息的 TS 或数据，以经再分多路复用器 30 再分多路复用成为输出的 TS(TS4 或 TS5)。同样，存储器件 40 可存储再分多路复用器 30 所产生的 TS 信息或数据，诸如从输入的 TS(TS1、TS2 或 TS3)中提取或复制的传输包，在再分多路复用器 30 处接收到的其它信息或由再分多路复用器 30 所产生的信息。

还示出一个或多个数据注入源 50 和一个或多个数据提取目的地 60。这些

源 50 和目的地 60 本身可以实现为 PC 兼容计算机。然而，源 50 还可以是诸如摄像机、视频磁带播放装置、通信解调器/接收机等器件，目的地 60 可以是显示接收器、视频磁带记录装置、通信调制器/发射机等器件。数据注入源 50 把 TS、ES 或其它数据提供给再分多路复用器 30，用以例如把它们再分多路复用成为输出的 TS，TS4 和/或 TS5。同样，数据提取目的地 60 接收来自再分多路复用器 30 的 TS、ES 或其它数据，例如从输入 TS(TS1、TS2 和/或 TS3)中所提取的。例如，可设置一用于产生每个输入的待再分多路复用的 TS(TS1、TS2 和 TS3)的数据注入源 50，可设置一用于接收每个输出的经再分多路复用的 TS(TS4 和 TS5)的数据提取目的地 60。

可把环境 10 看作一网络。在此情况下，可把环境 10 中的控制器 20、每个数据注入源 50、存储器件 40、数据提取目的地 60 和再分多路复用器 30 的每个“网络的黑盒”看作该通信网络的一个节点。每个节点可由同步或异步通信链路连接。此外，把器件 20、40、50 和 60 从再分多路复用器 30 分离仅仅是为了方便。在另一个实施例中，器件 20、40、50 和 60 是再分多路复用器 30 的一部分。

再分多路复用器的体系结构

图 2 示出再分多路复用器 30 的网络黑盒或节点 100(以下叫做“再分多路复用器节点”100)之一的基本体系结构。图 2 所示的特定再分多路复用器节点 100 可用作整个再分多路复用器 30。或者，从以下讨论中很明显的是，再分多路复用器节点 100 的不同部分可分布于分开的节点中，这些节点通过同步或异步通信链路相互互连。在又一个实施例中，具有与图 2 所示相同的体系结构的多个再分多路复用器节点 100 经由同步或异步通信链路相互互连，且可被编程为一致地行动。在这里把后两个实施例叫做网络分布式再分多路复用器。

示例地，再分多路复用器节点 100 是 Windows NT™ 兼容的 PC 计算机平台。再分多路复用器节点 100 包括一个或多个适配器 110。每个适配器 110 连到总线 130，示例的总线是 PCI 兼容总线。主存储器 120 也连到总线 130。诸如 Intel™ Pentium II™ 集成电路等处理器也连到总线 130。应注意，图 2 所示的单个总线体系结构可以是更复杂的多总线结构的简化表示。此外，可存在不止一个处理器 160，这些处理器在进行以下所述的处理功能时相互协作。

示例地，设有两个接口 140 和 150。这两个接口 140 和 150 连到总线 130，

虽然事实上它们也可直接连到 I/O 扩充总线(未示出)，继而此 I/O 扩充总线经由 I/O 桥路(未示出)连到总线 130。示例地，接口 140 是诸如以太网接口等异步接口。这意味着，不保证经由接口 140 发送的数据准确地在任何时间出现，该数据可能经历可变端-端延迟。另一方面，接口 150 是诸如 T1 接口等异步接口。使得在连到接口 150 的通信链路上的通信与接口 150 处所保持的时钟信号同步。在特定时间经由接口 150 发送数据，该数据经历恒定的端-端延迟。

图 2 还示出，再分多路复用器节点 100 可具有任选的加扰器/解扰器(它们可实现为加密器/解密器)170 和/或全球定位卫星(GPS)接收机 180。加扰器/解扰器 170 用于对传输包中的数据进行加扰或解扰。GPS 接收机 180 用于接收统一时钟信号，以使再分多路复用器节点 100 同步。以下更详细地描述这些器件的目的和操作。

每个适配器 110 是专门类型的同步接口。每个适配器 110 具有一个或多个数据链路控制电路 112、基准时钟发生器 113、一个或多个描述符和传输包高速缓冲存储器 114、任选的加扰器/解扰器 115 和一个或多个 DMA 控制电路 116。这些电路可以是一个或多个处理器的一部分。最好，它们使用有限状态自动机即在一个或多个 ASIC 或门阵列(PGA、FPGA 等)中实现。以下将描述这些电路中每一个电路的目的。

示例地，基准时钟发生器 113 是以 27MHZ 计数的 32 位翻转计数器。可在数据链路控制电路 112 处接收到基准时钟发生器 113 所产生的系统时间。此外，处理器 160 可如下直接访问基准时钟发生器 113。处理器 160 可从基准时钟发生器 113 的 I/O 寄存器读取当前系统时间。处理器 160 可把特定值装入基准时钟发生器 113 的同一 I/O 寄存器。最后，处理器 160 可在调节寄存器中设定基准时钟发生器的计数频率，从而基准时钟发生器 113 以特定范围内的频率进行计数。

高速缓冲存储器 114 的目的是暂时存储待输出的下一个或多个来自适配器 110 的待命输出的传输包或最近在适配器 110 处接收到的最后一个或多个传输包。使用高速缓冲存储器 114 使得可以最少的等待时间(最显著的是不发生通过总线 130 的传送等待时间)接收和存储或检索和输出传输包。高速缓冲存储器 114 还存储每个传输包的描述符数据。以下更详细地描述这些描述符的目的和结构。此外，高速缓冲存储器 114 存储在正常操作下可由处理器 160

下载和修改的过滤器映射(filter map)。示例地，高速缓冲存储器 114 还可存储入以下更详细描述的用于加扰或解扰的控制字信息。除了处理器 160 以外，高速缓冲存储器 114 还被数据链路控制电路 112、DMA 控制电路 116 和任选的加扰器/解扰器 115 访问。

众所周知，高速缓冲存储器 114 可拥有主存储器 120 中数据的全真或修改的拷贝。同样，在需要时，高速缓冲存储器 114 应获得主存储器中任何数据的修改拷贝而不是其拥有的旧拷贝。主存储器 120 也是如此。利用“所有权协议”，从而只有单个器件(诸如高速缓冲存储器 114 或主存储器 120)才有许可在任一时刻修改数据存储单元的内容。这里，当高速缓冲器存储器具有修改这些存储单元内容的专有控制时就说成高速缓冲存储器 114 获得对数据存储单元的控制。通常，高速缓冲存储器 114 获得对存储单元以及存储在其中的数据的全真复制拷贝的控制，修改其拷贝，但推迟把数据的修改写到主存储器直到以后。这意味着，当高速缓冲存储器把数据写到主存储器中的存储单元时，高速缓冲存储器 114 放弃对主存储器 120 的控制。

DMA 控制电路 116 用于在主存储器 120 与高速缓冲存储器 114 之间传送传输包数据和描述符。DMA 控制电路 116 可把足够数目的传输包(及其描述符)保存在高速缓冲存储器 114 中，以使数据链路控制电路 12 连续地(即，以连续的时隙)输出在输出 TS 中的传输包。DMA 控制电路 116 还可获得对高速缓冲存储器 114 中的足够数目的描述符存储单元及其指向的包存储单元的控制。DMA 控制电路 116 获得对高速缓冲存储器 114 的这些描述符和传输包存储单元的控制。这使得可在接收到入局的传输包(即，从连续时隙)时把它们连续地分配给描述符和传输包存储单元。

数据链路控制电路 112 用于接收来自入局 TS 的传输包或在出局 TS 上发送传输包。在接收到传输包时，数据链路控制电路 112 仅滤出和保留从入局 TS 中接收到的在可下载过滤器映射(由处理器 160 所提供)中所指定的选中传输包。数据链路控制电路 112 丢弃每个其它的传输包。数据链路控制电路 112 把下一未使用描述符分配给接收到的传输包，并把接收到的传输包存储在高速缓冲存储器 114 中，以传送到所分配的描述符指向的传输包存储单元。此外，数据链路控制电路 112 从基准时钟发生器 113 中获得相应于传输包的接收时间的基准时间。数据链路控制电路 112 把此时间作为接收时间标记记录在指向其中存储有传输包的传输包存储单元的描述符中。

在发送包时，数据链路控制电路 112 从高速缓冲存储器 114 检索出局传输包的描述符，并以在基准时钟发生器 113 近似等于各描述符中所指示的派送时间时发生的出局 TS 的时隙发送相应的传输包。此外，数据链路控制电路 112 在必要时在输出的传输包中进行任何最终的 PCR 校正，从而传输包中所指示的 PCR 与出局 TS 中的传输包的精确对准同步。

处理器 160 用于接收来自外部控制器 20(图 1)的控制指令，并把命令发送到适配器 110 和接口 140 和 150 以控制它们。与之响应，对于这些指令，处理器 160 产生 PID 过滤器映射并把它下载到高速缓冲存储器 114，或者修改已存在于高速缓冲存储器 114 中的 PID 过滤器，以便使数据链路控制电路 112 在选择性地提取想要的传输包时使用。此外，处理器 160 产生用于处理每个接收到的传输包(根据其 PID)的中断接收处理程序。接收中断处理程序可使处理器 160 重新映射传输包的 PID、估计传输包的离开时间、提取传输包中的信息以进行进一步处理等。此外，处理器 160 制定和执行发送中断程序，该程序使处理器对传输包进行适当地排序以输出，对每个传输包产生离开时间，粗略地校正传输包中的 PCR 以及把 PSI 插入输出的 TS 中。处理器 160 还可有助于以下更详细所述进行加扰和解扰。

主存储器120用于存储传输包及与其相关的描述符。如下组织主存储器120的存储单元。设置缓冲器 122，它包含用作传输包池的多个可再使用的传输包存储单元。把描述符存储单元 129 组织成为多个环 124。每个环 124 是一描述符存储单元 129 的序列，从起始存储器地址或环的顶部 124-1 到结尾存储器地址或环的底部 124-2。对从再分多路复用器节点 100 发送的每个出局 TS 设置一个环 124，对在再分多路复用器节点 100 处接收到的每个入局 TS 设置一个环 124。可如以下更详细所述来设置其它环 124。

在每个环 124 中实现一队列，这是通过给队列头部或队列中的第一个已使用/分配的描述符存储单元 129 指定一指针 124-3 并给队列尾部或队列中的最后一个已使用/分配的描述符存储单元 129 指定一指针 124-4 来实现的。把描述符存储单元 129 分配给以刚好接在尾部 124-4 后的未使用/未分配描述符存储单元 129 为起始的入局传输包。从头部 124-3 所指向的描述符存储单元 129 起始并按序进到尾部 124-4 的队列中检索出局传输包的描述符存储单元 129。每当到达处于环 124-2 末尾的描述符存储单元 129 的描述符时，以位于环 124-1 顶部的描述符存储单元 129 的描述符继续从描述符存储单元 129 中分配或检

索描述符。

如图所示，存储在每个描述符存储单元 129 中的每个描述符包括许多字段 129-1、129-2、129-3、129-4、129-5、129-6、129-7、129-8、129-9 和 129-10。简而言之，这些字段中每个字段的目的如下。字段 129-1 用于存储命令属性。处理器 160 可使用命令属性字段的各位来控制适配器 110 的传输包发送和描述符数据检索。例如，处理器 160 可在环 124 的底部 124-2 所指向的描述符存储单元 129 中的描述符的字段 129-1 中预设一位，以指示顶部指针 124-1 所指向的描述符存储单元 129 是接在底部指针 124-2 所指向的描述符存储单元 129 后。

字段 129-2 用于存储软件状态位。这些位既不能被适配器 110 访问也不能被它修改，这些位可被处理器 160 用于不涉及适配器 110 的任何目的。

字段 129-3 用于存储待输出的出局传输包的字节数(通常对 MPEG-2 传输包为 188 字节，但在描述符依据不同传输协议或 “集中”和“分散”支持(把包分成多个存储单元片或从存储在多个包存储单元的片进行装配)而指向包时，可把字节数设定为更大或更小的数目)。

字段 129-4 用于存储描述符所对应的传输包存储单元的指针。这在图 2 中利用从环 124 的描述符存储单元 129 中的描述符至传输包池 122 的指定存储单元的箭头来示出。

字段 129-5 用于存储接收到的入局传输包的接收时间或存储待发送的出局传输包的派送时间。

字段 129-6 用于存储可能发生的各种异常/差错。可使用该字段的位来指示总线 130 的差错、数据链路控制电路 112 连接到的通信链路上的数据链路差错、短或长包(少于或超过 188 字节)的接收等。

字段 129-7 用于存储指示描述符的不同状态方面的状态位，诸如描述符是否有效、是否无效地指向一出错包等。例如，假定多个器件必须相继处理描述符和/或其指向的包。在此情况下，最好提供四个状态位。可把前两位设定到值 0、1、2 或 3。值 0 指示描述符无效。值 1 指示描述符有效并可被最后一个器件处理，该器件必须处理描述符和/或其指向的包。值 2 指示描述符有效且可被倒数第二个器件处理，该器件必须处理描述符和/或其指向的包。值 3 指示描述符有效且可被倒数第三个器件处理，该器件必须处理描述符和/或其指向的包。后两位指示是否已从主存储器 120 中取得描述符送入高速缓冲

存储器 114 以及是否已在适配器 10 处完成对描述符的处理并把它存储在主存储器 120 中。可如以下更详细地所述来设置其它状态位。

字段 129-8 包含指示接收到的入局传输包的字节数的传送计数。

字段 129-9 用于存储加扰/解扰控制字或用于加扰或解扰的其它信息。例如，处理器 160 可在此字段 129-9 存储一控制字(加密/解密密钥)或存储在高速缓冲存储器 114 中的控制字表的基址。

字段 129-10 用于存储经调度估计的离开时间，实际离开时间或实际接收时间。如以下更详细所述，处理器 160 使用该字段对接收到的入局传输包进行排序以输出或记录入局传输包的接收时间。

示例地，为了在单个输入端处接收传输包，需要一数据链路控制电路 112、一 DMA 控制电路 116 和一环 124，为了从单个输出端发送传输包，需要一数据链路控制电路 112、一 DMA 控制电路 116 和一环 124。这里把存储在与输入端有关的队列中的描述符叫做接收描述符，把存储在与输出端有关的队列中的描述符叫做发送描述符。如下所述，以上所指的送入和输出端可以是数据链路控制电路 112 连接到的通信链路的输入或输出端或再分多路复用器节点 100 中的另一接口 140 或 150 的通信链路的输入或输出端。把适配器 110 示作仅有单个数据链路控制电路 112 和单个 DMA 控制电路 116。这只是为了说明——可在同一适配器 110 上设置多个数据链路控制电路 112 和 DMA 控制电路 116。或者，此外，可在再分多路复用器节点 100 中设置多个适配器 110。

基本传输包接收、再分多路复用和发送

现在考虑再分多路复用器节点 100 的操作。给操作人员提供如何操作再分多路复用器节点 100 的许多选择。在再分多路复用器节点 100 的第一种操作方式中，假设操作人员希望把两个 TS(即，TS1 和 TS2)的程序信息选择性地组合成为第三 TS，即 TS3。在此情况下，假设操作人员最初不知道这两个待再分多路复用的 TS(TS1 和 TS2)中包含什么程序，ES 还是 PID。此外，示例地，在第一适配器 110 处接收 TS1，在第二适配器 110 处接收 TS2，以及从同一再分多路复用器的节点 100 的第三适配器 110 发送 TS3。从以下的描述可以理解，TS1 和 TS2 中的每一个可改为经由同一节点或不同节点处的同步或异步接口接收，可经由任意结构的网络把 TS1 和 TS2 的选中部分传送到第三节点，以在第三节点处选择性地组合而形成 TS3。

可把依据此方式的操作归纳为(1)获取输入的待再分多路复的 TS(TS1 和

TS2)的内容信息(程序，ES、PAT、PMT、CAT、NIT 等及其 PID)；(2)把此内容信息报告给操作人员，从而操作人员可制定用户规定；以及(3)接收用于构成输出的经再分多路复用的 TS(TS3)的用户规定，并依据此用户规定从输入的待多路复用的 TS(TS1 和 TS2)的内容动态地构成经再分多路复用的 TS(TS3)。

为了使能内容信息的获取，传输处理器 160 把一接收队列分配给分别接收 TS(TS1 和 TS2)的第一和第二适配器 110 中的每一个。为了获取 TS(TS1 和 TS2)的内容，最初在 TS1 或 TS2 的适配器 110 处不丢弃传输包。因而，处理器 160 把一过滤器映射装入接收 TS(TS1 和 TS2)的第一和第二适配器 110 中每一个的高速缓冲存储器 114，使得可保留每个传输包并传送到主存储器 120。当在各适配器 110 处接收到 TS(例如，TS1)的每个传输包时，数据链路控制电路 112 把下一个未使用的描述符(接在存储在接收队列尾部 124-4 处的描述符存储单元中描述符后)分配给接收到的入局传输包。数据链路控制电路 112 把每个接收到的传输包存储在所分配的描述符指向的高速缓冲存储器 114 中的传输包存储单元中。

DMA 控制电路 116 把每个传输包写到主存储器 120 中池 122 的相应存储单元，并把分配给这些传输包的描述符的描述符数据写到接收队列的各描述符存储单元。此外，DMA 控制电路 116 可获得对接收队列中接着的几个未分配描述符存储单元 129(接在 DMA 控制电路 116 先前已获得控制的描述符 129 序列的存储单元后)、存储在其中的描述符的拷贝和这些描述符指向的传输包存储单元的控制。把对这些未使用、未分配的描述符和传输包存储单元的控制提供给高速缓冲存储器 114，以被数据链路控制电路 112 使用(即，分配给将来从 TS1 接收到的传输包)。

在 DMA 控制电路 116 把 $i \geq 1$ 个传输包和分配给它们的描述符的数据写到池 122 和接收队列中，DMA 控制电路 116 产生一中断。示例地，操作人员可使用控制器 20 来选择数字 i 并通过处理器 160 来设定。中断使得处理器 160 对每个接收到的传输包执行适当的接收“PID”处理器子程序(handler subroutine)。或者，可使用诸如轮询或基于定时器的过程等另一种技术启动处理器 160 对每个接收到的传输包执行接收 PID 处理器子程序。为了清楚，在这里使用中断范例来说明本发明。参考图 3，示例地，处理器 160 具有用于在再分多路复用会话(session)期间接收或发送 TS 的每个适配器 110(或其它器件)的一组 PID 处理器子程序。图 3 示出两种类型的 PID 处理器子程序组，

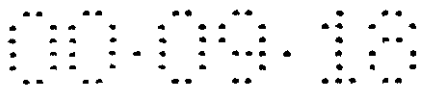
即接收 PID 处理器子程序组和发送 PID 处理器子程序组。每个 DMA 控制电路 116 产生可识别的不同中断，从而使处理器 160 确定使用哪一组 PID 处理器子程序。响应于 DMA 控制电路 116 的中断，处理器 160 执行步骤 S2，依据该步骤 S2，处理器 160 检查在中断适配器 110 的接收队列中最近存储的描述符所指向的每个传输包的 PID。对于每个 PID，处理器 160 咨询中断处理器 160 的适配器 110(或其它器件)所专用的接收 PID 处理器子程序 402 的指针表。

假设，接收 TS1 的第一适配器 110 中断处理器 160，在此情况下，处理器 160 确定咨询接收该 TS(TS1)的适配器 110 所专用的接收 PID 处理器子程序 402 的指针表。接收 PID 处理器子程序的指针表包括 8192 个表目，包括被每个可许可 PID(依据 MPEG-2，这些 PID 有 13 位)索引(index)的一个表目。每个被索引的表目包含将被处理器 160 执行的子程序的指针或地址 RIV0、RIV1、…、RIV8191。使用每个传输包的 PID，处理器 160 可给接收 PID 处理器子程序 402 的指针表的表目加上索引，以识别将对该特定传输包执行的子程序的指针。

利用指针表 402，把由各指针指向并被处理器 160 所执行的每个子程序具体地映射到每个 PID，以实现用户规定。有利的是，由指针表 402 依据用户规定预先确定每个子程序并进行简单的映射。每个子程序由一个或多个基本构件块处理的集合构成。基本构件块处理的一些例子包括：

(1) PAT 获取：最初，该处理包含在由 RIV0 指向的子程序即 PID 0x0000 的接收 PID 处理器子程序中。在执行该处理时，示例地，处理器 160 提取当前被处理的传输包中所携带的 PAT 部分(section)，并把此 PAT 部分装入保持在存储器中的 PAT 中。注意，可使用 PAT 的多个版本，因为 TS 中所携带的程序可随时间而改变。处理器 160 能识别 PAT 的不同版本并分开地聚集 PAT 的每个版本的拷贝并保持在主存储器 120 中。处理器 160 能根据包含在各种 PAT 部分中的信息在任何时间识别目前在使用 PAT 的哪一个版本。处理器 160 还使用每个更新的 PAT 部分中所携带的信息来识别此时 TS 中所携带的程序的程序号以及此类程序号的 PMT 部分的 PID 或程序定义。使用这些程序号，处理器 160 可修改接收 PID 处理器子程序的指针表 402，以插入适当 PID 的指针(给带有 PMT 部分的传输包作标签)，以执行包含用于获取 PMT 部分/程序定义的处理的子程序。

(2) PMT 部分/程序定义获取：在此处理中，处理器 160 提取包含在目前处理的传输包中的 PMT 部分或程序定义，并以所提取的程序定义或 PMT 部分



数据更新 PMT 的各部分。同 PAT 一样，可利用 PMT 的多个版本，处理器 160 可确定在哪一个 PMT 中存储所提取的 PMT 部分或程序定义数据。处理器 160 可使用 PMT 信息来更新 PID 过滤器映射(用来丢弃未包含在经再分多路复用的 TS 中的程序的传输包)，识别用于对 ES 进行解扰的控制字并选择用于处理包含在传输包(具有在 PMT 中所识别的 PID)中的 PCR 的子程序。

(3) PID 重新映射：这使得处理器 160 以不同的 PID 改写相应包的 PID。这是保证 PID 指派的唯一性所需的。即，如果要把携带不同内容(例如，不同 ES 的数据、不同 PSI 流的数据等)的传输包再分多路复用成为同一输出的再分多路复用 TS(并由其携带)，则 MPEG-2 要求给这些携带不同内容的传输包标上互不相同的 PID。否则，解码器或其它器件不能区分携带不同类型数据的传输包来进行提取、解码等。在 TS1 中使用某一 PID 给带有第一类型数据的传输包作标签并在 TS2 中使用同一 PID 给带有第二类型数据的传输包作标签是可能的。如果要把第一和第二类型的传输包包含在输出的再分多路复用的 TS(TS3)中，则应以新的 PID 对两种类型的传输包中的至少一种重新作标签以保证唯一性。

(4) 传输包丢弃：如名称所示，处理器 160 简单地丢弃这些传输包。为此，处理器 160 解除分配指向丢弃传输包的描述符。可通过处理器 160 调节该队列的描述符存储单元 129 中所存在的描述符的顺序，以除去删除的传输包的描述符(例如，处理器识别接在环 124 中的待删除传输包的描述符后的所有已分配的描述符，并把每一个移至刚好在前的那个描述符的描述符存储空间)，这样来实现描述符的解除分配。描述符的解除分配在接收队列中产生了用以重新分配的描述符存储空间 129。

(5) PCR 标志(flag)设定：PMT 对每个程序指示携带 PCR 的传输包的 PID。然而，这些传输包中只有一些携带有 PCR。简单地，这可以通过处理器 160 确定在传输包中是否设定适当的指示(传输包标题中的 adaption_field_control(适应_字段_控制)位和适应字段中的 PCR_flag(PCR_标志)位)来确定。如果处理器 160 确定存在 PCR，则处理器 160 在有关各包的描述符 129 的属性字段 129-1 中设定一 PCR 标志位。以下更详细地描述此属性标志位的目的。

此外，示例地，处理器 160 计算基准时钟发生器 113 相对于程序(其 PCR 为样本)的编码器系统定时时钟的当前的偏移。可通过以下公式来确定偏移：

偏移＝ΔRTS12-ΔPCR12Δ

ΔRTS12＝RTS2-RTS1；以及

ΔPCR12＝PCR1-PCR2

这里：ΔPCR12 是该程序的各相继 PCR 之差，

PCR2 是当前处理的传输包中的 PCR，

PCR1 是先前接收到的该程序的 PCR，

ΔRTS12 是相继接收时间标记之差；

RTS2 是对包含 PCR2 的当前所处理的传输包所记录的接收时间标记，以及

RTS1 是包含 PCR1 的传输包的先前接收时间标记。

在计算了偏移后，把 PCR1 和 PTS1 分别设定为等于 PCR2 和 PTS2。如下所述使用此偏移来调节 PCRΔ(在必要时)。

(6) 估计的离开时间计算：依据此处理，处理器 160 估计传输包的(理想)离开时间。示例地，此处理包含在接收中断处理程序中，以把每个接收到的入局传输包再分多路复用成为出局 TS。可从传输包的接收时间(字段 129-5 中)和再分多路复用节点 100 处的已知内部缓冲延迟来估计所估计的离开时间。处理器 160 把所期望的离开时间写入字段 129-10。

(7) 加扰/解扰控制字信息插入：通常，在加扰或解扰技术中，实际上需要动态变化的控制字(诸如加密或解密密钥)对传输包中的数据进行加扰或解扰。普通的加扰和解扰技术依据此使用奇数和偶数密钥，一个密钥用于对 ES 数据进行加密，在 TS 中同期地传送随后将使用的下一个密钥。然后，发送指示现在应使用最近传送的密钥的信号。加扰/解扰控制字可以是 ES 专用的或可用于一组 ES(整个“有条件访问系统”上)。可把解扰或加扰控制字保持在再分多路复用器节点 100 处的 PID 可编索引表中。如以下更详细所述，处理器 160 在执行此处理时可把控制字表的基址或控制字本身插入描述符的字段 129-9。

最初，处理器 160 选择用于获取每个接收到的 TS(TS1 和 TS2)的 PAT 并且其后丢弃每个经处理的传输包的 PID 处理程序。在接收 PAT 期间，获得诸如程序定义/PMT 部分、NIT 和 CAT 等带有其它 PSI 的 PID 的传输包以及诸如 ES 流、ECM 流、EMM 流等其它流的 PID 等。示例地，用于 PAT 的 PID 的接收 PID 处理器子程序选择用于获取 PMT、NIT、CAT 等的接收 PID 处理器子程序。这

可容易地通过使这些子程序可用并简单地改变指向这些 PID 处理器子程序的表 402 中表目的指针(可通过适当的识别 PID 编索引)来实现。注意，即使在接收 TS1 和 TS2 的传输包并进行处理时，也可动态地实行这种简单的 PID 处理器子程序选择处理。以下更详细地描述其优点。

最终，获取有关每个 TS(TS1 和 TS2)的足够数量的 PSI，以使操作人员产生将在再分多路复用 TS(TS3)中输出的信息的用户规定。示例地，处理器 160 例如使用异步接口 140 把获取的 PSI 信息发送到控制器 20。把用于选择用户规定的足够的信息发送到控制器 20。此信息可以是选择性的，例如，刚好是示出包含在其中的程序号的每个 TS 的通道映射及不同类型的 ES(以诸如视频、音频、第二音频呈现、关闭的图片说明文本等来描述)。或者，该信息可以是穷举的，例如包括每个程序的 PID、其 ES 的 ECM 等，控制器 20 以相干而有用的方式把该信息简单地显示给操作人员。

使用所提供的信息，操作人员产生用户规定以输出到待再分多路复用的 TS(TS3)中。此用户规定可指定：

(1) 待保留并在再分多路复用 TS(TS3)中输出的每个 TS(TS1 和 TS2)中的程序号，

(2) 待保留或丢弃的保留程序的 ES，

(3) 待解扰和/或加扰的 ES、ES 组、程序或程序组以及将在对每个 ES、ES 组、程序或程序组进行加扰时使用的控制字的源，

(4)待注入或包含在输出的再分多路复用 TS(TS3)中的任何新的 ECM 或 EMM，以及

(5)并非从以上选择中自动暗示的任何新 PSI 信息，这些选择诸如待置于输出的 TS(TS3)中的 NIT 或 CAT、待重新映射的特定 PID 及其应重新映射到的新 PID、指派给在再分多路复用器节点处产生并在 TS(TS3)中所携带的其它信息(例如，如下所述的突发数据)的 PID 等。

然后，例如经由异步接口 140 把用户规定从控制器 20 发送到再分多路复用器节点 100。

处理器 160 接收此用户规定，并通过对每个接收到的待再分多路复用的 TS(TS1 和 TS2)的适当 PID 选择适当的接收 PID 处理器子程序来响应。例如，对于标注包含待保留数据的传输包用的每个 PID，处理器 160 选择一子程序，在子程序中，处理器 160 插入估计离开时间的过程。对于标注包含被加扰数

据的传输包用的每个 PID，处理器 160 选择一子程序，该子程序包含用于选择适当的控制字并把该控制字插入与此传输包有关的描述符中的过程。对于标注包含 PCR 的传输包用的每个 PID，处理器 160 可选择一子程序，该子程序包含用于设定 PCR 标志并计算偏移等的过程。以下更详细地描述用户规定和/或 PSI 数据的动态调节。

处理器 160 把一发送队列分配给发送经再分多路复用的 TS 的每个器件，即输出 TS(TS3)的第三适配器 110。此外，处理器 160 把 PID 过滤器映射装入接收 TS(TS1 和 TS2)的第一和第二适配器 110 的每个高速缓冲存储器 114，此 TS(TS1 和 TS2)具有适当的值，这些值用于保留待在再分多路复用的 TS(TS3)中输出的那些传输包、用于保留包含 PSI 的其它传输包、用于保持 TS1 和 TS2 的内容的跟踪(track)以及用于丢弃每个其它(each other)传输包。

除了选择接收 PID 处理器子程序、分配发送队列以及装载适当的 PID 过滤器映射修改以外，示例地，处理器 160 对输出经再分多路复用的 TS 的每个适配器(或其它器件)选择一组发送 PID 处理器子程序。这如图 3 所示。发送 PID 处理器子程序以 PID 和发送 TS 为基础进行选择。如上所述，响应于接收可识别的中断(例如，来自发送诸如 TS3 等输出 TS 的适配器 110 的数据链路控制电路 112)，处理器 160 执行步骤 S4。在步骤 S4，处理器 160 检查来自接收队列(和/或包含还未被调度输出的传输包的描述符的其它可能的队列)的描述符，并识别指向将从中断适配器 110 输出的传输包的高达 $j \geq 1$ 个描述符。示例地，数目 j 可以是可编程的，且有利地可把它设定为等于从特定适配器 110(每次特定适配器 110 中断处理器 160 之间，从该特定适配器 110 发送一输出 TS)所发送的传输包的数目 k。

在执行步骤 S4 时，处理器 160 检查指向旨在特定输出 TS 的传输包的描述符的每个接收队列。处理器 160 通过咨询发送 PID 处理器子程序 404 的指针表来确定哪些传输包指针输出 TS。如同表 402，表 404 对于每个 PID 包括一个表目并用 0x0000 到 0x1FFF 对每个 PID 索引。每个被索引的表目包含 TIV0、TIV1、...、TIV8181(待响应于各 PID 执行的子程序)的指针或地址。处理器 160 依据从控制器 20 接收到的用户规定制定发送 PID 处理器子程序 404 的指针表，如以下所述来修改此指针表。

以下是可组合成发送 PID 处理器子程序的示例过程：

(1) 什么也没有(nothing)：如果不在向处理器 160 发出发送中断的器件

的再分多路复用 TS(或其它流)中输出当前传输包，则此传输包的 PID 映射到仅包含此过程的子程序。依据此过程，处理器 160 简单地跳过此传输包及其描述符。不把被检查的描述符计为待从中断处理器 160 的特定适配器 110 输出的 j 个传输包之一。

(2) 用于发送的顺序描述符：如果要在向处理器发出发送中断的器件的再分多路复用 TS(或其它流)中输出当前传输包，则此传输包的 PID 映射到包含此过程(以及可能的其它过程)的子程序。依据此过程，处理器 160 把一发送描述符分配给此传输包。然后，处理器 160 把指向该传输包的接收描述符中的相关信息拷贝到此新分配的发送描述符中。然后，以发送队列中与请求中断的器件有关的适当顺序对所分配的发送描述符进行排序，以进行发送。尤其是，处理器 160 把新分配的描述符指向的包的估计离开时间与记录在发送队列中的其它描述符中的实际派送时间(将发送传输包的实际时间)相比较。如果可能，把此描述符置于发送队列中实际派送时间比此描述符的估计离开时间晚的每个描述符前及实际派送时间比此描述符的估计离开时间早的每个描述符后。可通过把实际派送时间比待插入描述符的估计离开时间晚的发送描述符序列中的每个发送描述符拷贝到该队列中各后续的下一描述符存储单元 129 来实现此插入。然后，可把所分配的发送描述符的数据存入可通过拷贝该序列获得的描述符存储单元 129 中。

(3) 实际派送时间确定：处理器 160 可确定所分配的描述符指向的传输包的实际派送时间，此确定根据该传输包的估计离开时间。通过确定用于发送传输包(新分配和插入的发送描述符指向该传输包)的输出再分多路复用 TS(TS3)的传输包时隙来设定实际派送时间。即，选中在时间上最接近估计的离开时间的输出 TS(TS3)的传输包时隙。假定将在被选中的传输包时隙的时间(相对于适配器 110(这些适配器如下所述相互同步)的基准时钟发生器 113 所建立的内部基准时间)处输出该传输包。指派有关各传输包时隙的时间作为实际派送时间。然后，把此实际派送时间存入发送描述符的字段 129-5。如下所述，实际派送时间实际上是第三适配器 110(该适配器输出经再分多路复用的 TS，即 TS3)的数据链路控制电路 112 提交相应传输包以输出的近似时间。传输包的实际输出时间与处理器 160 所不知的外部时钟所建立的传输包时隙的对准有关。可实施如下所述的附加步骤，以消除此不对准而引起的 PCR 的抖动(dejitter)。

考虑从中接收包的 TS(即，TS1 或 TS2)的位速率可能不同于输出的 TS(即，TS3)的位速率。此外，将从内部缓冲传输包至一预定延迟(与接收和发送队列的长度有关)。然而，假设在输出的再分多路复用 TS(TS3)的同一传输包时隙接收到的不同 TS 的传输包之间不存在争用，则所有的传输包将在多路复用器假定 100 中引起近似相同的等待时间。由于平均等待时间是相同的，所以不会在传输包中引入抖动。

现在考虑在几乎相同的时间从不同 TS(即 TS1 和 TS2)中接收两个传输包并在再分多路复用的 TS(TS3)中输出这两个传输包的情况。 这两个传输包可能具有不同的估计离开时间，尽管如此，此估计离开时间相应于(在时间上最接近于)输出的再分多路复用 TS(TS3)的同一传输包时隙。把具有最早估计离开时间(或接收时间)的传输包指派给此时隙和此时隙的实际派送时间。给另一传输包指派输出的再分多路复用 TS(TS3)的下一传输包时隙及其实际派送时间。注意，指派给下一时隙的传输包所引起的等待时间不同于该程序的其它传输包所引起的平均等待时间。因而，示例地，处理器 160 设法消除此传输包引起的等待时间，包括调节此传输包的 PCR(如果其中包含 PCR)。

(4) PCR 偏移和等待时间调节：示例地，此过程包含在由包含 PCR 的传输包的 PID 来索引的表 404 的指针所指向的子程序中。处理器 160 确定仅在未把一传输包指派给在时间上最接近该传输包的估计离开时间的输出再分多路复用 TS(TS3)的传输包时隙(对该程序的其它传输包也如此)时以及在各接收描述符中设定 PCR 标志时才需要 PCR 等待时间调节。校正 PCR 中因指派给不理想时隙引起的时间偏离。此调节等于传输包的时隙偏离理想时隙的时隙数乘以时隙的时间。

如下所述调节所有 PCR 的偏移，除非输入和输出 TS 在时间上精确对准或此 PCR 是从异步通信链路接收到的。在前一情况下，内部时钟的偏移不影响输出 PCR 的定时。在后一情况下，如下所述使用不同的偏移调节。在其它所有的情况下，输出接收到的 PCR 的时间均受到接收该传输包的适配器 110 和发送此传输包的适配器 110 的基准时钟发生器 113 相对于 PCR 的程序时钟的偏移的影响。即，给包含 PCR 的传输包标上从基准时钟发生器 113 获得的接收时间标记。此接收时间标记用来确定估计离开时间和实际派送实际。如以下详细地所述，相对于发送 TS(TS3)的适配器 110 上的基准时钟发生器 113 和保持同步的所有适配器 110 的所有基准时钟发生器 113，依据传输包的实际派

送时间来派送传输包。然而，虽然基准时钟发生器 113 都相互同步，但基准时钟发生器 113 仍旧要经受相对于产生传输包及其 PCR 的编码器系统定时时钟的偏移。此偏移可影响从再分多路复用器节点 100 输出的输出再分多路复用 TS(诸如 TS3)中的每个 PCR 的时间。

依据本发明，再分多路复用器节点 100 校正这些偏移。如上所述，用于每个程序的 PCR 的接收处理器子程序的部分将保持对偏移的当前测量。保持测量基准时钟发生器 113 相对于每个程序的编码器系统定时时钟的偏移。对于每个 PCR，从 PCR 中减去 PCR 的程序的当前偏移(即，基准时钟发生器 113 与该程序的编码器系统定时时钟之间的偏移)。

通过如上所述分配队列、选择 PID 处理器子程序及修改 PID 过滤器映射，可如下进行再分多路复用。在第一适配器 110 的数据链路控制电路 112 处接收到 TS1 的传输包。同样，在第二适配器 110 的数据链路控制电路 112 处接收到 TS2 的传输包。第一和第二适配器 110 中每一个中的数据链路控制电路 112 咨询存储在当地的(thereat)高速缓冲存储器 114 中的本地 PID 过滤器映射，并选择性地丢弃具有指示将不保留传输包的 PID 的每个传输包。每个数据链路控制电路 112 从高速缓冲存储器 114 中检索下一未使用/未分配的描述符，并确定与此描述符有关的传输包存储单元。(如以上和以下所述，DMA 控制电路 116 连续地获得对指派给数据链路控制电路 112 的接收队列的一个或多个后续未使用/未分配描述符构成的序列及这些描述符指向的传输包存储单元的控制。)这些后续未使用、未分配描述符接在存储在尾部指针 129-4 所指向的描述符存储单元 129 中的描述符后，数据链路控制电路 112 可获得此尾部指针 129-4。(如上所述，如果尾部指针 129-4 等于环地址的底部 129-2，则被尾部指针 129-4 指向的描述符将具有由处理器 160 在字段 129-7 中所设定的描述符环命令位的结尾。这将使数据链路控制电路 112 使用环绕式寻址技术来分配存储在描述符存储等于 129 中环地址顶部 129-1 的描述符。)数据链路控制电路 112 获得相应于接收到传输包的第一字节的时间的基准时钟发生器 113 的时间，并把该值作为接收时间标记存储在所分配的描述符的字段 129-5 中。数据链路控制电路 112 把接收到的传输包的字节数存储在字段 129-8 中。此外，如果在接收传输包时发生任何差错(例如，丢失 TS1 的数据链路载波(carrier)、短包、长包、出错包等)，则数据链路控制电路 112 通过设定 129-6 的适当异常位来指示这些差错。然后，数据链路控制电路 112 在状态字段 129-7

中设定一位指示已处理描述符 129 或已处理描述符 129 的差错，并把该传输包存储在字段 129-4 中的指针所指向的高速缓冲存储器 114 中的传输包存储单元中。(注意，在长包的情况下，可把不止一个后续未使用未分配描述符构成的序列分配给接收到的传输包，可把过剩的数据存储在与这些描述符有关的包存储单元中。可在描述符中第一个的属性字段 129-1 中设定适当的集中/分散位，以指示这个包的数据超过了与描述符中第一个有关的单个传输包存储空间。还可在描述符中最后一个的属性字段 129-1 中设定相应的位，以指示它是多描述符传送中的最后一个描述符。这种长的包通常发生在适配器来自不同于 TS 的流的包时。)

DMA 控制电路 116 把此传输包写入主存储器 120 的传输包池 122 中其相应传输包存储单元内。DMA 控制电路 116 还把指向写入的传输包的描述符的数据写到指派给各适配器 110 的接收队列的各描述符存储单元 129。注意，DMA 控制电路 116 可通过确定哪些描述符的字段 129-7 中设定有处理完成状态位来识别要把哪些传输包写到主存储器 120 以及这些描述符指向的传输包存储单元。注意，DMA 控制电路 116 可在每次写完成时一个接一个地写入描述符和传输包的数据。或者，DMA 控制电路 116 可允许累积某阈值数目的传输包和描述符。然后，DMA 控制电路 116 写入 $i \geq 1$ 个多个完成的描述符和传输包的序列的数据。

在一个实施例中，把加扰器/解扰器电路 115 置于适配器 110 上。在此情况下，在 DMA 控制电路 116 把传输包的数据写到主存储器 120 前，加扰器/解扰器电路 115 对必须进行解扰的每个传输包进行解扰。以下对此进行更详细的描述。

当 DMA 控制电路 116 把描述符数据和传输包写到主存储器 130 时，DMA 控制电路 116 中断处理器 160。这些中断可由 DMA 控制电路 116 在把每 $i \geq 1$ 个描述符的数据写到主存储器 130 时启动。中断使得处理器 160 执行每个传输包(PID 和输入 TS 特定)的接收 PID 处理器子程序之一。如上所述，由表 402 中指针的适当变更来选择接收 PID 处理器子程序，从而处理器 160(尤其是)丢弃将不在再分多路复用 TS 中输出的传输包，把估计离开时间写入指向的输出传输包的描述符中并在指向包含 PCR 的传输包的描述符中设定 PCR 标志位。此外，选中的接收 PID 处理器子程序最好引起处理器 160 连续地获取和更新 PSI 表，调节 PID 过滤器映射，并选择实行某用户规定所需的附加接收 PID 处理

器子程序。例如，用户规定可指定将在再分多路复用 TS(TS3)中连续地输出的特定程序号。然而，由于达到事件边界使得构成该程序的 ES 经历改变。最好，处理器 160 将通过监测 PAT 和 PMT 的变化来检测 ES 构成中的这些改变，并将根据连续地在再分多路复用 TS(TS3)中输出选中程序的 ES 的需要来改变 PID 过滤器映射并选择接收 PID 处理器子程序，而无论该程序的构成是经常的。

在进行以上与接收传输包有关的功能的同时，第三适配器 110 上的 DMA 控制电路 116 和数据链路控制电路 112 也进行与在 TS3 中发送传输包有关的某些功能。每当此第三适配器 110 的数据链路控制电路 112 输出 $k \geq 1$ 个传输包时，数据链路控制电路 112 产生一发送中断。示例地，k 可由处理器 160 选择。在对输出的再分多路复用 TS(TS3)执行适当的发送 PID 处理器子程序的处理器 160 处接收此发送中断。尤其是，处理器 160 检查位于每个队列头部处的描述符，此每个队列包含指向将在 TS3 中输出的传输包的的描述符。如上所述，两个接收队列包含指向将在 TS3 中输出的传输包的描述符，包括一有关第一适配器 110(接收 TS1)的接收队列和一有关第二适配器 110(接收 TS2)的接收队列。如下所述，处理器 160 可分配包含指向待在 TS3 中输出的传输包的描述符的附加队列。处理器 160 识别指向待在 TS3 中输出的后续 j 个传输包的描述符。这是通过执行该组(与第三适配器 110 有关且可通过接收队列头部中的传输包的 PID 而索引)的发送 PID 处理器子程序来实现。如上所述，如果将不从第三适配器 110(产生中断)输出相应于处理器 160 所检查的队列中的一个描述符的传输包，则此传输包的 PID 将对什么也不作的第三适配器 110 的发送 PID 处理器子程序索引。如果要从第三适配器 110(产生中断)输出相应于处理器 160 所检查的队列中的一个描述符的传输包，则此传输包的 PID 将对一发送 PID 处理器子程序的指针索引，该发送 PID 处理器子程序将：(1)对该传输包分配一发送描述符，(2)按照正确的发送顺序对有关第三适配器 110 的发送队列中的发送描述符进行排序，(3)对所分配的描述符和传输包指派一实际派送时间，以及(4)在必要时对传输包的偏移和等待时间进行粗的 PCR 校正。示例地，处理器 160 检查(接收)队列中的描述符，直到识别指向将在 TS3 中输出或来自第三适配器 110 的传输包的 j 个描述符。从头部 124-3 到尾部 124-4 依次检查这些描述符。如果可获得具有候选描述符的多个队列用以检查，则处理器 160 可按照估计离开时间的顺序或可虑及描述符指向的传输包的内容的某些其它适当的顺序(如下所述)，以轮流(round-robin)方式来检查

这些队列。

DMA 控制电路 116 从主存储器 120 中检索有关 TS3 或第三适配器 110 的队列的 j≥1 个描述符构成的序列的数据。按照从头部指针 124-3 到尾部指针 124-4 的顺序，从队列的描述符存储单元 129 中检索这些描述符。DMA 控制电路 116 还从主存储器 120 中检索来自每个如此检索的描述符所指向的池 122 的传输包存储单元中的传输包。DMA 控制电路 116 把如此检索的描述符和传输包存入高速缓冲存储器 114 中。

数据链路控制电路 112 按照从头部指针 124-3 的顺序，从高速缓冲存储器 114 中连续检索发送队列中的每个描述符和该描述符指向的传输包存储单元中的传输包。当第三适配器 110 的基准时钟发生器 113 的时间等于被检索的描述符的派送时间字段 129-5 中所指示的时间时，数据链路控制电路 112 在 TS3 中发送该描述符(位于头部指针 124-3 所指向的存储单元中)指向的传输包。此派送时间只近似于发送时间，因为必须与 TS3 的传输包时隙边界对准地发送每个传输包。以处理器 160 不知道的外部时钟为基准来设定这种边界。注意，由于某种原因，每个传输包的 PCR 可能稍稍抖动。相应地，数据链路控制电路 112 此外最终还依据包含此 PCR 的传输包的精确发送时间来校正此 PCR。具体来说，此精确发送时间小于估计的传输包离开时间。数据链路控制电路 112 使用预先锁定于 TS3 的时隙边界的传输时隙边界时钟，以对所估计的 PCR 进行细调(即，通过把派送时间与实际发送时间之差加到传输包的 PCR)。注意，数据链路控制电路 112 可使用此描述符的 PCR 标志位来确定该传输包中是否存在 PCR(继而确定是否校正它)。

在发送一传输包后，数据链路控制电路 112 在指向所发送的传输包的描述符的字段 129-7 中设定适当的状态信息并解除分配该描述符。然后，DMA 控制电路 116 把此状态信息写入发送队列的适当的描述符存储单元中。

在另一种操作方式中，操作人员已完全知道待再分多路复用的输入 TS 的内容。在此情况下，操作人员简单地准备用户规定并把此用户规定从控制器 20 发送到再分多路复用器节点 100(或当多个节点在网络分布式再分多路复用器 100 中操作时发送到多个再分多路复用器节点 100)。尽管如此，最好，连续地获取有关待再分多路复用的输入 TS 的内容(诸如 PAT、PMT 等)的不同类型信息。这使得可即时地把此内容报告给操作人员(经由处理器 160 和控制器 20)，例如使得产生经修改的用户规定以及依据此修改的用户规定动态地调节

再分多路复用，而不必停止输入待再分多路复用的 TS、输出经再分多路复用的 TS 或上述再分多路复用器 100 的再分多路复用处理。

除了以上的基本再分多路复用功能以外，再分多路复用器节点 100 可进行更多的先进功能。以下单独地描述这些功能。

动态再分多路复用和程序专用信息插入

如上所述，操作人员可使用控制器 20 来产生指定保留或丢弃的程序和 ES、程序或 ES 的加扰或不加扰(或这两者)、PID 的重新映射等的用户规定说明。此外，处理器 160 最好连续地获取内容信息(例如，PAT、PMT、CAT、NIT、ECM 表等的数据)。这使得可对用户规定进行动态地实时或“飞行中(on the fly)”的修改，并依据新的用户规定而无缝地改变再分多路复用。具体来说，操作人员可改变用户规定并使再分多路复用器 30 依据新的用户规定无缝地切换到再分多路复用。无论如何，再分多路复用器 30 保证每个输出的经再分多路复用的 TS 始终是包含不中断的传输包序列或串的连续位流。因而，如此修改输出的再分多路复用的 TS 的内容，而不会在输出的再分多路复用 TS 中引入不连续，即在输出的传输包串中不发生中断或在输出的位流中不发生中止。

以上无缝修改可能受到使用可编程处理器 160 的影响，该处理器控制传输包在输入和输出适配器 110 或接口 140 和 150 与诸如解扰器/解扰器 170 等其它电路之间的流动。考虑到保留或丢弃不同组 ES 的选择可能简单地受到处理器 160 调节处理器 160 对每个 PID 选中的适当 PID 过滤器映射和 PID 处理器子程序的影响。处理器 160 通过改变响应于指派给这些 ES 或程序的 PID 所执行的 PID 处理器子程序而使它们包括适当的加扰或解扰过程(如以上和以下所述)，可实现对某些 ES 或程序是进行加扰还是解扰的选择。处理器 160 通过给新的输出端口分配发送描述符队列、对不需要的输出端口解除发送描述符队列的分配、对每个新输出端口的发送 PID 处理器子程序产生指针表 404 并丢弃每个被解除分配的发送队列的发送 PID 处理器子程序的每个指针表 404，可实现对输出端口的不同选择，以输出所输出的再分多路复用 TS 的不同组合。处理器 160 以相同的方式，通过分配和解除接收队列的分配并对被分配的接收队列的接收 PID 处理程序产生指针表 402 和丢弃被解除分配的接收队列的接收 PID 的处理程序的指针表 402，可实现对输入端口的不同选择。

除了选择用以输出的正确的传输包以外，示例地，再分多路复用器节点 100 还对每个输出的再分多路复用 TS 提供正确的 PSI。这是如下实现的。处理器

20(图 2)对输出的 TS 产生用户规定说明。考虑再分多路复用器节点 100 对两个 TS(即，TS1 和 TS2)进行再分多路复用以产生第三 TS(即，TS3)的上述例子。示例地，表 1 给出了 TS1 和 TS2 中每一个的内容。

表 1

程序	ES	PID	程序	ES	PID
A	视频 A	PID(VA)	E	视频 E	PID(VE)
A	音频 A	PID(AA)	E	音频 E	PID(AE)
A	数据 A	PID(DA)	PMT	Prog. Def. E	PID(e)
PMT	Prog. Def. A	PID(a)	F	视频 F	PID(VF)
B	视频 B	PID(VB)	F	音频 F	PID(AF)
B	音频 B	PID(AB)	F	数据 F	PID(DF)
PMT	Prog. Def. B	PID(b)	PMT	Prog. Def. F	PID(f)
C	视频 C	PID(VC)	G	视频 G	PID(VG)
C	音频 C	PID(AC)	G	音频 1 G	PID(A1G)
C	解密 C	PID(ECMC)	G	音频 2 G	PID(A2G)
PMT	Prog. Def. C	PID(c)	G	数据 G	PID(DG)
D	视频 D	PID(VC)	G	解密 G	PID(ECMG)
D	音频 1 D	PID(A1D)	PMT	Prog. Def. G	PID(g)
D	音频 2 D	PID(A2D)	PAT	PAT2	0x0000
D	数据 D	PID(DD)			
PMT	Prog. Def. D	PID(d)			
PMT	PAT 1	0x0000			

最好，控制器 20 对处理器 160 进行编程以使用如上所述的接收 PID 处理器子程序的获取过程来提取表 1 所示的信息。

假定用户规定说明指定仅保留程序 A、B、F 和 G 并把它们在再分多路复用的输出 TS(TS3)中输出。用户在控制器 20(图 1)处例如使用键盘/鼠标器 27(图 1)来指示此规定。控制器 20 确定此用户规定是否有效。尤其是，控制器 20 确定每个输出再分多路复用的 TS(诸如 TS3)是否有足够的带宽来输出所有被指定的程序 A、B、F 和 G 及有关的 PSI(即，程序定义 a、b、f、g 和如下

所述的新的替代 PAT3)。如果这些位速率信息不是已知的，则可从处理器 160 中获得。例如，处理器可执行 PID 处理器子程序，以从指派给带有每个程序的 PCR 的每个传输包的接收时间标记中确定每个程序的位速率(或传输包速率)。如上所述，总之处理器 160 为进行 PCR 调节的目的而获得这些信息。如果用户规定无效，则处理器 20 不下载此用户规定。如果规定有效，则控制器 20 把此用户规定下载到处理器 160。

假设，TS3 的带宽可满足用户规定。如果输入的 TS(TS1 和 TS2)的 PAT 和 PMT 还未被获取，则处理器 160 获取其 PAT 和 PMT。根据 PAT1 和 PAT2 中的信息，处理器 160 构成一替代的 PAT3，它仅包括指示有关程序 A、B、F 和 G 的程序定义 a、b、f 和 g 的 PID 的 PAT1 和 PAT2 的表目。再者，这可使用用于 PAT1 和 PAT2 的 PID 的适当 PID 处理器子程序来实现，且最好连续地执行以保证把在 PAT1 和 PAT2 中所反映的对程序的任何改变并入替代 PAT3 中。处理器 160 产生包含此新的替代 PAT3 的传输包序列并把它们存入包缓冲器 122。处理器 160 还产生指向带有 PAT3 的这些传输包的描述符的 PAT 队列，该队列最好实现为环 124。有利的是，PAT3 传输包的 PAT 描述符队列专用于仅存储替代 PAT 信息。此外，处理器 160 产生估计的离开时间并把它们存储在指向 PAT3 传输包的 PAT 队列的描述符中。

现在，处理器 160 可响应于发送中断以与任何接收队列相同的方式服务于 PAT3 描述符队列。即，当数据链路控制电路 112 发送 $k \geq 1$ 个包并中断处理器 160 时，处理器 160 将从 PAT3 队列以及接收队列中提取描述符。这里，把包含指向待输出传输包(发送队列中的发送描述符还未被分配)的所有队列统一地叫做“连接队列”。

然后，处理器 160 构成适当的过滤器映射并把一过滤器映射传送到接收 TS1 的第一适配器 110 并把第二过滤器映射传送到接收 TS2 的第二适配器 110。例如，第一过滤器映射可指示提取和保留具有 PID：PID(VA)、PID(DA)、PID(a)、PID(VB)、PID(AB)和 PID(b)(以及相应于 TS1 中的 PSI 的其它可能的 PID)的传输包。同样，第二过滤器映射可指示提取和保留具有 PID：PID：PID(VF)、PID(AF)、PID(DF)、PID(f)、PID(VG)、PID(A1G)、PID(A2G)、PID(DG)、PID(ECMG)和 PID(g)(以及相应于 TS2 中的 PSI 的其它可能的 PID)的传输包。对此响应，接收 TS1 和 TS2 的第一和第二数据链路控制电路 112 依据处理器 160 所提供的过滤器映射仅从 TS1 和 TS2 中提取这些传输包。如上所述，第一和第二数

据链路控制电路 112 把这些所提取的包存储在高速缓冲存储器 114 中并对其分配描述符。第一和第二 DMA 控制电路 116 周期性地把所提取的传输包及其描述符的数据写到主存储器 120。把由第一 DMA 控制电路 116 写入的描述符的数据存储在第一数据链路控制电路 112 的第一接收队列的各描述符存储单元 129 中，把由第二 DMA 控制电路 116 写入的描述符的数据存储在第二数据链路控制电路 112 的第二接收队列的描述符存储单元中。

此外，第三 DMA 控制电路 116 从有关 TS3 的发送队列中检索描述符及其相应的传输包，并把它们存储在高速缓冲存储器 114 中。第三数据链路控制电路 112 从高速缓冲存储器 114 中检索每个描述符并把它们在 TS3 中发送。第三数据链路控制电路 112 在发送了 $k \geq 1$ 个传输包后产生中断。这使得处理器 160 访问与第三数据链路控制电路 112 有关的发送队列的发送 PID 处理器子程序的指针表。在执行适当的发送 PID 处理器子程序时，处理器 160 把 TS3 中未使用的发送描述符分配给可获得的连接队列(即，第一接收队列、第二接收队列和 PAT3 队列)中的描述符，并从这些连接队列中拷贝如此分配的描述符的有关信息。按照有关接收描述符的估计派送时间的顺序在 TS3 发送队列中分配发送描述符。

注意，可动态的插入任何类型的 PSI，包括新的程序定义、EMM、ECM、CAT 或 NIT。

现在考虑产生新的用户规定说明同时依据前一个用户规定发生再分多路复用的情况。如上所述，控制器 20 开始验证是否有足够的带宽满足新的用户规定。如果有，则把新的用户规定下载到处理器 160。新的用户规定可能需要处理器 160 提取不同的程序和 ES，对 PID 进行不同地映射，或者产生：(a)新的 PSI、(b)带有此新的 PSI 的传输包以及(c)指向带有此新的 PSI 的传输包的描述符。在修改包含在 TS3 中的程序或 ES 的情况下，处理器 160 依据新的用户规定修改 PID 过滤器映射，以保留待保留的传输包并丢弃待丢弃的传输包。把这些新的过滤器映射传送到各高速缓冲存储器 114，这些高速缓冲存储器 114 立即动态地切换到依据新的用户规定提取传输包。处理器 160 还通过修改与新的待保留传输包的 PID 有关的接收 PID 处理器子程序指针表 402 的指针，为新的待保留传输包选择适当的接收 PID 处理器子程序。还可对由现在待被丢弃的传输包的 PID 索引的接收 PID 处理器子程序指针表 402 的指针进行修改。在新的 PID 重新映射的情况下，处理器 160 选择适当的子程序来执行新

的 PID 重新映射。

这种改变可能需要产生新的 PSI，例如新的 PAT。处理器 160 选择适当的 PID 处理器子程序来产生新的 PSI。例如，在新的 PAT 的情况下，PID 处理器子程序可由 TS1 和 TS2 的 PAT 的 PID 来触发。处理器 160 产生新的 PSI 并把此新的 PSI 插入传输包。把各 PSI 队列中的描述符分配给这些新的 PSI 传输包。处理器 160 停止服务于指向包含旧 PSI 的传输包的任何 PSI 描述符队列(即，进行刷新和从中传送传输包)，而是服务于新的 PSI 描述符队列。

在每一个改变时，即可获得每个新选中的 PID 处理器子程序(每个 PSI 插入修改或每个新的 PID 过滤器映射)时，适当的数据链路控制电路 112 或处理器 160 无缝地改变其操作。直到实行这些改变时，数据链路控制电路 112 或处理器 160 才在先前的用户规定下继续操作。在每个改变发生时进行排序时必须注意，从而输出的再分多路复用 TS 总是符合 MPEG-2。例如，最好延迟对 TS 中的 PID 映射、PID 过滤、程序、ES、ECM 等(它们影响 PMT 或 PAT)进行的任何改变，直到可在 TS 中输出 PMT(或其特定程序定义)和/或 PAT 的新版本以及在 TS 中指示切换到新的 PMT、程序定义或 PAT 的指示时。同样，如果包含或丢失一有条件访问系统的 EMM，则延迟此 EMM 的引入，直到在 TS 中可发送 CAT 的新版本。对资源的内部处理管理想要对改变进行附加的慎重排序，诸如在改变用于保留具有此 PID 的传输包的各适配器 110 的 PID 过滤器映射前，把一指针存储到待被保留的传输包(它先前被丢弃了)的 PID 索引的适当接收 PID 处理器子程序指针表中的接收 PID 处理器子程序等。

以下是依据新的用户规定修改再分多路复用的一个例子。假定用户提供了指示应丢弃程序 B 和 F 而应保留程序 C 和 D 的新的用户规定。与之响应，控制器 20 在依据新的用户规定修改再分多路复用的 TS(TS3)时，首先确定在输出的再分多路复用 TS(TS3)中是否有足够的带宽来容纳所有的新程序数据以及必须为其产生的新 PSI。假设有，则把此新的用户规定下载到再分多路复用器节点 100。处理器 160 修改第一适配器 110 中的 PID 过滤器映射，从而丢弃具有 PID(PID(VB)、PID(AB)、PID(b))的传输包，并保留具有 PID(PID(VC)、PID(AC)、PID(ECMC)、PID(c)、PID(VD)、PID(A1D)、PID(A2D)、PID(DD)和 PID(d)的传输包。同样，处理器 160 修改第二适配器 110 中的 PID 过滤器映射，从而丢弃具有 PID(PID(VF)、PID(AF)、PID(DF)和 PID(f))的传输包。处理器 160 选择包括用于 PID(PID(VC)、PID(AC)、PID(ECMC)、PID(c)、PID(VD)、PID(A1D)、

PID(A2D)、PID(DD)和 PID(d))的 PID 处理器子程序，包括用于 PID(PID(c)和 PID(d)中每一个的程序定义更新处理、用于 PID(ECMC)的控制字更新处理、用于程序 C 的每一个加扰 ES(例如，PID(VC))的解扰控制字信息插入处理。处理器 160 例如在执行第一和第二适配器 110 中每一个的 PID(0)的 PID 处理器子程序期间，还产生包括程序定义 a、b、c、d 和 g 的不同替代 PAT3。

现在考虑提供了指示应对程序 A 的 VA 视频 ES 进行加扰的另一个新用户规定的情况。再者，控制器 20 首先确定 TS3 中是否有足够的带宽来容纳带有 VA 的传输包的 ECM 和程序 A 的新程序定义。假设有，则把此新的用户规定下载到再分多路复用器节点 100。处理器 160 分配用于存储指向包含 VA 的 ECM 的传输包的描述符的队列。处理器 160 为 PID(VA)选择适当的 PID 处理器子程序，包括把加扰控制字插入指向包含 VA 的传输包的描述符中。处理器 160 还产生包含作为 VA 的 ECM 的控制字的传输包并分配指向这些传输包的描述符。这可使用定时器驱动的中断处理器子程序来实现。或者，由处理器 160 执行的附加硬件(未示出)或软件周期性地产生控制字并在这些控制字准备好时中断处理器 160。处理器 160 通过把一可获得的控制字置于一个或多个传输包中、把一 ECM 队列的 ECM 描述符分配给这些传输包并把新的控制字装入适当的控制字表中，对这些中断作出响应。此外，处理器 160 还选择用于 PID(a)的接收 PID 处理器子程序，包括提取程序定义 a 中的信息并添加有关 ECMA 的信息(例如，PID(ECMA)，其加密的 ES 等)。

加扰/解扰控制

与加扰和解扰有关的一个问题是对每个传输包选择正确的控制字或密钥。即，可以 PID 专用控制字或专用于一组 PID 的控制字对经加扰的传输包数据进行加扰。可在控制字时时改变时使用旋转控制字方案。简言之，可存在与每个 TS 有关的大量控制字(例如，密钥)并周期性地改变这些控制字。在解扰的情况下，必须提供连续地接收用于每个待解扰 ES 或 ES 组的控制字并在每一时刻选择适当的控制字的机构。在加扰的情况下，必须提供用于选择 ES 或 ES 组加扰用的正确控制字并足够超前于如此形成的任何经加扰的 ES 数据把此 ES 加扰用的控制字插入输出的再分多路复用 TS 中的机构。

可使用这些描述符及其在接收和发送队列中的排序来简化 TS 的加扰和解扰。尤其是，每个接收描述符具有其中可存储有关加扰或解扰的信息的字段 129-9，这些信息诸如在对传输包进行加扰时所使用的控制字或适当控制字表

(包含用于对传输包进行加扰或解扰的控制字)的指针。

首先考虑在对一传输包进行解扰时所执行的步骤。包含待解扰的传输包的 TS 包含带有 ECM(ES 专用有条件访问)和 EMM(专用于整个 ES 组的有条件访问)的传输包。在标有 EMM 所对应的 ES 组所独有的 PID 的传输包中携带有 EMM，在标有每个 ECM 所对应的特定 ES 所独有的 PID 的传输包中携带有 ECM。可参考 CAT 把 EMM 的 PID 同 EMM 所对应的特定 ES 组相关。可参考 PMT 把 ECM 的 PID 同 ECM 所对应的每个特定 ES 相关。处理器 160 选择 PID 处理器子程序，用以：

(1)恢复在 TS 中发送的每个 CAT 和 PMT 并识别当前所使用的 CAT 或 PMT 的版本，

(2)参考 PMT，恢复被携带有 ECM 所对应的 ES 的传输包的 PID 索引的 ECM 表。

接着，处理器 160 定义将在每个传输包和描述符上执行的一系列处理步骤。即，处理器 160 定义接收适配器 110 的数据链路控制电路 112、(任选)接收适配器 110 的解扰器 115、接收适配器 110 的 DMA 控制电路 116、(任选)解扰器 170 和处理器 160 可处理一接收描述符或一接收描述符指向的包的特定顺序。为此，处理器 160 可把适当的控制信息传送到装置 112、115 和 116 中的每一个，以使它们按照如下所述定义的一系列处理步骤的特定顺序来处理传输包及指向该传输包的描述符。

如果使用适配器 110 的解扰器 115，则如下定义该序列中的处理顺序。适配器 110 的数据链路控制电路 112 接收传输包并给未按照如上所述的 PID 过滤器映射丢弃的那些传输包中选中的传输包分配接收描述符。在把每个保留的传输包存入高速缓冲存储器 114 中后，示例地，数据链路控制电路 112 设定指向该传输包的描述符中的状态位 129-7 指示现在按照所定义的处理步骤序列的顺序可由下一个装置来处理该传输包。

解扰器 115 周期性地在高速缓冲存储器 114 中检查接着的一个或多个描述符，其状态位 129-7 被设定为指示已许可解扰器 115 修改传输包。示例地，解扰器 115 在已处理了 $m \geq 1$ 个描述符后访问高速缓冲存储器 114。解扰器 115 从先前被解扰器 115 访问的描述符开始依次访问高速缓冲存储器 114 的每个描述符，直到已访问了 $m \geq 1$ 个描述符或直到到达这一描述符，其状态位 129-7 被设定为指示已按照所定义的处理步骤序列的顺序对描述符及其指向的传输包执行了前一步骤的处理。

在处理描述符和传输包时，解扰器 115 使用当前被检查的描述符指向的传输包的 PID 来索引位于高速缓冲存储器 114 中的解扰映射。示例地，处理器 160 如下所述周期性地更新高速缓冲存储器 114 中的解扰映射。由位于描述符直到 129-9 中的基址来提供解扰映射的位置。示例地，处理器 160 在分配接收描述符队列时把解扰映射的基址装入每个描述符的直到 129-9。解扰映射的被索引表目指示是否对传输包进行加扰以及在加扰时可使用对传输包进行解扰的一个或多个控制字。解扰映射的被索引表目可包含相应于传输包的 PID 的控制字或指向其中存储有各控制字的存储单元的指针。如果解扰映射的被索引表目指示不对被访问的描述符指向的传输包进行解扰，则解扰器 115 简单地把描述符的状态位 129-7 设定为指示可按照所定义的处理步骤序列的顺序对描述符及其指向的传输包执行下一处理步骤。

如果解扰映射的被索引表目指示将要对传输包进行解扰，则解扰器 115 获得相应于该传输包的 PID 的控制字并使用该控制字对传输包数据进行解扰。注意，典型的解扰方案使用如上所述的旋转(即，奇数和偶数)控制字。在对一传输包进行解扰时所使用的正确的奇数或偶数控制字由传输包中的控制位来指示，诸如传输_加扰_控制位。解扰器 115 在对正确的控制字进行索引时使用这些位以及传输包的 PID。即，由 PID 和奇数/偶数指示符来索引由处理器 160 构成和保持的映射。然后，解扰器 115 把经解扰的传输包数据存储在由当前检查的描述符所指向的传输包存储单元，从而改写该传输包的预解扰数据。然后，解扰器 115 把该描述符的状态位 129-7 设定为指示可按照所定义的处理步骤序列的顺序对描述符及其指向的传输包执行下一处理步骤。

DMA 控制电路 116 周期性地把传输包数据和指向该传输包的描述符的数据从高速缓冲存储器 114 写到主存储器 130 的各个存储单元 122 和 129。这样做时，DMA 控制电路 116 周期性地检查高速缓冲存储器 114 中由接在 DMA 控制电路 116 所处理的最后一个描述符后(按照接收队列的顺序)的一个或多个描述符构成的序列。如果被检查的描述符的状态位 129-7 指示可对被检查的描述符执行 DMA 控制电路 116 的处理，则 DMA 控制电路 116 把该描述符中的适当状态位 129-7 设定为指示可按照所定义的处理步骤序列的顺序对此描述符及其指向的传输包执行下一处理步骤。然后，DMA 控制电路 116 把该描述符的数据及其指向的传输包的数据写到主存储器 130。然而，如果状态位 129-7 被设定为指示仍将对描述符执行 DMA 控制电路 116 所执行的处理之前的处理步骤，

则 DMA 控制电路 116 避免处理该描述符及其指向的传输包。示例地，在被使能时，DMA 控制电路 116 检查描述符，直到 DMA 控制电路 116 写入 i≥1 个描述符的序列及这些描述符指向的传输包的数据，或直到碰到这样的描述符，其状态位 129-7 指示仍旧按照所定义的处理步骤序列的顺序对该描述符执行前一处理步骤。每当 DMA 控制电路 116 传送了 i≥1 个传输包时，DMA 控制电路发出一中断。

处理器 160 通过执行适当的接收 PID 处理器子程序，响应于例如 DMA 控制电路 116 所发出的中断。处理器 160 从处理器 160 所处理的最后一个描述符开始检查相应于从中接收到中断的适配器 110 的接收队列中的一个或多个描述符。示例地，处理器 160 仅对这样的描述符执行适当的接收 PID 处理器子程序，这些描述符的状态位 129-7 被设定为指示可对描述符执行处理器 160 的处理。每当处理器 160 被中断时，示例地，处理器 160 处理描述符及其指向的传输包，直到对 i≥1 个传输包执行了 PID 处理器子程序或直到碰到这样的一个描述符，其适当的状态位 129-7 被设定为指示仍旧对该描述符执行前一处理步骤(按照所定义的处理步骤序列的顺序)的处理。。

在执行适当的接收 PID 处理器子程序期间，处理器 160 恢复所有 ES 的所有控制字并更新被解扰器 115(或如下所述的 170)所使用的解扰和控制字表或映射。在旋转控制字方案中，处理器 160 在控制字表或映射中保持用于每个 PID 的多个(即，奇数和偶数)密钥。处理器 160 还可进行使能对经解扰的传输包进行后续的加扰的处理(如下所述)。在处理了接收描述符后，处理器 160 通过把这些描述符的状态位 129-7 设定为指示描述符无效(继而数据链路控制电路 112 是下一个处理这些描述符的装置)、擦除或重新设定描述符的选中字段并使首部指针 123-3 进到下一描述符存储单元 129 来解除这些描述符的分配。

现在考虑在适配器 110 上未设置或不使用解扰器 115 的情况。取而代之，使用位于总线 130 上的解扰器 170。执行与先前非常类似的过程。然而，在此情况下，如此改变所定义序列的处理步骤的顺序，从而 DMA 控制电路 116 在数据链路控制电路之后并在解扰器之前处理描述符(及其相应的传输包)，解扰器 170 在 DMA 控制电路 116 之后但在处理器 160 之前处理描述符(及其相应的传输包)。继而，在数据链路控制电路 112 把描述符分配给传输包并设定适当的状态位 129-7 以使能对它们所执行的下一处理步骤后，DMA 控制电路 116 处理该描述符及其指向的传输包。如上所述，DMA 控制电路 116 把状态位 129-7

设定为指示可对描述符执行下一处理步骤，并把该传输包和描述符写到主存储器 130。

解扰器 170 周期性地检查接收队列中的描述符，以识别其状态位 129-7 被设定为指示可对描述符及其指向的传输包进行解扰处理(按照所定义的处理步骤序列的顺序)的描述符。解扰器 170 与以上对解扰器 115 所讨论的类似的方式处理这些被识别的传输包。在处理了这些传输包后，解扰器 170 把一个或多个状态位 129-7 设定为指示现在可对描述符及其指向的传输包执行下一个处理步骤(按照所定义的处理步骤序列的顺序)。

处理器 160 响应于 DMA 控制电路 116 所发出的中断进行如上所述的处理，包括执行适当的接收 PID 处理器子程序。最好，与中断处理器 160 的适配器 110 有关的接收队列的队列长度相对于解扰器 170 的处理时间足够长，从而使处理器 160 检查和处理解扰器 170 已完成处理的描述符。换句话说，处理器 160 和解扰器 170 最好不要尝试同时访问相同的描述符。再者，处理器 160 在接收队列中与解扰器 170 不同的点处开始处理描述符。

现在考虑有关加扰的处理。如同解扰处理，使用描述符中的状态位 129-7，按照所定义的处理步骤序列的顺序对每个描述符和这些描述符指向的传输包所进行的处理步骤进行排序。与解扰不同，加扰最好在处理器 160 已把发送描述符分配给待加扰的传输包后进行。这样，可以两种方式中的一种来使用控制字字段 129-9。如同解扰，可把加扰映射的基址置于控制字描述符字段 129-9 中。然而，由于加扰发生在处理器 160 处理发送队列中的描述符后，所以最好把正确的控制字本身置于控制字描述符字段 129-9 中。

首先考虑由发送适配器 110 的加扰器 115 进行加扰的加扰处理。处理器 160 获得包含控制字(最好被加密)的 ECM 传输包。把这些 ECM 传输包在各个相应的连接队列中排队，并对它们进行调度以在正确的时间输出。即，对这些 ECM 传输包进行调度，以足够超前于它们所解扰的传输包把这些 ECM 传输包插入输出的 TS 中，以使解码器在接收其所解扰的传输包前恢复该控制字。

在发送包含控制字的 ECM 传输包后的适当时间，处理器 160 改变控制字表以使用相应于最近发送的控制字的新密钥对数据进行加密。当从输出适配器发送传输包时，处理器 160 执行与被检查的连接队列中的描述符所指向的传输包的 PID 有关的发送 PID 处理器子程序。对于这些待加扰的传输包中的每一个，发送 PID 处理器子程序包括用于把控制字信息插入有关该传输包的

描述符中的处理。控制字信息可简单的是将在识别传输包加扰用控制字时所使用的加扰映射的基址。然而，控制字信息还可以是将在传输包加扰中所使用的正确的控制字。处理器 160 还可来回切换(toggle)传输包中的诸如传输_加扰_控制位等位，以指示应使用最近发送的控制字中的哪一个对解码器处的传输包进行解密或解扰。此外，示例地，处理器 160 把新分配的发送描述符的一个或多个状态位 129-7 设定为指示应对此发送描述符及其指向的传输包进行下一处理步骤(按照所定义的处理步骤序列的顺序)。

发送适配器 110 的 DMA 控制电路 116 周期性地从发送队列中检索描述符数据及这些描述符指向的传输包。这样做时，DMA 控制电路 116 检查接在 DMA 控制电路 116 把描述符数据传送到高速缓冲存储器 114 的最后一个描述符后面的发送队列中的描述符。DMA 控制电路 116 仅传送这些发送描述符的数据，这些发送描述符的状态位 129-7 被设定为指示现在可执行 DMA 控制电路 116 的处理(按照所定义的处理步骤序列的顺序)。例如，DMA 控制电路 116 可检查发送描述符，直到已识别许可 DMA 控制电路 116 处理的某一数目 $k \geq 1$ 个发送描述符，或直到识别了这样的一个描述符，其状态位 129-7 被设定为指示仍旧对发送描述符及其指向的传输包执行前一处理步骤。在把这些发送描述符的数据以及这些发送描述符指向的传输包传送到高速缓冲存储器 114 后，DMA 控制电路 116 把这些所传送的发送描述符的状态位 129-7 设定为指示可对这些发送描述符及其指向的传输包执行下一处理步骤(按照所定义的处理步骤序列的顺序)。

接着，加扰器 115 在高速缓冲存储器 114 内的描述符中周期性地检查要处理的一个或多个描述符的序列及其指向的传输包。加扰器 115 仅处理那些被访问的描述符，这些描述符具有被设定为指示可对这些描述符执行加扰处理步骤(按照所定义的处理步骤序列的顺序)的一个或多个状态 129-7。加扰器 115 访问控制字信息字段 129-9 并使用其中的信息对每个待加扰的传输包进行加扰。如上所述，可以两种方式中的一种来使用控制字信息。如果控制字信息是一加扰映射的基址，则加扰器 115 使用该基址和此传输包的 PID 信息对加扰映射进行索引。加扰映射被索引的表目指示是否要对传输包进行加扰以及在要进行加扰时加扰传输包用的控制字。或者，字段 129-9 中的控制字信息本身指示是否要对此传输包进行加扰以及在要进行加扰时加扰传输包所使用的控制字。如果不要对被处理描述符的传输包进行加扰，则加扰器 115 简

单地把适当的状态位 129-7 设定为指示现在可对发送描述符及其指向的传输包执行下一处理步骤(按照所定义的处理步骤序列的顺序)。如果要对被处理的描述符的传输包进行加扰，则加扰器首先对此传输包数据进行加扰，把此传输包存储在高速缓冲存储器中代替未加扰的传输包，然后设定适当的状态位 129-7。

数据链路控制电路 112 在高速缓冲存储器 114 内的发送描述符中周期性地检查发送描述符，这些描述符具有被设定为指示可对这些描述符执行数据链路控制电路 112 的处理的一个或多个状态位 129-7。对于这些发送描述符，数据链路控制电路 112 以这些描述符中所指示的实际派送时间发送这些描述符所指向的传输包。然后，数据链路控制电路 112 解除这些描述符的分配(并把状态位 129-7 设定为无效)。示例地，每当数据链路控制电路 112 发送 k≥1 个描述符的序列时，数据链路控制电路 112 产生被处理器 160 接收的发送中断。

在不存在或不使用加扰器 115 的情况下，示例地，取而代之使用加扰器 170。如此改变所述处理步骤的序列，从而加扰器 170 在处理器 160 后但在 DMA 控制电路 116 前处理每个发送描述符及其指向的传输包，DMA 控制电路 116 在加扰器 170 后但在数据链路控制电路 110 前处理每个发送描述符及其指向的传输包。

带宽优化

如上所述，通常在带有程序的 TS 中插入空传输包。这些空传输包的存在是由于程序编码器通常必须给每个程序分配多余的带宽。这是因为对时时产生的每个 ES 所产生的编码数据的数量只能控制这么多。没有此“开销带宽”，经编码的 ES 数据会频繁地超过对其所分配的带宽量，从而使得在 TS 中遗漏经编码的 ES 数据。或者，ES 编码器(尤其是视频 ES 编码器)不一定总是在发生传输包时隙时可获得被输出的数据。例如，特定图形的编码可能意外地占据比先前所预期的更长的时间，从而在产生经编码的视频 ES 数据时引起延迟。这些时隙被空的传输包所填充。

虽然在再分多路复用器节点 100 中必须容忍空传输包的存在，但想要减少这些浪费带宽的空传输包的数目。然而，在这样做时，不应改变每个程序的位速率且应使这些程序的端-端延迟保持恒定。依据一个实施例，利用了一种技术从而以其它待再分多路复用的传输包数据(如果这些其它传输包数据可

获得的话)来替换空传输包。这是如下实现的。

首先考虑处理器 160 现有多个连接队列，这些队列包含待调度传输包的描述符，即还未传送到发送队列的接收队列、PSI 队列、其它数据队列等中的描述符。如上所述，这些描述符可指向与接收到的入局 TS 有关的传输包或处理器 160 所产生的诸如 PAT 流、PMT 流、EMM 流、ECM 流、NIT 流、CAT 流等其它有关程序的流。然而，现有的可以是其它类型的待调度传输包及其描述符 129，诸如带有非时间敏感的“突发”或“竭尽全力(best effort)”的独有数据的传输包。例如，这些额外的传输包可包含事务处理计算机数据，例如在 Web 浏览器和 Web 服务器之间传送的数据。(再分多路复用器节点 100 可以是一服务器、一终端或简单地是一连到“因特网”的通信系统中的一个中间节点。可使用调制解调器、适配器 140 或 150 等来实现这种与因特网的连接。)这些数据没有恒定的端-端延迟的要求。而是，只要可获得带宽，这些数据就可以突发脉冲串(burst)来发送。

处理器 160 首先使得每个空的传输包被丢弃。这可通过由处理器 160 使用丢弃所有空传输包的接收 PID 处理器子程序来实现。示例地，该技术是在从适配器 110 以外(诸如接口 140 或 150)的装置接收到空传输包时使用的。或者，如果空传输包是从适配器 110 接收到的，则处理器 160 可把一 PID 过滤器映射提供给数据链路控制电路 112 以丢弃每个空传输包。接着，依据接收 PID 处理器子程序，给将在 TS 中输出的每个入局传输包指派一作为传输包的接收时间(记录在其描述符中)的函数的估计离开时间以及再分多路复用器节点 100 内的内部缓冲延迟。在包含待调度传输包的每个连接队列中，所指派的离开时间不可能是输出的 TS 的连续传输包发送时间(相应于相邻时隙)。再者，将在统一输出 TS 中输出的传输包的两个连续描述符的估计离开时间可能隔开输出的再分多路复用 TS(其中将发送这些传输包)的一个或多个传输包发送时间(或时隙)。

最好，把指向带有程序数据的传输包的描述符、指向带有 PSI、ECM 或 EMM 的传输包的描述符以及指向突发数据的描述符都保持在相互分开的连接队列中。在实现中，给每个连接队列指派一服务优先级，该优先级与在其中排队的描述符所指向的传输包中的数据类型有关。最好，给从多路复用器节点外部(例如，经由接收适配器 110 或接口 140 或 150)接收到的程序数据指派最高的优先级。还可给存储再分多路复用器节点 100 所产生的 PSI、ECM 或 EMM 流

的连接队列指派同一优先级。最后，给具有指向这种传输包(包含没有特定连续性、传播延迟或位速率要求的突发数据)的描述符的连接队列指派最低优先级。此外，与程序、PSI、ECM 和 EMM 数据不同，不给带有突发数据的传输包的描述符指派估计离开时间或不把该时间记录在其中。

在执行发送 PID 处理器子程序时，处理器 160 把有关待调度传输包的描述符从它们各自的连接队列传送到一发送队列。在这样做时，处理器 160 最好在对较低优先级的连接队列诉诸服务前，服务于给定优先级的每个连接队列(即，检查其中的描述符)。在检查描述符时，处理器 160 确定优先级高的连接队列(即，包含带有程序 PSI、ECM 或 EMM 数据的传输包的描述符)的任何被检查描述符是否指向必须在下一实际派送时间发送的传输包，此确定根据指派给这些传输包的估计离开时间。如果是这样，则处理器 160 把一发送描述符分配给每个这样的传输包，把连接队列描述符中的有关信息拷贝到所分配的发送队列描述符中，并给对发送描述符所分配的每个传输包指派适当的派送时间。如上所述，偶尔，两个或多个传输包竞争同一实际离开时间(即，输出的再分多路复用 TS 的同一传输包时隙)，在此情况下，把一传输包序列指派给连续时隙和实际离开时间。在必要时，对这些传输包进行 PCR 调节。

在其它时间，当处理器 160 服务于连接队列时，较高优先级的连接队列的传输包的估计离开时间都不会使处理器 160 把该传输包指派给输出的再分多路复用 TS 中的下一可获得的时隙和实际派送时间。通常，这会在输出的再分多路复用 TS 中产生空时隙。然而，在此情况下，处理器 160 最好服务于较低优先级的连接队列。处理器 160 检查较低优先级的连接队列(按照从头部指针 124-3 开始的顺序)，选择性地把发送描述符指派给这些被检查的描述符指向的一个或多个传输包构成的序列中的每一个，以及把被检查的描述符的有关信息拷贝到所分配的发送描述符上。处理器 160 选择性地把(另外的)空时隙分配给这些被检查的描述符指向的每个传输包，并把有关所指派的时隙的实际派送时间存储在所分配的相应发送描述符中。

偶尔，没有被优先级高或低的连接队列中的描述符所指的传输包可分配给输出的再分多路复用 TS 的时隙。这发生的原因可能是优先级高的传输包的估计离开时间都不相应于时隙的实际派送时间以及不对带有突发数据的传输包进行缓冲来等待再分多路复用器节点 100 处的发送。或者，对带有突发数据的传输包进行缓冲，但处理器 160 由于以下所讨论的原因而选择不在此特

定时刻对其指派发送描述符。在此情况下，发送队列中的描述符将具有相应于输出的再分多路复用 TS 的不连续传输包时隙序列的实际发送时间。当发送适配器 110 的数据链路控制电路 112 碰到这样的不连续时，数据链路控制电路 112 在未被指派传输包的每个空时隙发送一空的传输包(利用发送描述符实际派送时间)。例如，假设发送队列中与第一和第二传输包有关的两个连续描述符的派送时间指示将在第一传输包时隙处发送第一传输包以及将在第六传输包时隙处发送第二传输包。数据链路控制电路 112 在第一传输包时隙发送第一传输包。在第二、第三、第四和第五传输包时隙中的每一个时隙处，数据链路控制电路 112 自动地发送一空传输包。在第六传输包时隙处，数据链路控制电路 112 发送第二传输包。

注意，突发或竭尽全力数据通常没有严格的接收缓冲器约束。即，大多数突发或竭尽全力数据接收器和接收器应用程序不规定最大缓冲器尺寸，数据填充速率等。取而代之，可利用诸如发送控制协议(TCP)等传输协议，从而在接收器缓冲器填充时，接收器简单地丢弃随后接收到的数据。接收器不确认接收到所丢弃的包，源重新发送未被确认接收到的带有数据的包。这有效地节制了至接收器的有效数据发送速率。虽然此节制技术可有效地实现至接收器的准确数据发送速率，但它有两个问题。首先，网络必须支持双向通信。所有的有线电视网络中只有一部分且没有直接的广播卫星网络支持发送器与接收器之间(不存在电话返回路径)的双向通信。在任何支持双向通信的情况下，从接收器到发送器的返回路径的带宽基本上小于从发送器到接收器的前向路径，且该返回路径通常必须由多个接收器共享。因而，把 TCP 积极地用作节制机构利用了返回路径(它还必须用于其它接收器至发送器的通信)中的大部分。此外，不想要地浪费了发送被丢弃的传输包的前向路径的带宽。

最好，突发或竭尽全力数据的插入不应使这些缓冲器上溢。示例地，假设某一(或典型的)接收器缓冲器的占有率及其中的数据的未决性，PID 处理器子程序可控制插入突发数据的速率，以实现某一平均速率，从而不超过某一峰值速率或甚至简单地防止接收器缓冲器上溢。因而，甚至在处理器 160 可获得插入一个或多个空的传输包时隙的突发或竭尽全力数据(不能获得其它插入其中的数据)时，处理器 160 可选择仅把突发数据插入某些空的传输包时隙中，选择把突发数据插入交替或隔开的传输包时隙中，或选择不把突发数据插入任何空传输包时隙中，从而调整至假定的接收器突发数据缓冲器的数据

发送，或防止该缓冲器的上溢。此外，旨在多个不同接收器的传输包本身可以是交错的(interleaved)，而与它们何时产生无关，以保持接收器的某一数据发送速率。

在任何情况下，再分多路复用器节点 100 都提供了优化 TS 的带宽的简单方法。丢弃入局 TS 中的所有空传输包。如果可获得传输包，则把它们插入通常会分配给被丢弃的空传输包的时隙中。如果不能获得传输包，则通过正常派送时间指派过程给这些时隙留下空隙。如果传输包的派送时间都不指示应在输出的再分多路复用 TS 的下一可获得的时隙处发送该传输包，则数据链路控制电路 112 自动地把空传输包插入这一时隙。

这种带宽优化方案的优点是双重的。第一，就输出的再分多路复用 TS 而言实现了带宽增益。现在把通常在空的传输包上所浪费的带宽用于发送信息。第二，可在 TS 中输出竭尽全力或突发数据，而不对其具体地分配带宽(或分配少得多的带宽)。例如，假设一输出的再分多路复用 TS 的带宽为 20Mbits/sec。要对带有四个程序的 TS(每个为 5Mbits/sec)进行再分多路复用并把它们输出到 20Mbits/sec 的再分多路复用 TS。然而，可把带有这四个程序的 TS 中每一个带宽的大约 5%分配给空的包。这样，传送带有竭尽全力或突发数据的传输包可获得(名义上)高达 1Mbit/sec，然而对于端-端延迟的恒定性没有任何保证或限制。

对未定时的数据进行重新定时

如上所述，可经由异步接口 140 接收待再分多路复用的程序数据。因未把接口 140 及其所连的通信链路设计成在任何特定时间发送数据及将在所传送的数据中引入可变端-端延迟从而会引起问题。在比较中，可对经由异步通信链路(诸如连到接收适配器 110)在再分多路复用器节点 100 处接收到的程序数据进行这样的假设，即将输出其所有接收到的传输包而没有抖动。这是因为所有这些包在再分多路复用器节点 100 处都引起同一延迟(即，内部缓冲延迟)，或者如果它们不引起同一延迟(因如上所述的时隙竞争的结果)，则知道是附加的延迟，调节这些 PCR 以除去这些附加延迟所引入的任何抖动。此外，进一步校正 PCR 的内部时钟机构相对于每个程序的系统定时时钟的偏移，以及校正 PCR 的调度输出时间与实际输出时间之间相对于输出的 TS 的时隙边界的不对准。然而，在从接口 140 接收到传输包的情况下，在再分多路复用器节点 100 处以可变位速率以及不恒定的抖动时间接收这些传输包。因而，如

果把传输包的实际接收时间用作估计传输包的离开的基础，则抖动仍将保持。抖动的 PCR 不仅在解码器处造成解码和呈现不连续，而且它们还使缓冲器上溢和下溢。这是因为每个程序的位速率被仔细地调整(假设将把这些数据从解码器缓冲器中移去以用于解码，并相对于程序的系统定时时钟进行呈现)。

依据一个实施例，如下所述来克服这些问题。处理器 160 识别接收到的 TS 的每个程序的 PCR。使用这些 PCR，处理器 160 确定 PCR 对之间的每个程序的传输包的分段传输包速率。给定每个程序的每个(交错的)传输包序列的传输包速率，则处理器 160 可根据应接收到每个传输包的时间来指派估计离开时间。

示例地，当接口 140 接收到程序数据时，把接收到的程序数据从接口 140 传送到主存储器 120 的包缓冲器 122。具体来说，接口 140 以某种形式的接收队列来存储接收到的程序数据。最好，接收到的程序数据为传输包的形式。

接口 140 在接收到数据时周期性地中断处理器 160。接口 140 可在每当接收到任何数量的数据时中断处理器 160，或者可在接收到一定数量的数据后中断处理器 160。如适配器 100，专门为接口 140 设计接收 PID 处理器子程序指针表 402。被这些指针所指向的子程序在许多方面类似于被与接收适配器 110 有关的接收 PID 处理器子程序指针表中的指针所指向的子程序。然而，这些子程序至少在以下方面是不同的。首先，异步接口 140 不必分配具有图 2 所示格式的描述符，而且也不必以传输包的形式来接收程序数据。例如，程序数据可以是 PES 包数据或 PS 包数据。在这种情况下，示例地，处理器 160 对所保留的传输包的 PID 执行的子程序包括用于把程序数据插入传输包中的处理。此外，可设置把指派给该适配器 140 的队列的接收描述符分配给每个接收到的传输包。处理器 160 把指向相应传输包的存储单元的指针存储在每个所分配的描述符的指针字段 129-4 中。示例地，最初，使实际接收时间字段 129-5 是空白的。

此外，包含 PCR 的每个传输包还包括以下处理。在第一次接收到带有任何程序的 PCR 的传输包时，处理器 160 从任何适配器 110 的基准时钟发生器 113(或被同步锁定于适配器 110 的基准时钟发生器 113 的任何其它基准时钟发生器 113)中获得一时间标记。如下所述，基准时钟 113 被同步锁定。把获得的时间标记指派给第一个曾经接收到的带有一程序的 PCR 的传输包作为该传输包的接收时间。注意，在此第一接收到的带有 PCR 的传输包之前可能已

接收到其它待再分多路复用的传输包。可把再分多路复用器节点 100 处的已知内部缓冲延迟添加到此接收时间标记上，以产生被指派给传输包(包含一特定程序的第一曾经接收到的 PCR)的估计离开时间。

在接收到带有一特定程序的 PCR 的第二后续传输包后，处理器 160 可估计具有异步接口 140 接收的该程序的 PCR 之间的传输包速率。这是如下实现的。处理器 160 形成该程序的两个相继 PCR 之差。然后，处理器把此差值除以同一程序中包含第一 PCR 的传输包和包含该程序的第二 PCR 的传输包之间的传输包数目。这产生了该程序的传输包速率。处理器 160 通过把该程序的传输包速率乘以每个这样的传输包与包含第一 PCR 的传输包的偏移或偏离来估计程序的这些 PCR 之间程序的每个传输包的离开时间。此偏移是通过从计算估计离开时间的传输包队列位置中减去带有第一 PCR 的传输包的传输包队列位置来确定的。(注意，一传输包的队列位置相对于所有接收到的流的所有接收到的传输包。)然后，处理器 160 把指派给包含第一 PCR 的传输包的估计离开时间加到如此产生的积中。示例地，处理器 160 把每个这样的传输包的估计离开时间存入指向它们的描述符的字段 129-10。

在把一估计离开时间标记指派给一程序的传输包后，处理器 160 可丢弃不想要在 TS 中输出的传输包(依据用户规定)。然后，对 TS 中携带的每个程序的每一对相继 PCR 连续地重复以上过程。然后，在处理器 160 执行发送 PID 处理器子程序期间，可把具有估计离开时间的描述符的数据传送到适当的发送队列。注意，可能在接收到该程序的第一 PCR 之前接收到该程序最初的一些传输包。对于这些传输包，仅把传输包速率估计为该程序的第一和第二 PCR 之间的传输包速率(即使这些包不在第一和第二 PCR 的传输包之间)。然后，如上所述确定估计离开时间。

如同从诸如适配器 110 等同步接口接收到 PCR，对经由异步接口 140 接收到的 PCR，校正每个程序时钟与用来指派估计接收时间标记及输出传输包的本地基准时钟 113 之间的偏移。与从适配器 110 接收到的传输包不同，从接口 140 接收到的传输包没有为它们所记录的实际接收时间标记。这样，不存在可从中准确地测量偏移的有关每个传输包的基准时钟。取而代之，处理器 160 使用再分多路复用器节点 100 中的发送队列长度的测量或其中的当前延迟来估计偏移。理想地，发送队列长度应不从再分多路复用器节点 100 中的预定已知延迟而改变。发送队列长度的任何改变都是适配器 110 的基准时钟发生器 113

相对于该程序的程序时钟的偏移的指示。这样，处理器 160 依据当前发送队列长度与所期望的理想发送队列长度之差来向上或向下调节偏移的测量。例如，每当把一发送描述符分配给一传输包时，处理器 160 测量当前发送队列长度并把它从再分多路复用器节点 100 中的理想发送队列长度中减去。此差值就是偏移。使用如此计算的偏移来调节 PCR 和携带这些 PCR 的传输包的估计离开时间。即，从经由异步接口接收到的传输包的 PCR 中减去如此计算的偏移，该接口置于相应于该传输包的估计离开时间的时隙之后的时隙。同样，可在指派实际派送时间前从带有 PCR 的传输包的估计离开时间中减去此偏移。注意，此估计的偏移仅用于从异步接口 140 接收到的传输包，而不用于经由诸如适配器 110 等同步接口接收到的其它传输包。

现在考虑竞争的问题。当两个(或更多)接收到的传输包争着要指派给输出的再分多路复用 TS 的同一传输包时隙(和实际派送时间)时，把一传输包指派给该时隙，并把另一传输包指派给下一时隙。如果另一传输包包含 PCR，则以该 PCR 偏离其理想时隙的时隙数来调节此 PCR，以反映指派给后来的时隙。

辅助输出定时

如上所述，接口 140 在任何特定时间都不接收传输包。同样，接口 140 在任何特定时间都不发送传输包。然而，即使接口 140 及相连的通信链路都不提供恒定的端-端延迟，但想要尽可能地减少端-端延迟的变化。再分多路复用器节点 100 提供把这种变化减到最小的方式。

依据一个实施例，处理器 160 对将经由接口 140 输出的每个传输包分配被指派给接口 140 的发送队列的一个发送描述符。这可使用指派给接口 140 的输出端口的发送队列中适当的一组发送 PID 处理器子程序。此外，处理器 160 指派一用于管理从此接口 140 的数据输出的适配器 110。虽然，在技术上把发送队列“指派”给接口 140，但实际上，由被指派用来管理从接口 140 的输出的适配器 110 的 DMA 控制电路 116 获得对指派给接口 140 的描述符队列的描述符的控制。数据链路控制电路 112 访问这些描述符，如下所述，这些描述符可保存在高速缓冲存储器 114 中。因而，实际上，由检查该队列的数据链路控制电路 112 所产生的中断来触发指派给此队列并由处理器 160 来执行的这组发送 PID 处理器子程序。

如上所述，响应于此中断，处理器 160 检查待调度的描述符，即在连接队列中，从待从接口 140 的输出端口输出的这些连接队列中选择一个或多个

描述符并把发送描述符分配给位于有关接口 140 的输出端口的发送队列尾部处的连接队列的选中描述符。与输出如上所述的传输包不同，处理器 160 还可收集有关连接队列的选中描述符的传输包，且实际上把它们组织成为类似于队列的缓冲器(如果这些缓冲器是接口 140 所必需的话)。

如上所述，DMA 控制电路 116 获得对接在 DMA 控制电路 116 获得控制的最后一个描述符后的一个或多个描述符的序列的控制，该序列与接口 140 的输出端口有关。(注意，是否检索相应于这些描述符的传输包都无关。由于数据链路控制电路 112 控制接口 140 处的传输包输出，所以不从连到该数据链路接口 112 的输出接口中输出传输包。或者，数据链路控制电路 112 完全可如上所述进行操作，从而产生输出 TS 的镜象(mirror)拷贝。在此情况下，还必须提供可被适配器 110 访问的每个传输包的第二拷贝)。如上所述，数据链路控制电路 112 检索高速缓冲存储器 114 中的每个描述符，并根据记录在字段 129-5 中的指示派送时间来确定相对于基准时钟发生器 113 所指示的时间要在何时发送相应的传输包。大致在基准时钟发生器 113 的时间等于此派送时间时，数据链路控制电路 112 对处理器 160 产生一中断，以指示现在应发送传输包。这可以是与数据链路控制电路 112 在发送 $k \geq 1$ 个传输包时所产生的相同的中断。然而，此中断最好是每 $k=1$ 个传输包时产生的。与之响应，处理器 160 检查发送 PID 处理器子程序的适当指针表，并执行正确的发送 PID 处理器子程序。在执行此发送 PID 处理器子程序时，处理器 160 发出使接口 140 发送一传输包的命令或中断。这使得大致在基准时钟发生器 113 的当前时间与相应于传输包的描述符中写入的派送时间匹配时从接口 140 的输出端口发送刚好下一传输包。注意，某些总线和中断等待时间将发生在时间链路控制电路 112 发出中断和接口 140 输出传输包之间。此外，某些等待时间可能发生在与接口 140 相连的通信链路上(由于它因冲突等而变得繁忙)。通过处理器 160 慎重地选择传输包的派送时间，可在一定程度上适应平均数量的等待时间。无论如何，传输包的输出可相当接近于正确时间，纵使此接近程度小于使用适配器 110 或接口 150 可实现的程度。此外，处理器 160 还如上所述把一个或多个描述符传送到指派给接口 140 的输出端口的发送队列。

适配器间基准时钟锁定

利用多个时钟发生器的任何同步系统的特定问题在于每个发生器的时间或计数与其它时钟发生器不完全相同。更有甚者，每个时钟发生器的计数可

能偏离(例如，因制造公差、温度、功率变化等的结果)。这种担心在环境 10 中也存在。每个再分多路复用器节点 100、数据注入器(injector)50、数据提取器 60、控制器 20 等都可具有一基准时钟发生器，诸如再分多路复用器节点 100 中适配器 110 的基准时钟发生器 113。希望锁定同一 TS 信号流动路径中的至少每个节点 50、60 或 100 的基准时钟发生器，从而它们具有相同时间。

在广播环境中，使产生、编辑或发送程序信息的所有设备同步是有用的。在模拟广播中，这可使用黑短脉冲串(black burst)发生器或 SMPTE 时间代码发生器。这种同步使实时视频馈送能够无缝接合，且减少了有关把异步视频馈送耦合在一起的噪声。

在再分多路复用器节点 100 中，对同步的需求更为重要。这是因为根据一个基准时钟来调度接收到的传输包的离开，且实际上根据第二基准时钟来检索接收到的传输包的派送。假设再分多路复用器节点 100 中的传输包所引起的任何等待时间都相同。然而，此假设仅在估计包离开所依据的基准时钟与实际派送传输包所依据的基准时钟之间只存在可忽略的偏移时才有效。

依据一个实施例，提供了多种锁定即让基准时钟发生器 113 同步的技术。在每一种技术中，相对于“主”基准时钟发生器周期性地调节每个“从”基准时钟发生器的时间。

依据第一种技术，把一适配器 110 的一个基准时钟发生器 113 指定为主基准时钟发生器。每个其它的适配器 110 的每个其它的基准时钟发生器 113 指定为从基准时钟发生器。处理器 160 周期性地获得每个基准时钟发生器 113(包括主基准时钟发生器和从基准时钟发生器)的当前系统时间。示例地，这是使用在特定时间周期内“休眠”(即，空闲)、唤醒并使处理器 160 获得每个基准时钟发生器 113 的当前时间的过程来实现的。处理器 160 把每个从基准时钟发生器 113 的当前时间与主基准时钟发生器 113 的当前时间相比较。根据这些比较，处理器 160 调节每个从基准时钟发生器 113，以使它们相对于主基准时钟发生器 113 同步。此调节可简单地通过对基准时钟发生器 113 进行重新加载、把调节的时间值加到基准时钟发生器 113 的系统时间上或(过滤)加速或减缓电压控制振荡器(它把时钟脉冲提供给基准时钟发生器 113 的计数器)的脉冲来实现的。最后一种调节形式类似于 MPEG-2 系统规定中所述的锁相回路反馈调节。

现在考虑主基准时钟发生器和从基准时钟发生器不在同一节点中而是通

过一通信链路相连的情况。例如，主基准时钟发生器可位于第一再分多路复用器节点 100 中，从基准时钟发生器可位于第二再分多路复用器 100 中，其中第一和第二再分多路复用器节点通过一在第一和第二再分多路复用器节点 100 的各适配器 110 之间延伸的通信链路相连。周期性地，响应于定时器过程，处理器 160 发出获得主基准时钟发生器 113 的当前时间的命令。适配器 100 通过把此当前时间提供给处理器 160 来响应。然后，处理器 160 经由此通信链路把此当前时间发送到每个其它的从基准时钟。然后，例如如上所述调节从基准时钟。

应注意，可把任何时间源或时间服务器用作主基准时钟发生器。经由具有恒定端-端延迟的专用通信链路把该主基准时钟发生器的时间发送到包含从基准时钟的每个其它节点。

如果一再分多路复用器 30 的两个或多个节点 20、40、50、60 或 100 在地理上分开很大的距离，则使每个节点的基准时钟发生器与任何其它节点的基准时钟发生器同步是不可能的。这是因为在一通信链路上发送的任何信号可能经历某些有限的传播延迟。这种延迟在传输包尤其是带有同步时间标记的传输包的发送中引起等待时间。取而代之，使用一与再分多路复用器 30 的每个节点等距离的基准时钟源是可能的。众所周知，美国政府保持地面和卫星基准时钟发生器。这些源在公知的载波信号上可靠地发送时间。诸如再分多路复用器节点 100 等每个节点可设有诸如 GPS 接收器 180 等接收器，该接收器能接收广播的基准时钟。每个节点 20、40、50、60 或 100 处的处理器 160(或其它电路)从接收器 180 周期性地获得基准时钟。处理器 160 可把获得的时间传送到适配器 110，以装入基准时钟发生器 113 中。然而，最好，处理器 160 向适配器 110 发出获得基准时钟发生器 113 的当前时间的命令。然后，处理器 160 发出根据从接收器 180 获得的时间与基准时钟发生器 113 的当前时间之间的不同来调节(例如，加速或减缓)基准时钟发生器 113 的电压控制振荡器的命令。

网络化再分多路复用

给出如上所述的操作，可在一网络上分配再分多路复用的各种功能。例如，可通过各种通信链路、适配器 110 和接口 140 和 150 使多个再分多路复用器节点 100 互连。可由控制器 20(图 1)控制这些再分多路复用器节点 100 中的每一个以共同起到单个再分多路复用器 30 的作用。

为了方便或灵活性起见，可能想要这种网络分布式再分多路复用器 30。例如，可把一再分多路复用器节点 100 连到多个文件服务器或存储器件 40(图 1)。可把第二再分多路复用器节点 100 连到多个其它输入源，诸如摄像机或解调器/接收器。可把每个其它的再分多路复用器节点 100 连到一个或多个发送器/调制器或记录器。或者，可如此连接再分多路复用器节点 100 以提供冗余功能继而在一再分多路复用器节点 100 失效或把它有目的地从服务中取出的情况下的容错。

考虑图 3 所示的第一网络再分多路复用器 30'。在此情况下，经由诸如 100 BASE-TX 以太网等异步网络把多个再分多路复用器节点 100'、100"、100'"相互连接起来。前两个再分多路复用器节点 100'、100"中的每一个接收四个 TS(TS10-TS13 或 TS14-TS17)并产生单个再分多路复用的输出 TS(TS18 或 TS19)。第三再分多路复用器节点 100'"接收 TS(TS18 和 TS19)并产生输出再分多路复用 TS(TS20)。在图 3 所示的例子中，再分多路复用器节点 100'"经由其适配器 100(图 2)接收从解调器/接收器实时发送的 TS(TS10-TS13)。另一方面，再分多路复用器 100"经由同步接口 150(图 2)从一存储器件接收预先存储的 TS(TS14-TS17)。再分多路复用器节点 100'和 100"中的每一个经由至再分多路复用器节点 100'"的异步(100 BASE-TX 以太网)接口 140(图 2)的异步(100 BASE-TX 以太网)接口 140(图 2)把其各自的输出再分多路复用 TS 即 TS18 或 TS19 发送到再分多路复用器节点 100'"。有利的是，再分多路复用器节点 100'和 100"中的每一个使用上述辅助输出定时技术，把这种通信所引起的端-端延迟的变化减到至少。在任何情况下，再分多路复用器节点 100'"使用上述对未定时数据进行重新定时的技术来估计 TS18 和 TS19 中每个程序的位速率并对 TS18 和 TS19 去抖动(dejitter)。

任选地，在系统 30'的至少一个通信链路上还可包括突发器件 200。例如，如同在 LAN 中，可与其它终端一起共享通信媒体来进行普通数据处理。然而，还为了把数据注入 TS(例如，TS20)中和/或从中提取数据而设置突发器件 200。例如，突发器件 200 可以是提供因特网访问的服务器、Web 服务器、Web 终端等。

当然，这仅仅是网络分布式再分多路复用器的一个例子。其它配置也是可能的。例如，连接有这些节点的网络的通信协议可以是 TAM、DS3 等。

应注意网络分布式再分多路复用器 30'的两个重要的性质。第一，在所示

的特定网络中，任何输入端口可接收来自任何输出端口的诸如突发数据或 TS 数据等数据。即，再分多路复用器节点 100′可接收来自再分多路复用器节点 100″或 100‴或突发器件 200 的数据，再分多路复用器节点 100″可接收来自再分多路复用器节点 100′或 100‴或突发器件 200 的数据，再分多路复用器节点 100‴可接收来自再分多路复用器 100′或 100″或突发器件 200 中任一个的数据，突发器件 200 可接收来自再分多路复用器节点 100′、100″或 100‴中任一个的数据。第二，执行数据提取和丢弃的再分多路复用器节点，即再分多路复用器节点 100‴可在同一通信链路上接收来自不止一个源即再分多路复用器 100′或 100″或突发器件 200 的数据。

由于这两个性质，所以从源节点到再分多路复用器内的目的地节点的传输包的“信号流动模式”与节点连接的网络拓扑结构无关。换句话说，网络分布式再分多路复用器 30′中的传输包所遍历的节点和通信链路路径与节点通过通信链路的精确物理连接无关。因而，可使用非常普通的网络拓扑结构——可以基本上任意的拓扑结构(总线、环、链、树、星形等)来连接再分多路复用器节点 100，而这些再分多路复用器节点仍能对 TS 进行再分多路复用，以实现实际任何类型的节点-节点信号流动模式。例如，节点 100′、100″、100‴和 200 以总线拓扑结构连接。还可实现以下用于所发送的数据的信号流动模式中的任一种：从节点 100′到节点 100″，然后到节点 100‴；从节点 100′和 100‴中的每一个平行地到达节点 200；从节点 200 和 100′平行地到达节点 100″，然后从节点 100″到节点 100‴等。在此类型的发送中，可能需要时分多路复用来交错不同组通信节点之间的信号流。例如，在图 3 所示的信号流中，在共享的通信媒体上时分多路复用 TS18 和 TS19。

以上讨论旨在对本发明进行说明。本领域内的技术人员可设计出大量的替代实施例，而不背离以下权利要求书的精神和范围。

说 明 书 附 图

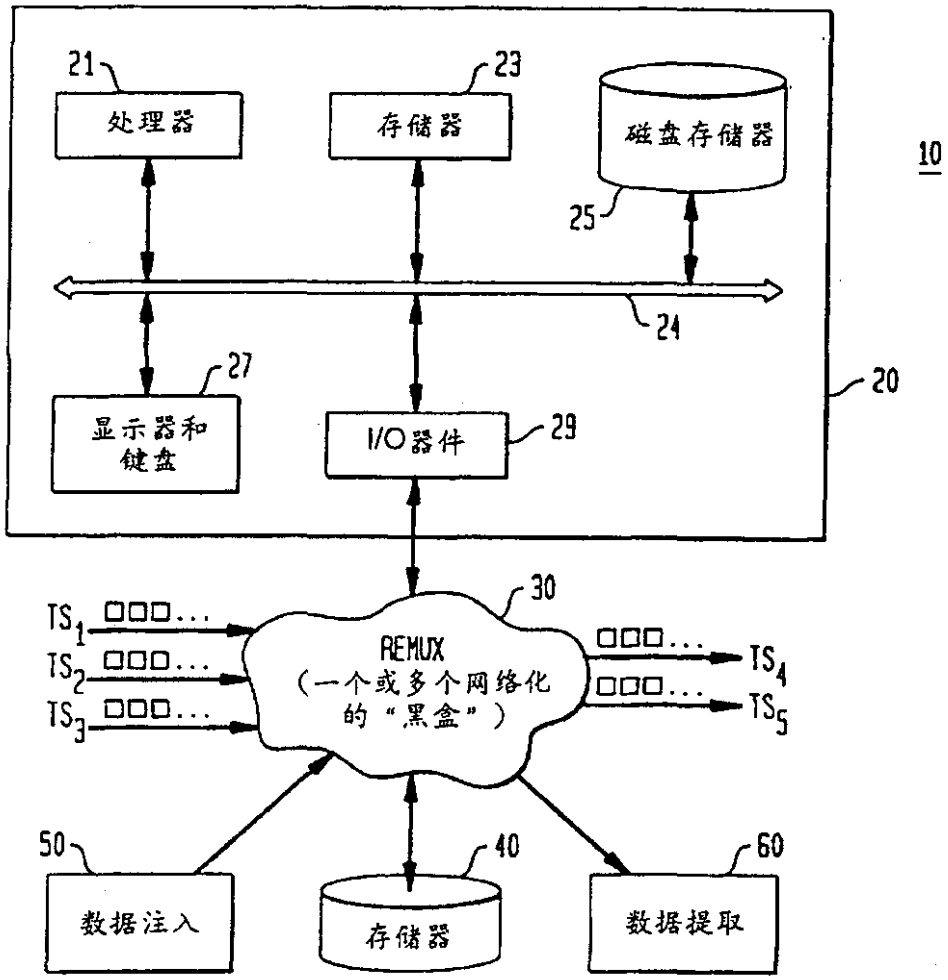

21
处理器
23
存储器
磁盘存储器
10
25
24
27
显示器和
键盘
I/O器件
29
20
30
REMUX
（一个或多个网络化
的“黑盒”）
TS1
TS2
TS3
TS4
TS5
50
数据注入
40
存储器
60
数据提取

图 1

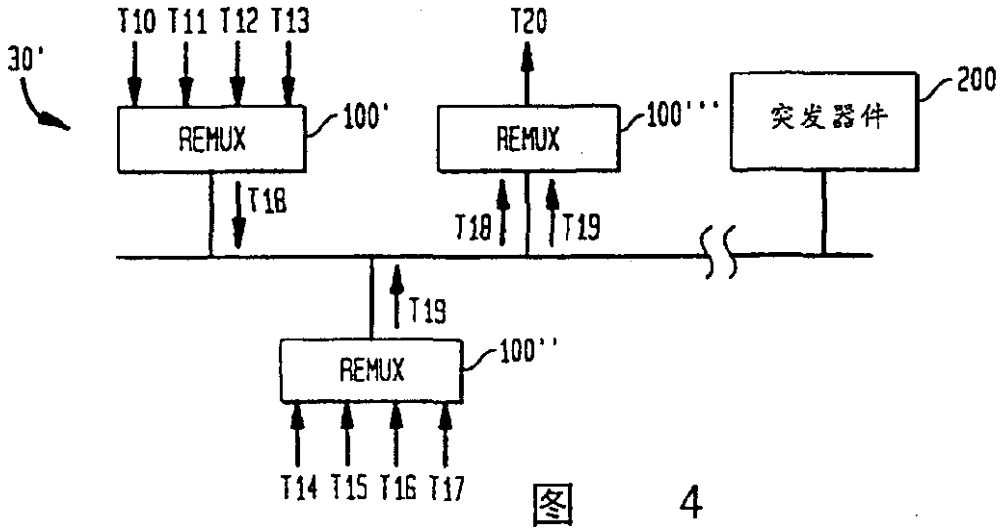

30'
T10 T11 T12 T13
REMUX
100'
T18
T20
REMUX
100'''
T18
T19
突发器件
200
T19
REMUX
100''
T14 T15 T16 T17

图 4

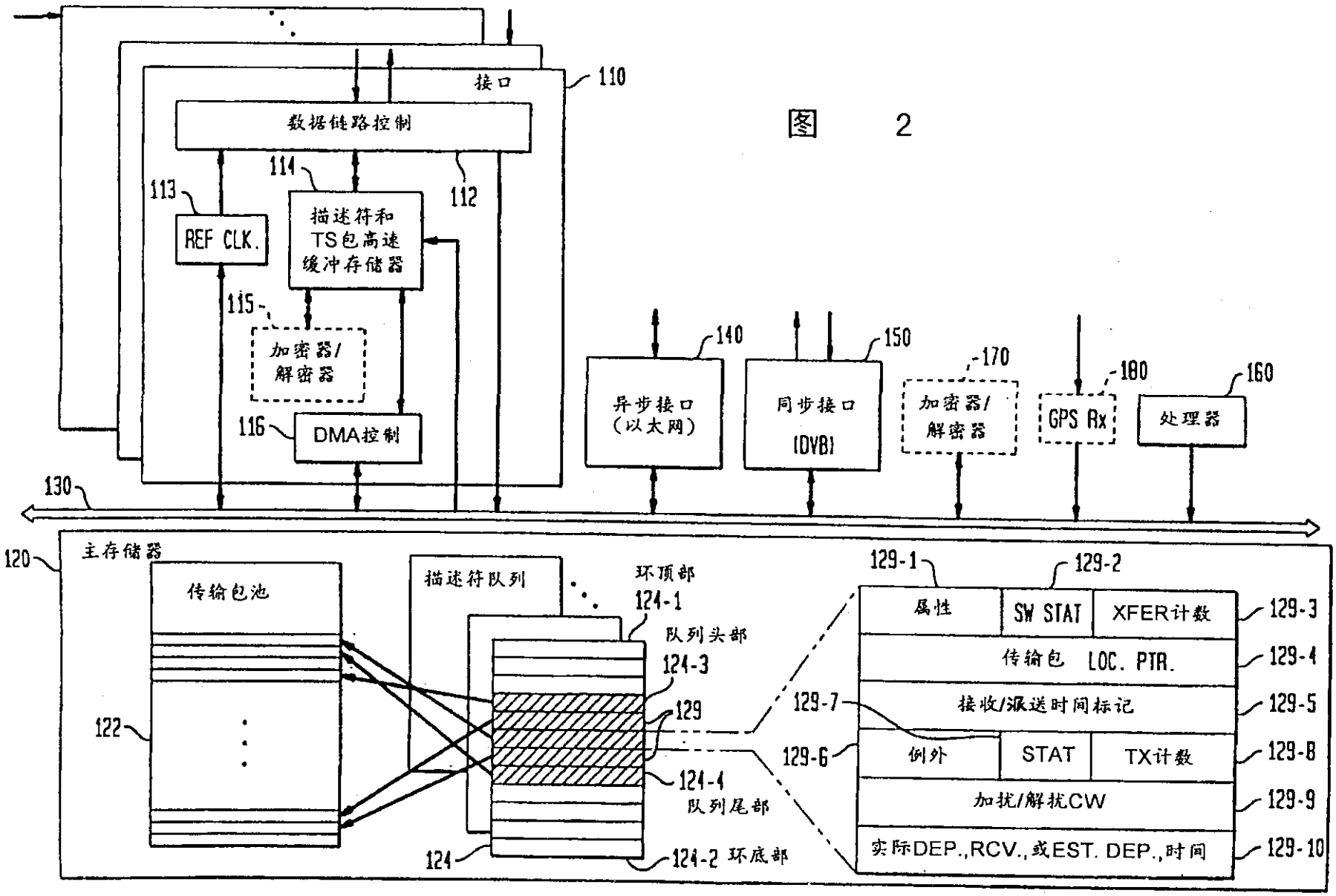
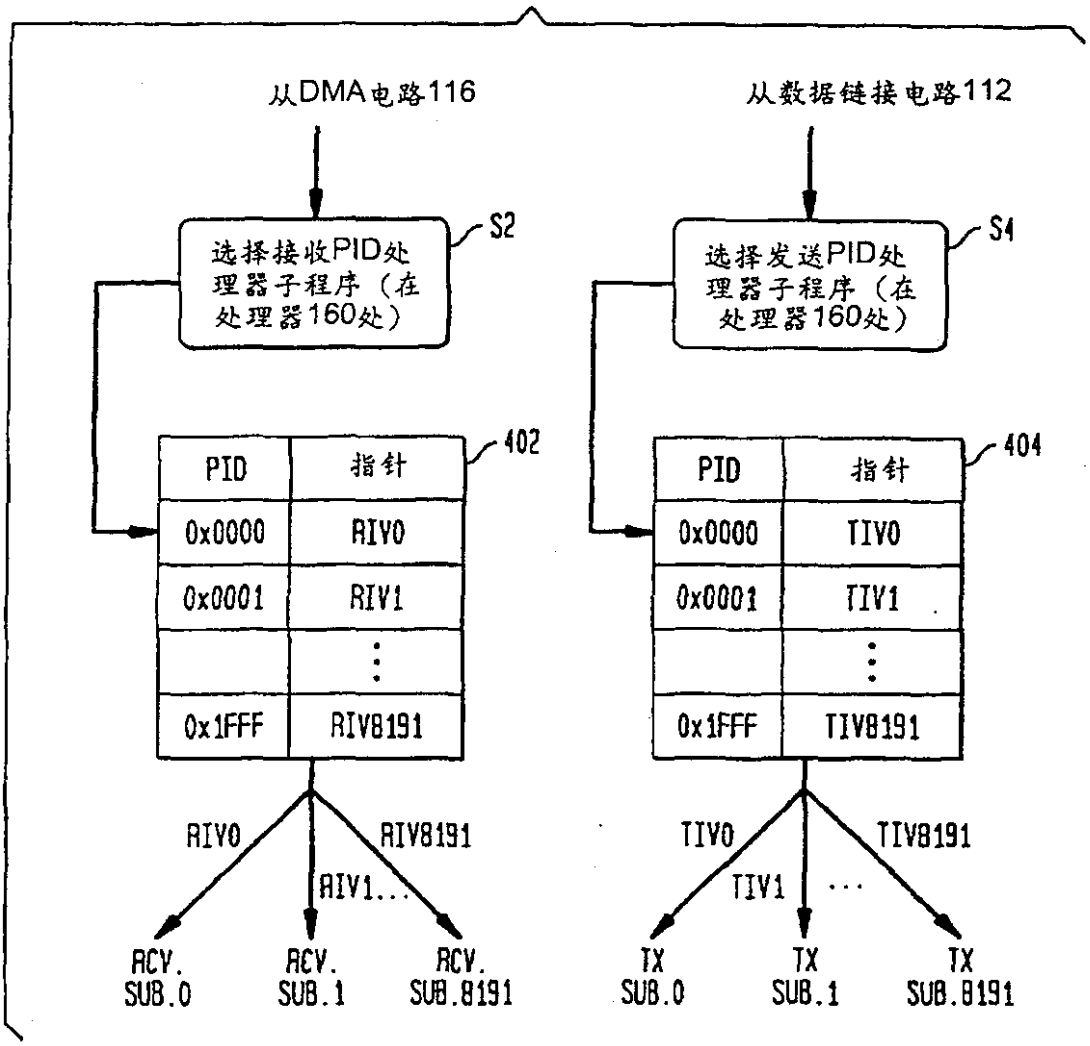

接口
110
数据链路控制
114
113
112
REF CLK.
描述符和
TS包高速
缓冲存储器
115
加密器/
解密器
116
DMA控制
140
异步接口
(以太网)
150
同步接口
(DVB)
170
加密器/
解密器
180
GPS Rx
160
处理器
130
120
主存储器
传输包池
描述符队列
环顶部
124-1
队列头部
124-3
129
122
124-4
队列尾部
124
124-2 环底部
129-1
129-2
属性
SW STAT
XFER计数
129-3
传输包 LOC. PTR.
129-4
129-7
接收/派送时间标记
129-5
129-6
例外
STAT
TX计数
129-8
加扰/解扰CW
129-9
实际DEP.,RCV.,或EST. DEP.,时间
129-10

图 2

图 3


从DMA电路116
从数据链接电路112
选择接收PID处理器子程序（在处理器160处）
S2
选择发送PID处理器子程序（在处理器160处）
S4
PID
指针
0x0000
RIV0
0x0001
RIV1
0x1FFF
RIV8191
402
PID
指针
0x0000
TIV0
0x0001
TIV1
0x1FFF
TIV8191
404
RIV0
RIV1...
RIV8191
TIV0
TIV1
TIV8191
RCV. SUB.0
RCV. SUB.1
RCV. SUB.8191
TX SUB.0
TX SUB.1
TX SUB.8191