



US 20140289380A1

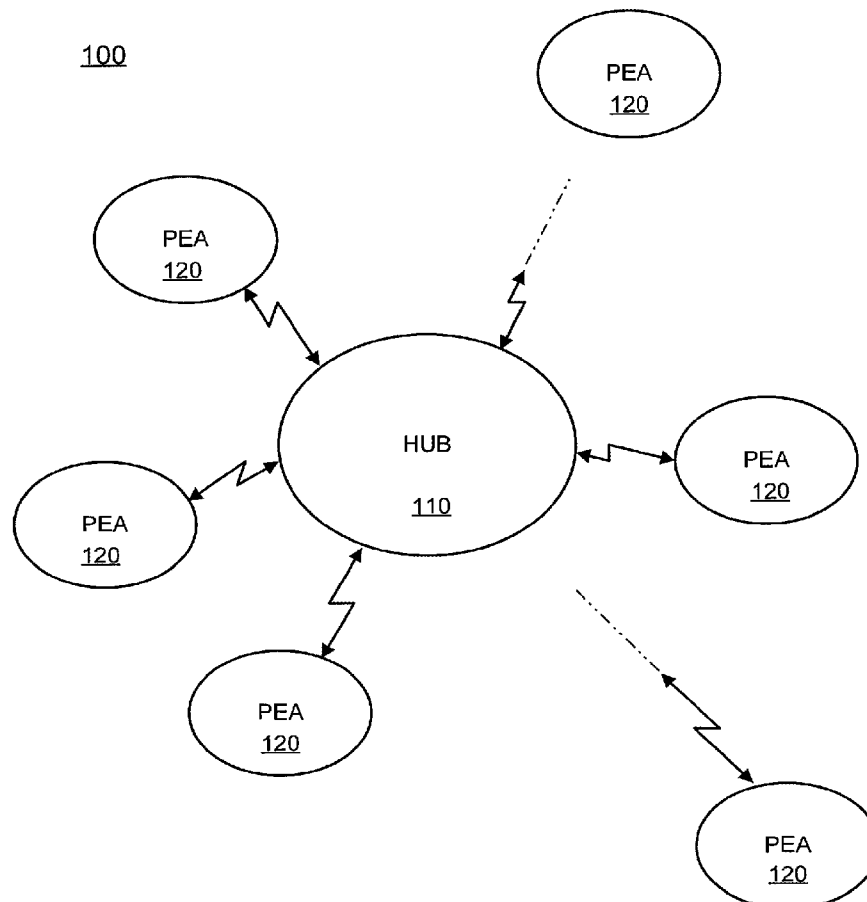
(19) **United States**(12) **Patent Application Publication**  
**Donaghey**(10) **Pub. No.: US 2014/0289380 A1**(43) **Pub. Date: Sep. 25, 2014**(54) **PERSONAL AREA NETWORK APPARATUS**(71) Applicant: **Tri-County Excelsior Foundation,**  
Marshall, TX (US)(72) Inventor: **Robert J. Donaghey,** Lexington, MA  
(US)(73) Assignee: **Tri-County Excelsior Foundation,**  
Marshall, TX (US)(21) Appl. No.: **14/216,427**(22) Filed: **Mar. 17, 2014****Related U.S. Application Data**

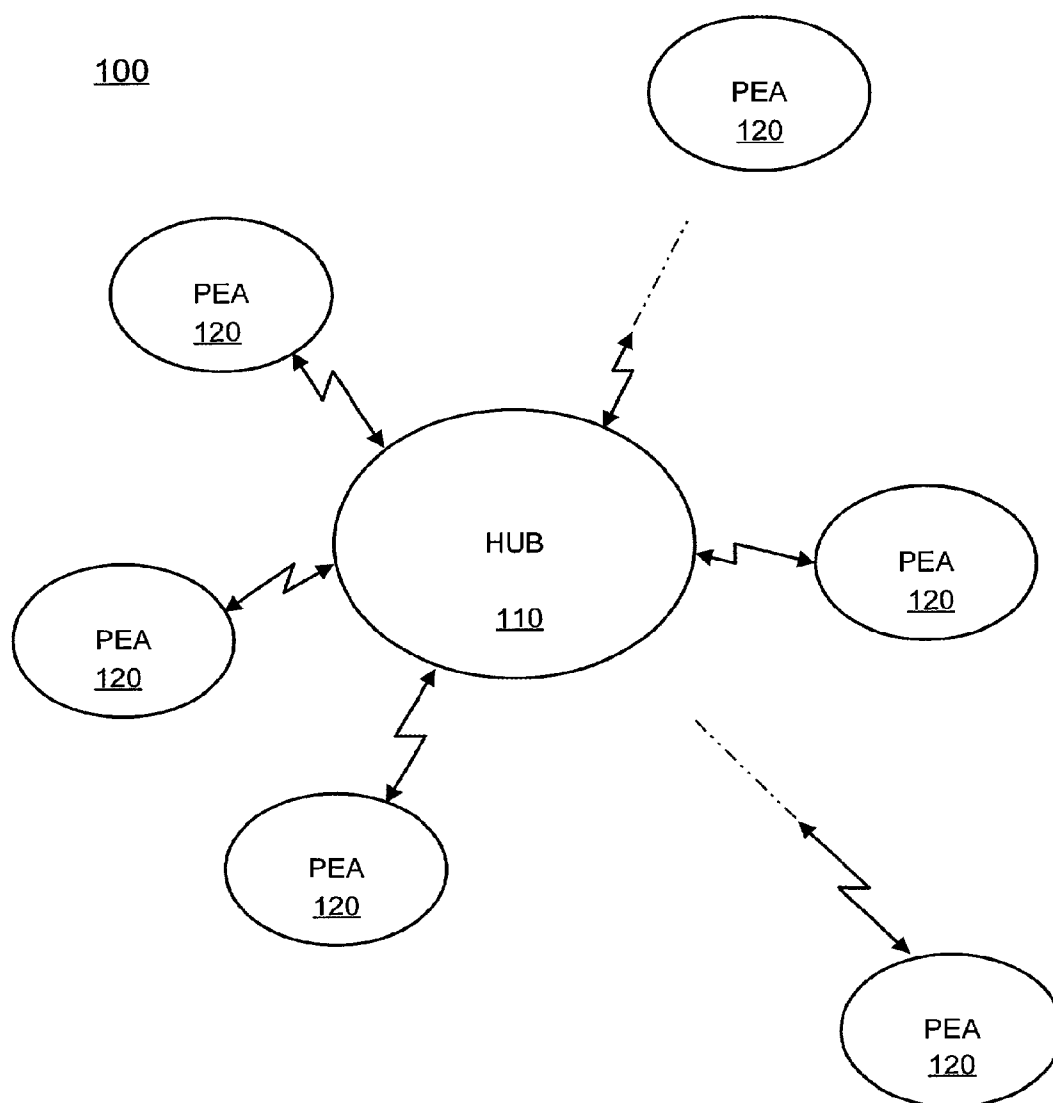
(63) Continuation of application No. 13/430,650, filed on Mar. 26, 2012, now Pat. No. 8,675,590, which is a continuation of application No. 12/699,846, filed on Feb. 3, 2010, now Pat. No. 8,149,829, which is a continuation of application No. 11/728,246, filed on Mar. 23, 2007, now Pat. No. 7,756,129, which is a continuation of application No. 10/894,406, filed on Jul. 19,

2004, now Pat. No. 7,218,633, which is a continuation of application No. 09/535,591, filed on Mar. 27, 2000, now Pat. No. 6,804,232.

**Publication Classification**(51) **Int. Cl.**  
**H04W 60/00** (2006.01)  
**H04W 76/02** (2006.01)  
**H04L 29/12** (2006.01)  
(52) **U.S. Cl.**  
CPC ..... **H04W 60/00** (2013.01); **H04L 61/2069**  
(2013.01); **H04W 76/02** (2013.01)  
USPC ..... **709/221**(57) **ABSTRACT**

A network (100) includes a hub device (110) and at least one unattached peripheral device (120). The unattached peripheral device (120) transmits an attach request to the hub device (110) with a selected address, receives a new address from the hub device to identify the unattached peripheral device (120), and communicates with the hub device (110) using the new address.



**FIG. 1**

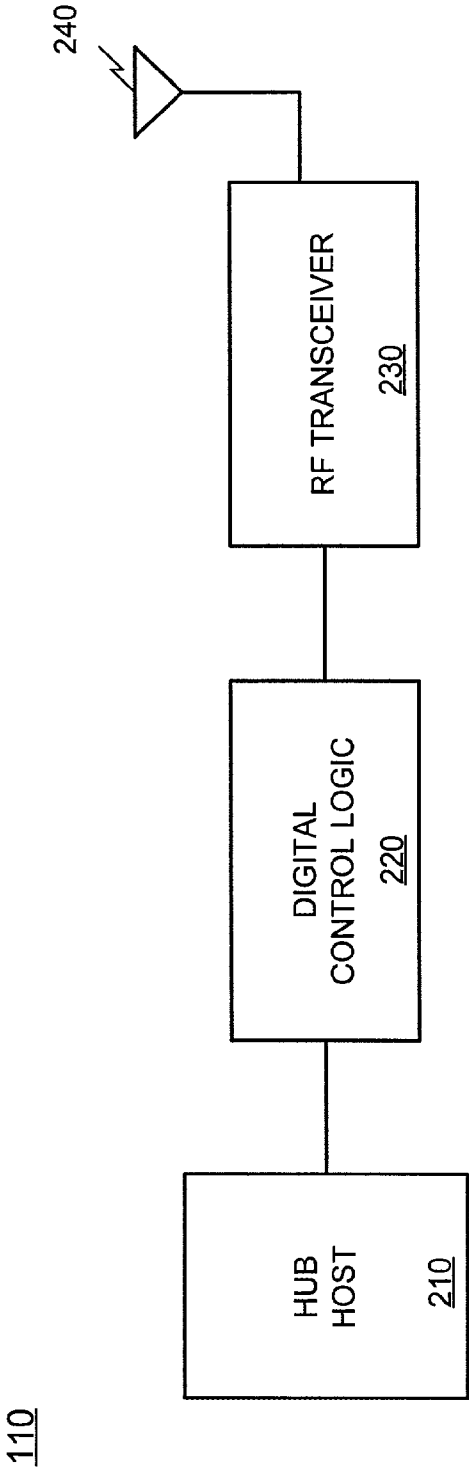


FIG. 2

120

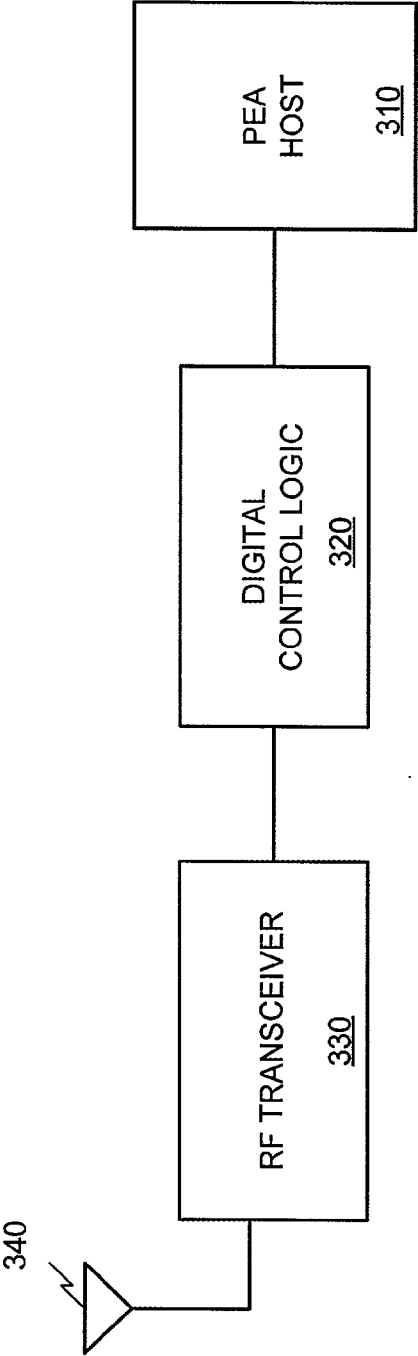


FIG. 3

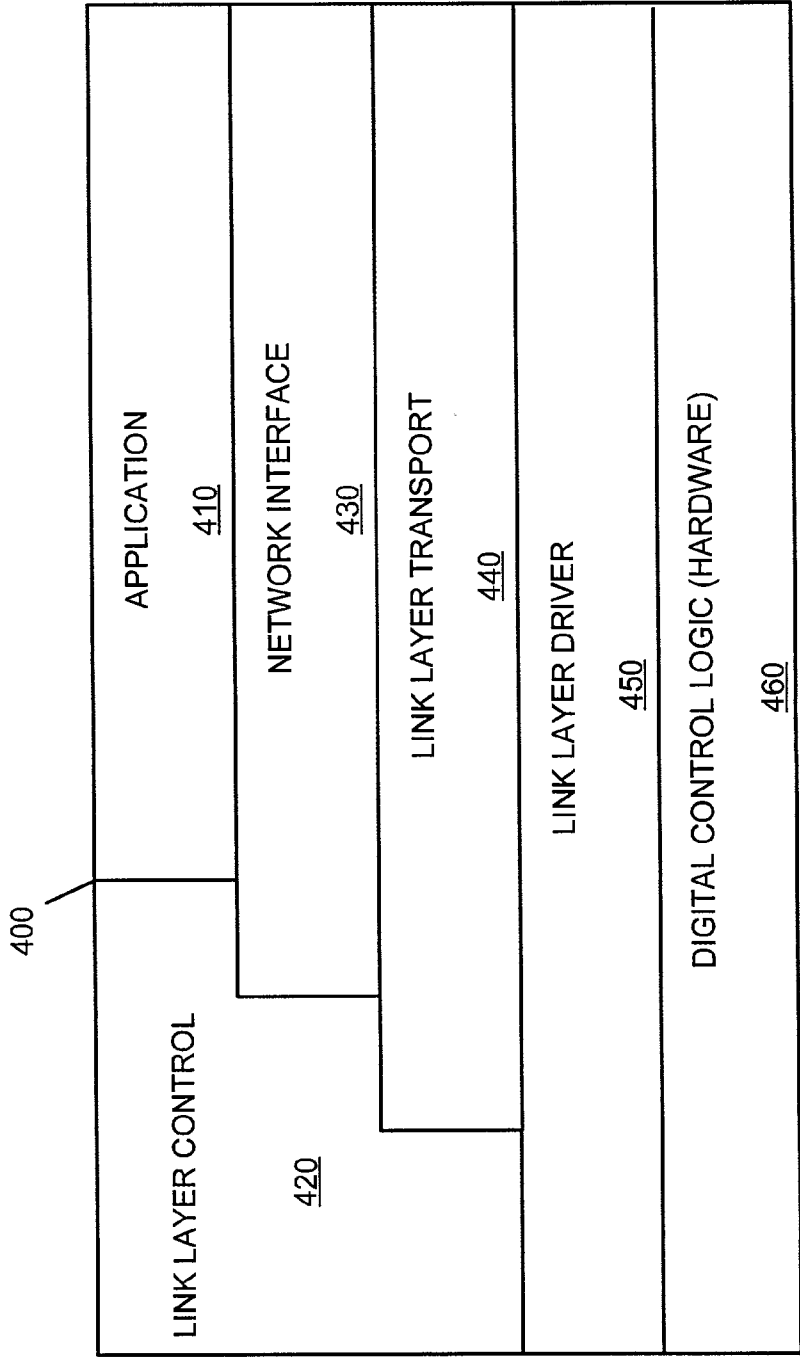
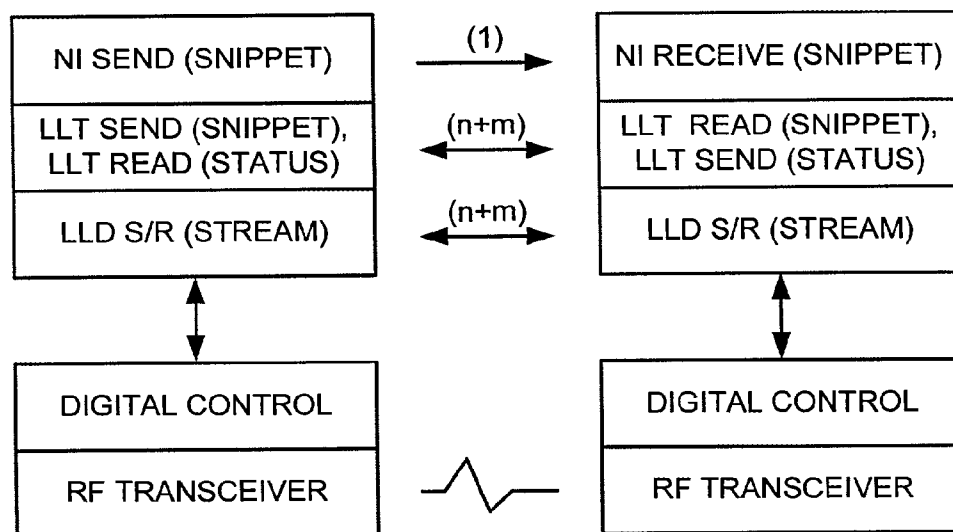


FIG. 4

**FIG. 5**

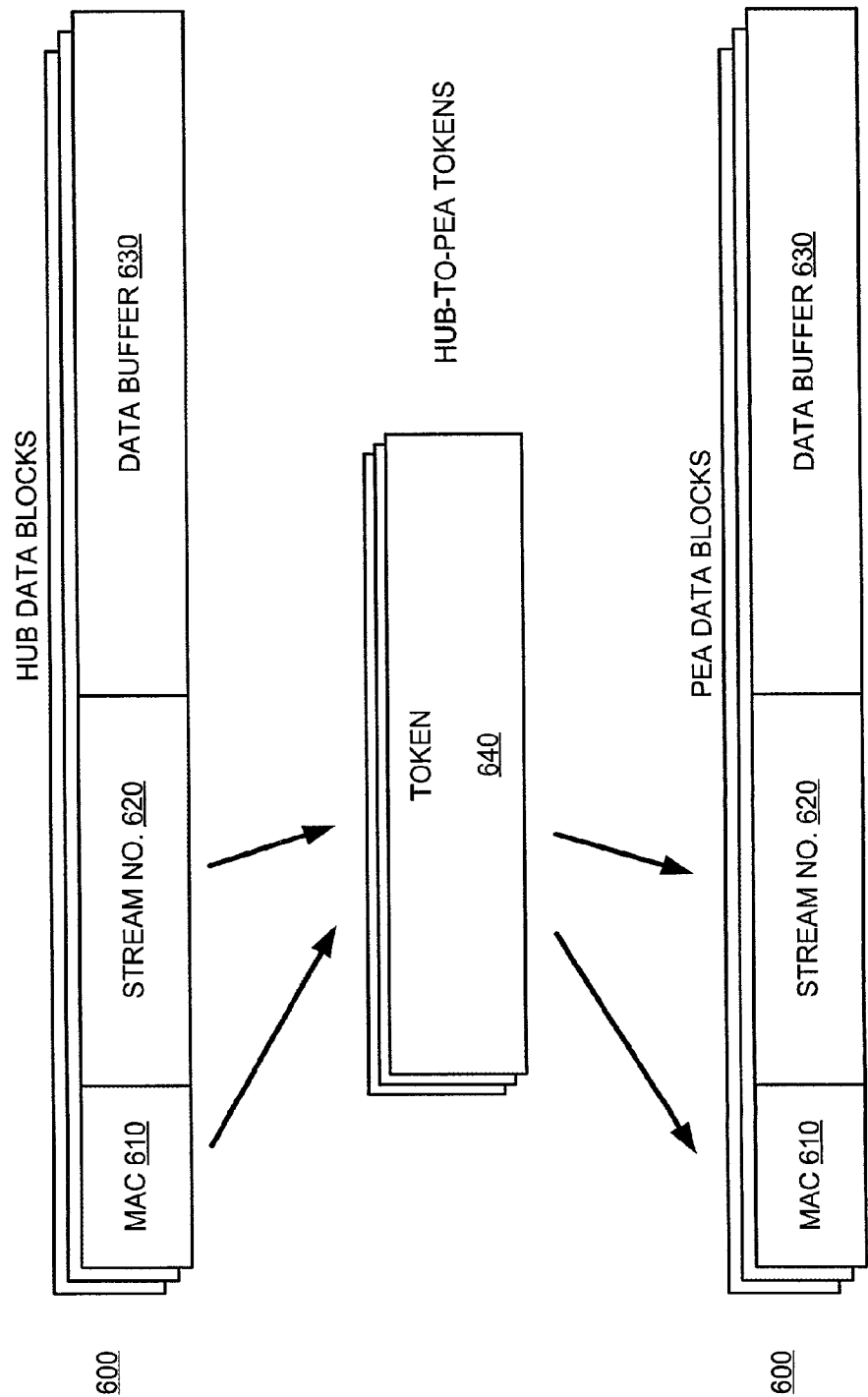
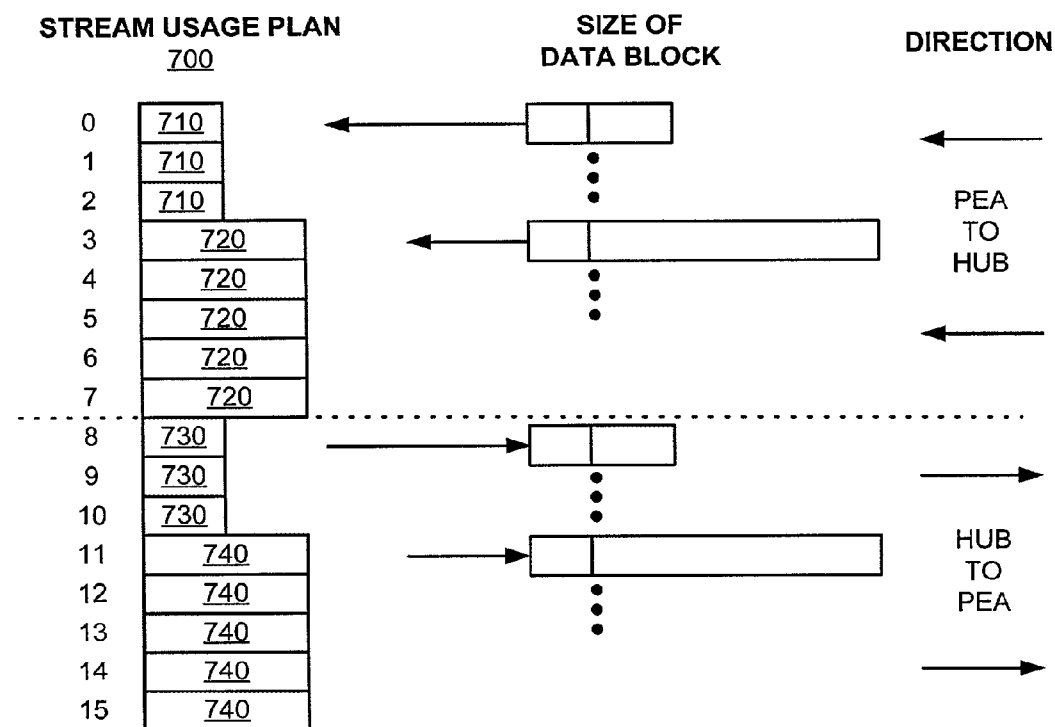
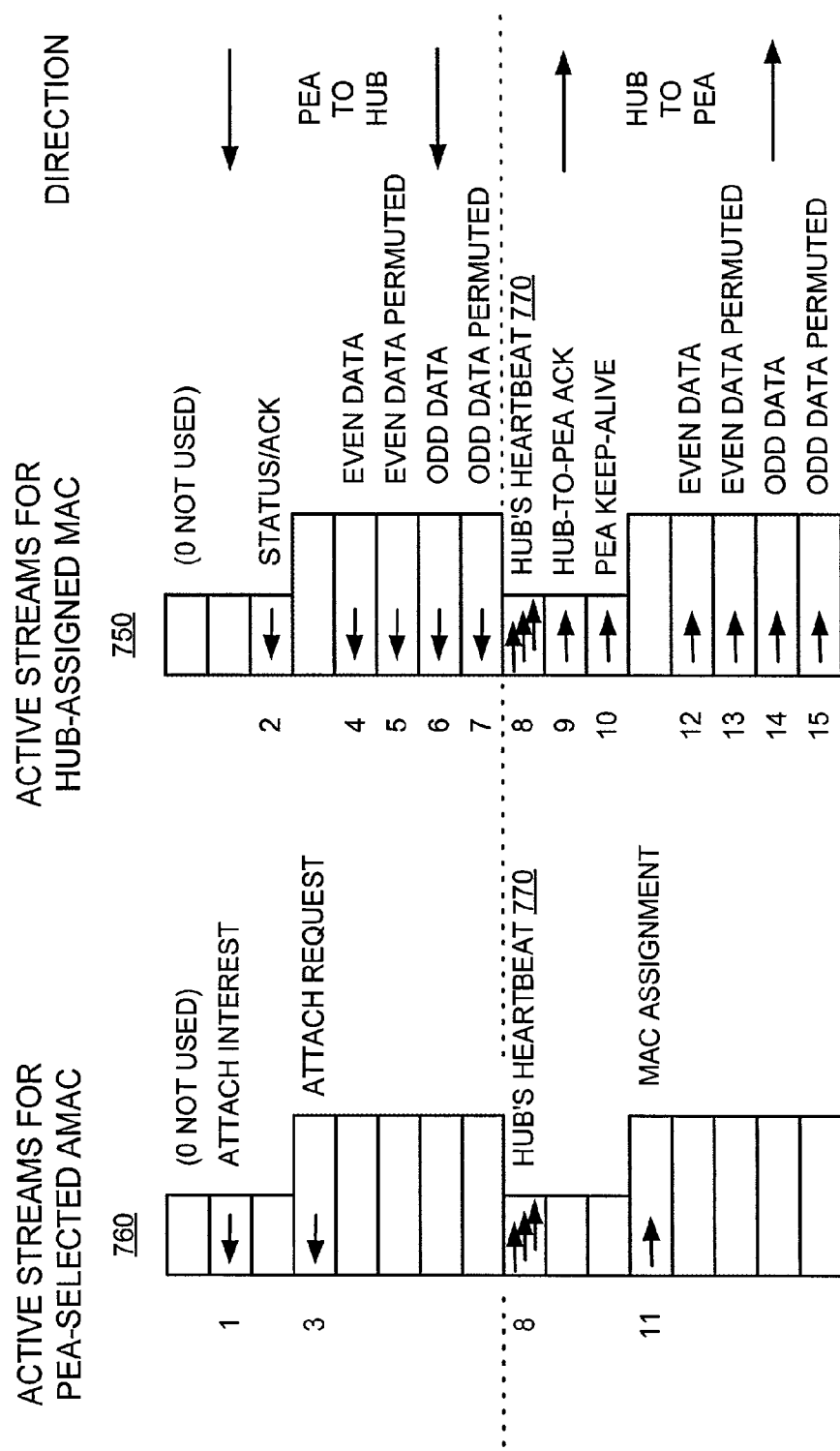


FIG. 6

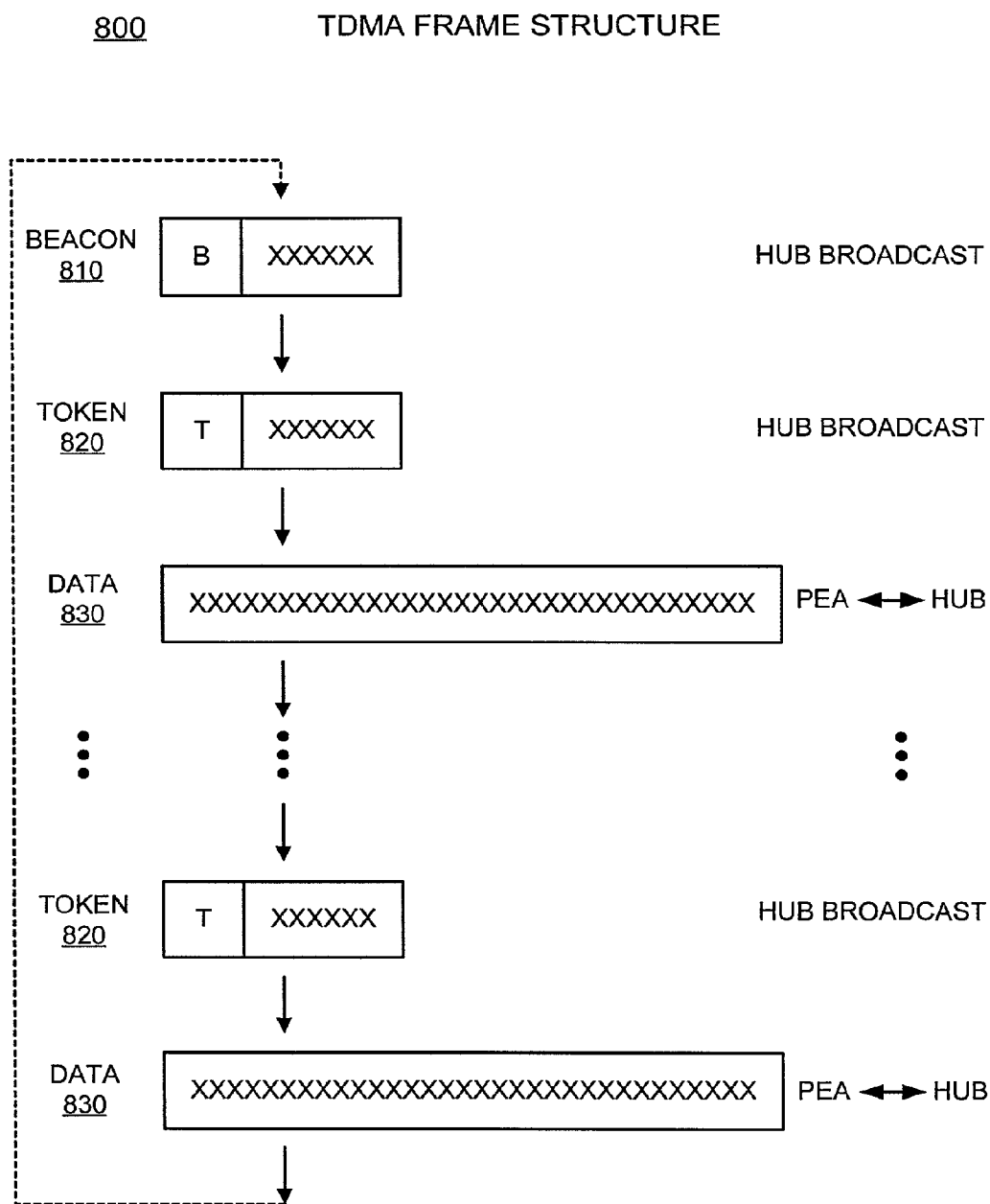


**FIG. 7A**





**FIG. 7B**



**FIG. 8**

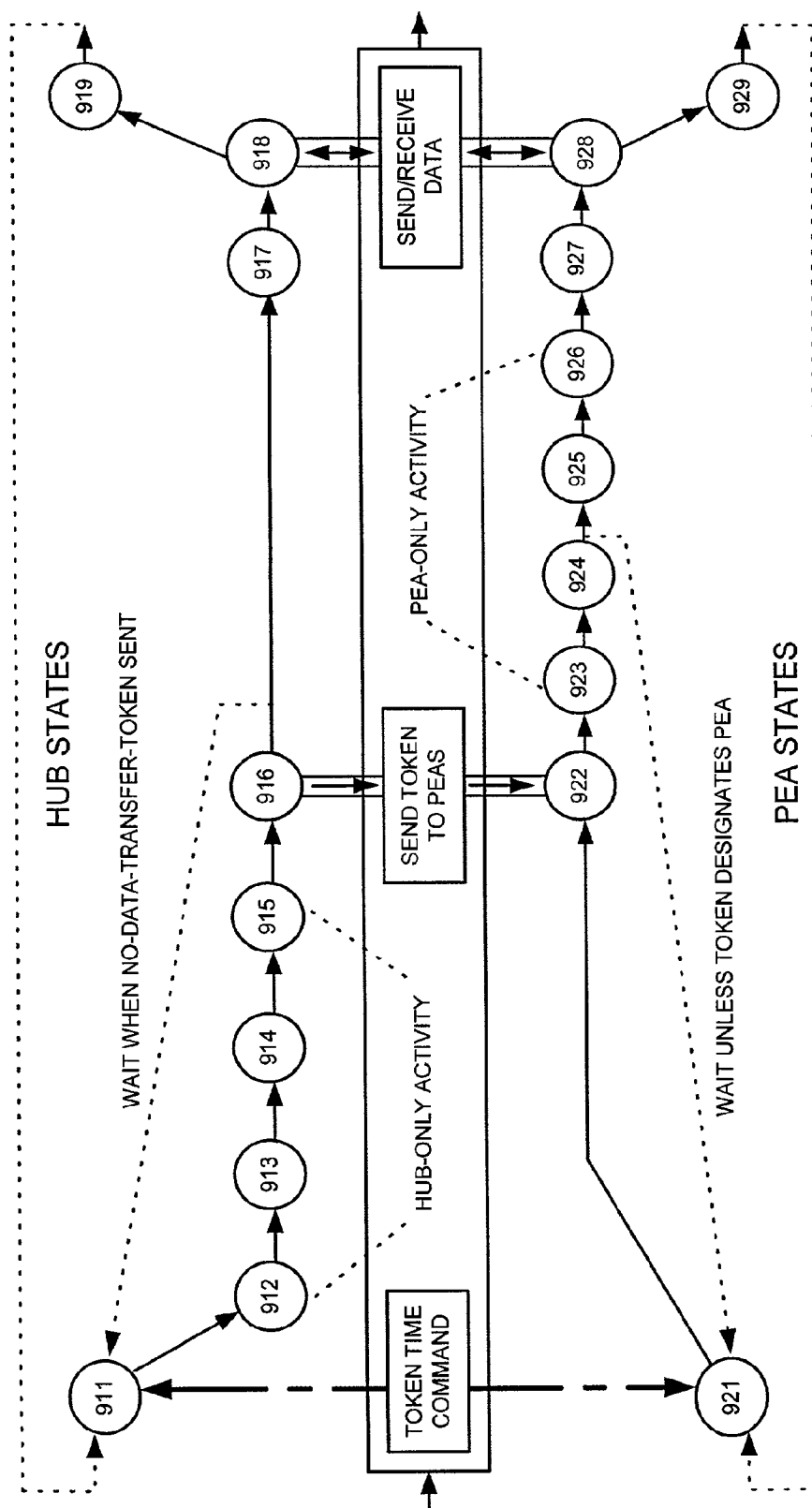
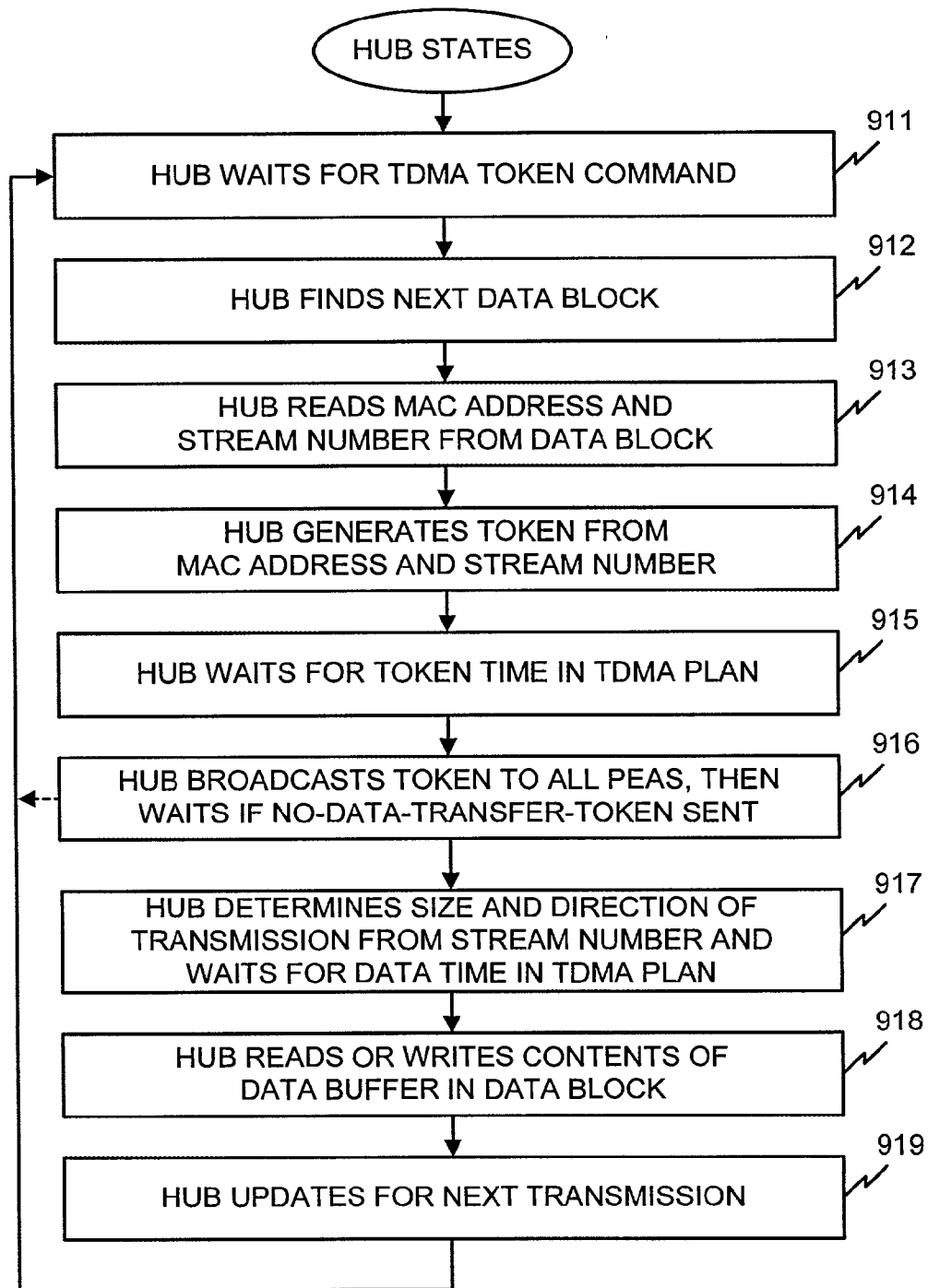
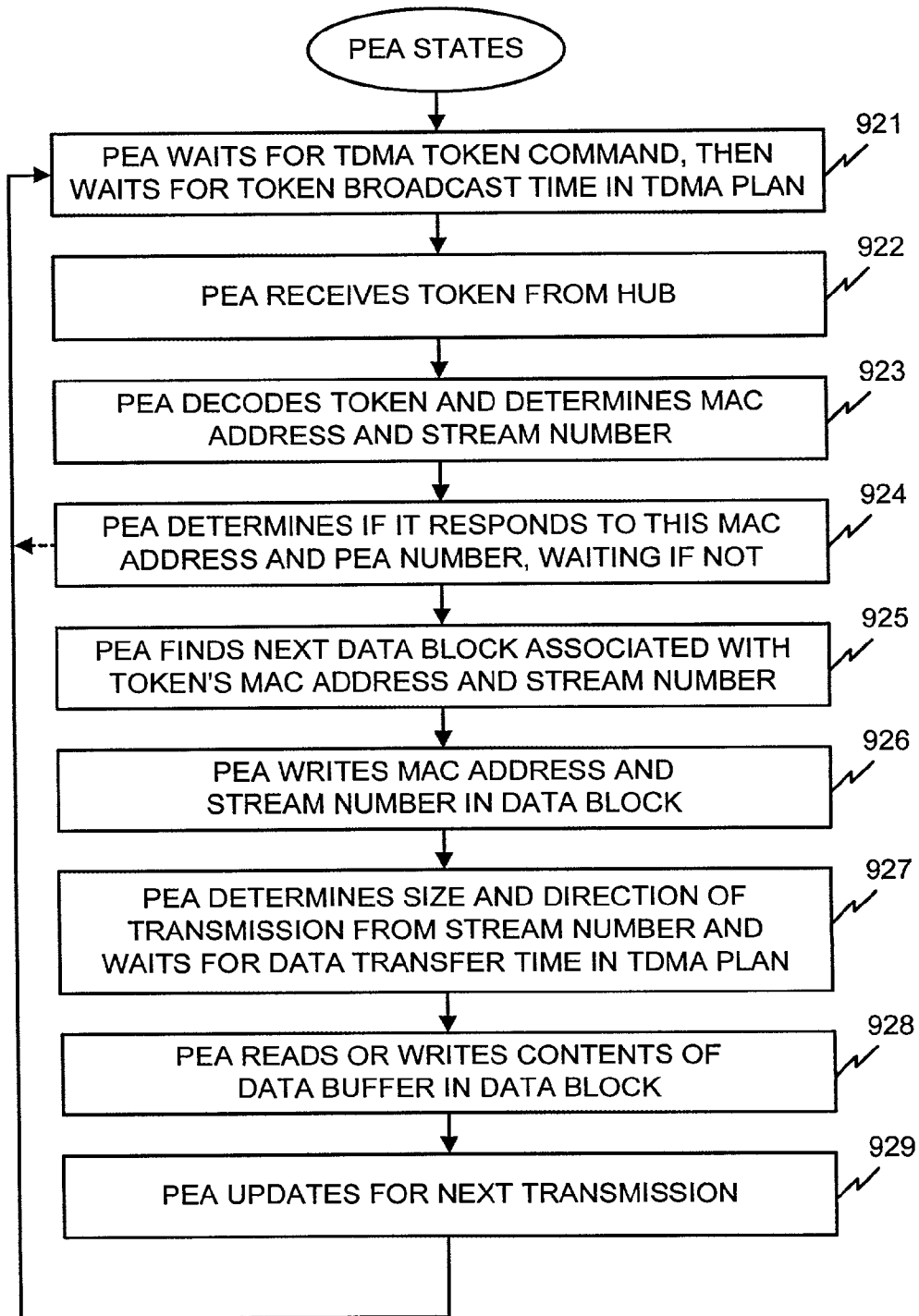


FIG. 9A



**FIG. 9B**

**FIG. 9C**

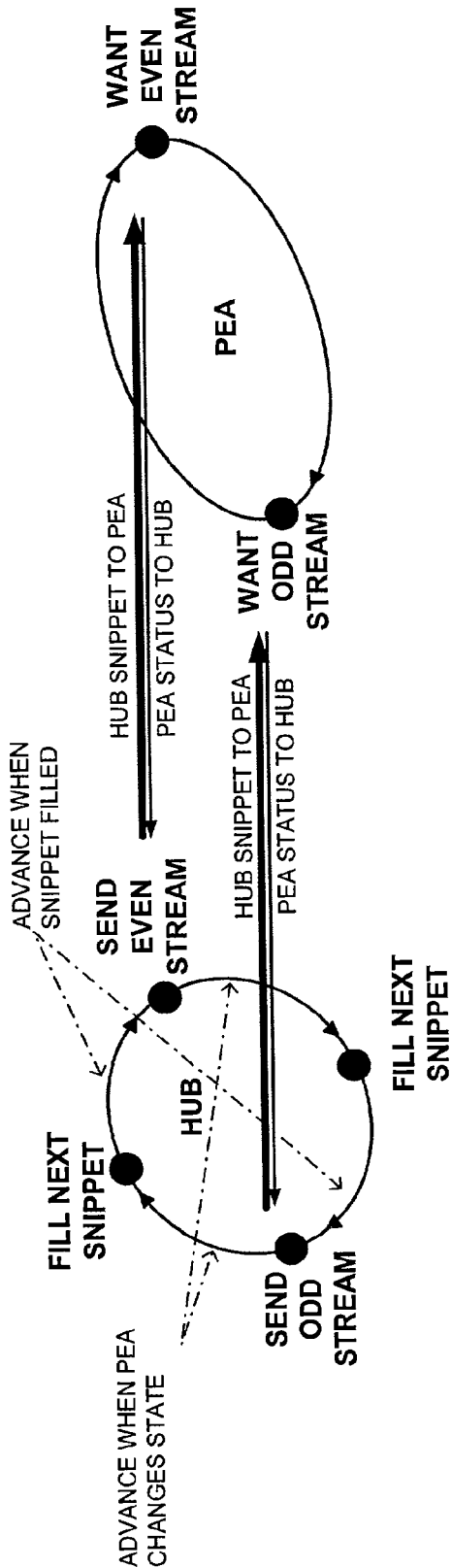


FIG. 10A

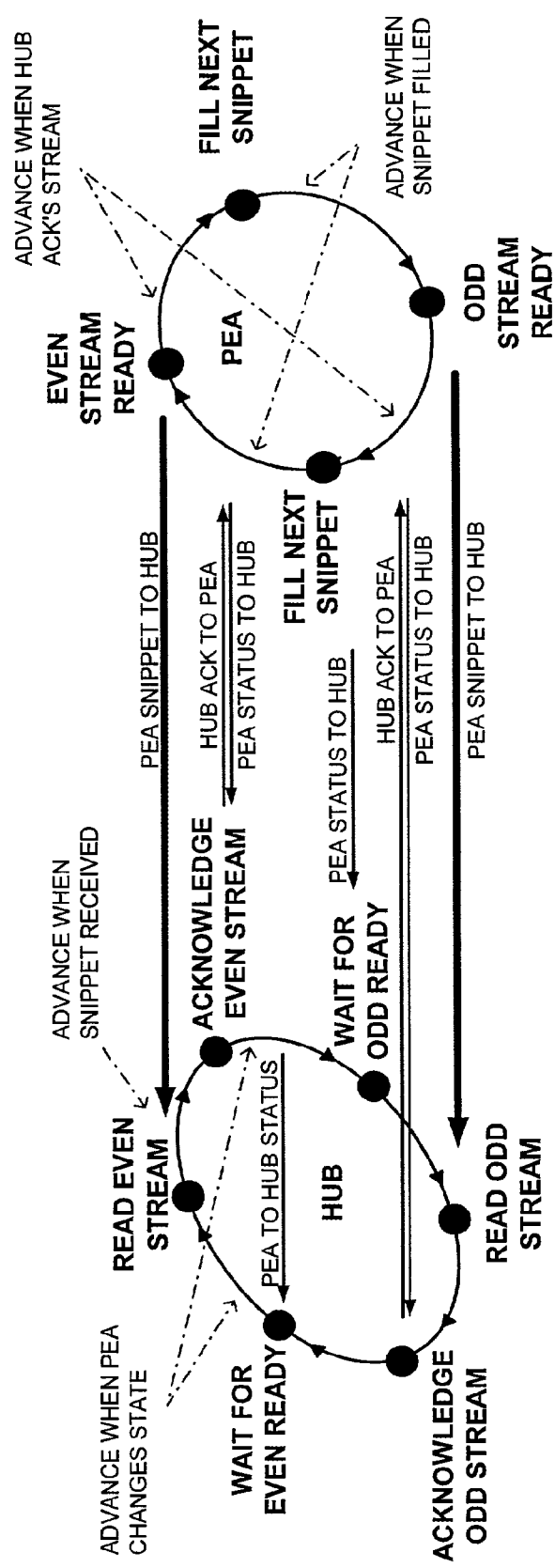
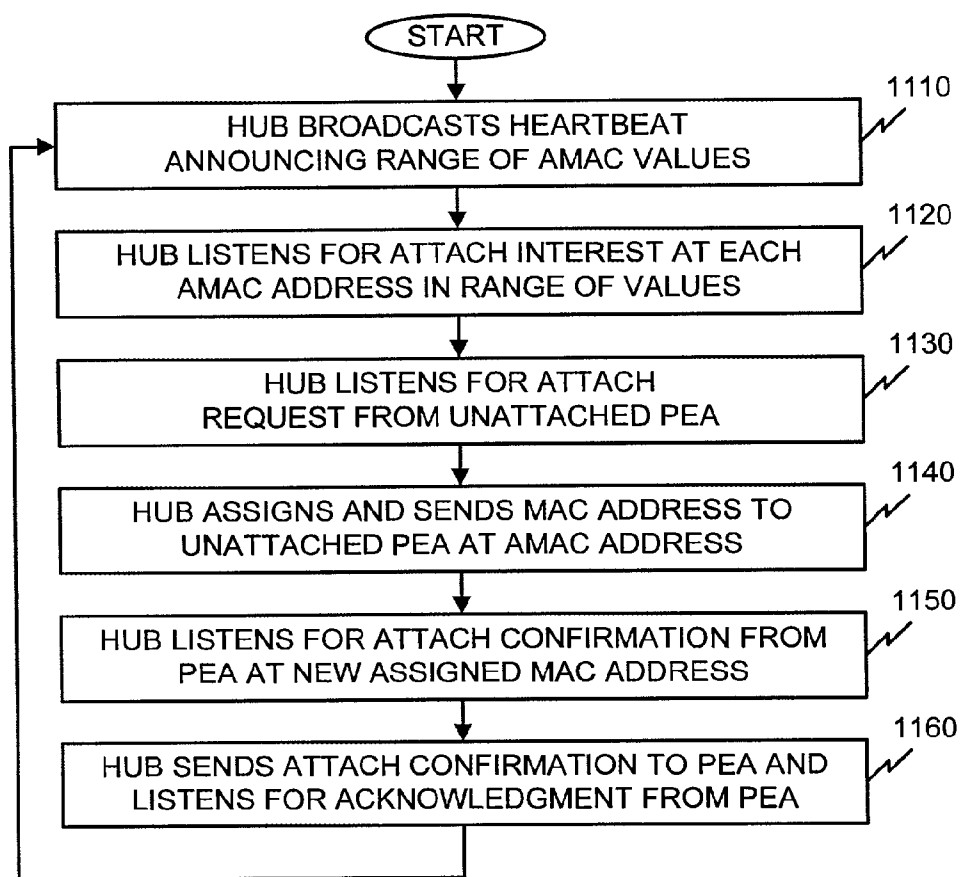


FIG. 10B

**FIG. 11**



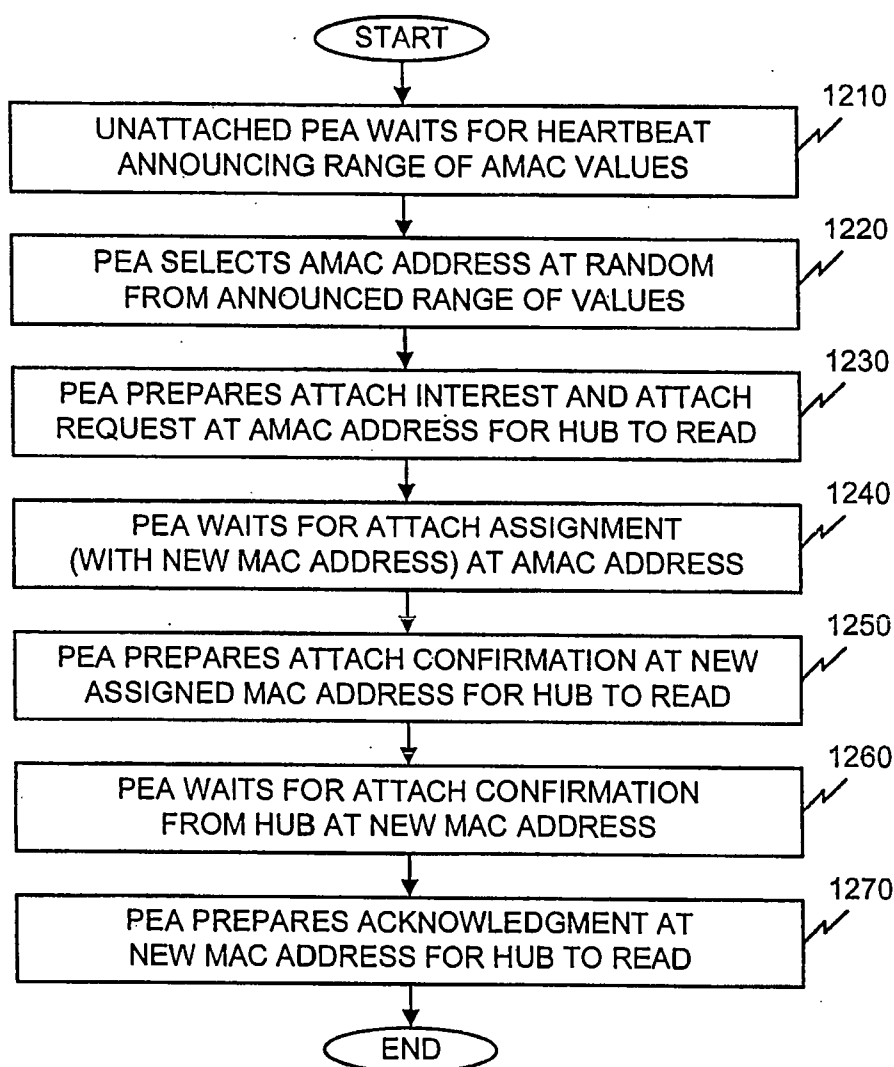
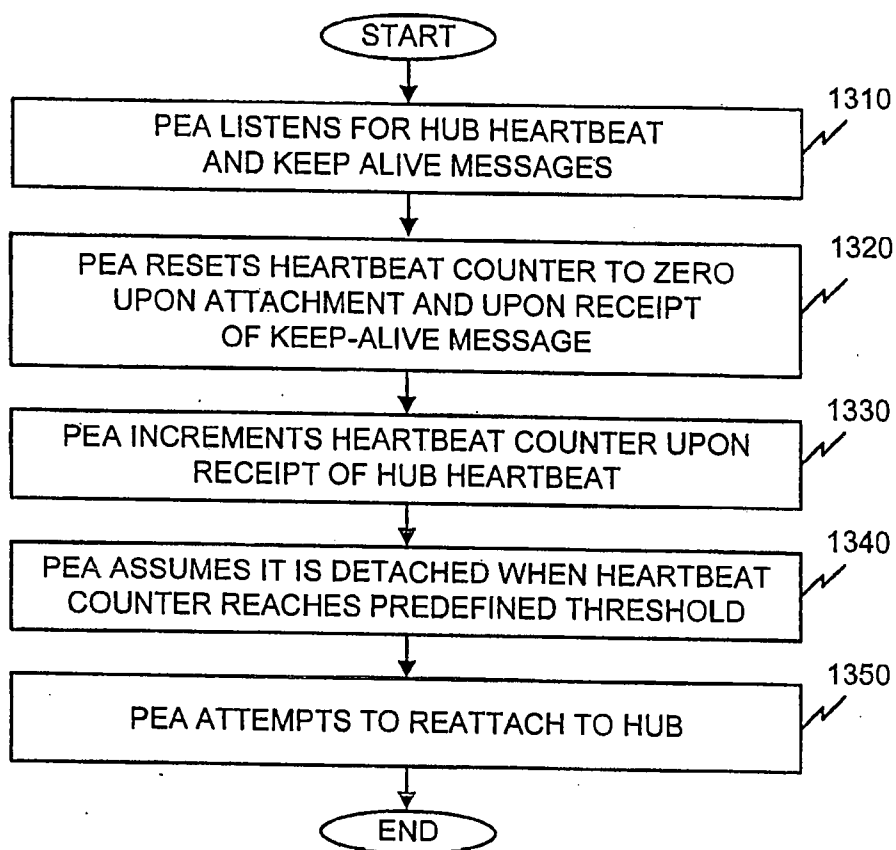


FIG. 12

**FIG. 13**

## PERSONAL AREA NETWORK APPARATUS

### RELATED APPLICATIONS

[0001] This application is a continuation of U.S. patent application Ser. No. 13/430,650 filed on Mar. 26, 2012, which is a continuation of U.S. patent application Ser. No. 12/699,846 filed on Feb. 3, 2010, now U.S. Pat. No. 8,149,829, which is a continuation of U.S. patent application Ser. No. 11/728,246 filed on Mar. 23, 2007, now U.S. Pat. No. 7,756,129, which is a continuation of U.S. patent application Ser. No. 10/894,406 filed on Jul. 19, 2004, now U.S. Pat. No. 7,218,633, which is a continuation of U.S. patent application Ser. No. 09/535,591 filed on Mar. 27, 2000, now U.S. Pat. No. 6,804,232, which is related to U.S. patent application Ser. No. 09/536,191 filed on Mar. 27, 2000, all of which are incorporated herein by reference in their entirety for all purposes.

### BACKGROUND AND FIELD OF THE INVENTION

#### [0002] A. Field of the Invention

[0003] The present invention relates to networking and, more particularly, to a data network.

#### [0004] B. Description of Related Art

[0005] Over the last decade, the size and power consumption of digital electronic devices has been progressively reduced. For example, personal computers have evolved from laptops and notebooks into hand-held or belt-carriable devices commonly referred to as personal digital assistants (PDAs). One area of carriable devices that has remained troublesome, however, is the coupling of peripheral devices or sensors to the main processing unit of the PDA. Generally, such coupling is performed through the use of connecting cables. The connecting cables restrict the handling of a peripheral in such a manner as to lose many of the advantages inherent in the PDA's small size and light weight. For a sensor, for example, that occasionally comes into contact with the PDA, the use of cables is particularly undesirable.

[0006] While some conventional systems have proposed linking a keyboard or a mouse to a main processing unit using infrared or radio frequency (RF) communications, such systems have typically been limited to a single peripheral unit with a dedicated channel of low capacity.

[0007] Based on the foregoing, it is desirable to develop a low power data network that provides highly reliable bidirectional data communication between a host or server processor unit and a varying number of peripheral units and/or sensors while avoiding interference from nearby similar systems.

### SUMMARY OF THE INVENTION

[0008] Systems and methods consistent with the present invention address this need by providing a wireless personal area network that permits a host unit to communicate with peripheral units with minimal interference from neighboring systems.

[0009] A system consistent with the present invention includes a hub device and at least one unattached peripheral device. The unattached peripheral device transmits an attach request to the hub device with a selected address, receives a new address from the hub device to identify the unattached peripheral device, and communicates with the hub device using the new address.

[0010] In another implementation consistent with the present invention, a method for attaching an unattached

peripheral device to a network having a hub device connected to multiple peripheral devices, includes receiving an attach request from the unattached peripheral device, the attach request identifying the unattached peripheral device to the hub device; generating a new address to identify the unattached peripheral device in response to the received attach request; sending the new address to the unattached peripheral device; and sending a confirmation message to the unattached peripheral device using the new address to attach the unattached peripheral device.

[0011] In yet another implementation consistent with the present invention, a method for attaching an unattached peripheral device to a network having a hub device connected to a set of peripheral devices, includes transmitting an attach request with a selected address to the hub device; receiving a new address from the hub device to identify the unattached peripheral device; and attaching to the network using the new address.

### BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings:

[0013] FIG. 1 is a diagram of a personal area network (PAN) in which systems and methods consistent with the present invention may be implemented;

[0014] FIG. 2 is a simplified block diagram of the Hub of FIG. 1;

[0015] FIG. 3 is a simplified block diagram of a PEA of FIG. 1;

[0016] FIG. 4 is a block diagram of a software architecture of a Hub or PEA in an implementation consistent with the present invention;

[0017] FIG. 5 is an exemplary diagram of communication processing by the layers of the software architecture of FIG. 4;

[0018] FIG. 6 is an exemplary diagram of a data block architecture within the DCL of the Hub and PEA in an implementation consistent with the present invention;

[0019] FIG. 7A is a detailed diagram of an exemplary stream usage plan in an implementation consistent with the present invention;

[0020] FIG. 7B is a detailed diagram of an exemplary stream usage assignment in an implementation consistent with the present invention;

[0021] FIG. 8 is an exemplary diagram of a time division multiple access (TDMA) frame structure in an implementation consistent with the present invention;

[0022] FIG. 9A is a detailed diagram of activity within the Hub and PEA according to a TDMA plan consistent with the present invention;

[0023] FIG. 9B is a flowchart of the Hub activity of FIG. 9A;

[0024] FIG. 9C is a flowchart of the PEA activity of FIG. 9A;

[0025] FIGS. 10A and 10B are high-level diagrams of states that the Hub and PEA traverse during a data transfer in an implementation consistent with the present invention;

[0026] FIGS. 11 and 12 are flowcharts of Hub and PEA attachment processing, respectively, consistent with the present invention; and

[0027] FIG. 13 is a flowchart of PEA detachment and reattachment processing consistent with the present invention.

## DETAILED DESCRIPTION

[0028] The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims.

[0029] Systems and methods consistent with the present invention provide a wireless personal area network that permits a host device to communicate with a varying number of peripheral devices with minimal interference from neighboring networks. The host device uses tokens to manage all of the communication in the network, and automatic attachment and detachment mechanisms to communicate with the peripheral devices.

## Network Overview

[0030] A Personal Area Network (PAN) is a local network that interconnects computers with devices (e.g., peripherals, sensors, actuators) within their immediate proximity. These devices may be located nearby and may frequently or occasionally come within range and go out of range of the computer. Some devices may be embedded within an infrastructure (e.g., a building or vehicle) so that they can become part of a PAN as needed.

[0031] A PAN, in an implementation consistent with the present invention, has low power consumption and small size, supports wireless communication without line-of-sight limitations, supports communication among networks of multiple devices (over 100 devices), and tolerates interference from other PAN systems operating within the vicinity. A PAN can also be easily integrated into a broad range of simple and complex devices, is low in cost, and is capable of being used worldwide.

[0032] FIG. 1 is a diagram of a PAN 100 consistent with the present invention. The PAN 100 includes a single Hub device 110 surrounded by multiple Personal Electronic Accessory (PEA) devices 120 configured in a star topology. Other topologies may also be possible. Each device is identified by a Media Access (MAC) address.

[0033] The Hub 110 orchestrates all communication in the PAN 100, which consists of communication between the Hub 110 and one or more PEA(s) 120. The Hub 110 manages the timing of the network, allocates available bandwidth among the currently attached PEAs 120 participating in the PAN 100, and supports the attachment, detachment, and reattachment of PEAs 120 to and from the PAN 100.

[0034] The Hub 110 may be a stationary device or may reside in some sort of wearable computer, such as a simple pager-like device, that may move from peripheral to peripheral. The Hub 110 could, however, include other devices.

[0035] The PEAs 120 may vary dramatically in terms of their complexity. A very simple PEA might include a movement sensor having an accelerometer, an 8-bit microcontroller, and a PAN interface. An intermediate PEA might include a bar code scanner and its microcontroller. More complex PEAs might include PDAs, cellular telephones, or even desktop PCs and workstations. The PEAs may include stationary devices located near the Hub and/or portable devices that move to and away from the Hub.

[0036] The Hub 110 and PEAs 120 communicate using multiplexed communication over a predefined set of streams. Logically, a stream is a one-way communications link

between one PEA 120 and its Hub 110. Each stream has a predetermined size and direction. The Hub 110 uses stream numbers to identify communication channels for specific functions (e.g., data and control).

[0037] The Hub 110 uses MAC addresses to identify itself and the PEAs 120. The Hub 110 uses its own MAC address to broadcast to all PEAs 120. The Hub 110 might also use MAC addresses to identify virtual PEAs within any one physical PEA 120. The Hub 110 combines a MAC address and a stream number into a token, which it broadcasts to the PEAs 120 to control communication through the network 100. The PEA 120 responds to the Hub 110 if it identifies its own MAC address or the Hub MAC address in the token and if the stream number in the token is active for the MAC address of the PEA 120.

## Exemplary Hub Device

[0038] FIG. 2 is a simplified block diagram of the Hub 110 of FIG. 1. The Hub 110 may be a battery-powered device that includes Hub host 210, digital control logic 220, radio frequency (RF) transceiver 230, and an antenna 240.

[0039] Hub host 210 may include anything from a simple microcontroller to a high performance microprocessor. The digital control logic (DCL) 220 may include a controller that maintains timing and coordinates the operations of the Hub host 210 and the RF transceiver 230. The DCL 220 is specifically designed to minimize power consumption, cost, and size of the Hub 110. Its design centers around a time-division multiple access (TDMA)-based network access protocol that exploits the short range nature of the PAN 100. The Hub host 210 causes the DCL 220 to initialize the network 100, send tokens and messages, and receive messages. Responses from the DCL 220 feed incoming messages to the Hub host 210.

[0040] The RF transceiver 230 includes a conventional RF transceiver that transmits and receives information via the antenna 240. The RF transceiver 230 may alternatively include separate transmitter and receiver devices controlled by the DCL 220. The antenna 240 includes a conventional antenna for transmitting and receiving information over the network.

[0041] While FIG. 2 shows the exemplary Hub 110 as consisting of three separate elements, these elements may be physically implemented in one or more integrated circuits. For example, the Hub host 210 and the DCL 220, the DCL 220 and the RF transceiver 230, or the Hub host 210, the DCL 220, and the RF transceiver 230 may be implemented as a single integrated circuit or separate integrated circuits. Moreover, one skilled in the art will recognize that the Hub 110 may include additional elements that aid in the sending, receiving, and processing of data.

## Exemplary PEA Device

[0042] FIG. 3 is a simplified block diagram of the PEA 120. The PEA 120 may be a battery-powered device that includes a PEA host 310, DCL 320, RF transceiver 330, and an antenna 340. The PEA host 310 may include a sensor that responds to information from a user, an actuator that provides output to the user, a combination of a sensor and an actuator, or more complex circuitry, as described above.

[0043] The DCL 320 may include a controller that coordinates the operations of the PEA host 310 and the RF transceiver 330. The DCL 320 sequences the operations necessary in establishing synchronization with the Hub 110, in data

communications, in coupling received information from the RF transceiver 330 to the PEA host 310, and in transmitting data from the PEA host 310 back to the Hub 110 through the RF transceiver 330.

[0044] The RF transceiver 330 includes a conventional RF transceiver that transmits and receives information via the antenna 340. The RF transceiver 330 may alternatively include separate transmitter and receiver devices controlled by the DCL 320. The antenna 340 includes a conventional antenna for transmitting and receiving information over the network.

[0045] While FIG. 3 shows the exemplary PEA 120 as consisting of three separate elements, these elements may be physically implemented in one or more integrated circuits. For example, the PEA host 310 and the DCL 320, the DCL 320 and the RF transceiver 330, or the PEA host 310, the DCL 320, and the RF transceiver 330 may be implemented as a single integrated circuit or separate integrated circuits. Moreover, one skilled in the art will recognize that the PEA 120 may include additional elements that aid in the sending, receiving, and processing of data.

#### Exemplary Software Architecture

[0046] FIG. 4 is an exemplary diagram of a software architecture 400 of the Hub 110 in an implementation consistent with the present invention. The software architecture 400 in the PEA 120 has a similar structure. The software architecture 400 includes several distinct layers, each designed to serve a specific purpose, including: (1) application 410, (2) link layer control (LLC) 420, (3) network interface (NI) 430, (4) link layer transport (LLT) 440, (5) link layer driver (LLD) 450, and (6) DCL hardware 460. The layers have application programming interfaces (APIs) to facilitate communication with lower layers. The LLD 450 is the lowest layer of software. Each layer may communicate with the next higher layer via procedural upcalls that the higher layer registers with the lower layer.

[0047] The application 410 may include any application executing on the Hub 110, such as a communication routine. The LLC 420 performs several miscellaneous tasks, such as initialization, attachment support, bandwidth control, and token planning. The LLC 420 orchestrates device initialization, including the initialization of the other layers in the software architecture 400, upon power-up.

[0048] The LLC 420 provides attachment support by providing attachment opportunities for unattached PEAs to attach to the Hub 110 so that they can communicate, providing MAC address assignment, and initializing an NI 430 and the layers below it for communication with a PEA 120. The LLC 420 provides bandwidth control through token planning. Through the use of tokens, the LLC 420 allocates bandwidth to permit one PEA 120 at a time to communicate with the Hub 110.

[0049] The NI 430 acts on its own behalf, or for an application 410 layer above it, to deliver data to the LLT 440 beneath it. The LLT 440 provides an ordered, reliable “snippet” (i.e., a data block) delivery service for the NI 430 through the use of encoding (e.g., 16-64 bytes of data plus a cyclic redundancy check (CRC)) and snippet retransmission. The LLT 440 accepts snippets, in order, from the NI 430 and delivers them using encoded status blocks (e.g., up to 2 bytes of status information translated through Forward Error Correction (FEC) into 6 bytes) for acknowledgments (ACKs).

[0050] The LLD 450 is the lowest level of software in the software architecture 400. The LLD 450 interacts with the DCL hardware 460. The LLD 450 initializes and updates data transfers via the DCL hardware 460 as it delivers and receives data blocks for the LLT 440, and processes hardware interrupts. The DCL hardware 460 is the hardware driven by the LLD 450.

[0051] FIG. 5 is an exemplary diagram of communication processing by the layers of the software architecture 400 of FIG. 4. In FIG. 5, the exemplary communications involve the transmission of a snippet from one node to another. This example assumes that the sending node is the Hub 110 and the receiving node is a PEA 120. Processing begins with the NI 430 of the Hub 110 deciding to send one or more bytes (but no more than will fit) in a snippet. The NI 430 exports the semantics that only one transaction is required to transmit these bytes to their destination (denoted by “(1)” in the figure). The NI 430 sends a unique identifier for the destination PEA 120 of the snippet to the LLT 440. The LLT 440 maps the PEA identifier to the MAC address assigned to the PEA 120 by the Hub 110.

[0052] The LLT 440 transmits the snippet across the network to the receiving device. To accomplish this, the LLT 440 adds header information (to indicate, for example, how many bytes in the snippet are padded bytes) and error checking information to the snippet, and employs reverse-direction status/acknowledgment messages and retransmissions. This is illustrated in FIG. 5 by the bidirectional arrow between the LLT 440 layers marked with “(n+m).” The number n of snippet transmissions and the number m of status transmissions in the reverse direction are mostly a function of the amount of noise in the wireless communication, which may be highly variable. The LLT 440 may also encrypt portions or all of the snippet using known encryption technology.

[0053] The LLT 440 uses the LLD 450 to provide a basic block and stream-oriented communications service, isolating the DCL 460 interface from the potentially complex processing required of the LLT 440. The LLT 440 uses multiple stream numbers to differentiate snippet and status blocks so that the LLD 450 need not know which blocks contain what kind of content. The LLD 450 reads and writes the hardware DCL 460 to trigger the transmission and reception of data blocks. The PEA LLT 440, through the PEA LLD 450, instructs the PEA DCL 460 which MAC address or addresses to respond to, and which stream numbers to respond to for each MAC address. The Hub LLT 440, through the Hub LLD 450, instructs the Hub DCL 460 which MAC addresses and stream numbers to combine into tokens and transmit so that the correct PEA 120 will respond. The Hub DCL 460 sends and receives (frequently in a corrupted form) the data blocks across the RF network via the Hub RF transceiver 230 (FIG. 2).

[0054] The Hub LLT 440 employs FEC for status, checksums and error checking for snippets, and performs retransmission control for both to ensure that each snippet is delivered reliably to its client (e.g., PEA LLT 440). The PEA LLT 440 delivers snippets in the same order that they were sent by the Hub NI 430 to the PEA NI 430. The PEA NI 430 takes the one or more bytes sent in the snippets and delivers them in order to the higher-level application 410, thereby completing the transmission.

### Exemplary DCL Data Block Architecture

[0055] FIG. 6 is an exemplary diagram of a data block architecture 600 within the DCL of the Hub 110 and the PEA 120. The data block 600 contains a MAC address 610 designating a receiving or sending PEA 120, a stream number 620 for the communication, and a data buffer 630 which is full when sending and empty when receiving. As will be described later, the MAC address 610 and stream number 620 form the contents of a token 640. When the LLD 450 reads from and writes to the hardware DCL 460, the LLD 450 communicates the MAC address 610 and stream number 620 with the data buffer 630. When a PEA 120 receives a data block, the DCL 460 places the MAC address 610 and stream number 620 contained in the preceding token 640 in the data block 600 to keep track of the different data flows.

### Exemplary Stream Architecture

[0056] The LLD 450 provides a multi-stream data transfer service for the LLT 440. While the LLT 440 is concerned with data snippets and status/acknowledgements, the LLD 450 is concerned with the size of data blocks and the direction of data transfers to and from the Hub 110.

[0057] FIG. 7A is a detailed diagram of an exemplary stream usage plan 700 in an implementation consistent with the present invention. A single stream usage plan may be predefined and used by the Hub 110 and all PEAs 120. The PEA 120 may have a different set of active streams for each MAC address it supports, and only responds to a token that specifies a MAC address of the PEA 120 and a stream that is active for that MAC address. In an implementation consistent with the present invention, every PEA 120 may support one or more active Hub-to-PEA streams associated with the Hub's MAC address.

[0058] The stream usage plan 700 includes several streams 710-740, each having a predefined size and data transfer direction. The plan 700 may, of course, have more or fewer entries and may accommodate more than the two data block sizes shown in the figure. In the plan 700, streams 0-2 (710) are used to transmit the contents of small data blocks from the PEA 120 to the Hub 110. Streams 3-7 (720) are used to transmit the contents of larger data blocks from the PEA 120 to the Hub 110. Streams 8-10 (730), on the other hand, are used to transmit the contents of small data blocks from the Hub 110 to the PEA 120. Streams 11-15 (740) are used to transmit the contents of larger data blocks from the Hub 110 to the PEA 120.

[0059] To avoid collisions, some of the streams are reserved for PEAs desiring to attach to the network and the rest are reserved for PEAs already attached to the network. With such an arrangement, a PEA 120 knows whether and what type of communication is scheduled by the Hub 110 based on a combination of the MAC address 610 and the stream number 620.

[0060] FIG. 7B is a detailed diagram of an exemplary stream usage assignment by the LLT 440 in an implementation consistent with the present invention. The LLT 440 assigns different streams to different communication purposes, reserving the streams with small block size for status, and using the streams with larger block size for snippets. For example, the LLT 440 may use four streams (4-7 and 12-15) for the transmission of snippets in each direction, two for odd parity snippets and two for even parity snippets. In other

implementations consistent with the present invention, the LLT 440 uses different numbers of streams of each parity and direction.

[0061] The use of more than one stream for the same snippet allows a snippet to be sent in more than one form. For example, the LLT 440 may send a snippet in its actual form through one stream and in a form with bytes complemented and in reverse order through the other stream. The alternating use of different transformations of a snippet more evenly distributes transmission errors among the bits of the snippet as they are received, and hence facilitates the reconstruction of a snippet from multiple corrupted received versions. The receiver always knows which form of the snippet was transmitted based on its stream number.

[0062] The LLT 440 partitions the streams into two disjoint subsets, one for use with Hub 110 assigned MAC addresses 750 and the other for use with attaching PEAs' self-selected MAC addresses (AMACs) 760. Both the LLT 440 and the LLD 450 know the size and direction of each stream, but the LLT 450 is responsible for determining how the streams are used, how MAC numbers are assigned and used, and assuring that no two PEAs 120 respond to the same token (containing a MAC address and stream number) transmitted by the Hub 110. One exception to this includes the Hub's use of its MAC address to broadcast its heartbeat 770 (described below) to all PEAs 120.

### Exemplary Communication

[0063] FIG. 8 is an exemplary diagram of a TDMA frame structure 800 of a TDMA plan consistent with the present invention. The TDMA frame 800 starts with a beacon 810, and then alternates token broadcasts 820 and data transfers 830. The Hub 110 broadcasts the beacon 810 at the start of each TDMA frame 800. The PEAs 120 use the beacon 810, which may contain a unique identifier of the Hub 110, to synchronize to the Hub 110.

[0064] Each token 640 (FIG. 6) transmitted by the Hub 110 in a token broadcast 820 includes a MAC address 610 (FIG. 6) and a stream number 620 for the data buffer 630 transfer that follows. The MAC address 610 and stream number 620 in the token 640 together specify a particular PEA 120 to transmit or receive data, or, in the case of the Hub's MAC address 610, specify no, many, or all PEAs to receive data from the Hub 110 (depending on the stream number). The stream number 620 in the token 640 indicates the direction of the data transfer 830 (Hub 110 to PEA 120 or PEA 120 to Hub 110), the number of bytes to be transferred, and the data source (for the sender) and the appropriate empty data block (for the receiver).

[0065] The TDMA plan controls the maximum number of bytes that can be sent in a data transfer 830. Not all of the permitted bytes need to be used in the data transfer 830, however, so the Hub 110 may schedule a status block in the initial segment of a TDMA time interval that is large enough to send a snippet. The Hub 110 and PEA 120 treat any left over bytes as no-ops to mark time. Any PEA 120 not involved in the data transfer uses all of the data transfer 830 bytes to mark time while waiting for the next token 640. The PEA 120 may also power down non-essential circuitry at this time to reduce power consumption.

[0066] FIG. 9A is an exemplary diagram of communication processing for transmitting a single data block from the Hub 110 to a PEA 120 according to the TDMA plan of FIG. 8. FIGS. 9B and 9C are flowcharts of the Hub 110 and PEA 120

activities, respectively, of FIG. 9A. The reference numbers in FIG. 9A correspond to the flowchart steps of FIGS. 9B and 9C.

[0067] With regard to the Hub activity, the Hub 110 responds to a token command in the TDMA plan [step 911] (FIG. 9B) by determining the location of the next data block 600 to send or receive [step 912]. The Hub 110 reads the block's MAC address 610 and stream number 620 [step 913] and generates a token 640 from the MAC address and stream number using FEC [step 914]. The Hub 110 then waits for the time for sending a token 640 in the TDMA plan (i.e., a token broadcast 820 in FIG. 8) [step 915] and broadcasts the token 640 to the PEAs 120 [step 916]. If the stream number 620 in the token 640 is zero (i.e., a NO-DATA-TRANSFER token), no PEA 120 will respond and the Hub 110 waits for the next token command in the TDMA plan [step 911].

[0068] If the stream number 620 is non-zero, however, the Hub 110 determines the size and direction of the data transmission from the stream number 620 and waits for the time for sending the data in the TDMA plan (i.e., a data transfer 830) [step 917]. Later, when instructed to do so by the TDMA plan (i.e., after the PEA 120 identified by the MAC address 610 has had enough time to prepare), the Hub 110 transmits the contents of the data buffer 630 [step 918]. The Hub 110 then prepares for the next token command in the TDMA plan [step 919].

[0069] With regard to the PEA activity, the PEA 120 reaches a token command in the TDMA plan [step 921] (FIG. 9C). The PEA 120 then listens for the forward error-corrected token 640, having a MAC address 610 and stream number 620, transmitted by the Hub 110 [step 922]. The PEA 120 decodes the MAC address from the forward error-corrected token [step 923] and, if it is not the PEA's 120 MAC address, sleeps through the next data transfer 830 in the TDMA plan [step 924]. Otherwise, the PEA 120 also decodes the stream number 620 from the token 640.

[0070] All PEAs 120 listen for the Hub heartbeat that the Hub 110 broadcasts with a token containing the Hub's MAC address 610 and the heartbeat stream 770. During attachment (described in more detail below), the PEA 120 may have two additional active MAC addresses 610, the one it selected for attachment and the one the Hub 110 assigned to the PEA 120. The streams are partitioned between these three classes of MAC addresses 610, so the PEA 120 may occasionally find that the token 640 contains a MAC address 610 that the PEA 120 supports, but that the stream number 620 in the token 640 is not one that the PEA 120 supports for this MAC address 610. In this case, the PEA 120 sleeps through the next data transfer 830 in the TDMA plan [step 924].

[0071] Since the PEA 120 supports more than one MAC address 610, the PEA 120 uses the MAC address 610 and the stream number 620 to identify a suitable empty data block [step 925]. The PEA 120 writes the MAC address 610 and stream number 620 it received in the token 640 from the Hub 110 into the data block [step 926]. The PEA 120 then determines the size and direction of the data transmission from the stream number 620 and waits for the transmission of the data buffer 630 contents from the Hub 110 during the next data transfer 830 in the TDMA plan [step 927]. The PEA 120 stores the data in the data block [step 928], and then prepares for the next token command in the TDMA plan [step 929].

[0072] FIGS. 9A-9C illustrate communication of a data block from the Hub 110 to a PEA 120. When the PEA 120 transfers a data block to the Hub 110, similar steps occur

except that the Hub 110 first determines the next data block to receive (with its MAC address 610 and stream number 620) and the transmission of the data buffer 630 contents occurs in the opposite direction. The Hub 110 needs to arrange in advance for receiving data from PEAs 120 by populating the MAC address 610 and stream number 620 into data blocks with empty data buffers 630, because the Hub 110 generates the tokens for receiving data as well as for transmitting data.

[0073] FIGS. 10A and 10B are high-level diagrams of the states that the Hub 110 and PEA 120 LLT 440 (FIG. 4) go through during a data transfer in an implementation consistent with the present invention. FIG. 10A illustrates states of a Hub-to-PEA transfer and FIG. 10B illustrates states of a PEA-to-Hub transfer.

[0074] During the Hub-to-PEA transfer (FIG. 10A), the Hub 110 cycles through four states: fill, send even parity, fill, and send odd parity. The fill states indicate when the NI 430 (FIG. 4) may fill a data snippet. The even and odd send states indicate when the Hub 110 sends even numbered and odd numbered snippets to the PEA 120. The PEA 120 cycles through two states: want even and want odd. The two states indicate the PEA's 120 desire for data, with 'want even' indicating that the last snippet successfully received had odd parity. The PEA 120 communicates its current state to the Hub 110 via its status messages (i.e., the state changes serve as ACKs). The Hub 110 waits for a state change in the PEA 120 before it transitions to its next fill state.

[0075] During the PEA-to-Hub transfer (FIG. 10B), the Hub 110 cycles through six states: wait/listen for PEA-ready-to-send-even status, read even, send ACK and listen for status, wait/listen for PEA-ready-to-send-odd status, read odd, and send ACK and listen for status. According to this transfer, the PEA 120 cannot transmit data until the Hub 110 requests data, which it will only do if it sees from the PEA's status that the PEA 120 has the next data block ready.

[0076] The four listen for status states schedule when the Hub 110 asks to receive a status message from the PEA 120. The two 'send ACK and listen for status' states occur after successful receipt of a data block by the Hub 110, and in these two states the Hub 110 schedules both the sending of Hub status to the PEA 120 and receipt of the PEA status. The PEA status informs the Hub 110 when the PEA 120 has successfully received the Hub 110 status and has transitioned to the next 'fill' state.

[0077] Once the PEA 120 has prepared its next snippet, it changes its status to 'have even' or 'have odd' as appropriate. When the Hub 110 detects that the PEA 120 has advanced to the fill state or to 'have even/odd,' it stops scheduling the sending of Hub status (ACK) to the PEA 120. If the Hub 110 detects that the PEA 120 is in the 'fill' state, it transitions to the following 'listen for status' state. If the PEA 120 has already prepared a new snippet for transmission by the time the Hub 110 learns that its ACK was understood by the PEA 120, the Hub 110 skips the 'listen for status' state and moves immediately to the next appropriate 'read even/odd' state. In this state, the Hub 110 receives the snippet from the PEA 120.

[0078] The PEA 120 cycles through four states: fill, have even, fill, and have odd (i.e., the same four states the Hub 110 cycles through when sending snippets). The fill states indicate when the NI 430 (FIG. 4) can fill a data snippet. During the fill states, the PEA 110 sets its status to 'have nothing to send.' The PEA 120 does not transition its status to 'have even' or 'have odd' until the next snippet is filled and ready to send to the Hub 110. These two status states indicate the parity of the

snippet that the PEA 120 is ready to send to the Hub 110. When the Hub 110 receives a status of ‘have even’ or ‘have odd’ and the last snippet it successfully received had the opposite parity, it schedules the receipt of data, which it thereafter acknowledges with a change of status that it sends to the PEA 120.

#### Exemplary Attachment Processing

[0079] The Hub 110 communicates with only attached PEAs 120 that have an assigned MAC address 610. An unattached PEA can attach to the Hub 110 when the Hub 110 gives it an opportunity to do so. Periodically, the Hub 110 schedules attachment opportunities for unattached PEAs that wish to attach to the Hub 110, using a small set of attach MAC (AMAC) addresses and a small set of streams dedicated to this purpose.

[0080] After selecting one of the designated AMAC addresses 610 at random to identify itself and preparing to send a small, possibly forward error-corrected, “attach-interest” message and a longer, possibly checksummed, “attach-request” message using this AMAC and the proper attach stream numbers 620, the PEA 120 waits for the Hub 110 to successfully read the attach-interest and then the attach-request messages. Reading of a valid attach-interest message by the Hub 110 causes the Hub 110 believe that there is a PEA 120 ready to send the longer (and hence more likely corrupted) attach-request.

[0081] Once a valid attach-interest is received, the Hub 110 schedules frequent receipt of the attach-request until it determines the contents of the attach-request, either by receiving the block intact with a valid checksum or by reconstructing the sent attach-request from two or more received instances of the sent attach-request. The Hub 110 then assigns a MAC address to the PEA 120, sending the address to the PEA 120 using its AMAC address.

[0082] The Hub 110 confirms receipt of the MAC address by scheduling the reading of a small, possibly forward error-corrected, attach-confirmation from the PEA 120 at its new MAC address 610. The Hub 110 follows this by sending a small, possibly forward error-corrected, confirmation to the PEA 120 at its MAC address so that the PEA 120 knows it is attached. The PEA 120 returns a final small, possibly forward error-corrected, confirmation acknowledgement to the Hub 110 so that the Hub 110, which is in control of all scheduled activity, has full knowledge of the state of the PEA 120. This MAC address remains assigned to that PEA 120 for the duration of the time that the PEA 120 is attached.

[0083] FIGS. 11 and 12 are flowcharts of Hub and PEA attachment processing, respectively, consistent with the present invention. When the Hub 110 establishes the network, its logic initializes the attachment process and, as long as the Hub 110 continues to function, periodically performs attachment processing. The Hub 110 periodically broadcasts heartbeats containing a Hub identifier (selecting a new heartbeat identifier value each time it reboots) and an indicator of the range of AMACs that can be selected from for the following attach opportunity [step 1110] (FIG. 11). The Hub 110 schedules an attach-interest via a token that schedules a small PEA-to-Hub transmission for each of the designated AMACs, so unattached PEAs may request attachment.

[0084] Each attaching PEA 120 selects a new AMAC at random from the indicated range when it hears the heartbeat. Because the Hub 110 may receive a garbled transmission whenever more than one PEA 120 transmits, the Hub 110

occasionally indicates a large AMAC range (especially after rebooting) so that at least one of a number of PEAs 120 may select a unique AMAC 610 and become attached. When no PEAs 120 have attached for some period of time, however, the Hub 110 may select a small range of AMACs 610 to reduce attachment overhead, assuming that PEAs 120 will arrive in its vicinity in at most small groups. The Hub 110 then listens for a valid attach-interest from an unattached PEA [step 1120]. The attach-interest is a PEA-to-Hub message having the AMAC address 610 selected by the unattached PEA 120.

[0085] Upon receiving a valid attach interest, the Hub 110 schedules a PEA-to-Hub attach-request token with the PEA’s AMAC 610 and reads the PEA’s attach-request [step 1130]. Due to the low-power wireless environment of the PAN 100, the attach-request transmission may take more than one attempt and hence may require scheduling the PEA-to-Hub attach-request token more than once. When the Hub 110 successfully receives the attach-request from the PEA, it assigns a MAC address to the PEA [step 1140]. In some cases, the Hub 110 chooses the MAC address from the set of AMAC addresses.

[0086] The Hub 110 sends the new MAC address 610 in an attach-assignment message to the now-identified PEA 120, still using the PEA’s AMAC address 610 and a stream number 620 reserved for this purpose. The Hub 110 schedules and listens for an attach-confirmation response from the PEA 120 using the newly assigned MAC address 610 [step 1150].

[0087] Upon receiving the confirmation from the PEA 120, the Hub 110 sends its own confirmation, acknowledging that the PEA 120 has switched to its new MAC, to the PEA 120 and waits for a final acknowledgment from the PEA 120 [step 1160]. The Hub 110 continues to send the confirmation until it receives the acknowledgment from the PEA 120 or until it times out. In each of the steps above, the Hub 110 counts the number of attempts it makes to send or receive, and aborts the attachment effort if a predefined maximum number of attempts is exceeded. Upon receiving the final acknowledgment, the Hub 110 stops sending its attach confirmation, informs its NI 430 (FIG. 4) that the PEA 120 is attached, and begins exchanging both data and keep-alive messages (described below) with the PEA 120.

[0088] When an unattached PEA 120 enters the network, its LLC 420 (FIG. 4) instructs its LLT 440 to initialize attachment. Unlike the Hub 110, the PEA 120 waits to be polled. The PEA 120 instructs its DCL 460 to activate and associate the heartbeat stream 770 (FIG. 7B) with the Hub’s MAC address and waits for the heartbeat broadcast from the Hub 110 [step 1210] (FIG. 12). The PEA 120 then selects a random AMAC address from the range indicated in the heartbeat to identify itself to the Hub 110 [step 1220]. The PEA 120 instructs its DCL 460 to send an attach-interest and an attach-request data block to the Hub 110, and activate and associate the streams with its AMAC address [step 1230]. The PEA 120 tells its driver to activate and respond to the selected AMAC address for the attach-assignment stream.

[0089] The unattached PEA 120 then waits for an attach-assignment with an assigned MAC address from the Hub 110 [step 1240]. Upon receiving the attach-assignment, the PEA 120 finds its Hub-assigned MAC address and tells its driver to use this MAC address to send an attach-confirmation to the Hub 110 to acknowledge receipt of its new MAC address [step 1250], activate all attached-PEA streams for its new MAC address, and deactivate the streams associated with its AMAC address.



[0090] The PEA 120 waits for an attach confirmation from the Hub 110 using the new MAC address [step 1260] and, upon receiving it, sends a final acknowledgment to the Hub 110 [step 1270]. The PEA 120 then tells its NI 430 that it is attached.

[0091] The PEA 120, if it hears another heartbeat from the Hub 110 before it completes attachment, discards any prior communication and begins its attachment processing over again with a new AMAC.

#### Exemplary Detachment and Reattachment Processing

[0092] The Hub 110 periodically informs all attached PEAs 120 that they are attached by sending them 'keep-alive' messages. The Hub 110 may send the messages at least as often as it transmits heartbeats. The Hub 110 may send individual small, possibly forward error-corrected, keep-alive messages to each attached PEA 120 when few PEAs 120 are attached, or may send larger, possibly forward error-corrected, keep-alive messages to groups of PEAs 120.

[0093] Whenever the Hub 110 schedules tokens for PEA-to-Hub communications, it sets a counter to zero. The counter resets to zero each time the Hub 110 successfully receives a block (either uncorrupted or reconstructed) from the PEA 120, and increments for unreadable blocks. If the counter exceeds a predefined threshold, the Hub 110 automatically detaches the PEA 120 without any negotiation with the PEA 120. After this happens, the Hub 110 no longer schedules data or status transfers to or from the PEA 120, and no longer sends it any keep-alive messages.

[0094] FIG. 13 is a flowchart of PEA detachment and reattachment processing consistent with the present invention. Each attached PEA 120 listens for Hub heartbeat and keep-alive messages [step 1310]. When the PEA 120 first attaches, and after receiving each keep-alive message, it resets its heartbeat counter to zero [step 1320]. Each time the PEA 120 hears a heartbeat, it increments the heartbeat counter [step 1330]. If the heartbeat counter exceeds a predefined threshold, the PEA 120 automatically assumes that the Hub 110 has detached it from the network 100 [step 1340]. After this happens, the PEA 120 attempts to reattach to the Hub 110 [step 1350], using attachment processing similar to that described with respect to FIGS. 11 and 12.

[0095] If the Hub 110 had not actually detached the PEA 120, then the attempt to reattach causes the Hub 110 to detach the PEA 120 so that the attempt to reattach can succeed. When the PEA 120 is out of range of the Hub 110, it may not hear from the Hub 110 and, therefore, does not change state or increment its heartbeat counter. The PEA 120 has no way to determine whether the Hub 110 has detached it or how long the Hub 110 might wait before detaching it. When the PEA 120 comes back into range of the Hub 110 and hears the Hub heartbeat (and keep-alive if sent), the PEA 120 then determines whether it is attached and attempts to reattach if necessary.

#### CONCLUSION

[0096] Systems and methods consistent with the present invention provide a wireless personal area network that permit a host device to communicate with a varying number of peripheral devices with minimal power and minimal interference from neighboring networks by using a customized

TDMA protocol. The host device uses tokens to facilitate the transmission of data blocks through the network.

[0097] The foregoing description of exemplary embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. The scope of the invention is defined by the claims and their equivalents.

What is claimed is:

1. In a personal area network having at least one peripheral device not communicably attached to the network and a hub device connected to the network, a method for attaching the peripheral device to the network, comprising:

transmitting, by the peripheral device, an attachment request to attach to the network with a selected address; receiving the attachment request by the hub device; generating a new address for the peripheral device in response to the received attachment request; sending the new address from the hub device to the peripheral device using the selected address; further communicating between the hub device and the peripheral device using the new address; and broadcasting, by the hub device, a heartbeat signal to begin an attach cycle.

2. The method of claim 1, further comprising: abandoning a current attach attempt by the peripheral device upon receipt of the heartbeat signal prior to receiving the new address from the hub device, and beginning a new attempt to attach to the network.

3. The method of claim 1, further comprising: broadcasting, by the hub device, a range of available addresses.

4. The method of claim 3, further comprising: selecting, by the peripheral device, an address from the range of available addresses to identify the peripheral device to the hub device in the attachment request.

5. The method of claim 4, further comprising: listening, by the hub device, for an address in the range of available addresses, for a signal from an unattached peripheral device.

6. The method of claim 1, further comprising: attaching the peripheral device to the network using the new address.

7. The method of claim 1, wherein the network further includes at least one other peripheral device communicably coupled to the hub device; and

wherein the method further comprises:

determining, by the hub device, whether a transmission from the other peripheral device was successful; and detaching the other peripheral device when the transmission fails to be successful in a predetermined number of attempts.

8. The method of claim 1, wherein the network further includes at least one other peripheral device communicably coupled to the hub device; and

wherein the method further comprises:

transmitting, by the hub device, a heartbeat signal to each of the other peripheral devices; and periodically transmitting, by the hub device, a keep-alive signal to each of the other peripheral devices.

9. The method of claim 8, further comprising: receiving the heartbeat signal at the other peripheral devices;

receiving the keep-alive signal at the other peripheral devices; and  
detaching from the network when a predetermined number of heartbeat signals has been received prior to the keep-alive signal.

**20.** The method of claim **1**, wherein the network further includes at least one other peripheral device communicably coupled to the hub device; and

wherein the method further comprises:

receiving, by the hub device, an attach request from one of the other peripheral devices, and

detaching the other peripheral device in response to receipt of the attach request from the other peripheral device.

**11.** A hub device in a network having a plurality of attached peripheral devices and an unattached peripheral device, comprising:

a memory having instructions for:

broadcasting a range of available addresses,

generating a new address to identify the unattached peripheral device in response to receipt of an attach request from the unattached peripheral device, the attach request identifying the unattached peripheral device to the hub device,

sending the new address to the unattached peripheral device using the selected address,

receiving confirmation from the unattached peripheral device using the new address, and

sending a confirmation message to the unattached peripheral device using the new address; and

a processor that executes the instructions in the memory.

**12.** The hub device of claim **11**, wherein the memory further includes instructions for:

broadcasting a heartbeat signal to begin an attach cycle.

**13.** The hub device of claim **12**, wherein the instructions for broadcasting include:

broadcasting a new range of available addresses with each heartbeat signal.

**14.** The hub device of claim **11**, wherein the memory further includes instructions for:

listening, for each of the addresses in the range of available addresses, for a signal from the unattached peripheral device.

**15.** The hub device of claim **11**, wherein the memory further includes instructions for:

determining whether a transmission from one of the attached peripheral devices was successful, and  
detaching the attached peripheral device when the transmission fails to be successful in a predetermined number of attempts.

**16.** A method for attaching an unattached peripheral device to a network having a hub device connected to a set of peripheral devices, the method, performed by the hub device, comprising:

receiving an attach request from the unattached peripheral device, the attach request identifying the unattached peripheral device to the hub device;

generating a new address to identify the unattached peripheral device in response to the received attach request;

sending the new address to the unattached peripheral device; and

sending a confirmation message to the unattached peripheral device using the new address to attach the unattached peripheral device.

**17.** The method of claim **16**, further comprising:

broadcasting a range of available addresses from which the unattached peripheral device selects to use when sending the attach request to the hub device.

**18.** The method of claim **17**, further comprising:

listening, for each of the addresses in the range of available addresses, for a signal from the unattached peripheral device;

determining whether a transmission from the unattached peripheral device at one of the addresses was successful; and

discarding an attach attempt by the unattached peripheral device at the one address when the transmission fails to be successful in a predetermined number of attempts.

**19.** The method of claim **16**, further comprising:

attaching the unattached peripheral device to the network to create a newly attached peripheral device capable of communicating with the hub device.

**20.** A computer-readable medium that stores instructions executable by a hub device to cause the hub device to perform a method for attaching an unattached peripheral device to a network, the method comprising:

broadcasting a range of available addresses;

listening, for each of the available addresses, for an attach request from an unattached peripheral device;

generating a new address to identify the unattached peripheral device in response to receipt of the attach request from the unattached peripheral device, the attach request identifying the unattached peripheral device to the hub device;

sending the new address to the unattached peripheral device;

receiving confirmation from the peripheral device using the new address; and

attaching the unattached peripheral device using the new address.

**21.** An unattached peripheral device in a network having a plurality of peripheral devices connected to a hub device, comprising:

a memory having instructions for:

selecting an address from a range of available addresses broadcast by the hub device to identify the unattached peripheral device to the hub device,

transmitting an attach request with the selected address to the hub device,

receiving a new address from the hub device to identify the unattached peripheral device,

sending a confirmation message with the new address to the hub device, and

attaching to the hub device using the new address; and

a processor that executes the instructions in the memory.

\* \* \* \* \*