(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 May 2001 (03.05.2001)

PCT

(10) International Publication Number
WO 01/31408 A1

(51) International Patent Classification[7]: G05B 19/18

(21) International Application Number: PCT/US00/29550

(22) International Filing Date: 27 October 2000 (27.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/161,901	27 October 1999 (27.10.1999)	US
60/162,989	1 November 1999 (01.11.1999)	US
60/162,802	1 November 1999 (01.11.1999)	US
60/162,801	1 November 1999 (01.11.1999)	US
60/182,864	16 February 2000 (16.02.2000)	US
60/185,192	25 February 2000 (25.02.2000)	US

(71) Applicant *(for all designated States except US)*: **ROY-G-BIV CORPORATION** [US/US]; Suite 4, 401 Bingen Point Way, Bingen, WA 98605 (US).

(72) Inventors; and

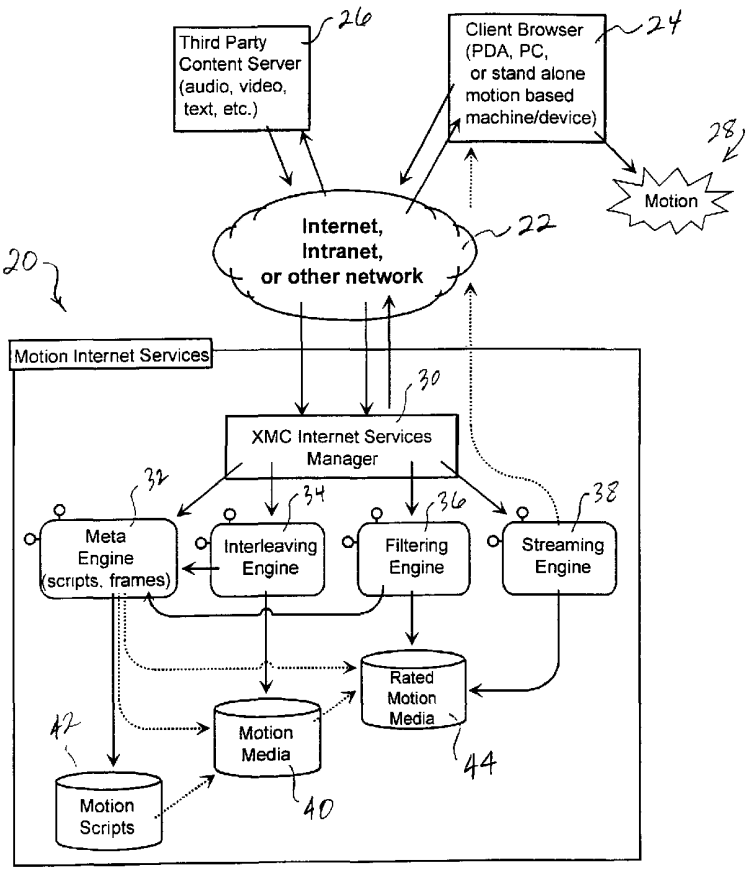
(75) Inventors/Applicants *(for US only)*: **BROWN, David, W.** [US/US]; Suite 4, 401 Bingen Point Way, Bingen, WA 98605 (US). **CLARK, Jay, S.** [US/US]; 670 North 34th #208, Seattle, WA 98103 (US).

(74) Agent: **SCHACHT, Michael, R.**; Hughes & Schacht, P.S., Suite 1, 2801 Meridian Street, Bellingham, WA 98225 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ,

[Continued on next page]

(54) Title: SYSTEMS AND METHODS FOR GENERATING AND COMMUNICATING MOTION DATA THROUGH A DISTRIBUTED NETWORK



(57) Abstract: A control software system (20) of figure (1) is adapted to generate, distribute and connect motion content in the form of motion media over a distributed network (22) from and to a client browser (24) and a content server (26) for operating a target device. The control software system (20) generates the motion media based on a motion program generated at the content server (26). The control software system (20) distributes the motion media to the client browser (24) associated with the target motion device.

WO 01/31408 A1

NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, 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.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEMS AND METHODS FOR GENERATING AND COMMUNICATING MOTION DATA THROUGH A DISTRIBUTED NETWORK

RELATED APPLICATIONS

This application claims priority of U.S. Provisional Patent Application Serial Nos. 60/161,901 filed on 10/27/99; 60/162,989 filed on 11/1/99; 60/162,802 filed on 11/1/99; 60/162,801 filed on 11/1/99; 60/182,864 filed on 2/16/00; and 60/185,192 which was filed on 2/25/00.

TECHNICAL FIELD

The present invention relates to motion control systems and, more particularly, to a software system that facilitates the creation and distribution of motion control software.

BACKGROUND OF THE INVENTION

The purpose of a motion control device is to move an object in a desired manner. The basic components of a motion control device are a controller and a mechanical system. The mechanical system translates signals generated by the controller into movement of an object.

While the mechanical system commonly comprises a drive and an electrical motor, a number of other systems, such as hydraulic or vibrational systems, can be used to cause movement of an object

based on a control signal. Additionally, it is possible for a motion control device to comprise a plurality of drives and motors to allow multi-axis control of the movement of the object.

The present invention is of particular importance in the context of a target device or system including at least one drive and electrical motor having a rotating shaft connected in some way to the object to be moved, and that application will be described in detail herein. But the principles of the present invention are generally applicable to any target device or system that generates movement based on a control signal. The scope of the present invention should thus be determined based on the claims appended hereto and not the following detailed description.

In a mechanical system comprising a controller, a drive, and an electrical motor, the motor is physically connected to the object to be moved such that rotation of the motor shaft is translated into movement of the object. The drive is an electronic power amplifier adapted to provide power to a motor to rotate the motor shaft in a controlled manner. Based on control commands, the controller controls the drive in a predictable manner such that the object is moved in the desired manner.

These basic components are normally placed into a larger system to accomplish a specific task. For example, one controller may operate in conjunction with several drives and motors in a multi-axis system for moving a tool along a predetermined path relative to a workpiece.

Additionally, the basic components described above are often used in conjunction with a host computer or programmable logic controller (PLC). The host computer or PLC allows the use of a high-level programming language to generate control commands that are

passed to the controller. Software running on the host computer is thus designed to simplify the task of programming the controller.

Companies that manufacture motion control devices are, traditionally, hardware oriented companies that manufacture software dedicated to the hardware that they manufacture. These software products may be referred to as low level programs. Low level programs usually work directly with the motion control command language specific to a given motion control device. While such low level programs offer the programmer substantially complete control over the hardware, these programs are highly hardware dependent.

In contrast to low-level programs, high-level software programs, referred to sometimes as factory automation applications, allow a factory system designer to develop application programs that combine large numbers of input/output (I/O) devices, including motion control devices, into a complex system used to automate a factory floor environment. These factory automation applications allow any number of I/O devices to be used in a given system, as long as these devices are supported by the high-level program. Custom applications, developed by other software developers, cannot be developed to take advantage of the simple motion control functionality offered by the factory automation program.

Additionally, these programs do not allow the programmer a great degree of control over the each motion control device in the system. Each program developed with a factory automation application must run within the context of that application.

In this overall context, a number of different individuals are involved with creating a motion control system dedicated to performing a particular task. Usually, these individuals have specialized backgrounds that enable them to perform a specific task in the overall process of creating a motion control system. The need thus exists for

systems and methods that facilitate collaboration between individuals of disparate, complimentary backgrounds who are cooperating on the development of motion control systems.

Conventionally, the programming and customization of motion systems is very expensive and thus is limited to commercial industrial environments. However, the use of customizable motion systems may expand to the consumer level, and new systems and methods of distributing motion control software, referred to herein as motion media, are required.

PRIOR ART

A number of software programs currently exist for programming individual motion control devices or for aiding in the development of systems containing a number of motion control devices.

The following is a list of documents disclosing presently commercially available high-level software programs: (a) Software Products For Industrial Automation, iconics 1993; (b) The complete, computer-based automation tool (IGSS), Seven Technologies A/S; (c) OpenBatch Product Brief, PID, Inc.; (d) FIX Product Brochure, Intellution (1994); (e) Paragon TNT Product Brochure, Intec Controls Corp.; (f) WEB 3.0 Product Brochure, Trihedral Engineering Ltd. (1994); and (g) AIMAX-WIN Product Brochure, TA Engineering Co., Inc. The following documents disclose simulation software: (a) ExperTune PID Tuning Software, Gerry Engineering Software; and (b) XANALOG Model NL-SIM Product Brochure, XANALOG.

The following list identifies documents related to low-level programs: (a) Compumotor Digiplan 1993-94 catalog, pages 10-11; (b) Aerotech Motion Control Product Guide, pages 233-34; (c) PMAC Product Catalog, page 43; (d) PC/DSP-Series Motion Controller C

Programming Guide, pages 1-3; (e) Oregon Micro Systems Product Guide, page 17; (f) Precision Microcontrol Product Guide.

The Applicants are also aware of a software model referred to as WOSA that has been defined by Microsoft for use in the Windows programming environment. The WOSA model is discussed in the book Inside Windows 95, on pages 348-351. WOSA is also discussed in the paper entitled WOSA Backgrounder: Delivering Enterprise Services to the Windows-based Desktop. The WOSA model isolates application programmers from the complexities of programming to different service providers by providing an API layer that is independent of an underlying hardware or service and an SPI layer that is hardware independent but service dependent. The WOSA model has no relation to motion control devices.

The Applicants are also aware of the common programming practice in which drivers are provided for hardware such as printers or the like; an application program such as a word processor allows a user to select a driver associated with a given printer to allow the application program to print on that given printer.

While this approach does isolates the application programmer from the complexities of programming to each hardware configuration in existence, this approach does not provide the application programmer with the ability to control the hardware in base incremental steps. In the printer example, an application programmer will not be able to control each stepper motor in the printer using the provided printer driver; instead, the printer driver will control a number of stepper motors in the printer in a predetermined sequence as necessary to implement a group of high level commands.

The software driver model currently used for printers and the like is thus not applicable to the development of a sequence of control commands for motion control devices.

The Applicants are additionally aware of application programming interface security schemes that are used in general programming to limit access by high-level programmers to certain programming variables. For example, Microsoft Corporation's Win32 programming environment implements such a security scheme. To the Applicants' knowledge, however, no such security scheme has ever been employed in programming systems designed to generate software for use in motion control systems.

SUMMARY OF THE INVENTION

The present invention is a system for generating and distributing motion media for motion control systems. A control software system is connected to a network such as the Internet. The control software system distributes motion media to clients through the network. Content servers connected to the network create application programs that can be directly transmitted to the clients as motion media or may be processed by a control command generating system that generates hardware independent motion media.

The control software system may include one or more of the following: a services manager module, a meta engine module, an interleaving engine module, a filtering engine module, and/or a streaming engine module. The software system further comprise memory for storing motion scripts, motion media, and/or rated motion media.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system interaction map of an exemplary control software system constructed in accordance with the principles of the present invention;

FIG. 2 is a block diagram depicting how the control software system of FIG. 1 can communicate with clients;

FIGS. 3-8 are module interaction maps depicting how the modules of the motion control system interact under various scenarios.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. 1 of the drawing, shown at 20 therein is a control software system adapted to generate, distribute, and collect motion content in the form of motion media over a distributed network 22 from and to a client browser 24 and a content server 26.

The distributed network 22 can be any conventional computer network such as a private intranet, the Internet, or other specialized or proprietary network configuration such as those found in the industrial automation market (e.g., CAN bus, DeviceNet, FieldBus, ProfiBus, Ethernet, Deterministic Ethernet, etc). The distributed network 22 serves as a communications link that allows data to flow among the control software system 20, the client browser 24, and the content server 26.

The client browsers 24 are associated with motion systems or devices that are owned and/or operated by end users. The client browser 24 includes or is connected to what will be referred to herein as the target device. The target device may be a hand-held PDA used to control a motion system, a personal computer used to control a motion system, an industrial machine, an electronic toy or any other type of motion based system that, at a minimum, causes physical motion. The client browser 24 is capable of playing motion media from any number of sources and also responds to requests for motion data from other sources such as the control software system 20. The exemplary client browser 24 receives motion data from the control software system 20.

The target device forming part of or connected to the client browser 24 is a machine or other system that, at a minimum, receives motion content instructions to run (control and configuration content) and query requests (query content). Each content type causes an

action to occur on the client browser 24 such as changing the client browser's state, causing physical motion, and/or querying values from the client browser. In addition, the target device at the client browser 24 may perform other functions such as playing audio and/or displaying video or animated graphics.

The term "motion media" will be used herein to refer to a data set that describes the target device settings or actions currently taking place and/or directs the client browser 24 to perform a motion-related operation. The client browser 24 is usually considered a client of the host control software system 20; while one client browser 24 is shown, multiple client browsers will commonly be supported by the system 20. In the following discussion and incorporated materials, the roles of the system 20 and client browser 24 may be reversed such that the client browser functions as the host and the system 20 is the client.

Often, but not necessarily, the end users will not have the expertise or facilities necessary to develop motion media. In this case, motion media may be generated based on a motion program developed by the content providers operating the content servers 26. The content server systems 26 thus provides motion content in the form of a motion program from which the control software system 20 produces motion media that is supplied to the client browser 24.

The content server systems 26 are also considered clients of the control software system 20, and many such server systems 26 will commonly be supported by the system 20. The content server 26 may be, but is not necessarily, operated by the same party that operates the control software system 20. Exhibit 1 attached hereto and incorporated by reference herein further describes the use of content servers in communications networks.

As briefly discussed above, the motion media used by the client browser 24 may be created and distributed by other systems and

methods, but the control software system 20 described herein makes creation and distribution of such motion media practical and economically feasible.

Motion media comprises several content forms or data types, including query content, configuration content, control content, and/or combinations thereof. Configuration content refers to data used to configure the client browser 24. Query content refers to data read from the client browser 24. Control content refers to data used to control the client browser 24 to perform a desired motion task as schematically indicated at 28 in FIG. 1.

Content providers may provide non-motion data such as one or more of audio, video, Shockwave or Flash animated graphics, and various other types of data. In a preferred embodiment, the control software system 20 is capable of merging motion data with such non-motion data to obtain a special form of motion media; in particular, motion media that includes non-motion data will be referred to herein as enhanced motion media.

The present invention is of particular significance when the motion media is generated from the motion program using a hardware independent model such as that disclosed in U.S. Patent Nos. 5,691,897 and 5,867,385 issued to the present Applicant, and the disclosure in these patents is incorporated herein by reference. However, the present invention also has application when the motion media is generated, in a conventional manner, from a motion program specifically written for a particular hardware device.

As will be described in further detail below, the control software system 20 performs one or more of the following functions. The control software system 20 initiates a data connection between the control software system 20 and the client browser 24. The control software system 20 also creates motion media based on input, in the

form of a motion program, from the content server system 26. The control software system 20 further delivers motion media to the client browser 24 as either dynamic motion media or static motion media. Dynamic motion media is created by the system 20 as and when requested, while static motion media is created and then stored in a persistent storage location for later retrieval.

Referring again to FIG. 1, the exemplary control software system 20 comprises a services manager 30, a meta engine 32, an interleaving engine 34, a filtering engine 36, and a streaming engine 38. In the exemplary system 20, the motion media is stored at a location 40, motion scripts are stored at a location 42, while rated motion data is stored at a location 44. The storage locations may be one physical device or even one location if only one type of storage is required.

Not all of these components are required in a given control software system constructed in accordance with the present invention. For example, if a given control software system is intended to deliver only motion media and not enhanced motion media, the interleaving engine 34 may be omitted or disabled. Or if the system designer is not concerned with controlling the distribution of motion media based on content rules, the filtering engine 36 and rated motion storage location 44 may be omitted or disabled.

The services manager 30 is a software module that is responsible for coordinating all other modules comprising the control software system 20. The services manager 30 is also the main interface to all clients across the network.

The meta engine 32 is responsible for arranging all motion data, including queries, configuration, and control actions, into discrete motion packets. The meta engine 32 further groups motion packets into motion frames that make up the smallest number of motion

packets that must execute together to ensure reliable operation. If reliability is not a concern, each motion frame may contain only one packet of motion data – i.e. one motion instruction. The meta engine 32 still further groups motion frames into motion scripts that make up a sequence of motion operations to be carried out by the target motion system. These motion packets and motion scripts form the motion media described above. The process of forming motion frames and motion scripts is described in more detail in Exhibit 2, which is attached hereto and incorporated herein by reference.

The interleaving engine 34 is responsible for merging motion media, which includes motion frames comprising motion packets, with non-motion data. The merging of motion media with non-motion data is described in further detail in Exhibit 3, which is attached hereto and incorporated by reference.

Motion frames are mixed with other non-motion data either on a time basis, a packet or data size basis, or a packet count basis. When mixing frames of motion with other media on a time basis, motion frames are synchronized with other data so that motion operations appear to occur in sync with the other media. For example, when playing a motion/audio mix, the target motion system may be controlled to move in sync with the audio sounds.

After merging data related to non-motion data (e.g., audio, video, etc) with data related to motion, a new data set is created. As discussed above, this new data set combining motion media with non-motion data will be referred to herein as enhanced motion media.

More specifically, the interleaving engine 34 forms enhanced motion media in one of two ways depending upon the capabilities of the target device at the client browser 22. When requested to use a non-motion format (as the default format) by either a third party content site or even the target device itself, motion frames are injected into the

non-motion media. Otherwise, the interleaving engine 34 injects the non-motion media into the motion media as a special motion command of 'raw data' or specifies the non-motion data type (ie 'audio-data', or 'video-data'). By default, the interleaving engine 34 creates enhanced motion media by injecting motion data into non-motion data.

The filtering engine 36 injects rating data into the motion media data sets. The rating data, which is stored at the rating data storage location 44, is preferably injected at the beginning of each script or frame that comprises the motion media. The client browser 22 may contain rating rules and, if desired, filters all received motion media based on these rules to obtain filtered motion media.

In particular, client browser 22 compares the rating data contained in the received motion media with the ratings rules stored at the browser 22. The client browser 22 will accept motion media on a frame by frame or script basis when the ratings data falls within the parameters embodied by the ratings rules. The client browser will reject, wholly or in part, media on a frame by frame or script basis when the ratings data is outside the parameters embodied by the ratings rules.

In another embodiment, the filtering engine 36 may be configured to dynamically filter motion media when broadcasting rated motion data. The modification or suppression of inappropriate motion content in the motion media is thus performed at the filtering engine 36. In particular, the filtering engine 36 either prevents transmission of or downgrades the rating of the transmitted motion media such that the motion media that reaches the client browser 22 matches the rating rules at the browser 22.

Motion media is downgraded by substituting frames that fall within the target system rating rules for frames that do not fall within the target system's rating. The filtering engine 36 thus produces a

data set that will be referred to herein as the rated motion media, or rated enhanced motion media if the motion media includes non-motion data.

The streaming engine 38 takes the final data set (whether raw motion scripts, enhanced motion media, rated motion media, or rated enhanced motion media) and transmits this final data set to the client browser 22. In particular, in a live-update session, the final data set is sent in its entirety to the client browser 22 and thus to the target device associated therewith. When streaming the data to the target device, the data set is sent continually to the target device. Optionally, the target system will buffer data until there is enough data to play ahead of the remaining motion stream received in order to maintain continuous media play. This is optional for the target device may also choose to play each frame as it is received yet network speeds may degrade the ability to play media in a continuous manner. This process may continue until the motion media data set ends, or, when dynamically generated, the motion media may play indefinitely.

Referring now to FIGS. 2, depicted therein is a block diagram illustrating the various forms in which data may be communicated among the host system software 20 and the target device at the client browser 22. Before any data can be sent between the host and the target, the network connection between the two must be initiated. There are several ways in which this initiation process takes place. As shown in FIG.2, this initiation process may be accomplished by broadcasting, live update, and request broker.

In addition, FIG. 2 also shows that, once the connection is initiated between the host and target systems, the content delivery may occur dynamically or via a static pool of already created content. When delivering dynamic content, the content may be sent via requests from a third party content site in a slave mode, where the

third party requests motion media from the host on behalf of the target system. Or the dynamic content may be delivered in a master mode where the target system makes direct requests for motion media from the host where the motion services reside.

In the following discussion, the scenario maps depicted in FIGS. 3-8 will be explained in further detail. These scenario maps depict a number of scenarios in which the control software system 20 may operate.

Referring initially to FIG. 3, depicted therein is a scenario map that describes the broadcasting process in which the host sends information across the network to all targets possible, notifying each that the host is ready to initiate a connection to transmit motion media. Broadcasting consists of initiating a connection with a client by notifying all clients of the host's existence via a connectionless protocol by sending data via the User Diagram Protocol (or UDP). The UDP is a connectionless protocol standard that is part of the standard TCP/IP family of Internet protocols. Once notified that the host has motion media to serve, each target can then respond with an acceptance to complete the connection. The broadcasting process is also disclosed in Exhibits 1 and 4, which are attached hereto and incorporated herein by reference.

The following steps occur when initiating a connection via broadcasting.

First, before broadcasting any data, the services manager 30 queries the meta engine 32 and the filter engine 36 for the content available and its rating information.

Second, when queried, the filter engine 36 gains access to the enhanced or non-enhanced motion media via the meta engine 32. The filtering engine 36 extracts the rating data and serves this up to the internet services manager 30.

Third, a motion media descriptor is built and sent out across the network. The media descriptor may contain data as simple as a list of ratings for the rated media served. Or the descriptor may contain more extensive data such as the type of media categories supported (i.e., medias for two legged and four legged toys available). This information is blindly sent across the network using a connectionless protocol. There is no guarantee that any of the targets will receive the broadcast. As discussed above, rating data is optional and, if not used, only header information is sent to the target.

Fourth, if a target receives the broadcast, the content rating meets the target rating criteria, and the target is open for a connection, the connection is completed when the target sends an acknowledgement message to the host. Upon receiving the acknowledgement message, the connection is made between host and target and the host begins preparing for dynamic or static content delivery.

Referring now to FIG. 4, depicted therein is a scenario map illustrating the process of live update connection initiation. A live update connection is a connection based on pre-defined criteria between a host and a target in which the target is previously registered or "known" and the host sends a notification message directly to the known target. The process of live update connection initiation is also disclosed in Exhibit 1 and in Exhibit 5, which is attached hereto and incorporated herein by reference.

The following steps take place when performing a live-update.

First, the internet services manager 30 collects the motion media and rating information. The motion media information collected is based on information previously registered by a known or pre-registered target. For example, say the target registers itself as a two-

legged toy – in such a case the host would only collect data on two-legged motion media and ignore all other categories of motion media.

Second, when queried, the filtering engine 36 in turn queries the meta engine 32 for the raw rating information. In addition, the meta engine 32 queries header information on the motion media to be sent via the live update.

Third, the motion media header information along and its associated rating information are sent to the target system. If rating information is not used, only the header information is sent to the target.

Fourth, the target system either accepts or rejects the motion media based on its rating or other circumstances, such as the target system is already busy running motion media.

FIG. 5 describes the process of request brokering in master mode in which the target initiates a connection with the host by requesting motion media from the host.

First, to initiate the request broker connection, the target notifies the host that it would like to have a motion media data set delivered. If the target supports content filtering, it also sends the highest rating that it can accept (or the highest that it would like to accept based on the target system's operator input or other parameters) and whether or not to reject or downgrade the media based on the rating.

Second, the services manager 30 queries the meta engine 32 for the requested media and then queries the filter engine 36 to compare the requested rating with that of the content. If the rating does not meet the criteria of the rating rules, the Filter Engine uses the content header downsizing support info to perform Rating Content Downsizing.

Third, the meta engine 32 collects all header information for the requested motion media and returns it to the services manager 30.

Fourth, if ratings are supported, the meta engine 32 also queries all raw rating information from the rated motion media 44. When ratings are used, the rated motion media 44 is used exclusively if available. If the media is already rated, the rated media is sent out. If filtering is not supported on the content server the rating information is ignored and the Raw Motion Scripts or Motion Media data are used.

Fifth, the motion media header information and rating information (if available) are sent back to the requesting target device, which in turn either accepts the connection or rejects it. If accepted, a notice is sent back to the services manager 30 directing it to start preparing for a content delivery session.

FIG. 6 describes request broker connection initiation in slave mode. In slave mode connection initiation, the target initiates a connection with the third party content server 26, which in turn initiates a connection with the host on behalf of the target system. Request brokering in slave mode is similar to request brokering in master mode, except that the target system communicates directly with a third party content server 26 instead of with the host system.

Slave mode is of particular significance when the third party content site is used to drive the motion content generation. For example, motion media may be generated based on non-motion data generated by the third party content site. A music site may send audio sounds to the host system, which in turn generates motions based on the audio sounds.

The following steps occur when request brokering in slave mode.

First, the target system requests content from the third party content server (e.g., requests a song to play on the toy connected to, or part of the target system).

Second, upon receiving the request, the third party content server locates the song requested.

Third, the third party content server 26 then sends the song name, and possibly the requested associated motion script(s), to the host system 20 where the motion internet service manager 30 resides.

Fourth, upon receiving the content headers from the third party content server 26, the services manager 30 locates the rating information (if any) and requested motion scripts.

Fifth, rating information is sent to the filtering engine 36 to verify that the motion media is appropriate and the requested motion script information is sent to the meta engine 32.

Sixth, the filtering engine 36 extracts the rating information from the requested motion media and compares it against the rating requirements of the target system obtained via the third party content server 26. The meta engine also collects motion media header information.

Seventh, the meta engine 32 extracts rating information from the rated motion media on behalf of the filtering engine 36.

Eighth, either the third party content server is notified, or the target system is notified directly, whether or not the content is available and whether or not it meets the rating requirements of the target. The target either accepts or rejects the connection based on the response. If accepted, the motion internet services begin preparing for content delivery.

FIG. 7 describes how the host dynamically creates motion media and serves it up to the target system. Once a connection is initiated between host and target, the content delivery begins. Dynamic content delivery involves actually creating the enhanced motion media in real time by mixing motion scripts (either pre-created scripts or dynamically generated scripts) with external media (ie audio,

video, etc). In addition, if rating downgrading is requested, the media is adjusted to meet the rating requirements of the target system.

The following steps occur when delivering dynamic content from the host to the target.

In the first step, either content from the third party content server is sent to the host or the host is requested to inject motion media into content managed by the third party content server. The remaining steps are specifically directed to the situation in which content from the third party content server is sent to the host, but the same general logic may be applied to the other situation.

Second, upon receiving the content connection with the third party content server, the services manager 30 directs the interleaving engine 34 to begin mixing the non-motion data (ie audio, video, flash graphics, etc) with the motion scripts.

Third, the interleaving engine 34 uses the meta engine 32 to access the motion scripts. As directed by the interleaving engine 34, the meta engine 32 injects all non-motion data between scripts and/or frames of motion based on the interleaving algorithm (ie time based, data size based or packet count based interleaving) used by the interleaving engine 34. This transforms the motion media data set into the enhanced motion media data set.

Fourth, if ratings are used and downgrading based on the target rating criteria is requested, the filtering engine 36 requests the meta engine 32 to select and replace rejected content based on rating with an equal operation with a lower rating. For example, a less violent move having a lower rating may be substituted for a more violent move having a higher rating. The rated enhanced data set is stored as the rated motion media at the location 44. As discussed above, this step is optional because the service manager 30 may not support content rating.

Fifth, the meta engine 32 generates a final motion media data set as requested by the filtering engine 36.

Sixth, the resulting final motion media data set (containing either enhanced motion media or rated enhanced motion media) is passed to the streaming engine 38. The streaming engine 38 in turn transmits the final data set to the target system.

Seventh, in the case of a small data set, the data may be sent in its entirety before actually played by the target system. For larger data sets (or continually created infinite data sets) the streaming engine sends all data to the target as a data stream.

Eighth, the target buffers all data up to a point where playing the data does not catch up to the buffering of new data, thus allowing the target to continually run motion media.

FIG. 8 describes how the host serves up pre-created or static motion media to the target system. Static content delivery is similar to dynamic delivery except that all data is prepared before the request is received from the target. Content is not created on the fly, or in real time, with static content.

The following steps occur when delivering static content from the host to the target.

In the first step, either motion media from the third party content server 26 is sent to the host or the host is requested to retrieve already created motion media. The remaining steps are specifically to the situation in which the host is requested to retrieve already created motion media, but the same general logic may be applied to the other situation.

Second, upon receiving the content connection with the third party content server, the services manager 30 directs the meta engine 32 to retrieve the motion media.

Third, the meta engine 32 retrieves the final motion media data set and returns the location to the services manager 30. Again, the final motion set may include motion scripts, enhanced motion media, rated motion media, or enhanced rated motion media.

Fourth, the final data motion media data set is passed to the streaming engine 38, which in turn feeds the data to the target system.

Fifth, again in the case of a small data set, the data may be sent in its entirety before actually played by the target system. For larger data sets (or continually created infinite data sets) the streaming engine sends all data to the target as a data stream.

Sixth, the target buffers all data up to a point where playing the data does not catch up to the buffering of new data, thus allowing the target to continually run motion media.

The control software system 20 described herein can be used in a wide variety of environments. The following discussion will describe how this system 20 may be used in accordance with several operating models and in several exemplary environments. In particular, the software system 20 may be implemented in the broadcasting model, request brokering model, or the autonomous distribution model. Examples of how each of these models applies in a number of different environments will be set forth below.

The broadcast model, in which a host machine is used to create and store a large collection of data sets that are then deployed out to a set of many target devices that may or may not be listening, may be used in a number of environments. The broadcast model is similar to a radio station that broadcasts data out to a set of radios used to hear the data transmitted by the radio station.

The broadcasting model may be implemented in the several areas of industrial automation. For example, the host machine may be used to generate data sets that are used to control machines on the

factory floor. Each data set may be created by the host machine by translating engineering drawings from a known format (such as the data formats supported by AutoCad or other popular CAD packages) into the data sets that are then stored and eventually broadcast to a set of target devices. Each target device may be the same type of machine. Broadcasting data sets to all machines of the same type allows the factory to produce a larger set of products. For example, each target device may be a milling machine. Data sets sent to the group of milling machines would cause each machine to simultaneously manufacture the same part thus producing more than one of the same part simultaneously thus boosting productivity.

Also, industrial automation often involves program distribution, in which data sets are translated from an engineering drawing that is sent to the host machine via an Internet (or other network) link. Once received the host would translate the data into the type of machine run at one of many machine shops selected by the end user. After translation completes, the data set would then be sent across the data link to the target device at the designated machine shop, where the target device may be a milling machine or lathe. Upon receiving the data set, the target device would create the mechanical part by executing the sequence of motions defined by the data set. Once created the machine shop would send the part via mail to the user who originally sent their engineering drawing to the host. This model has the benefit of giving the end user an infinite number of machine shops to choose from to create their drawing. On the other hand, this model also gives the machine shops a very large source of business that sends them data sets tailored specifically for the machines that they run in their shop.

The broadcasting model of the present invention may also be of particular significance during environmental monitoring and sampling.

For example, in the environmental market, a large set of target devices may be used in either the monitoring or collection processes related to environmental clean up. In this example, a set of devices may be used to stir a pool of water along different points on a river, where the stirring process may be a key element in improving the data collection at each point. A host machine may generate a data set that is used to both stir the water and then read from a set of sensors in a very precise manner. Once created the data set is broadcast by the host machine to all devices along the river at the same time to make a simultaneous reading from all devices along the river thus giving a more accurate picture in time on what the actual waste levels are in the river.

The broadcasting model may also be of significance in the agriculture industry. For example, a farmer may own five different crop fields that each requires a different farming method. The host machine is used to create each data set specific to the field farmed. Once created, the host machine would broadcast each data set to a target device assigned to each field. Each target device would be configured to only listen to a specific data channel assigned to it. Upon receiving data sets across its assigned data channel, the target device would execute the data set by running each meta command to perform the tilling or other farming methods used to harvest or maintain the field. Target devices in this case may be in the form of standard farming equipment retrofitted with motors, drives, a motion controller, and an software kernel (such as the XMC real-time kernel) used to control each by executing each meta command. The farming operations that may be implemented using the principles of the present invention include watering, inspecting crops, fertilizing crops and/or harvesting crops.

The broadcasting model may also be used in the retail sales industry. For example, the target devices may be a set of mannequins that are employ simple motors, drives, a motion controller, and a software kernel used to run meta commands. The host machine may create data sets (or use ones that have already been created) that are synchronized with music selections that are about to play in the area of the target mannequins. The host machine is then used to broadcast the data sets in a manner that will allow the target device to dance (or move) in a manner that is in sync with the music playing thus giving the illusion that the target device is dancing to the music. This example is useful for the retailer for this form of entertainment attracts attention toward the mannequin and eventually the clothes that it wears. The host machine may send data sets to the target mannequin either over a hard wire network (such as Ethernet), across a wireless link, or some other data link. Wireless links would allow the mannequins to receive updates while still maintaining easy relocation.

The broadcasting model may also be used in the entertainment industry. One example is to use the present invention as part of a biofeedback system. The target devices may be in the form of a person, animal or even a normally inanimate object. The host machine may create data sets in a manner that creates a feedback loop. For example a band may be playing music that the host machine detects and translates into a sequence of coordinated meta commands that make up a stream (or live update) of data. The data stream would then be broadcast to a set of target devices that would in-turn move in rhythm to the music. Other forms of input that may be used to generate sequences of meta commands may be some of the following: music from a standard sound system; heat detected from a group of people (such as a group of people dancing on a dance floor); and/or

the level of noise generated from a group of people (such as an audience listening to a rock band).

The broadcasting model may also have direct application to consumers. In particular, the present invention may form part of a security system. The target device may be something as simple as a set of home furniture that has been retrofitted with a set of small motion system that is capable of running meta commands. The host machine would be used to detect external events that are construed to be compromising of the residence security. When detected motion sequences would be generated and transmitted to the target furniture, thus giving the intruder the impression that the residence is occupied thus reducing the chance of theft. Another target device may be a set of curtains. Adding a sequence of motion that mimic that of a person repeatedly pulling on a line to draw the curtains could give the illusion that a person was occupying the residence.

The broadcasting model may also be applied to toys and games. For example, the target device may be in the form of an action figures (such as GI Joe, Barbie and/or Start Wars figures). The host machine in this case would be used to generate sequences of motion that are sent to each target device and then played by the end user of the toy. Since the data sets can be hardware independent, a particular data set may work with a wide range of toys built by many different manufacturers. For example, GI Joe may be build with hardware that implements motion in a manner that is very different from the way that Barbie implements or uses motion hardware. Using the motion kernel to translate all data from hardware independent meta commands to hardware specific logic use to control each motor, both toys could run off the same data set. Combining this model with the live updates and streaming technology each toy could receive and run the same data set from a centralized host.

The request brokering model also allows the present invention to be employed in a number of environments. Request brokering is the process of the target device requesting data sets from the host who in turn performs a live update or streaming of the data requested to the target device.

Request brokering may also be applied to industrial automation. For example, the present invention implemented using the request brokering model may be used to perform interactive maintenance. In this case, the target device may be a lathe, milling machine, or custom device using motion on the factory floor. When running data sets already broadcast to the device, the target device may be configured to detect situations that may eventually cause mechanical breakdown of internal parts or burnout of electronic parts such as motors. When such situations are detected, the target device may request for the host to update the device with a different data set that does not stress the parts as much as those currently being executed. Such a model could improve the lifetime of each target device on the factory floor.

Another example of the request brokering model in the industrial automation environment is to the material flow process. The target device in this example may be a custom device using motion on the factory floor to move different types of materials into a complicated process performed by the device that also uses motion. Upon detecting the type of material the target device may optionally request a new live update or streaming of data that performs the operations special to the specific type of material. Once requested, the host would transmit the new data set to the device that would in turn execute the new meta commands thus processing the material properly. This model would extend the usability of each target device for each could be used on more than one type of material and/or part and/or process.

The request brokering model may also be applied to the retail industry. In one example, the target device would be a mannequin or other target device use to display or draw attention to wares sold by a retailer. Using a sensor to detect location within a building or other space (i.e. a global positioning system), the target device could detect when it is moved from location to location. Based on the location of the device, it would request for data sets that pertain to its current location by sending a data request to the host pertaining to the current location. The host machine would then transmit the data requested. Upon receiving the new data, the device would execute it and appear to be location aware by changing its behavior according to its location.

The request brokering model may also be applied to toys and games or entertainment industry. Toys and entertainment devices may also be made location aware. Other devices may be similar to toys or even a blend between a toy and a mannequin but used in a more adult setting where the device interacts with adults in a manner based on the device's location. Also biofeedback aware toys and entertainment devices may detect the tone of voice used or sense the amount of pressure applied to the toy by the user and then use this information to request a new data set (or group of data sets) to alter its behavior thus appearing situation aware. Entertainment devices may be similar to toys or even mannequins but used in a manner to interact with adults based on biofeedback, noise, music, etc.

The autonomous distribution model may also be applied to a number of environments. The autonomous distribution model is where each device performs both host and target device tasks. Each device can create, store and transmit data like a host machine yet also receive and execute data like a target device.

In industrial automation, the autonomous distribution model may be implemented to divide and conquer a problem. In this application, a

set of devices is initially configured with data sets specific to different areas making up the overall solution of the problem. The host machine would assign each device a specific data channel and perform the initial setup across it. Once configured with its initial data sets, each device would begin performing their portion of the overall solution. Using situation aware technologies such as location detection and other sensor input, each target device would collaborate with one another where their solution spaces cross or otherwise overlap. Each device would not only execute its initial data set but also learn from its current situation (location, progress, etc) and generate new data sets that may either apply to itself or transmitted to other devices to run.

In addition, based on the devices situation, the device may request new data sets from other devices in its vaccinate in a manner that helps each device collaborate and learn from one another. For example, in an auto plant there may be one device that is used to weld the doors on a car and another device used to install the windows. Once the welding device completes welding it may transmit a small data set to the window installer device thus directing it to start installing the windows. At this point the welding device may start welding a door on a new car.

The autonomous distribution model may also be applied to environmental monitor and control systems. For example, in the context of flow management, each device may be a waste detection device that as a set are deployed at various points along a river. In this example, an up-stream device may detect a certain level of waste that prompts it to create and transmit a data set to a down-stream device thus preparing it for any special operations that need to take place when the new waste stream passes by. For example, a certain type of waste may be difficult to detect and must use a high precision and complex procedure for full detection. An upstream device may

detect small traces of the waste type using a less precise method of detection that may be more appropriate for general detection. Once detecting the waste trace, the upstream device would transmit a data set directing the downstream device to change to its more precise detection method for the waste type.

In agriculture, the autonomous distribution model has a number of uses. In one example, the device may be an existing piece of farm equipment used to detect the quality of a certain crop. During detection, the device may detect that the crop needs more water or more fertilizer in a certain area of the field. Upon making this detection, the device may create a new data set for the area that directs another device (the device used for watering or fertilization) to change it's watering and/or fertilization method. Once created the new data set would be transmitted to the target device.

The autonomous distribution model may also be applied to the retail sales environments. Again, a dancing mannequin may be incorporated into the system of the present invention. As the mannequin dances, it may send data requests from mannequins in its area and alter its own meta commands sets so that it dances in better sync with the other mannequins.

Toys and games can also be used with the autonomous distribution model. Toys may work as groups by coordinating their actions with one another. For example, several Barbie dolls may interact with one another in a manner where they dance in sequence or play house.

From the foregoing, it should be clear that the present invention may be embodied in forms other than those described above. The scope of the present invention should thus be determined by the claims ultimately allowed and not the foregoing detailed discussion of the preferred embodiments.

EXHIBIT 1

XMC Web Models

Community and Scheduling

Author: Dave Brown
Create Date: February 22, 2000 Save Date: XXX 0, 0000 Print Date: February 25, 2000
Project: $/prjcmpnt/xmc/v.100/persdev/doc/_devpers/web_models/webmodels
Document: Document2
Description:
Revisions:

ROY-G-BIV®
Software for a spectrum of ideas™

Table of Contents

1 • Overview

This document describes web models that are used to serve content, such as motion control instructions, audio, video, computer instructions and other media, in ways that are useful in that they allow users to collaborate with others when creating and using content as well as schedule times when content is to be played on content players.

There are three main chapters in this document that describe networked content models that allow a group of content users to collaborate as a community as well as networked content models that allow users to schedule when content is played. The chapters in this document are as follows:

- **Chapter 1 - Overview**; this chapter.
- **Chapter 2 - Community Model**; describes the network community model that allows several content users and/or creators collaborate on the creation or use of content.
- **Chapter 3 - Scheduling Model**; describes the network scheduling model that allows user to schedule when content is to be played.

2 • Community Model

The community model is designed to be a meeting place where more than one user can collaborate, share and discuss content. Users upload content that they create for use by (or sale to) others. Other users may merely download (or purchase) new content that others create for use on their local players where a player may be a toy, or other electronic device capable of running the content.

The following diagram shows the community model.

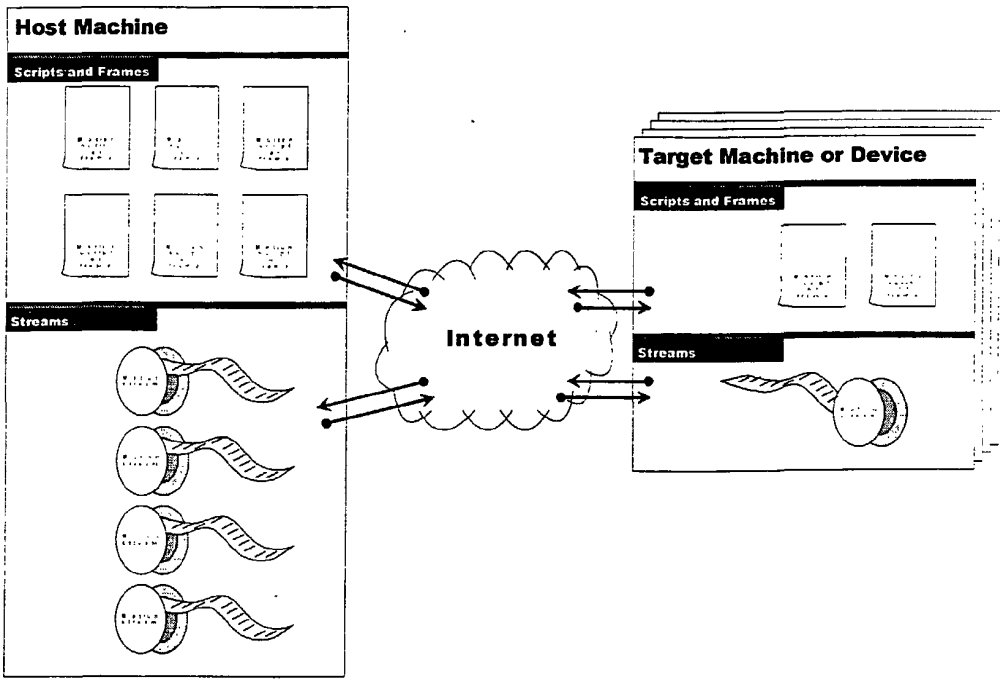


Figure 1 Community Model.

When using this content model users may either work as an individual where content is uploaded and downloaded across a network (ie the Internet, a local network, etc.) by a single person, or the model may be used by a group where each action performed when running the content is synchronized among all members of the group. The following sections describe each of these ideas in more detail.

Individual Sessions

Individual sessions involve a single user who downloads content for playing and/or uploads content that they create. To play content, the user browses the content list via a directory list, web browser, or other means of displaying the titles of each content script or content stream that is to be played.

The user may also create content of their own that they wish to share with others. Once created, the user uploads their content to the content site on the network.

Group Sessions

Group sessions use similar methods of uploading and downloading as described in the individual sessions with a new addition – users in the group are able to collaborate with one another in real-time. For example a set of users may optionally choose to run the same content script or stream. Using content synchronization, described below, the content running on each users machine or device is synchronized thus giving each end user the same experience.

NOTE: *when combined with scheduling, a group of devices may be synchronized with one other without requiring the intervention of the user. The chapter on the Scheduling Model below describes this in more detail.*

Content Synchronization

Sometimes is useful to have each machine or device play the same content script or stream and remain in sync with one another. For example, two users may own a toy doll that when directed to play a certain content script, both dolls sing and dance at the same time. In order to give each user a similar experience, both user devices are synchronized with one another either by the host machine or by communicating with one another. When synchronized each device runs the same content at the same time. If both dolls were placed side by side they would dance in a synchronized fashion when running the same content. It is not as important that each device run the same content, but when they do run the actions run are run in a manner that is in sync with the other device.

Host-to-Device synchronization is a synchronization model driven by the host (which could easily be a target device playing the host roll), where the host broadcasts content to other target devices. While broadcasting, the content data is injected with synchronization packets that each device uses to adjust the rate in which they play the content data received. Usually the host-to-device model is used when running a stream of content data on many target devices.

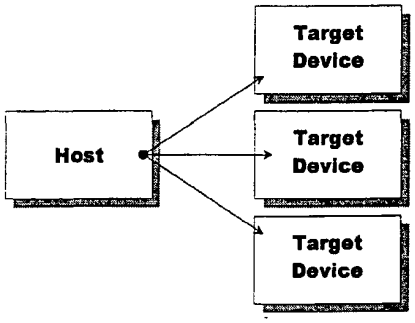


Figure 2 Host-to-Device Model.

When using the device-to-device synchronization model, a certain device requests that another device start a synchronization session. During the synchronization session, both devices periodically send synchronization packets to one another thus allowing each to change their individual play rates accordingly. Usually device-to-device synchronization is used when each device plays a content script that has been downloaded from the host.

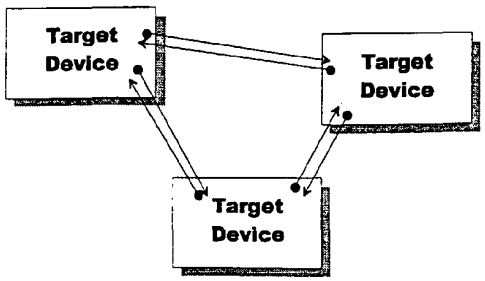


Figure 3 Device-to-Device Model.

Host-to-Device Synchronization

When using host-to-device synchronization, the host machine generates synchronization packets and injects them into the stream that is being played by each of the target machines.

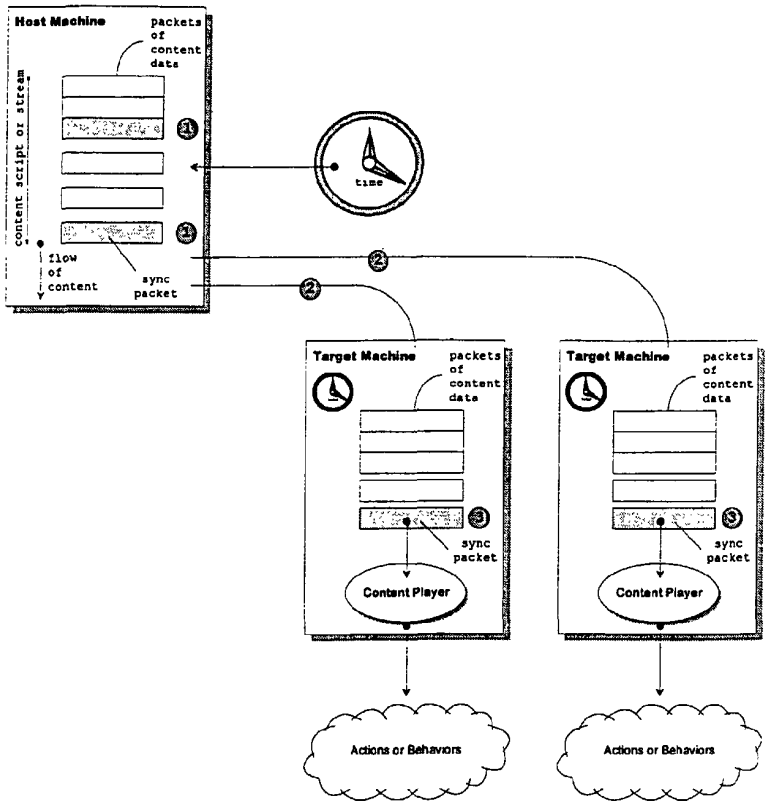


Figure 4 Host to Device Synchronization.

The following steps occur during host-to-device synchronization.

Step	Description
1	Using either a time signature, a packet sequence number, or some other reference data, the host periodically builds each synchronization packet. Each synchronization packet is then injected at pre-defined intervals into the content data.
2	The content data is sent to one or more target devices in a broadcast fashion where all data is sent out to all at the same time or as close to the same time as possible using a round robin approach where each packet is sent to all targets before the next packet is sent out.
3	Upon receiving each packet, the target devices buffer each packet until the synchronization packet is received. Upon receiving the synchronization packet, the remaining packets are processed by the content player thus causing movement or other actions to occur.

Device-to-Device Synchronization

When using device-to-device synchronization one device sends a synchronization packet to another requesting that it run a script or stream in sync with the requesting device.

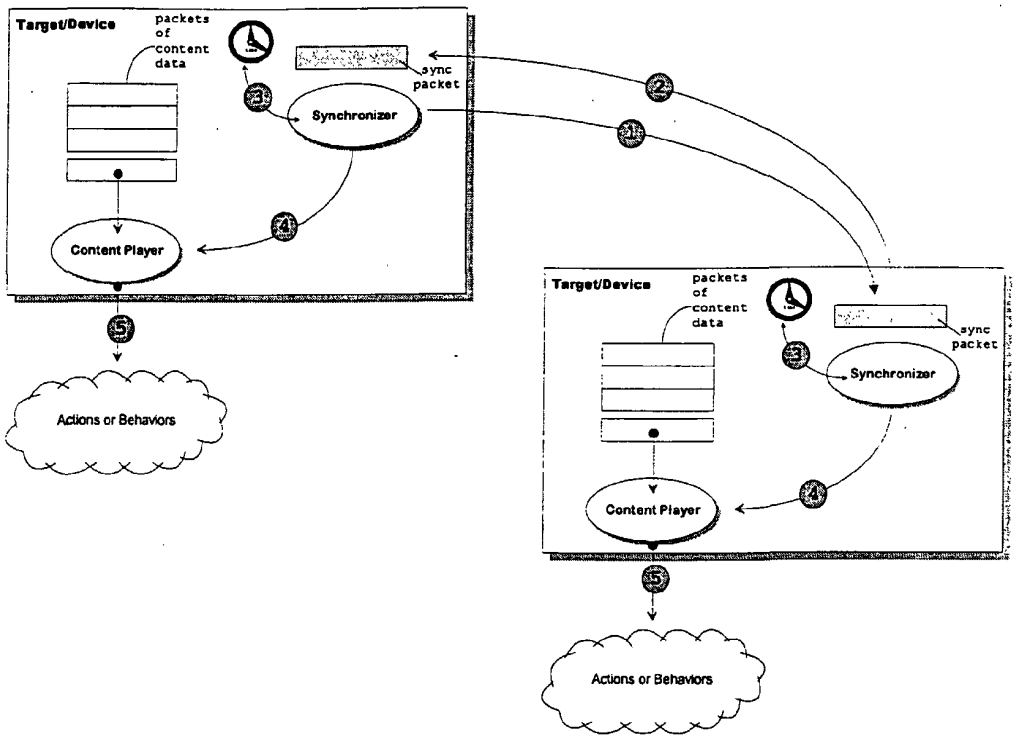


Figure 5 Device-to-Device Synchronization.

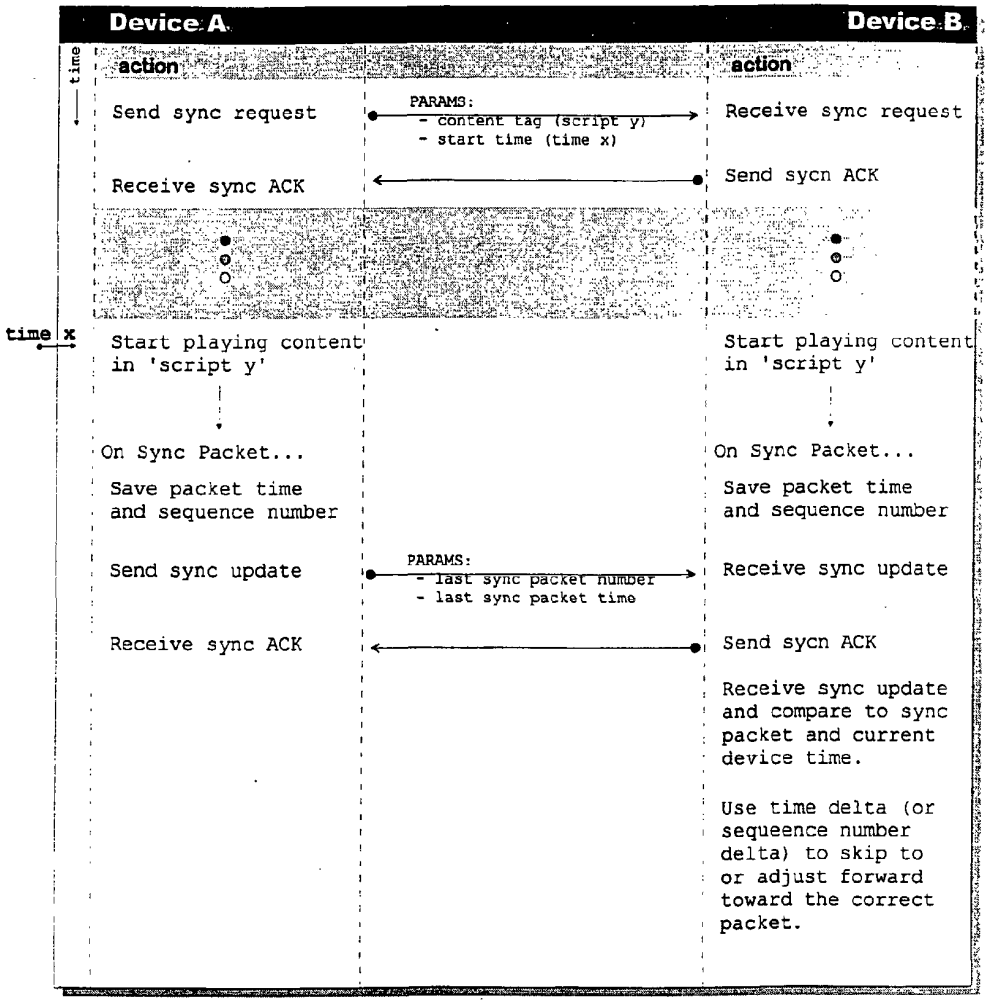
When performing device-to-device synchronization, the following steps occur.

Step	Description
1	To start the sync process, a device requests another device to start a synchronization session. In the request, the original device sends the script or stream name (that is to be run) and the start time.
2	Upon receiving the sync request, the receiving device responds with an acknowledgement message (ACK).
3	Each device waits for the start time, specified in the sync request, to strike.
4	When the time hits, both devices direct their content players to start running the content.
5	Running the content causes motions and other actions to occur.

NOTE: *Device-to-device synchronization assumes that all device clocks have been synchronized at least once with an outside reference point. For example, all devices may be requested (by a central host) to update their internal clocks with a GMT web server, or with the central host machine's clock.*

Synchronization Handshaking

The following shows an example synchronization handshaking session where Device A first requests that Device B synchronize with it at a certain time **X** with script **Y**.



When device B becomes out of sync with device A, it must either adjust its play rate of content packets or if the data contains motion, adjust the velocity of the moves (if any) caused by running the content data.

Play Rate Synchronization

The general algorithm used to decide how to adjust the play rate is as follows:

```
TimeA  = time of sync packet from deviceA
TimeB  = time of sync packet from deviceB
Tdelta = TimeA - TimeB

If (Tdelta > 0)
    Increase the Play Rate

Else If (Tdelta < 0)
    Decrease the Play Rate

Else
    Don't change the Play Rate
```

There are many ways to adjust the play rate and usually the method chosen will depend on the type of content that is used.

If the content contains dimensional point data (such as **<x,y,z>** for thee dimensional points), new points may be inserted within the point set thus causing each move to take slightly more time thus slowing down the play rate. In the same example speeding up the play rate is accomplished by skipping point data.

In another example, if the content contains motion instructions that involve causing moves at a certain velocity, the actual velocities may be altered causing the move directed by the instruction to complete in a shorter or longer amount of time. For example, to increase the play rate with this method, the velocity of a move would be increased slightly, which would in turn cause the motion instruction to complete in a shorter amount of time. Usually move instructions are accompanied with a 'wait for move' instruction which causes the instruction processor to wait until the move is complete before running the next instruction.

3 • Scheduling Models

Scheduling is used both direct the host to start broadcasting content and/or the device to start running content that is already downloaded or being broadcast. Host scheduling involves scheduling times where the host machine is to carry out certain actions such as initiating a broadcast secession, etc. Target scheduling involves scheduling each target device to begin running content at certain pre-determined points in time.

Host Scheduling and Broadcasting

With host scheduling, the host machine is configured to carry out certain operations at pre-determined points in time. For example, much like Television programming, the host machine may be configured to schedule broadcasting certain content streams at certain points in time.

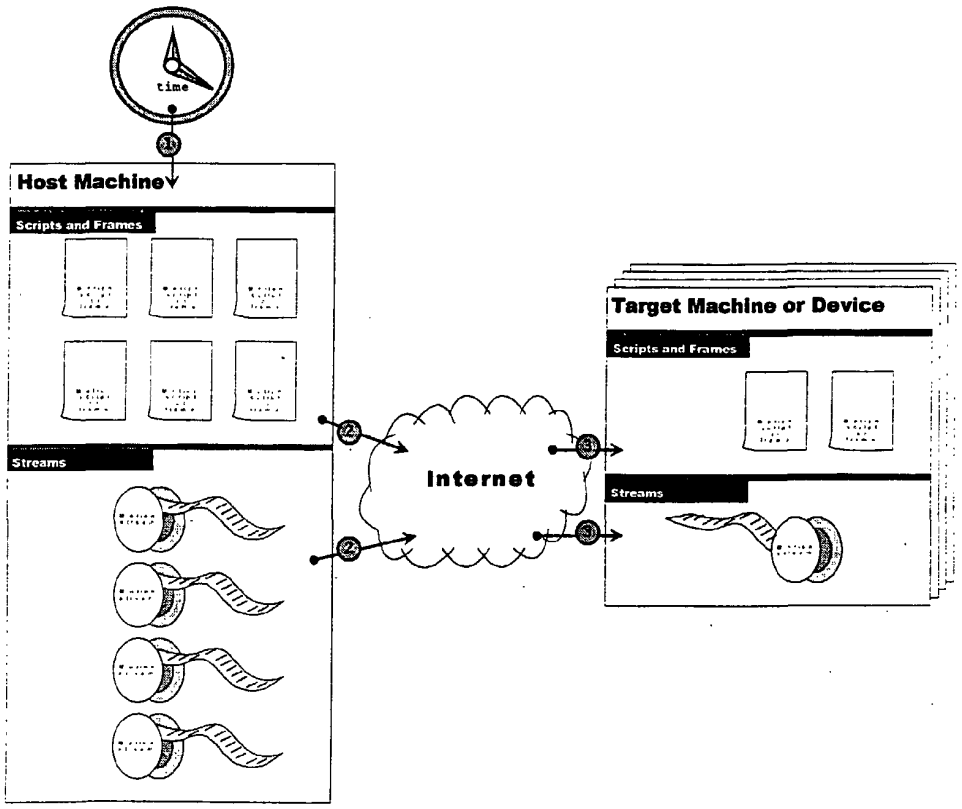


Figure 6 Host Scheduling Model

The following steps occur during host scheduling.

Step	Description
1	At certain pre-defined points in time, the host start performing the pre-determined actions such as broadcasting stream data.
2	At the pre-determined times, the host starts broadcasting content to the network (ie internet, or internal network).
3	Target devices that are configured to 'tune-into' certain broadcast channels run the content as it is received.

As an example of host scheduling, a host machine may be configured with several content streams that contain interleaved motion/audio data. At each pre-determined 'scheduled' time, the host starts broadcasting the interleaved motion/audio data to any devices that may be listening. As an example, the listening device may be a dancing mannequin that plays music as it dances thus giving the appearance that the mannequin is dancing to the music.

Target Scheduling

Target scheduling involves the target device being programmed to request and run content from the host (or run data sets already downloaded) at certain scheduled times.

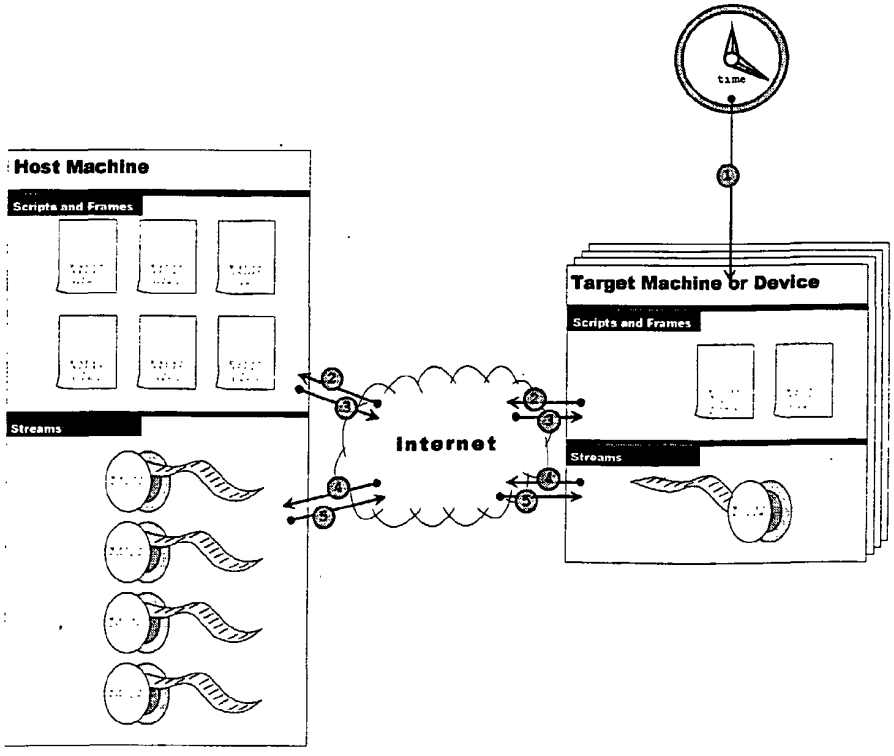


Figure 7 Target Scheduling Model.

The following steps occur during target based scheduling.

Step	Description
1	The target device is programmed to wait for a certain point in time to hit.
2	When the scheduled time hits, the target device's scheduler wakes up and queries the host across the network for a content script.
3	Upon receiving the content script, the target device begins running the content.
4	When the scheduled time hits, the target device's scheduler may also query the host across the network for a content stream.
5	Upon receiving the content stream, the target device starts running each instruction.

As an example of target based scheduling, the owner of a motion-based toy would go to a web site and select a certain motion script or stream to run on a certain data (i.e. a parent may select the Macarena dance and run it on their child's birthday as a surprise). Once scheduled, on the child's birthday, the toy would automatically connect to the host web site, download the data and start running the dance.

EXHIBIT 2

XMC Extensions

Scripts and Framing of Motion Sequences

Author: Dave Brown
Create Date: October 26, 1999 Save Date: XXX 0, 0000 Print Date: October 27, 1999
Project: $/prjcmpnt/xmc/v.100/int/persdev/doc/des/script_frames
Document: Document2
Description:
Revisions:

ROY-G-BIV®
Software for a spectrum of ideas™

Table of Contents

1 • Overview

Many applications that use motion control often use several sequences of basic motion operations to perform the operations carried out by the machine. The creation of each motion sequence is usually created off-line and then downloaded to the machine on which it is run. This model becomes fairly limited, and almost unusable, when the connection between the host machine (used to create and store the motion sequence) and the target device (the consumer of the motion data) becomes intermittent or unreliable.

This document describes the process of breaking up each sequence of basic motion operations, or motion scripts, into small frames of motion operations that must be sent in their entirety before actually being run on the target device. Breaking a script of motion operations into small frames is very important for it is easier to send small data packets across an unreliable or intermittent data line than it is to send large data packets.

2 • System Design

The meta scripting system is made up of scripts, frames and meta commands, where each frame is a set of meta commands and each script is a set of motion frames.

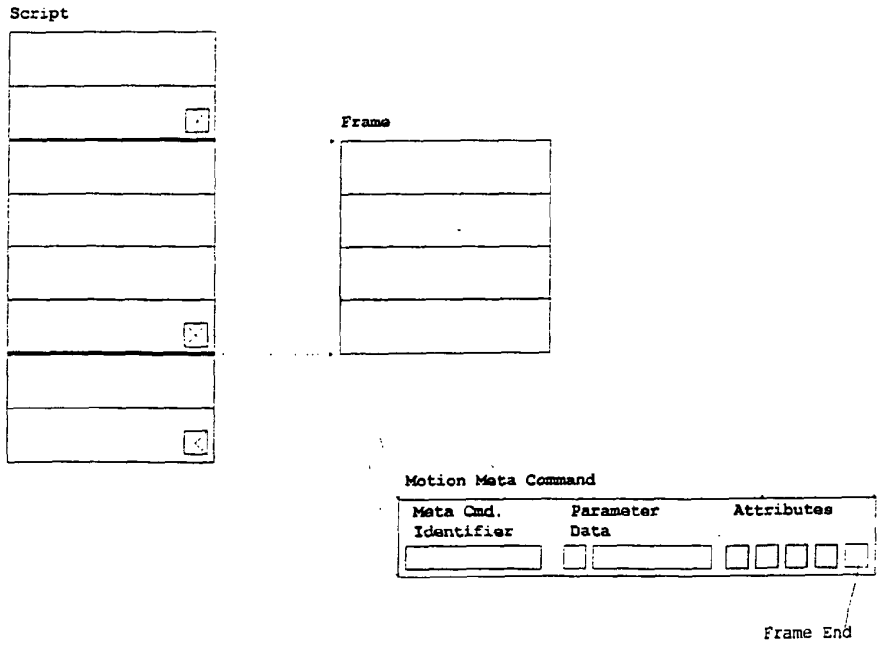


Figure 1 Meta Scripting System

The following pieces make up the motion script and framing system.

Item	Description
Meta Command	A meta command is an atomic script element that describes a simple motion control operation that is to be executed by the target device.
Motion Frame	A motion frame is a set of meta commands that end with a meta command that has its 'Frame End' attribute set to TRUE. All other previous meta commands in the frame must have this attribute set to FALSE. The script designer is recommended to build each frame making up a script in such a manner that it will run successfully on the device in its entirety and place the device in a safe state upon its completion. For example the designer should end all frames that contain move operations with a Wait operation that waits for all motion to complete) before continuing processing other motion operations located within another frame.

Motion Script

A motion script is a set of motion frames that define a certain operation carried out by the target device. The script will execute one frame at a time on the device.

Constructing the script with frames of meta commands is recommended for doing so allows the designer to somewhat detach the download of the script to the device from the execution of those scripts on the target device. For example, while the host is downloading a large script, the target device may actually start running the first frames in the script even before receiving the entire script. This is an option that the device may elect to take, but is not required.

Meta Commands

Each meta command contains all information necessary to describe a basic motion operation carried out by the script engine running on the target device. The script engine may opt to use one of many different forms of implementations. For example, the hardware independent XMC Motion Control system may be used as an implementation, or the script engine may just as well use a hardware dependent implementation for each operation. The meta command data allows the designer to separate the data describing the motion control operation from the actual implementation.

Definition

```
typedef struct XMCMetaCommand
{
    DWORD       dwMetaCmdID;
    LPVARIANT   pParamData;
    DWORD       dwParamDataCount;
    DWORD       dwFlags;
};
```

dwMetaCmdID – The Meta command identifier, which is a unique identifier that corresponds to a certain basic motion operation implementation that is to be run when this meta command is encountered by the script engine running on the target device.

pParamData – array of VARIANT [1]structures that each describe a single parameter used when running the meta command.

dwParamDataCount – number of elements within the *pParamData* array of elements.

dwFlags – set of attribute flags that describe the meta command and how it is to be processed. The following flags are supported.

Flag:	Description:
XMC_MF_FRAMEEND	End meta command within the current

[1] For more information on the VARIANT structure see the Microsoft® Win32® Reference Guide or the Microsoft® MSDN® on-line help.

frame of commands.

Notes

Out of the complete set of meta commands, several are defined with the 'Frame End' attribute set to TRUE by default. Defaulting several key meta commands to be frame end commands allows the system designer to quickly build meta scripts without having to worry about the framing mechanics used by the script engine. However, if they choose to have more direct control over command framing, they can easily change the *dwFlags* field of any command and enable or disable its XMC_MF_FRAMEEND attribute.

Motion Frames

A motion frame is a set of meta commands where only one, the end meta command, has its 'Frame End' attribute set to TRUE – the 'Frame End' attribute for all other commands is set to FALSE. The main purpose of the frame is to provide to the script engine on the target device a sequence of motion operations that can run in a reliable manner even if the data link between the host machine and device becomes intermittent or is clipped. The target device's script engine will only run a frame once it is received in its entirety.

Definition

Each frame is actually a contiguous set of elements within an array of XMC Meta Commands where the last element in the set has the 'Frame End' attribute set to TRUE. The next frame starts immediately after the last element in the previous frame.

Motion Scripts

A motion script is a set of meta commands that defines a sequence of basic motion operations to be run by the target device. Each script is made up of one or more frames of motion. When sending a script to the target device the data is sent a frame at a time. Upon receiving the script data, the target device will only run each frame of motion only after each complete frame has been received.

3 • General Use

In the meta scripting system, there are two steps that take place: a.) Building the scripts, and b.) Running the scripts. The following sections describe each of these in more detail.

Building Scripts

Building of scripts involves selecting from the set of meta commands available to build scripts of motion sequences that define the motion operations. The following diagram shows the details of this process.

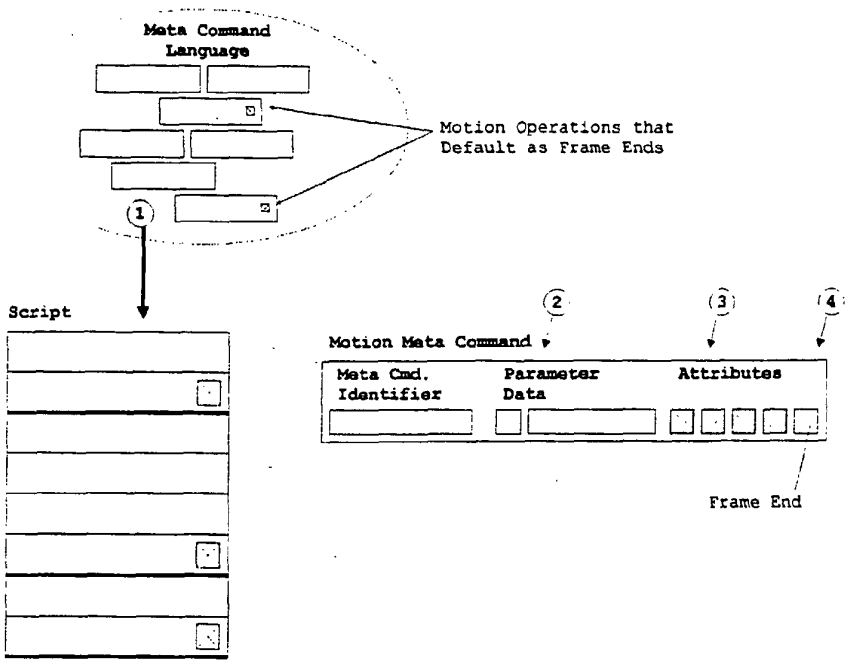


Figure 2 Building Script Data

The steps making up this process are as follows.

Step	Description
1	Select meta commands from the supported set of meta commands making up the scripting language to build each script in a manner that defines the sequence of motions to be run on the target device.
2	Set the parameter data for the meta command.
3	Set the general attributes (if any) supported by the meta command.

Set (or unset) the 'Frame End' attribute for the meta command. This step is optional for each meta command is defined with a default for this attribute which value depends on the command.

Once built, the script data can either be stored to a persistent medium and/or run on the target device.

Running Frames

To run the script data on the target device, the data must first be transferred to the device, which in turn then runs the data as each frame is received.

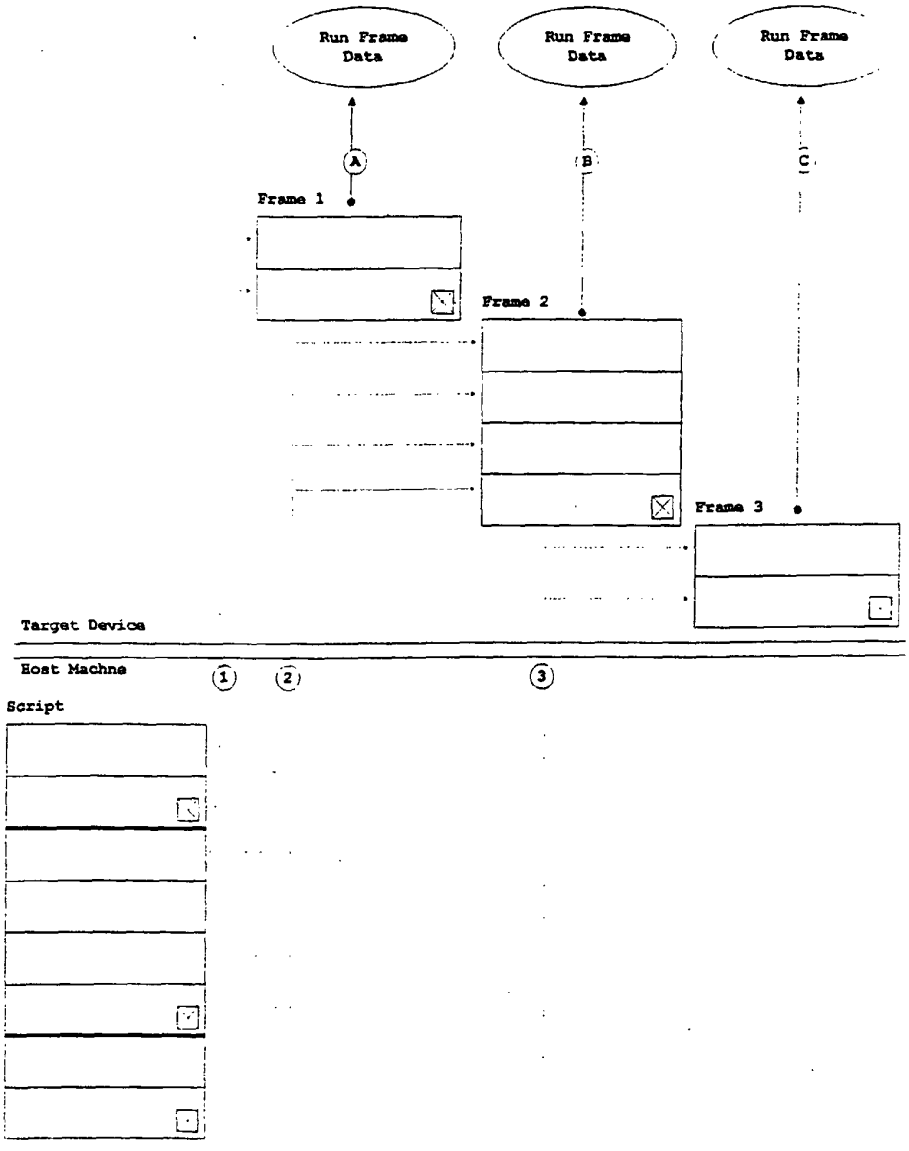


Figure 3 Running Script Data

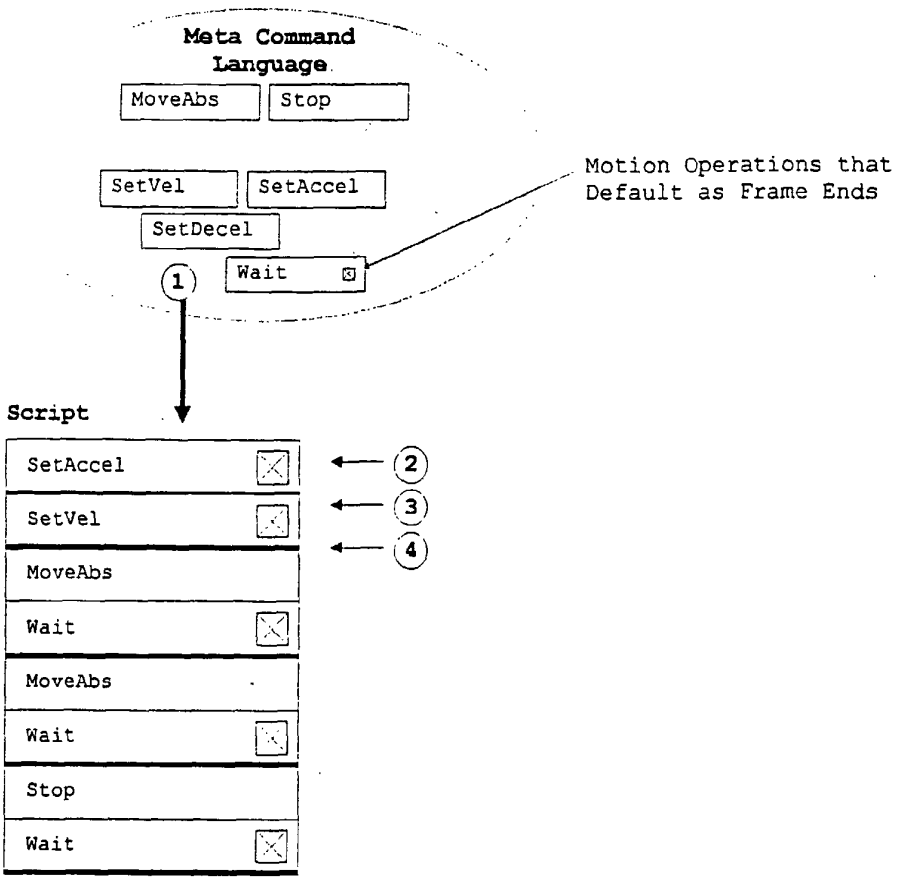
The steps involved when running the script data are as follows.

Step	Description
1	Each frame is transferred to the device, in this case, the first frame in the script is transferred.

2	The second frame in the script is transferred.
3	The third frame in the script is transferred.
A	Once received, the first frame in the script is run.
B	Once received and the first frame is done running, the second frame is run.
C	Once received and the second frame is done running, the third frame is run.

4 • Example

In the following example, a script containing a sequence of motions is built. The motion sequence below is used to set the acceleration and velocity, make two absolute moves, and then stop all motion (if any).



The following steps were used to build this script.

Step	Description
1	Meta commands are selected from the meta language and organized in the desired execution order in the script.
2	All parameter data is entered for each meta command.
3	All general attributes (if any) are set for each meta command.
4	The 'Frame End' attribute is set on several commands to make the download/run process more efficient. In this example this attribute was set on the 'SetVel' and 'SetAccel' meta commands for their default

setting for the 'Frame End' attribute is FALSE. The 'Wait' command, on the other hand has a default setting of 'TRUE' for the device must wait for a motion to complete before completing most operations.

Running the Data

When running the data, the host sends each motion frame to the device starting with the first one. The device, in turn, runs each frame as it is received. The following download/run process takes pace with the motion data.

a.) Download Frame 1 - { SetAccel }

b.) Download Frame 2 - { SetVel }

c.) Download Frame 3 - { MoveAbs, Wait }

d.) Download Frame 4 - { MoveAbs, Wait }

e.) Download Frame 5 - { Stop, Wait }

During the download process, the device may start running each frame as it is received. The following table shows an example of how the download and run sequences may actually overlap.

Frame	Meta Commands	Host Machine	Target Device
1	{ SetAccel }	Download Frm 1	Waiting for input
2	{ SetVel }	Download Frm 2	Run Frm 1
3	{ MoveAbs, Wait }	Download Frm 3	Run Frm 2
4	{ MoveAbs, Wait }	Download Frm 4	Run Frm 3
5	{ Stop, Wait }	Download Frm 5	Pending on Frm 3

EXHIBIT 3

Packet Count Based Packing

When building the interleaved data stream with count based packing, a specified count of packets from the current data type (ie motion, audio, video, etc) are placed in the target interleaved data stream before adding packets from other data types. After switching to the new data type a specified count of packets from the new data type are placed in the data stream. A specified count of packets of each data type are placed into the target data stream until no data remains in any of the original data specific data streams.

Count-based packing is a packet selection method where packets are selected from the data source until a specified number of packets are packed from the data source into the target interleaved data stream. Upon reaching or exceeding the specified packet count, packets are then selected from another data source.

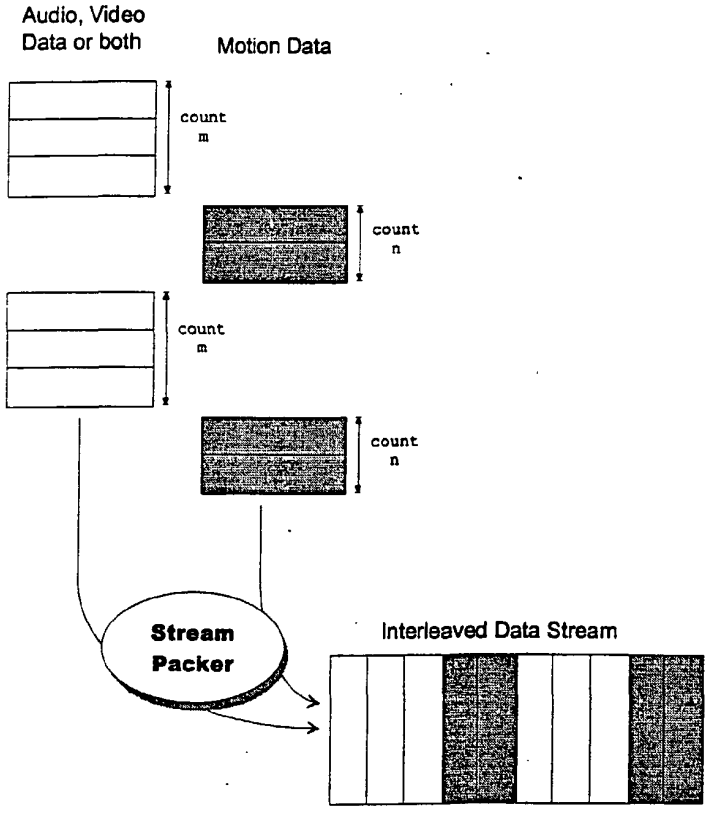


Figure 4 Count Based Packing.

Count based packing is used to ensure that a specific count of packets from each data source are grouped together in the target interleaved data stream.

Combination Packing

Combination packing is the use of a different packing algorithm for each data source. For example, when loading data from the motion data source size-based packing may be used, yet when loading data from the audio or video data source time-based packing may be used. The mix of different methods is optional and provided to help better synchronize the data.

Table of Contents

1 • Overview

This document describes the interleaving general model and algorithms used to mix motion data with other types of media such as audio or video. The intent of mixing such data is to allow motion instructions to play in sync with other media. For example, motion instructions may direct an object to move in sync with a certain musical song giving the appearance that the object is dancing to the music played.

The following chapters make up this document:

Chapter 1 – Overview; this chapter.

Chapter 2 – Interleaving Model; describes how interleaving works and the packing methods used to build the target data stream.

Chapter 3 – General Algorithms; describes different packing and unpacking algorithms.

2 • Interleaving Model

Interleaving is the process of merging two data sources, from two different data types into a single data stream that is then transferred to the target player. The target player, in turn plays each data set concurrently as the data is received.

Interleaving is a technology designed to synchronize two data types so that when they are played the end results (ie musical sounds and motion driven movements) are synchronized with one another. For example, interleaving allows music data and motion instructions to be mixed in a manner that when played on a robotic device, the device dances in sync with the music.

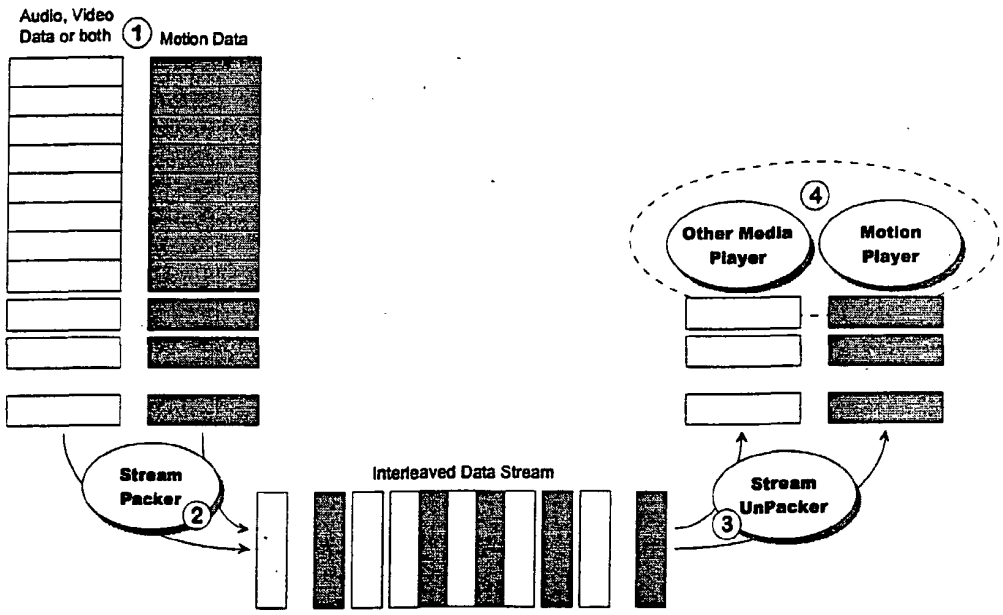


Figure 1 General Interleaving Model.

The following steps occur during the interleaving process.

Step	Description
1	The process starts with two data sources that are to be merged together.
2	Next, the Stream Packer takes each data source and packs each into the interleaved data stream using a packing algorithm to alternate the selection of data from each stream.
3	When used, the interleaved data stream is unpacked using the Stream Unpacker which is used to extract each data packet and pass each to the appropriate data player (or appropriate data player module in a single player unit). The data is passed to each player based on the data packet type. For example, the motion data is passed to the motion player, whereas the audio and/or video data is passed to the

	audio and video players respectively.
	The data is played concurrently and in sequence synchronizing the end results. For example, playing motion data at the same time as playing audio data causes the motions to be in sync with the audio sounds.

Packing Methods

The packing method, used by the Stream Packer, determines how the motion media is mixed with other media types such as audio or video data. There are four main methods used to mix the data during the packing process.

Time Based Packing

Time based packing is the process of selecting data from each data stream based on a pre-specified time quantum. A different time quantum may be used with each data stream.

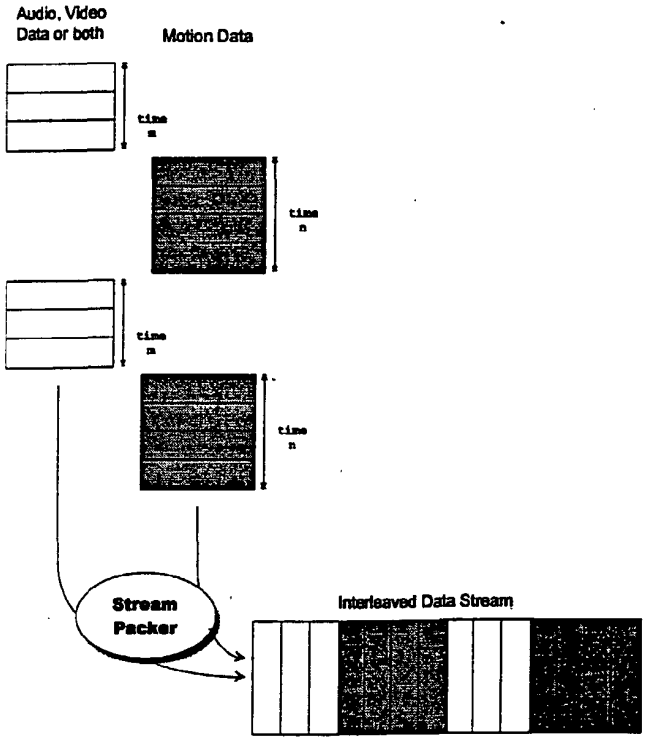


Figure 2 Time Based Packing

Upon selecting the data source to select packets from, the time quantum is reset to its pre-specified value associated with the data source. Packets are then pulled from the data source until the time quantum expires.

Packet Size Based Packing

Size-based packing is a packet selection method where packets are selected from the data source until a specified number of bytes is packed from the data source into the target interleaved data stream. Upon reaching or exceeding the specified size, packets are then selected from another data source.

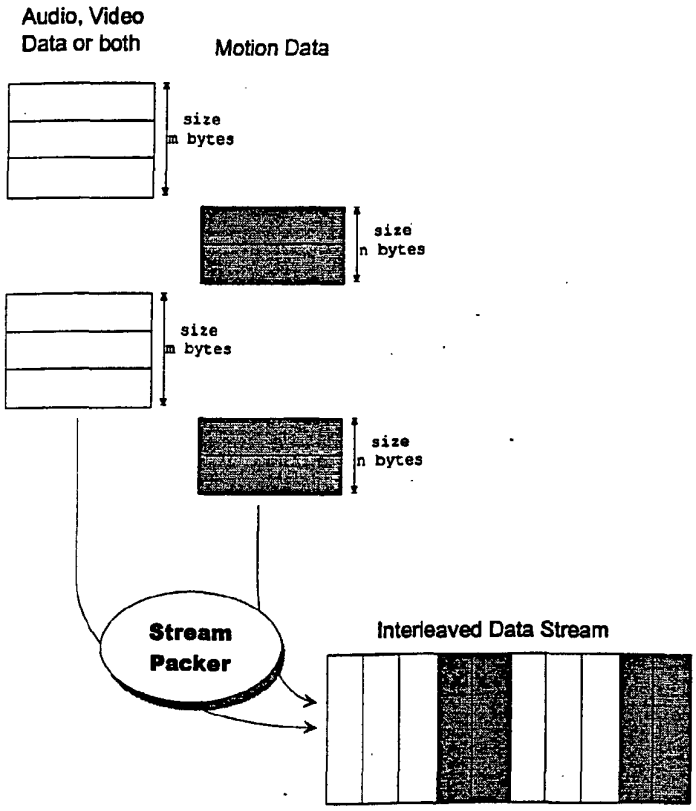


Figure 3 Size Based Packing.

Size based packing is used to ensure that each set of packets from each data source is packed in the target stream as a group of packets whose data size is at least a pre-specified size.

EXHIBIT 4

Motion Network Models

XMC Motion Network Models

Author: Dave Brown
Create Date: October 31, 1999 Save Date: XXX 0, 0000 Print Date: November 1, 1999
Project: $/prjcmpnt/xmc/v.100/int/persdev/doc/_devpers/xmcmeta/des/network_models
Document: Document2
Description:
Revisions:

ROY-G-BIV®
Software for a spectrum of ideas™

Table of Contents

1 • Overview

When using the scripting & framing and live update and streaming technologies the host machine – target device relationship may take many forms. This document details out several of these relationships and how they may be used by the end user.

The general relationship is that of a host machine connected to a target device by a data link of some sort. Where the host is responsible for creating and storing data, the target device is responsible for consuming the data sent to it by the host across the data link.

With this organization, there are three main models:

- **Broadcasting** – this model is defined as a host machine sending data out enabling several devices to pick up the data and execute it. The data is sent out much in the same way that a radio station broadcasts a radio signal to many radio devices.
- **Request Brokering** – this model inverts the broadcast model in that data is only sent to each device after the device makes a data request. The host machine acts as a data broker in that it only sends data to a device when requested.
- **Autonomous Distribution** – this model is a mix of both the broadcast and request-brokering model in that each device plays the role of both the host machine and the target device. In this model each device is capable of broadcasting data to all other devices as well as broker data requests from each. Each device also plays the role of the data consumer in that each device executes the data sets that it either requests or broadcasts that it is *tuned-in* to execute.

This document describes each of these models along with the basic model. The following chapters make up the document.

- **Chapter 1 – Overview**; this chapter.
- **Chapter 2 – Basic Model**; describes the general relationship between the host machine, data link and target device.
- **Chapter 3 – Network Models**; describes several different uses of the basic model.

2 • Basic Model

The basic model involves a host machine connected to a target device where the host is responsible for creating and storing the data sets, the data link is responsible for transferring the data to the target device, and the target device is responsible for consuming the data by executing it. The basic model may optionally have a user interface on the host and/or target side to enable the end user to configure each.

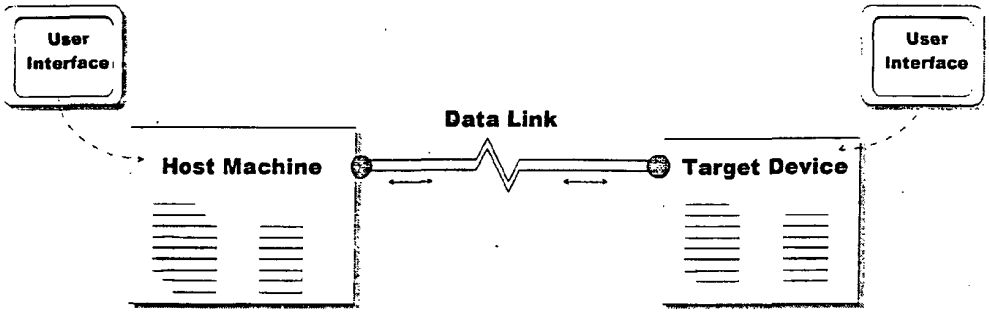


Figure 1 Basic Model.

It should be noted that even though the picture above shows both the host and target as separate physical machines, it is entirely possible that a single device provide both sets of functionality. This arrangement is used in one of the network models described later in this document.

Responsibility of each Item

Each of the items making up the basic model are described as follows:

- **Host Machine** – The host machine creates and stores each data set in either a live update or stream form. Data sets can be organized in script and frame format or just as a sequence of meta commands that describe intended operations for the target device to carry out. Even though the focus of this technology is on that of a sequence of *motion control* operations, the operations may also describe any operation used to control the target device such as the following:
    - Controlling and/or querying digital or analog IO lines on the device.
    - Controlling and/or querying a video camera, digital still camera or other vision-based sensor.
    - Controlling and/or querying any of a large array of sensors such as accelerometers, thermocouples, pressure transducers, etc.
    - Controlling and/or querying internal logic sequences (such as soft logic) or other algorithm running on the device.

- Controlling and/or querying motion control operations.

- **Data Link** – The data link is the medium across which the data is sent from the host machine to the target device. The data link may be in many different forms such as the following:
  - Ethernet based network link (or other physical wire based network such as TokenRing).
  - Wireless link (infrared, high bandwidth wireless or satellite).
  - Physical backplane (ISA or PCI bus). When host and target device are implemented as a single machine the back-plane acts as the data link between the logic implemented by each.
- **Target Device** – The target device is the consumer of the data. To consume the data the target device executes the set of logical machine instructions associated with each meta command making up the data set.

Data Channels

Each data link can be designated as a channel in its entirety, or be segmented into several channels. To segment a data link into a channel, each *packet* of data is marked with its channel number before sent across the data link. When received the target device can then discern whether or not the data item is actually on the data channel that it is currently listening to. If the entire data link is designated as a single channel, the marking of each data packet is not required.

Each packet of data is a set of one or more frames of meta commands along with additional packet attributes such as the data channel number or sequencing number for the packet.

3 • Network Models

There are several important uses of the basic model, namely: broadcasting, request brokering, and autonomous distribution. The following sections describe each of these in detail.

Broadcasting

Broadcasting is a variation of the basic model where the host machine sends data sets across one or many data links to one or many different devices simultaneously. The host machine may not even be aware that any devices are *listening* for data sets, but this is not required in the broadcast model. In this model, the host machine is mainly concerned with sending data sets out across the data link(s) without caring who is on the other end of those links.

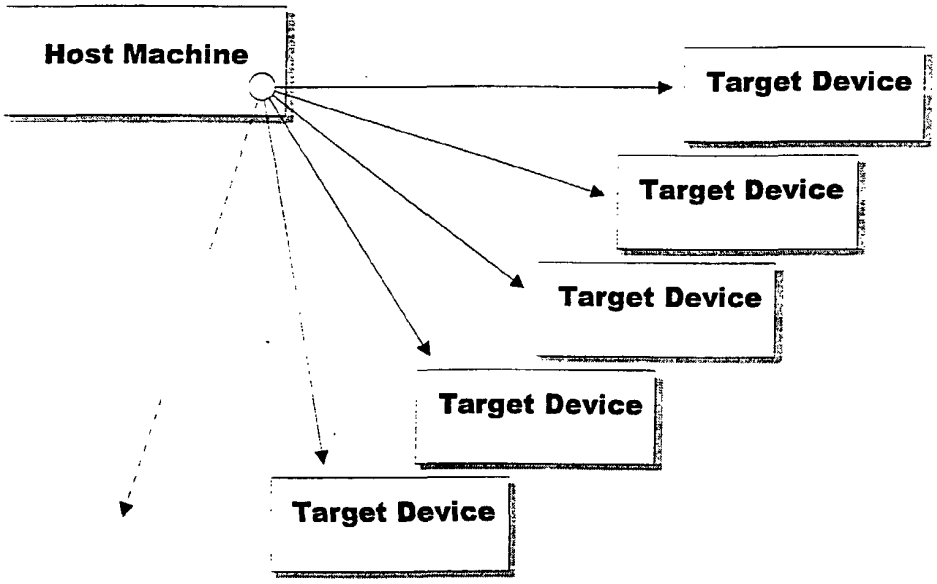


Figure 2 Broadcast Network Model

On the other side of the data link the target devices receive data that they are *tuned* to listen to. For example, each device may be tuned to listen to a single *channel* of data (i.e. a certain portion of a data link, or even just a specific data link chosen out of a set of data links). When listening to a data channel, the device then executes all data received by running all machine logic that it has associated with each meta command in the data set.

Request Brokering

In the request broker model, one or more target device request data updates from the host machine. Each request may occur simultaneously or at different times. The host machine waits idle until it receives a request for a data set or connection to a data channel. Once requested, the host machine associates the data set and/or data channel with the requesting device and begins transmitting data across the data link.

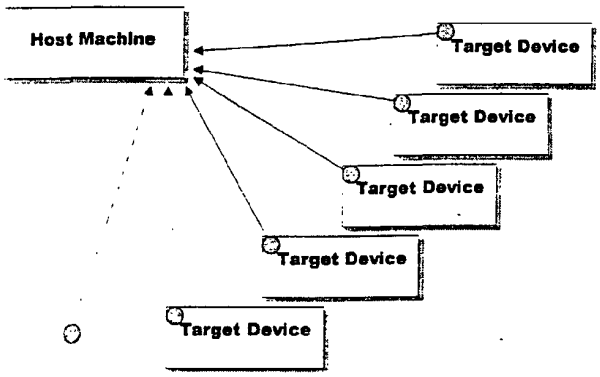


Figure 3 Request Brokering Network Model

Upon receiving the data requested, the target device runs each meta command within the data by executing the logical machine instructions that it has associated to each.

Autonomous Distribution

The autonomous distribution is a mix of the broadcast and request broker models in that each target device internally plays the role of both the host machine and target device. As host machine each device creates, stores and transmits data sets to other devices, and as a target device it either consumes its own data sets or other data sets received from other devices.

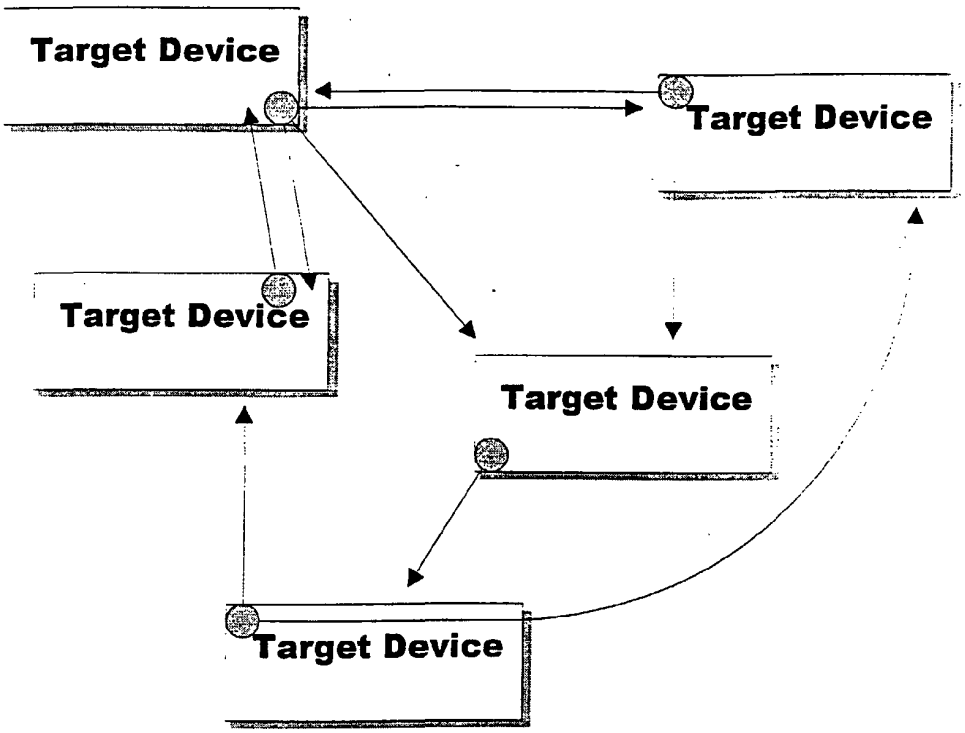


Figure 4 Autonomous Distribution

With this model target devices can work together as a community and be used to divide and conquer to break down a difficult task.

EXHIBIT 5

Data Updating

XMC Motion Live Update, Update Requesting and Streaming

Author: Dave Brown
Create Date: October 28, 1999 Save Date: XXX 0, 0000 Print Date: October 30, 1999
Project: $\prjcmpnt\xmc\v.100\int\persdev\doc_devpers\xmcmeta\dataupdate
Document: Document2
Description:
Revisions:

ROY-G-BIV®
Software for a spectrum of ideas™

Table of Contents

1 • Overview

Data updating is the process of transferring data sets from the machine used to create and store the data set to a target device used to consume the data. This document details out the process that takes place when making these data transfers, or otherwise know as *live updates.*

Both the host machine and target device play integral roles in any interactive system. In an interactive system, the host is defined as the machine (or device) that creates and stores data sets that are to be later run on the target device. The target device (which may actually be he same host machine or a separate independent machine or device) is the consumer of the data sets. Consuming, or playing, the data is the process of executing the instructions or meta codes making up the data set to cause an event (either external or internal).

For example, the host machine may be used to generate frames of motion meta commands that make up the scripts defining desired behaviors for a particular target device. When received, the target device plays each frame of motion meta commands by executing (or running) the action associated with each thus causing motion actions to occur. In this example, one machine could perform both host and device activities or they could also be on separate machines, connected by a data link (i.e. tethered line, network connection, wireless connection, the Internet, etc).

The following chapters make up this document:

- **Chapter 1 – Overview**; this chapter.
- **Chapter 2 – Data Update Models**; describes the different data update models used to perform data updates.
- **Chapter 3 – Initiating Data Updates**; describes the different methods of initiating a data update session.

2 • Data Update Models

There are three variations of the general *live update* model used to transfer data from the host to the target device and involve the following: scheduling, requesting and streaming 'live update' data. The following sections detail out the general live update model and each of these three variations of it. The data transferred can take many forms including: only motion data (where the data is a sequence of basic motion control operations), a mix of motion and music data (where the motion data may choreographed to the music data), or motion, music and video data combined together.

Live Update

Live updating is the process of transferring data from a host (a location used to create and store data) to a target (a machine that is used to execute the data). The location for both host and target operations may be on the same machine or on separate machines as described below.

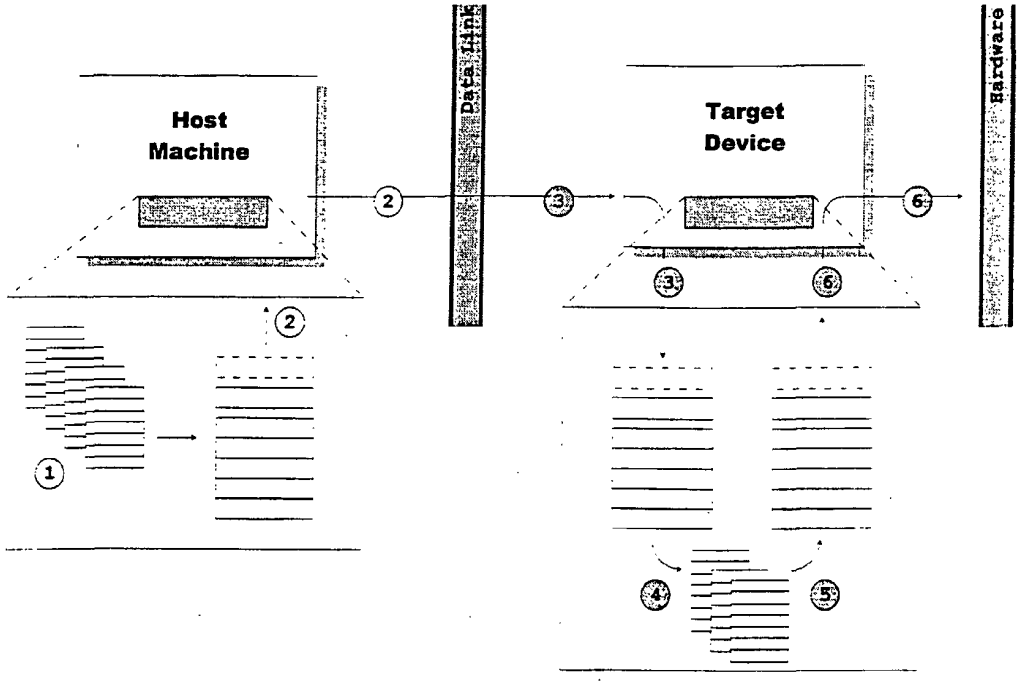


Figure 1 Live Update Process

The following steps take place during the live update process.

Step	Description
1	The first step in the live update process is that of the host machine

selecting from a library of scripts that are to be transferred to the target device.

2. Once selected, each frame of motion sequences (where a frame is defined as a set of motion meta commands, that describe basic motion operations, with the last motion meta command marked as the 'frame end' attribute). See 'Scripting and Framing' design document for details. It is possible, however not recommended to also send each script of motion data to the target device one meta command at a time. This is not recommended for doing so could place the target device in an unpredictable state if the data link connecting the host machine and target device is clipped.

3. The target device stores each frame received in a temporary location until all frames making up the script are received.

4. Once the script is received in its entirety, it is stored in a *ready* queue signifying that it is ready for execution.

5. When the script becomes the target script (i.e. the script selection logic of the device selects the script via a time quantum, programmed logic, or basic sequential order of each script) the device initiates executing the script.

6. To execute the script, the target device executes each **meta command** in the sequential order defined by the script. At this point the frames are not used. Execution of a meta command is the process of running the motion logic associated with each meta command.

Streaming

Streaming is the process of continually running through the live update process described in the previous section. Instead of sending scripts of data, the data set plays much like a continuous musical song, where frames of motion are continually sent to the target device, which in turn stores each frame received and runs each in the sequence received.

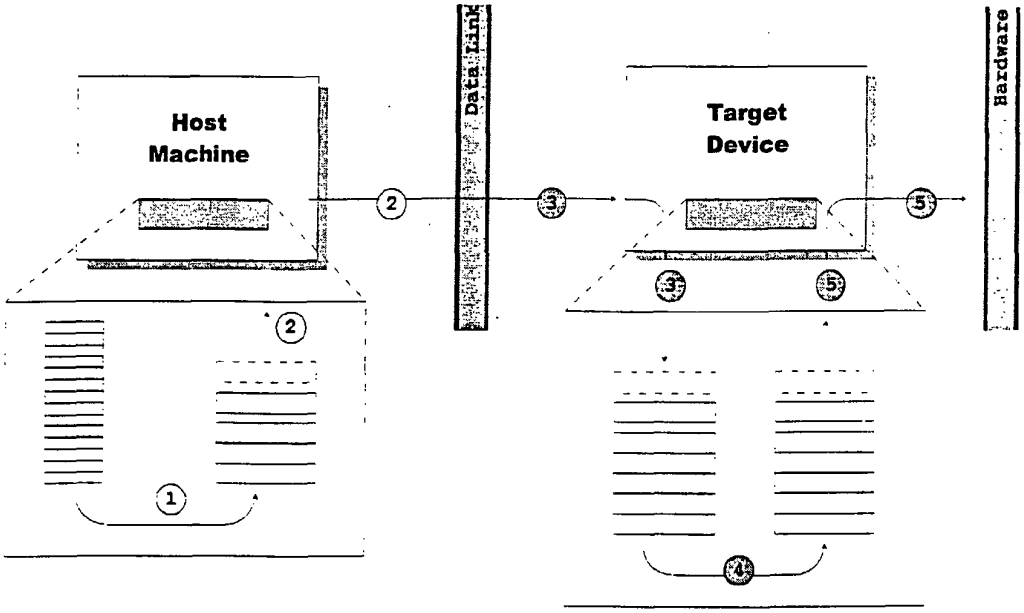


Figure 2 Streaming.

The following steps take place during the live update process.

Step	Description
1	When streaming, the host machine sends each frame of a large set of frames (or a set of frames that is continually being created or generated by the host machine) to the send queue.
2	Next, each frame (made up of one or more meta commands that can only be run by the target device as a set) is sent to the target device.
3	Upon receiving each frame, the target device adds them to a receive queue. Note, it is possible that the frames received may be received out of sequence. In such a case, the frames are inserted into the receive queue in a manner that maintains their original sequence sent by the host machine. More than often the sequencing is performed by the underlying network protocol, but this is not required for the host and target device may implement their own sequencing mechanism.
4	Unlike the like update method that waits until a full script of data is received before continuing, each frame is immediately passed to the target device output queue in preparation for execution.

5 Each **meta command** within the output queue is executed by running the motion logic associated with each command.

Pull Updates

Pull updates are a version of the *live update* model where the target, instead of the host, initiates the live update process. After encountering a pre-determined event (or set of events) the device requests for a live update from the host machine. Upon receiving such notification, the host machine then runs through the update process to transfer the requested data back to the target device.

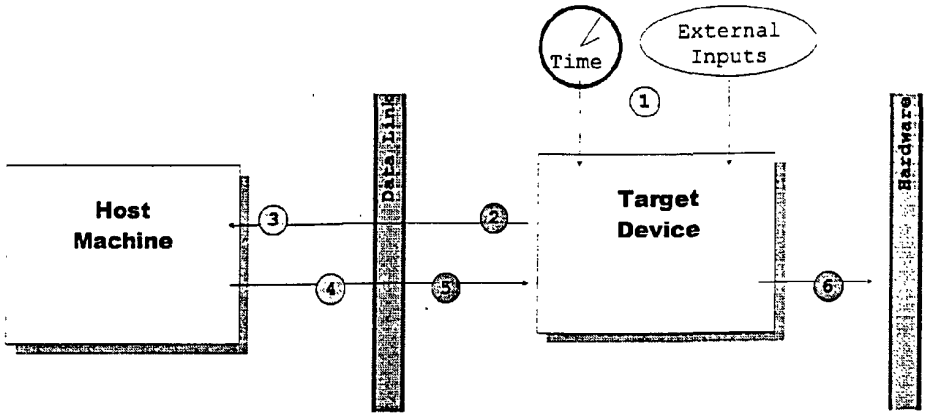


Figure 4 Pull Updating.

The following steps take place during the live update process.

Step	Description
1	Unlike Push updates, the target device initiates a Pull data update after either an external event occurs, a specified time increment passes or an internal logical algorithm dictates.
2	When the data update trigger event occurs, the target device fires a *data update request* to the host machine to initiate the update process.
3	Upon receiving the data update request the host machine begins preparing the data for either a live update or data streaming data transfer.
4	When ready the frames of motion meta commands are transferred to the target device across the data link connection.
5	When received, the device process the data much in the same way that it does in the Push data update.
6	The data received is run either as a live update or a stream by executing each meta command making up the frames of data. Executing the meta command consists of running the motion logic associated with each meta command, which manipulates, monitors and/or controls the target hardware.

What is claimed is:

1. A method of communicating motion data through a distributed network, the method comprising the steps of:

connecting a control software system, a content server, and a client browser to a network;

creating a motion program at the content server;

transferring the motion program from the content server to the control software system;

generating motion media at the control software system based on the motion program; and

transmitting the motion media to the client browser to operate a target device associated with the client browser based on the motion media.

2. A method as recited in claim 1, further comprising the steps of:

creating non-motion data at the content server;

combining the non-motion data with motion media at the control software system to obtain enhanced motion media; and

transmitting the enhanced motion media to the client browser.

3. A method as recited in claim 1, further comprising the steps of:

storing ratings rules associated with the client browser; and

modifying the motion media based on which the target device operates based on the ratings rules.

4. A method as recited in claim 1, in which the motion program is independent of the target device and the step of generating

the motion media further comprises the step of converting the motion program into the motion media of a type associated with the target device.

5. A method as recited in claim 1, in which the motion program is of a type associated with the target device.

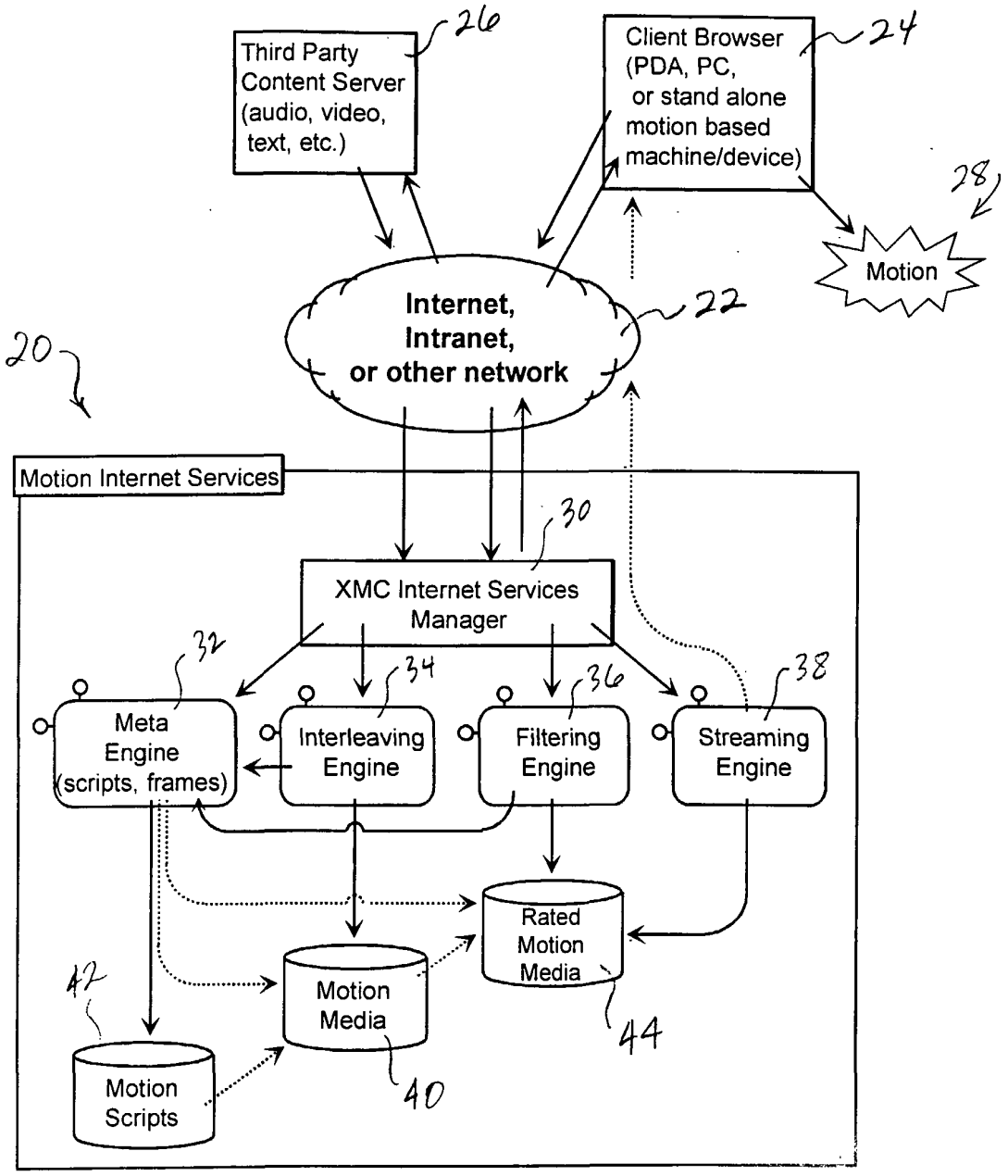

Third Party Content Server (audio, video, text, etc.)
26
Client Browser (PDA, PC, or stand alone motion based machine/device)
24
28
Motion
Internet, Intranet, or other network
22
20
Motion Internet Services
30
XMC Internet Services Manager
32
34
36
38
Meta Engine (scripts, frames)
Interleaving Engine
Filtering Engine
Streaming Engine
Rated Motion Media
44
42
Motion Media
40
Motion Scripts

FIG. 1

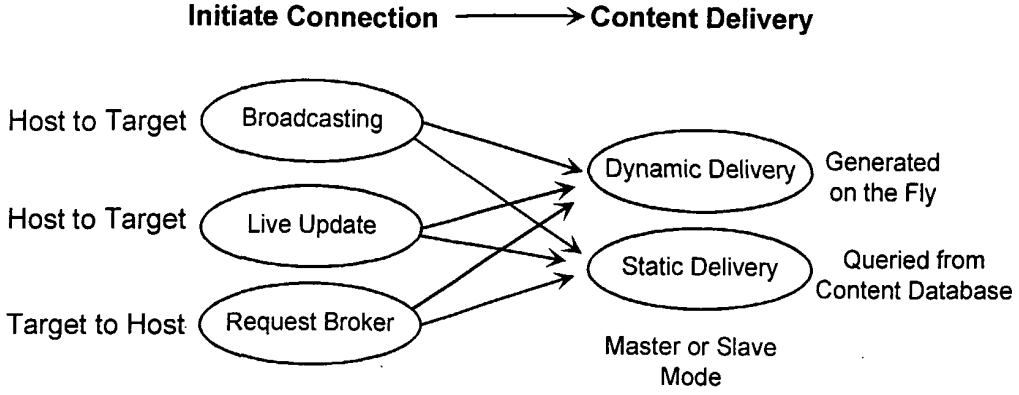

Initiate Connection
Content Delivery
Host to Target
Broadcasting
Host to Target
Live Update
Target to Host
Request Broker
Dynamic Delivery
Generated on the Fly
Static Delivery
Queried from Content Database
Master or Slave Mode

FIG. 2

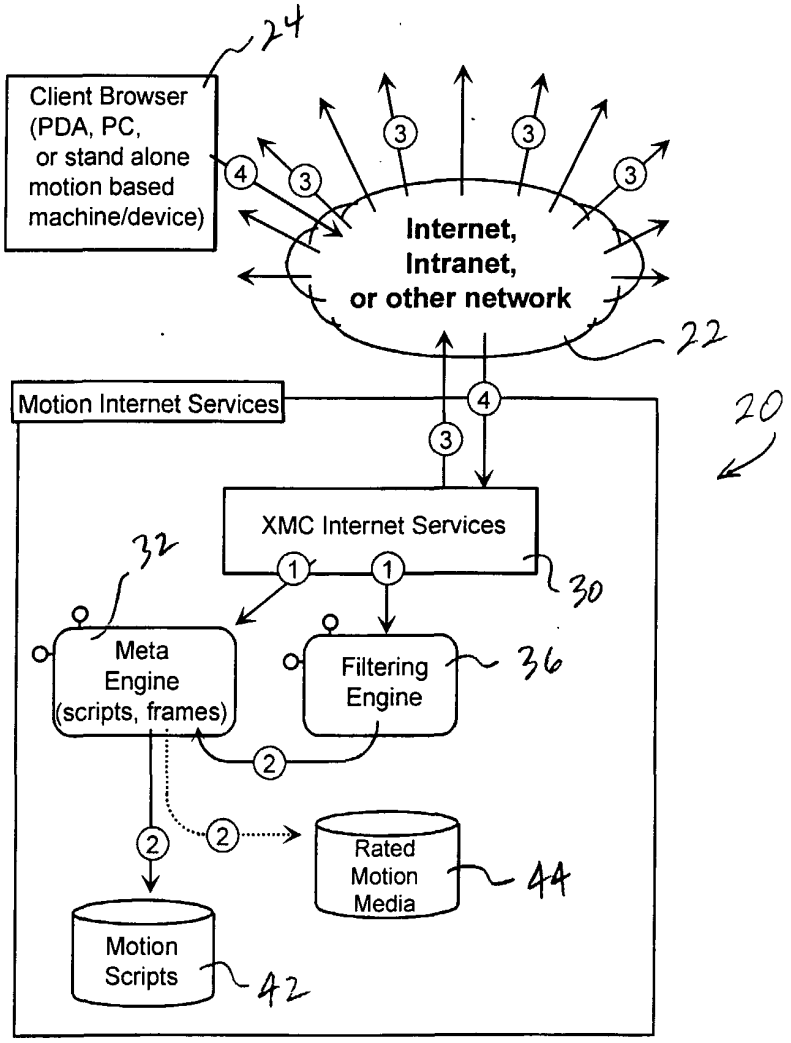

24
Client Browser (PDA, PC, or stand alone motion based machine/device)
Internet, Intranet, or other network
22
20
Motion Internet Services
XMC Internet Services
30
32
Meta Engine (scripts, frames)
Filtering Engine
36
Rated Motion Media
44
Motion Scripts
42

FIG. 3

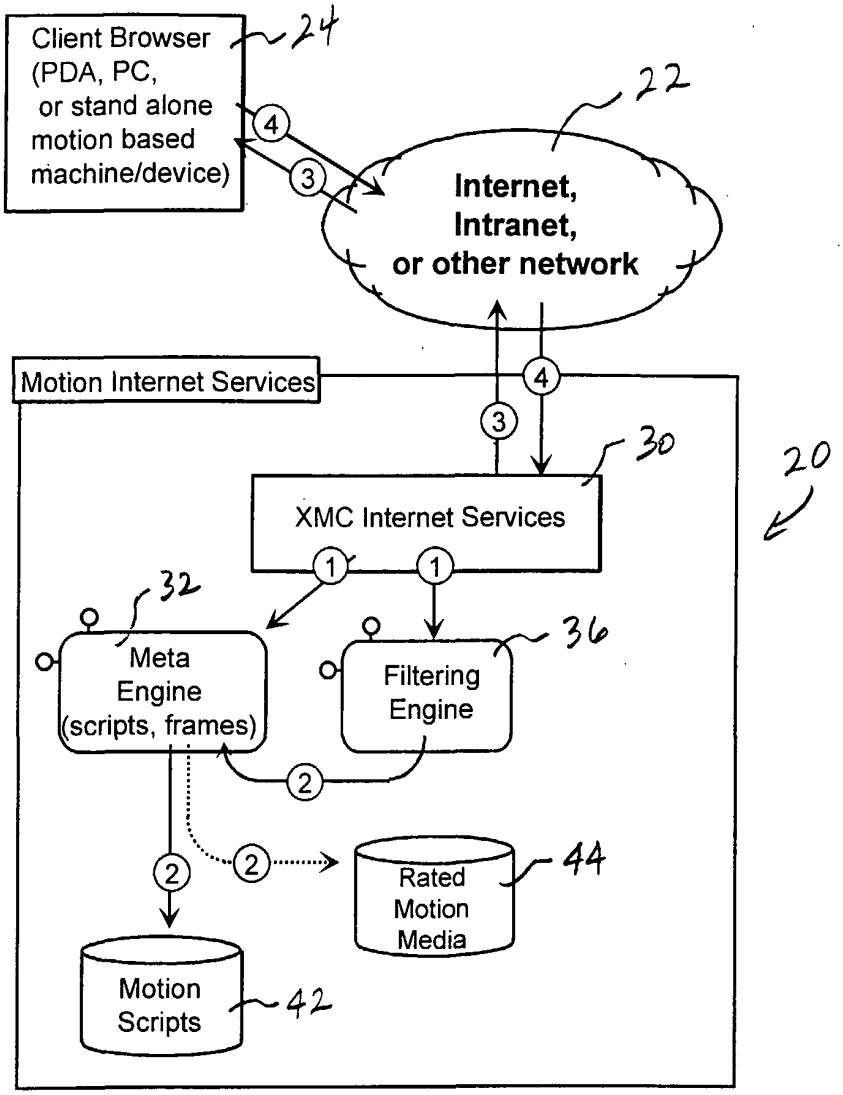

Client Browser
(PDA, PC,
or stand alone
motion based
machine/device)
24
22
Internet,
Intranet,
or other network
Motion Internet Services
20
30
XMC Internet Services
32
Meta
Engine
(scripts, frames)
36
Filtering
Engine
44
Rated
Motion
Media
Motion
Scripts
42

FIG. 4

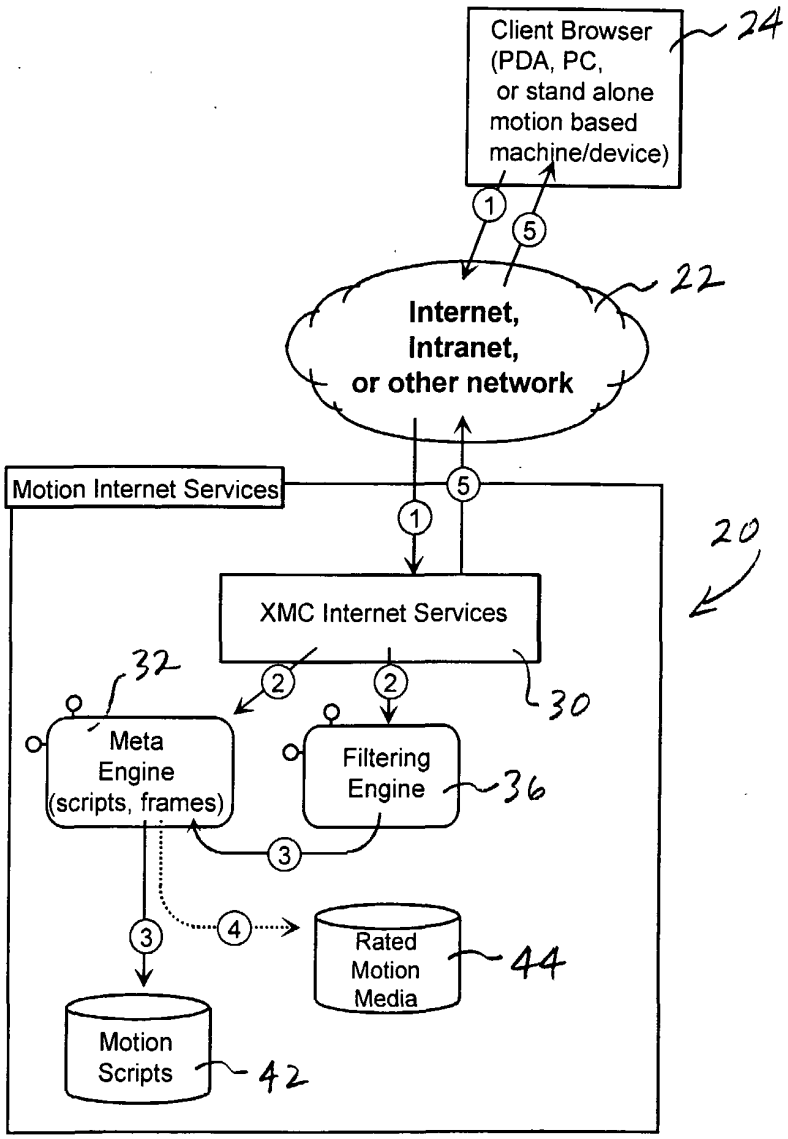

Client Browser
(PDA, PC,
or stand alone
motion based
machine/device)
24
1
5
22
Internet,
Intranet,
or other network
Motion Internet Services
5
1
20
XMC Internet Services
2
2
30
32
Meta
Engine
(scripts, frames)
Filtering
Engine
36
3
3
4
Rated
Motion
Media
44
Motion
Scripts
42

FIG. 5

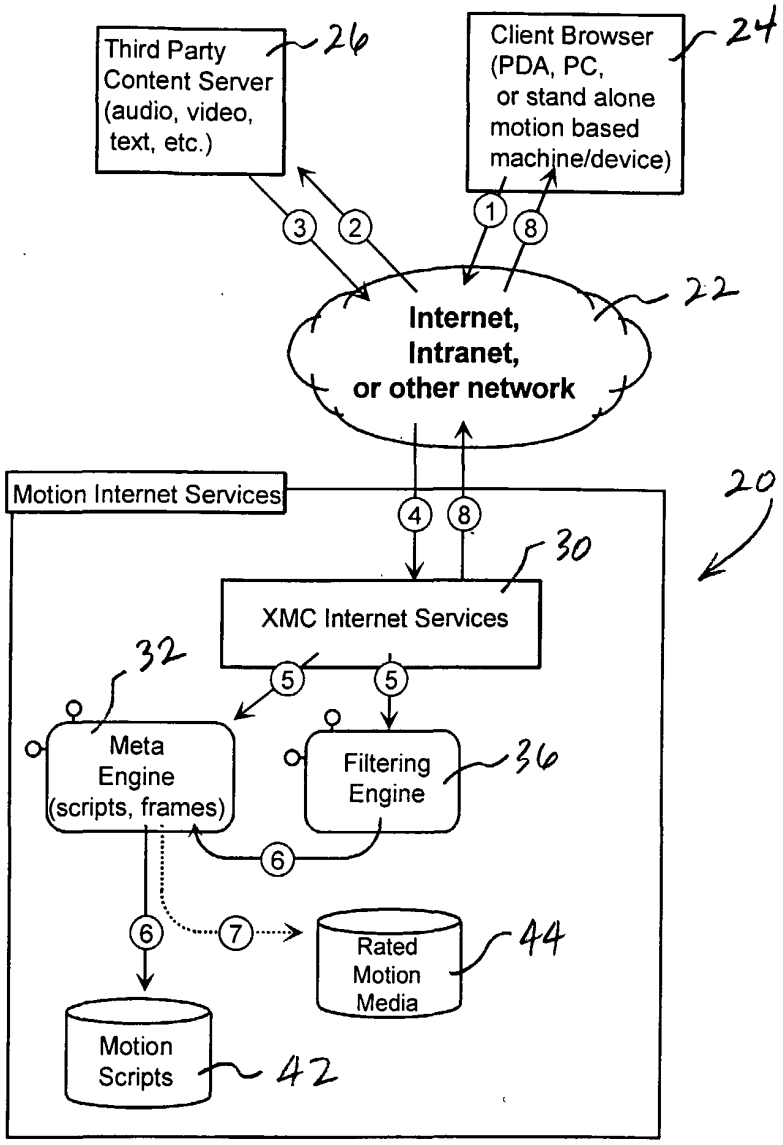

Third Party Content Server (audio, video, text, etc.)
26
Client Browser (PDA, PC, or stand alone motion based machine/device)
24
1
2
3
8
Internet, Intranet, or other network
22
Motion Internet Services
20
4
8
30
XMC Internet Services
5
5
32
Meta Engine (scripts, frames)
Filtering Engine
36
6
6
7
Rated Motion Media
44
Motion Scripts
42

FIG. 6

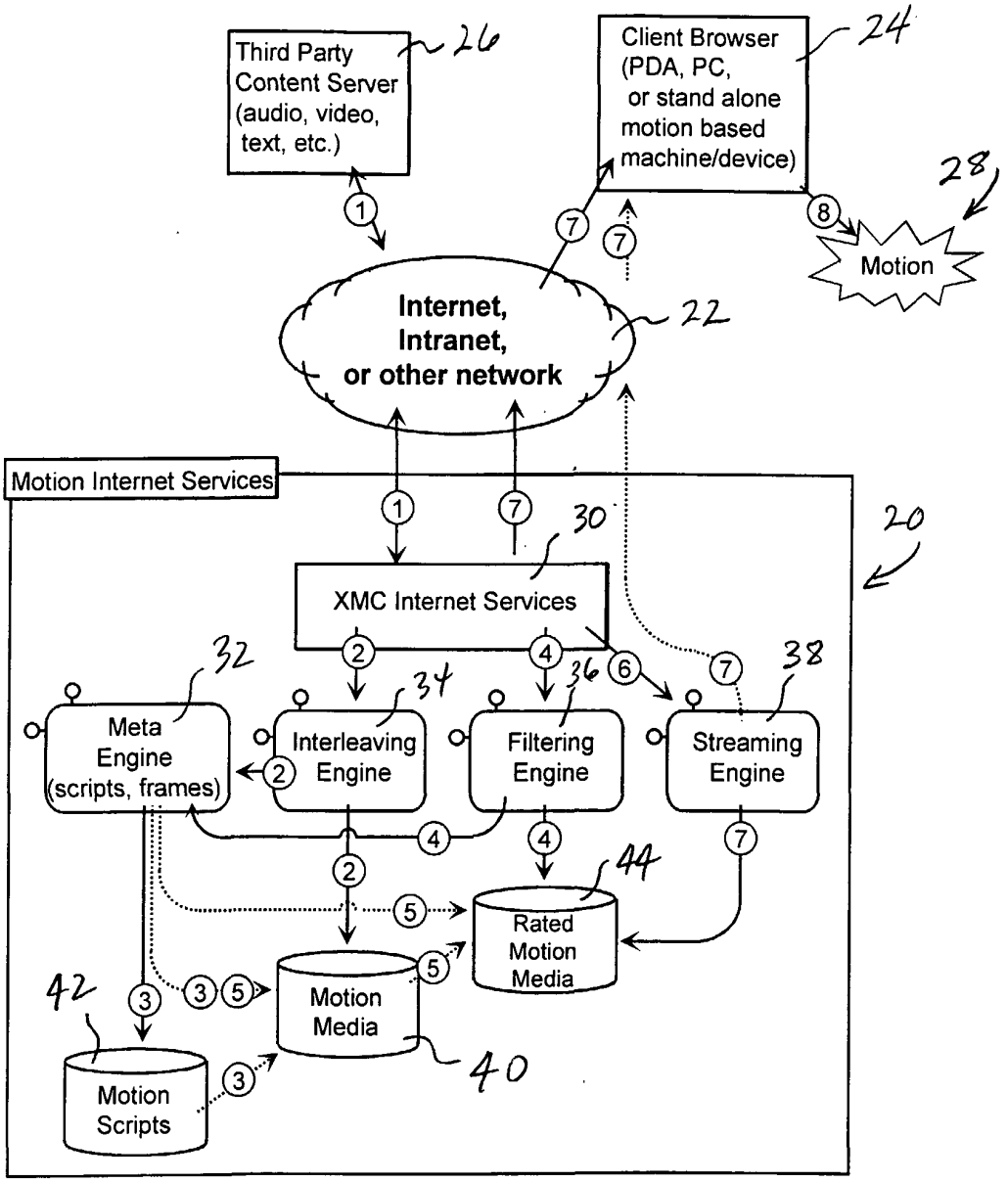

Third Party Content Server (audio, video, text, etc.)
26
Client Browser (PDA, PC, or stand alone motion based machine/device)
24
28
Motion
Internet, Intranet, or other network
22
Motion Internet Services
20
30
XMC Internet Services
32
Meta Engine (scripts, frames)
34
Interleaving Engine
36
Filtering Engine
38
Streaming Engine
44
Rated Motion Media
Motion Media
40
42
Motion Scripts

FIG. 7

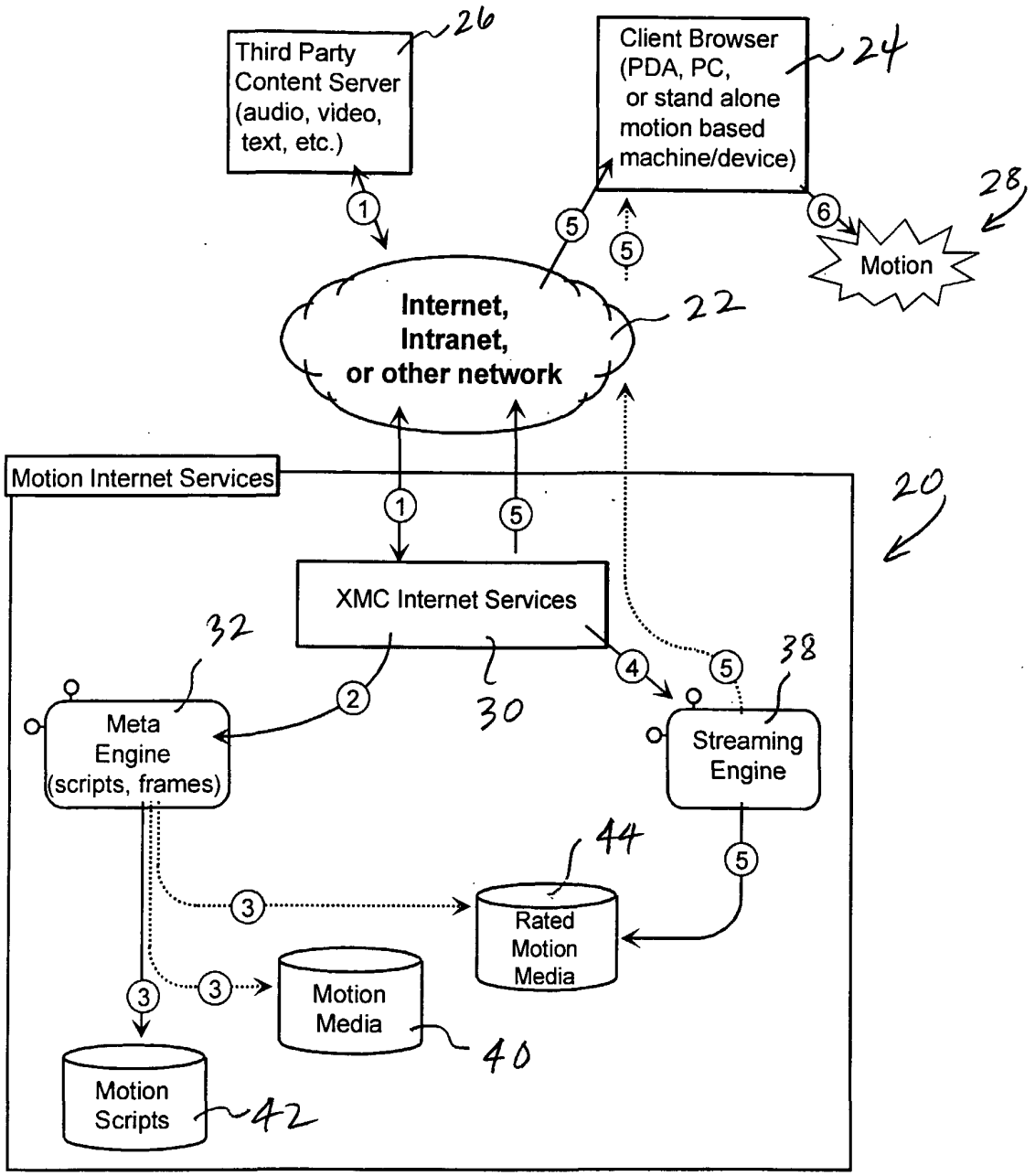

Third Party Content Server (audio, video, text, etc.)
26
Client Browser (PDA, PC, or stand alone motion based machine/device)
24
28
Motion
Internet, Intranet, or other network
22
Motion Internet Services
20
XMC Internet Services
30
32
Meta Engine (scripts, frames)
38
Streaming Engine
44
Rated Motion Media
Motion Media
40
Motion Scripts
42
1
2
3
4
5
6

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/29550

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G05B 19/18
US CL :700/65

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 700/65, 48, 49

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

West

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,600,373 A (CHUI et al) 04 February 1997, see, figures 4a to 15.	1-5
Y	US 5,666,161 A (KOHIYAMA et al.) 09 September 1997, see, figures 1A to 11 and abstract.	1-5
Y	US 5,790,178 A (SHIBATA et al.) 04 August 1998, see, abstract and col. 2 .	1-5
Y	US 5,818,537 A (ENOKIDA et al) 06 October 1998, see, figures 1-2 and col. 2 to col. 4.	1-5
Y	US 5,821,987 A (LARSON) 13 October 1998, see, col. 3 to col. 6.	1-5
Y	US 5,924,013 A (GUIDO et al) 13 July 1999, see, abstract and figures 1-3.	1-5

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

18 DECEMBER 2000

Date of mailing of the international search report

31 JAN 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

RAMESH PATEL

Telephone No. (703) 308-6673

Form PCT/ISA/210 (second sheet) (July 1998)★