(57) Abrégé/Abstract:
A system for remotely updating software on at least one electronic device connected to a network. The electronic devices have a non-volatile re-writable storage unit divided into at least two partitions, one of which will contain core firmware and the other of which will contain auxiliary software. When an update is received at the device, the updated core firmware is written to overwrite the partition in the re-writable storage unit which contained the auxiliary software. When this is completed and verified, the previous version of the core firmware stored in the storage unit is disabled from execution by the device. Next, the updated auxiliary software is written to overwrite the old version of the core firmware. When this write is complete, the device determines a suitable time for it to be rebooted to execute the updated software. In another embodiment, the present core firmware in the device is copied from the partition it is in to the other partition, overwriting the auxiliary software stored there. The new core firmware received to update the device is overwritten into the first partition, the old copied core firmware being present in case of an upgrade failure, and upon a successful update of the first partition, the auxiliary software is written to the second partition, overwriting the copied old core firmware. In this manner, the position of the core firmware and auxiliary software within the partitions is preserved during normal operation of the device.
ABSTRACT

A system for remotely updating software on at least one electronic device connected to a network. The electronic devices have a non-volatile re-writable storage unit divided into at least two partitions, one of which will contain core firmware and the other of which will contain auxiliary software. When an update is received at the device, the updated core firmware is written to overwrite the partition in the re-writable storage unit which contained the auxiliary software. When this is completed and verified, the previous version of the core firmware stored in the storage unit is disabled from execution by the device. Next, the updated auxiliary software is written to overwrite the old version of the core firmware. When this write is complete, the device determines a suitable time for it to be rebooted to execute the updated software. In another embodiment, the present core firmware in the device is copied from the partition it is in to the other partition, overwriting the auxiliary software stored there. The new core firmware received to update the device is overwritten into the first partition, the old copied core firmware being present in case of an upgrade failure, and upon a successful update of the first partition, the auxiliary software is written to the second partition, overwriting the copied old core firmware. In this manner, the position of the core firmware and auxiliary software within the partitions is preserved during normal operation of the device.
Software Update Method, Apparatus and System

FIELD OF THE INVENTION

The present invention relates generally to a method, apparatus and system for updating software in a remotely located electronic device. More specifically, the present invention relates to a method, system and apparatus for updating software in remotely located electronic devices connected to a network where the devices can recover from an update failure and complete the update through the network.

BACKGROUND OF THE INVENTION

Many common electronic devices include re-writable memory which allows software and/or data in the device to be changed and/or replaced. Presently, such re-writable memory is typically Flash memory, or equivalents, although other types of memory or storage can be employed. Flash memory is a type of solid-state memory that is nonvolatile, in that it does not lose its data when the power is turned off, and yet is re-writable to contain different data. Flash memory is popular because it is compact, reasonably durable, fast and re-writable. For example, cellular phones use Flash memory to hold software implementing telephone features, speed dial numbers, ringing tones, firmware updates, etc. So, as new features or bug fixes are implemented, the firmware in the electronic device can be updated.

However Flash memory, and the like, are not without disadvantages. One disadvantage is that Flash memory is relatively expensive. In devices where the manufacturer needs to keep the consumer costs low, the devices must be engineered to minimize the amount of Flash memory required.

While the ability to update firmware or other software or data in a deployed device is clearly desirable, upgrading Flash memory in an electronic device is not always simple. For example, when most cellular phones require a firmware or other software update, the cell phone must be taken into a local service center where the software can be updated by attaching the cell phone through a wired data link to an update station holding the updated software. If there is a problem in transferring the new software from the update session, resulting in the device being placed into an unknown or inoperable state, the device can be reset and the new software can simply be transferred again until the transfer completes and the device functions properly.

However, this is seldom an attractive option as it requires the active participation and/or cooperation of the user who must visit the service center. With some devices, such as a wireless
local loop subscriber station, taking the subscriber station into a service center means that, in addition to the inconvenience of the trip to the service center, the residence where the subscriber station is normally located is temporarily without telephone or data services.

Prior attempts have been made to provide updates to non-volatile re-writable memory in devices through the network to which they are attached. For example, a cell phone can receive software updates for its Flash memory through the wireless network which services it. However, problems exist with such techniques in that, should the transmission fail or be corrupted for any reason, the device may be left in an unknown or inoperative state. In such a case, unlike the example above with the service center, attempts to resend the update software could be impossible and the user would be left with an inoperable device until it was returned to a service center.

One prior solution to this problem has been to provide separate banks of re-writable memory. U.S. Patent 6,023,620 to Hansson teaches a cell phone system wherein half the re-writable memory is a bank used to maintain the current version of the software and while the updated software is downloaded into a bank comprising the other half. Once the cell phone determines that the transfer has been successful, by verifying a CRC, etc., the cell phone switches to using the updated bank of memory and the bank containing the old software is available to receive the next update. A similar prior art solution is taught in U.S. Patent 6,275,931 to Narayanaswamy et al. These arrangements prevent a non-recoverable error from occurring during a update, but require twice as much of the expensive re-writable memory. Further, the entire contents of the bank of memory must be downloaded through the network. Thus, even if only a few bytes of data need to be transmitted, the full contents of the bank must be sent through the network using bandwidth and requiring transmission time which could otherwise be usefully used.

Also, the prior art update methods discussed above typically require the cooperation and/or participation of the user of the device, either requiring them to visit the service center or requiring them to accept and/or initiate the transfer of update data through the network. A crucial update, such as one which will improve stability or capacity in the network for the network operator, may be refused or otherwise delayed by some users due to the inconvenience to them. Additionally, with the prior art methods the updating of each device in the network must be performed separately. Thus, the update software must be transmitted each time a device is updated, utilizing a large amount of bandwidth (which is often a scarce and/or valuable resource, especially in a wireless network). Thus, upgrading the devices in an entire network can take an undesirably large amount of time and resources.
It is desired to have a method and system of reliably updating software and/or data in non-volatile re-writable memory of devices connected to a network which does not require double the amount of non-volatile re-writable memory in the devices and which can be achieved automatically and/or in parallel on multiple devices.

5 SUMMARY OF THE INVENTION

It is an object of the present invention to provide a novel method, system and apparatus for updating, through a network, software in electronic devices which obviates or mitigates at least one of the above-identified disadvantages of the prior art.

According to a first aspect of the present invention, there is provided a system for remotely updating at least one electronic device across a communications link, where said system comprises:

- an update server, operable to transfer an update to the at least one electronic device across the communications link, the update comprising core firmware and auxiliary software;
- a volatile memory to temporarily store the transfer received from the update server;
- a non-volatile re-writable storage unit within said at least one electronic device divided into at least first and second partitions, the first partition storing one of a version of core firmware and auxiliary software and the second of the partitions storing the other of a version of core firmware and auxiliary software; and

an update client executing on the device and operable:

(i) to overwrite the version of the auxiliary software stored in one of the first and second partitions with the received updated core firmware stored in the volatile memory and to verify the success of this write;

(ii) to configure the device to execute the core firmware stored in (i) upon the next reboot of the device;

(iii) to overwrite the version of the core firmware stored in the other of the first and second partition with the received updated auxiliary software store in the volatile memory and to verify the success of this write; and

(iv) to reboot the device to execute the updated core firmware and updated auxiliary software.

According to another aspect of the present invention, there is provided a method of updating software in a plurality of remote devices connected to a network, comprising the steps of:

(i) placing an update onto an update server, the update comprising at least a core firmware update;
(ii) identifying the devices connected to the network to be updated;

(iii) transferring the update from the update server to the identified devices through the network, each identified device verifying the reception of the update, requesting retransmission of and receiving any previously incorrectly received portion of the update;

(iv) writing and verifying the core firmware portion of the received update into a partitioned non-volatile re-writable storage unit, the core firmware portion overwriting a partition containing a previously stored version of software while ensuring that a valid copy of the previous version of the core firmware is always present in the storage unit;

(v) identifying the verified updated core firmware partition as being the valid core firmware to be used by the device and identifying the previous version of the core firmware as being unusable; and

(vi) rebooting the device to load and execute the updated software.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the present invention will now be described, by way of example only, with reference to the attached Figures, in which:

Figure 1 is a diagram of a network permitting upgrading of electronic devices in accordance with an embodiment of the invention;

Figure 2 is a diagram of an update station, in accordance with an embodiment of the invention;

Figure 3 is a diagram of an updateable electronic device, in accordance with an embodiment of the invention, including a memory unit;

Figures 4a, 4b and 4c are diagrams of the memory unit in the electronic device of Figure 3, in accordance with an embodiment of the invention;

Figures 5a, 5b and 5c are diagrams of memory unit in the electronic device of Figure 3, in accordance with another embodiment of the present invention;

Figure 6 is a schematic diagram of the hierarchy of an update system in accordance with an embodiment of the invention; and

Figure 7 is a flowchart of an embodiment of the update process of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to Figure 1, a wireless network, enabling the upgrading software of at least one electronics device, is indicated generally at 20. Network 20 includes at least one update station,
which in this example is a radio base station 24, operable to transmit software updates across a bi-
directional communication link 32. Network 20 also includes at least one electronic device, such as
subscriber stations 28a, 28b, ..., 28n for a voice and data capable wireless local loop. Subscriber
stations 28 can be the customer premises equipment in a wireless local loop for voice and data, or
can be wireless point of sale terminals, or any other electronic devices such as a PDA, laptop
computer, etc. capable of communicating through communication link 32.

The number ‘n’ of subscriber stations serviced by a base station 24 can vary depending upon
the amount of radio bandwidth available and/or the configuration and requirements of the
subscriber stations 28. For the purposes of clarity, references to the electronic device being updated
will be made only to subscriber stations 28, however other electronic devices able to receive
software updates across a communication link are within the scope of the invention. Examples of
such other electronic devices can include PDAs with a modem, cellular phones, cable modems, etc.
and other examples of suitable electronic devices will occur to those of skill in the art.

In a presently preferred embodiment, base station 24 is connected to at least one
telecommunications network, such as a land line-based switched data network 30, a public switched
telephone network 33, etc. by an appropriate gateway and one or more backhauls 34. Backhaul
connections 34 can be links such as T1, T3, E1, E3, OC3 or other suitable land line link, or can be a
satellite or other radio or microwave channel link or any other link suitable for operation as a
backhaul as will occur to those of skill in the art. Base station 24 can also include, or be connected
to by a backhaul or other means, a software update server 36 which contains software loads for
subscriber stations 28. Base station 24 is also connected to a subscriber database, located in update
server 36 or in a separate database server (not shown) provided for this purpose elsewhere, such as
in a centralized networks operation center (discussed below), which holds records of the current
software loads of subscriber stations 28.

In the illustrated embodiment of wireless network 20, communications link 32 is established
between base station 24 and each subscriber station 28 via radio, although over physical means of
connection, including wireline, infrared and ultrasonic means can be employed. Communications
link 32 can carry voice and data information between base station 24 and respective subscriber
stations 28a, 28b ... 28n as needed. Communications link 32 can be implemented with a variety of
multiplexing techniques, including TDMA, FDMA, CDMA, OFDM or hybrid systems such as
GSM, etc. Furthermore, communications link 32 can be arranged into different channels carrying
different data types such as voice communications or data transmissions and/or being employed for
various purposes such as end user communications or control of network 20. In a present embodiment, data transmitted over communications link 32 is transmitted as packets, which can be of any suitable type, but IP (Internet Protocol) packets are presently employed.

Figure 2 shows an example of base station 24 in greater detail. Base station 24 comprises an antenna 40, or antennas, for receiving and transmitting radio-communications over communications link 32. In turn, antenna 40 is connected to a radio 44 and a modem 48. Modem 48 is connected to a microprocessor-router assembly 52 such as a Pentium III™ processor system manufactured by Intel and associated devices. It will be understood that microprocessor-router assembly 52 can include multiple microprocessors, as desired and/or that the router can be provided as a separate unit, if desired. The router implemented within microprocessor-router assembly 52 is connected to a backhaul 34 in any suitable manner to connect base station 24 to an appropriate telecommunications network 30, 33. Base station 24 can also be connected directly, or through a backhaul 34, to an update server 36, as mentioned above.

Referring now to Figure 3, an example of a subscriber station 28 is shown in greater detail. Subscriber station 28 comprises an antenna 60, or antennas, for receiving and transmitting radio-communications over communications link 32. In turn, antenna 60 is connected to a radio 64 and a modem 68, which in turn is connected to a microprocessor-assembly 72 and memory unit 78. Microprocessor-assembly 72 can include, for example, a StrongARM processor manufactured by Intel, and performs a variety of functions, including implementing A/D-D/A conversion, filters, encoders, decoders, data compressors, de-compressors and/or packet disassembly. As seen in Figure 3, microprocessor-assembly 72 interconnects modem 68 and one or more ports 76, for connecting subscriber station 28 to data devices and telephony devices. An example of a telephony device would be a telephone, or the like, which is operable to receive voice received over communications link 32. Examples of a data devices include personal computers, personal digital assistants or the like which are operable to use data received over communications link 32. Accordingly, microprocessor-assembly 72 is operable to process data between ports 76 and modem 68.

Memory unit 78 consists of two principle components, the first being a volatile random access memory, RAM 82, such as Dynamic RAM (DRAM) or synchronous DRAM (SDRAM), that is used for running software for the operations of subscriber station 28. The second is a non-volatile, re-writable storage unit, RSU 86, which can be Flash memory etc., that is used to store a copy of software that is not lost when subscriber station 28 is without power. Memory unit 78 is
-7-

Operable to contain and run all the necessary software for the proper and desired functioning of subscriber station 28. Necessary software can include, but is not limited to, boot software, file and operating systems, software applications, radio resource management software, device drivers, and data files.

Microprocessor-assembly 72 is operable to copy software and data from RSU 86 to RAM memory 82 as needed. As known to those of skill in the art, RAM memory 82 is typically faster than Flash memory or other types of re-writable memory used in RSU 86 and thus software is typically decompressed, if stored in a compressed form to reduce the size requirements of RSU 86, and copied from RSU 86 to RAM memory 82 before execution, although this is not always required. In some circumstances, which will be apparent to those of skill in the art, software can be directly executed from RSU 86. In other circumstances, the software in RSU 86 may not be compressed in RSU 86, thus not requiring decompression before being copied to RAM memory 82.

Microprocessor-assembly 72 is also operable to write software and data from RAM memory 82 to RSU 86 as necessary.

In a present embodiment, RAM memory 82 consists of 32 Mbytes of SDRAM. However, the quantity and type of RAM memory is not particularly limited and other quantities and types of RAM memory, such as DDR RAM or Rambus RAM or others are also within the scope of the invention. Similarly, in a present embodiment, RSU 86 consists of Flash memory with 8 Mbytes of storage space, the lesser amount of Flash memory being, at least in part, due to the significant expense of such Flash memory compared to RAM memory.

Flash memory is typically organized into rows and columns of 'blocks', which can hold a number of bits of information. A block is the smallest amount of information that can be erased from the memory at a time and a typical block could hold 256 Kbytes of information, the memory having to be erased before it can written to. Many Flash memory devices allow writing of data to erased blocks in data words (16 bits) and if two devices are arranged in parallel, as with a present embodiment of the invention, data can be written as double words (32 bits). There are several standards for Flash memory, but all work in generally the same way and, in addition, other types of non-volatile and re-writable memory with varying amounts of storage space are within the scope of the invention. The term "overwritten" as used herein, is intended to comprise the necessary operations for placing new contents into a non-volatile memory, over previous contents and can include the steps of first erasing the memory before writing new contents to it. Other examples of non-volatile and re-writable memory include conventional IDE and SCSI hard drives, optical
memory storage devices, EPROMS, etc. Other types of non-volatile and re-writable memory will occur to those of skill in the art.

Referring now to Figure 4a, RAM 82 and RSU 86 in memory unit 78 are shown in more detail. In the present embodiment, the Flash memory of RSU 86 is divided into logical partitions. As known to those of skill in the art, logical partitions are treated for most purposes as if they were separate discrete memory devices. For example, a hard drive with two partitions will appear to the computer as two separate hard drives. Also known to those of skill in the art, logical partitions typically can be added, removed, or resized in order to provide flexibility within the storage device.

In a presently preferred embodiment, RSU 86 is divided into three partitions, namely partition 104, partition 108 and partition 112.

Partition 104 contains the boot-loading firmware for subscriber station 28. This boot-loading firmware is the first piece of software accessed at startup (boot) by micro-processor-assembly 72 and directs microprocessor assembly 72 to decompress and copy the core firmware (discussed below) into RAM memory 82 and to commence executing it. Partition 104 is at a fixed memory address (typically the first memory block of Flash memory), so that it is always the first set of instructions accessed upon boot-up by microprocessor-assembly 72. In a presently preferred embodiment, partition 104 is relatively small and contains two hundred and fifty-six Kbytes of storage space.

In the presently preferred embodiment of the invention, the boot-loading firmware is provided in partition 104 to avoid the necessity of providing an additional ROM or other non-volatile storage package in subscriber station 28. However, as will be apparent to those of skill in the art, if such ROM or other non-volatile storage package is provided in a subscriber station 28, partition 104 can be placed therein and omitted from RSU 86 which would then be arranged into two partitions 108 and 112. Of course, by providing partition 104 in RSU 86, it is possible to update the boot-loading firmware, if desired, by overwriting it with a new version received through network 20 in a manner similar to that disclosed below.

Partition 108 contains the software required to give at least minimum functionality to the device, in this case subscriber station 28, and this software is referred to herein as core firmware. The core firmware is responsible for providing the basic operations of subscriber station 28, and includes the operating/file system and necessary device drivers. These basic operations can include memory management, task handling, managing files, input/output, etc. and at least the minimum amount of functionality required to allow subscriber station 28 to communicate with base station 24.
(but not necessarily enough functionality to provide any end user services).

In a presently preferred embodiment, the operating system included in the core firmware includes the Linux kernel, version 2.4, as an operating and file system and the boot-loading software in partition 104 is a Linux boot loader.

In the present embodiment, the Linux kernel and other operating system components are compressed, using CRAMFS, to reduce its overall storage requirements in RSU 86. CRAMFS is a well known Linux tool, and documentation is available from a variety of sources and the use of CRAMFS will be apparent to those of skill in the art.

At start up, the boot loader will commence reading sequentially from the start of RSU 86 to locate the superblock which is used in CRAMFS to indicate the start of a valid kernel copy, which is also the start of core firmware partition 108. When the loader locates the start of the core firmware partition 108, the offset of this start and the size of partition 108 are passed as boot parameters to the Linux kernel and the kernel copy is then read from RSU 86, decompressed and copied into RAM 82 as the operating system starts.

While Linux is presently preferred, other operating systems and operating system versions are within the scope of the invention. The location of partition 108 within RSU 86 is not particularly limited and can occupy any continuous set of block addresses after partition 104 (if present) as the boot loader searches the contents of RSU 86 until the start of a valid core firmware partition 108 is located.

Partition 112 contains the balance of software stored in the device, and this software is hereinafter referred to as the auxiliary software. The auxiliary software can include optional device drivers, user applications, system software applications and data files. The auxiliary software is not particularly limited and can include such software and/or end user applications as telephone call processing software, voice and/or audio codecs, optional device drivers, software filters, firewalls, utilities, help files, subscriber data files, digital media files, and other such applications and data files as will occur to those of skill in the art. Generally, the auxiliary software stored in partition 112 is not required for subscriber station 28 to communicate with base station 24, although the auxiliary software stored in partition 112 may be required to enable subscriber station 28 to be enabled to make or receive voice calls, end user data connections (such as http browser sessions) or other end user functions. The location of partition 112 within RSU 86 is not particularly limited and need only occupy a contiguous set of blocks.

As shown in Figure 4a, partition 108 is smaller in size than partition 112, although the
difference in size can be relatively small (as little as one memory block). In a present embodiment, partition 108 occupies up to 3.75 Mbytes of storage space, which is a little less than half the 8 Mbyte size of RSU 86 and partition 112 is up to 4 Mbyte in size.

When it is desired to update the core firmware in partition 108, the replacement core firmware is transferred from an update server 36, as described in more detail below, over communications link 32 to subscriber station 28. In a present embodiment of the invention, the core firmware is received and stored in RAM memory 82 at subscriber station 28 until the entire transfer of the core firmware and the auxiliary software to subscriber station 28 has been completed, although it is also contemplated that, if desired, the core firmware could be transferred and installed before transferring the auxiliary software. Depending upon the size of the update and/or the size of RAM memory 82, it may be required to terminate any processes running on subscriber station 28 which require large amounts of RAM memory 82. As will be discussed below, the ability to terminate such processes is one of the status criteria considered before it is decided to update a subscriber station 28.

It is contemplated that a variety of techniques can be used to transfer the core firmware and/or auxiliary software from update server 36 to a subscriber station 28, such as transmission of the software in packets via UDP/IP or TCP/IP. As communications link 32 and/or the physical media used to transfer the software can be subject to faults and/or errors, the correctness of the received transfer of the software is verified before use. The particular method used to verify this correctness is not particularly limited and checksums, CRCs, digital signatures, etc. can be employed on all, or portions, of the transfer, as will be apparent to those of skill in the art. If the received contents are not correct, and contain one or more errors, the software or appropriate portions of it, can be retransmitted from update server 36 to the subscriber station 28 until a complete correct copy is received at subscriber station 28.

Once a complete correct copy of the updated/replacement core firmware, i.e. - the "new" core firmware, is received at subscriber station 28 and stored in RAM 82, the update process continues by overwriting the new core firmware onto the auxiliary software in partition 112 in RSU 86. In order to perform this overwriting, any remaining processes which were executing on subscriber station 28 and which require read access to the auxiliary software in partition 112 are terminated. Once these processes, if any, are terminated the new core firmware is copied from RAM 82, overwriting part of partition 112, to form a new core partition 108’ as shown in Figure 4b.
-11-

It is also contemplated that, to reduce the memory required in RAM memory 82, the updated core firmware can be written in increments as it is received, for example in update blocks of 256 Kbytes, provided that the received update can be verified as having been properly received before writing. In such a case, as a given amount of update data is received at subscriber station 28, it is temporarily stored and verified in RAM 82 and then written to RSU 86 while the next received update overwrites the previous received update which had been temporarily stored in RAM 82. In this manner, various applications and/or processes running from RAM 82 may not necessarily have to be terminated, as discussed below, during the update process, at least until the subscriber station 28 is rebooted.

As shown in Figure 4b, the overwriting is commenced at a determined offset from the beginning of partition 112 so that the end of partition 108' coincides with the former end of partition 112. In the example above, where RSU 86 is 8 megabytes in total size and partition 104 is 0.25 megabytes in size, partition 112 is 4 megabytes in size and partition 108 is 3.75 megabytes, the offset at which new partition 108' is overwritten onto partition 112 is 4.25 megabytes from the start of RSU 86, assuming partition 104 is in fact present at the beginning of RSU 86.

After partition 108' is written, its contents are next verified by subscriber station 28 which reads back the contents from partition 108' and compares them to the copy of the new core firmware in RAM 82. If partition 108' is written in smaller portions from RAM 82 as received, the writing of these smaller portions is verified to that stored in RAM 82 before the next received portion overwrites the last portion temporarily stored in RAM 82.

In either case, if the contents read from partition 108' cannot be verified, the writing of partition 108', or the relevant portion of partition 108', is performed again and the verification/rewrite process is repeated until the contents are verified.

When the contents of partition 108' have been verified as having been written correctly, partition 108' is identified to subscriber station 28 as being the most recent core firmware partition 108' and original core partition 108 is then disabled from being executed by subscriber station 28 by being marked "invalid". In the presently preferred embodiment of the invention, wherein the core partitions 108 and 108' include a CRAMFS formatted Linux kernel, etc., the invalid partition 108 is identified by overwriting its superblock. Once the superblock of partition 108 is overwritten, the boot loader on the next reboot of subscriber station 28 will only locate the superblock of partition 108', which is the most recent valid core firmware partition, and subscriber station 28 will boot from partition 108'.
As will be apparent from Figure 4b, this results in invalid partition 108 and a remaining portion of partition 112 no longer containing useful data. Accordingly, auxiliary software partition 112' is created, as shown in Figure 4c, in the memory space of partition 108 and that portion of partition 112. Subscriber station 28 can then copy the auxiliary software from RAM 82 (or download it to RAM 82 from update server 36 if this has not yet been performed) into new auxiliary software partition 112'. Once the writing of the new auxiliary software into partition 112' has been verified, in a manner similar to that described above for the core firmware, subscriber station 28 can commence execution of any desired auxiliary software.

As will be apparent from the above description, a valid copy of the core firmware is always present in RSU 86, even if subscriber station is turned off, looses power, or otherwise requires a reboot, during the update process. In the event that the update process has commenced with a new core firmware partition 108' being written over auxiliary software partition 112 when the subscriber station 28 is turned off before the write of partition 108' has been completed and verified, when the subscriber station 28 is again turned on it will boot from old partition 108, which is still intact, and the absence of a valid auxiliary partition 112 is noted and the entire update process will either restart, or a transfer of valid auxiliary software can be requested from update server 36 and stored in RSU 86 as a restored partition 112 and the update process deferred until later. This latter option will be selected, for example, when network 20 is being heavily used and the capacity to perform an update is not readily available.

Assuming a successful update has been performed, whenever it is desired to again update core firmware in subscriber station 28, a new core firmware partition 108 will overwrite a portion of auxiliary software partition 112' and a new auxiliary software partition 112 will overwrite old core firmware partition 108' and the remaining part of old auxiliary software partition 112' to again obtain the configuration shown in Figure 4a. Each subsequent update of core firmware will result in the overwriting of the existing auxiliary software partition by the new core firmware partition and the overwriting of the old core firmware partition and the remaining portion of the auxiliary software partition with a new auxiliary software partition.

As shown in Figure 5a, if it is desired to have the location of partitions 108 and 112 be constant in RSU 86 during normal operations, the "old" core partition 108 can be copied over the "old" auxiliary partition 112 to form a copied old partition 108''. The writing of this copied old partition 108'' is then verified and, once verified, original partition 108 is marked invalid by overwriting its superblock or by any other suitable means. Should the subscriber station 28 be re-
booted at this point for any reason, the boot loader will locate and use the contents of copied partition 108".

Next, the “new” core firmware is overwritten onto original partition 108 from RAM 82, as shown in Figure 5b, and verified. The superblock of the new core firmware for original partition 108 is not written to original partition 108 until the contents of the remainder of the partition 108 are verified, after which the superblock will be written and verified, and then the superblock of copied partition 108" will be overwritten. In this manner, a reboot or other event requiring loading of the core firmware prior to verification of the write of new partition 108 will instead employ copied partition 108".

Finally, the new auxiliary software is copied from RAM 82 to form auxiliary partition 112, as shown in Figure 5c and this partition is verified before being used and subscriber station 28 is then once again in its normal operating configuration. While this process does result in partitions 108 and 112 always having the same positions in RSU 86, it does require additional write and verification operations to be performed to copy the contents of partition 108 to partition 108",

which adds to the time required to perform the update.

In a present embodiment of the invention, updating of software in subscriber stations 28 is a managed process. This is especially important in a critical network or an “always on” network, such as wireless local loop providing a primary telephone line replacement. Figure 6 shows the management system hierarchy of network 20 which includes a network operations center (NOC) 200, a radio sector manager 204 for each sector in a base station 24 in network 20 and a subscriber station update client 208 for each subscriber station 28 served by network 20.

Network operations center 200 is a centralized facility operated by the operator of network 20 and, in addition to managing the updating of subscriber stations throughout network 20, can also perform other network management functions such as OAM&P, etc. Radio sector managers 204 are preferably located in each base station 24 of network 20 and can be co-located with and/or implemented within update server 36. If a base station 24 is an omni-directional (single sector) base station, only a single radio sector manager 204 will be provide in that base station 24. It is contemplated that, more commonly, a base station 24 will employ non-omni-directional (beam-forming) antennas which provide for increased densities of subscriber stations 28 that can be served by a base station 24. For example, if sixty degree beam-forming antennas are employed, a base station 24 can be configured into six different radio sectors, each sector serving a subset of the total number of subscriber stations 28 served by that base station 24. In such a case, six radio sector
managers 204 would be provided at base station 24. Finally, each subscriber station 28, as part of its core firmware, includes an update client 208 which executes on the subscriber station 28.

In Figure 6, the radio sector managers 204 illustrated in dashed line are intended to represent a plurality of such radio sector managers 204 and their associated update clients as NOC 200 can serve several hundreds of radio sector managers 204 and, through them, several thousand or more subscriber stations 28 through their update clients 208.

When an update to the auxiliary software or core firmware of subscriber stations 28 has been created and the operator of network 20 wishes to implement it, the network operations center 200 will determine the present version of the software loaded into each subscriber station 28 being served by network 20. This determination can be performed in a variety of manners, as will be apparent to those of skill in the art. In a present embodiment of the invention, the update client 208 of each subscriber station 28 reports to the radio sector manager 204 serving it the present version of the software loaded into the subscriber station 28 each time the subscriber stations 28 is turned on and/or after each time an update is performed. The radio sector managers 204 forward this information to the network operations center 200 where it is stored in a suitable database. As will be apparent to those of skill in the art, a variety of other techniques can be employed to determine the present software load of each subscriber station 28, including polling of the update clients 208 at appropriate intervals, etc.

Once the present software load of each subscriber station in network 20, or in a subset of interest of the subscriber stations 28 in network 20 (for example, the network operator can be interested in only updating those subscriber stations 28 in a particular city), has been determined, network operations center 200 determines the subscriber stations 28 that should be updated. In most circumstances, it will be desired to update all subscriber stations 28 in network 20, but it is also contemplated that it can be desired to update selected subsets of subscriber stations 28, or even individual subscriber stations 28.

The network operations center 200 ensures that the desired updated software is available on the update server, or servers, 36 which serve the radio sector managers 204 with subscriber stations 28 to be updated, transferring the updated software to the update servers 36 as necessary. Next, network operations center 200 instructs the radio section managers 204 with subscriber stations 28 to be updated of the identity of those subscriber stations 28 and instructs them to initiate the updates.

Once a radio sector manager 204 receives update instructions from the network operations
center 200, it determines the activity level and or status of the subscriber stations 28 it manages that are to be updated. Ideally, updates are only performed when the capacity they require on communications link 32 is not otherwise required and/or subscriber stations 28 are not executing end user processes that will have to be interrupted for the update process. Accordingly, radio sector manager 204 first determines that it can spare and/or has capacity on communications link 32 to transmit the update. It is contemplated that typically, such updates will be performed late at night or early in the morning when communications link 32 is underutilized by end users. However, it is also contemplated that an essential update, such as one required to stabilize operation of network 20 or to provide enhanced security, etc. can be assigned an update priority by network operations center 200 which instructs radio sector managers 204 to perform the update as soon as possible, terminating or interrupting other uses of the capacity of communications link 32 and/or interrupting, degrading or suspending end user activities at the subscriber stations 28 to be updated.

Once it has been determined by radio sector manager 204 that it has capacity on communications link 32 to transmit the update to subscriber stations 28, or in the event of a priority update that it has made capacity, it queries the update client 208 in each of those subscriber stations 28 it is to update to determine the status of those subscriber stations 28. This status indicates the level and/or type of activity presently taking place on that subscriber station 28.

For example, a subscriber station 28 can indicate that it has been idle (no end user activity) for ten minutes, or that it is currently conducting an http session for an end user, etc. As the transferred download of the update is typically first stored in RAM memory 82, if the necessary amount of memory (this amount can be a predefined amount for all updates, or can be provided by the radio sector manager 204 in its status query to the update clients 208) selected to store the download is not available in RAM memory 82, the subscriber station 28 will include this information in its status report to radio sector manager 204. Alternatively, the update client 208 in subscriber station 23 can determine if it can terminate processes and/or applications using RAM memory 82 to free memory space and will do so, if possible, before sending the status response to radio sector manager 204.

Radio sector manager 204 examines the status responses received from each subscriber station 28 to be updated and determines which subscriber stations 28 can be included in the update at this time. The update client 208 of each of these subscriber stations 28 then receives an update information transmission from radio sector manager 204 informing the subscriber stations 28 that they are to undergo an update and indicating the channel of communications link 32 that the update
is to be transmitted on.

Radio sector manager 204 then initiates a multicast transmission of the update from update server 36 to the subscriber stations 28 which have been instructed to process the update. In a present embodiment of the invention, the update is transmitted via UDP over IP as a multicast transmission and each transmitted packet includes a CRC checksum to verify correct receipt and a sequence number or other unique identifier of the packet so that each subscriber station 28 can determine if it has correctly received all necessary packets. In the event that one or more packets have been received incorrectly or missed altogether by a subscriber station 28, such subscriber stations 28 can send a retransmit request to radio sector manager 204 while the transmission is in progress and the requested packet can be retransmitted. Preferably, this retransmission is performed over the multicast channel and is available to all subscriber stations 28 being updated (in case more than one subscriber station 28 requires the retransmission of the same packet) although it is also contemplated that such retransmissions can be made to a subscriber station over a dedicated channel on communications link 32 established with the subscriber station 28 by radio sector manager 204 for that purpose.

In the presently preferred embodiment, once the entire update has been downloaded and verified, the update client in the subscriber station must determine when to perform the update of RSU 86. As the update will require a reboot (restart) of the subscriber station 28, update client 208 attempts to select a time for the update when minimal, if any, service interruption to the end users will occur. Again, it is contemplated that such updates will typically be performed late at night or early in the morning or at any other time when the likelihood of end user use of the subscriber station 28 is low.

However, the update client 208 in a subscriber station 28 can also make an intelligent decision on when to perform the update by determining what end user activities are occurring and/or the time since the last end user activity. For example, a subscriber station 28 which last made an end user voice or data connection more than twenty minutes ago, can make a reasonable assumption that it will be unused by an end user for the next several minutes while the update is performed.

Again, it is also contemplated that some updates will have sufficient importance to the operations of network 20 that they will have a priority assigned to them which enables the update client 208 to terminate end user activities on the subscriber station in order to ensure the update is performed.
When the update client 208 has determined that the update can be performed, the process discussed above is performed overwriting auxiliary software partition 112 with new firmware partition 108’, or copied partition 108, etc.

Once a successful update has been performed and the subscriber station 28 has rebooted and is executing the new core firmware and/or auxiliary software, an update status message is sent to radio sector manager 204 informing it that the update has been completed and verifying the version numbers of the software being executed by the subscriber station 28. Radio sector manager 204 then updates its records of the subscriber stations 28 which have been updated and those which still require updating.

Radio sector managers 204 then repeat the process, through one or more iterations, for the remaining subscriber stations 28 to be updated. It is contemplated that an update will not be commenced until a pre-selected proportion of the subscriber stations 28 to be updated in a radio sector are available for the update. For example, it can be selected that radio sector manager 204 will not commence the update unless at least 50% of the subscriber stations 28 to be updated in its sector are available for updating. If this threshold is not reached, the update will be delayed until the threshold can be met or until the network operations center lowers the threshold (eg. to 35%) or raises the priority of the update so that subscriber stations 28 are forced to implement the update.

Network operations center 200 is advised of the status of the update by the radio sector managers 204. Thus, network operations center 200 can determine the number of subscriber stations 28 that have been updated and the number that remain to be updated. If network operations center 200 observes that the update is being performed for fewer subscriber stations 28 than it desires, it can apply priority to the update to force subscriber stations 28 to ready themselves for the update, etc.

While it is contemplated that in most circumstances the core firmware and auxiliary software updates will be transmitted as one update, it is also contemplated that in some circumstances it may be desired to transmit the core firmware update first and, after the subscriber stations 28 have successfully installed that update, the auxiliary software update will be transmitted. It is also contemplated that in some circumstances only the auxiliary software be desired to be updated. In such a case, the core firmware is not updated and the updated auxiliary software is written over the existing auxiliary software partition 112.

Figure 7 shows a flowchart of an embodiment of the update process described above. When the network operations center 200 wishes to update devices in network 20, at step 300 the devices
which require the update to be installed are determined. At step 304 the update is transferred or otherwise made available to the update server 36, or servers, from which the update will be transferred to the devices to be updated.

At step 308, each radio sector manager 204 determines which of the devices it serves are available for updating. At step 312, the radio sector manager 204 instructs those devices that they will be updated and provides the details of the update communication, such as the multicast parameters, etc.

At step 316, the update is transmitted to the devices being updated. Each intended device verifies the reception of the transmitted update, either when the transmission is completed or as the transmission is occurring, and at step 320, devices which have received a portion of the transmission which is in error or have missed reception of a portion of the transmission advise the radio sector manager 204 of this fact and the radio sector manager 204 which will cause those portions to be retransmitted.

At step 324, once the devices have a correct copy of the update, the devices determine the appropriate time to perform the update. As mentioned above, this determination can be trivial (i.e. – perform the update regardless of the status of the device) or can be made depending upon the status of the device, including factors such as current processes executing on the device, the time since an end user process was executed, etc.

When the update and rebooting of the device is achieved, at step 328 the device will notify the radio sector manager 204 managing it that it has been updated and can provide the details of its present software load.

At step 332, each radio sector manager 204 determines if any devices it manages remain to be updated. If such devices do remain, steps 308 through 328 are performed again, as necessary. If no such devices remain, the update process completes at step 336.

While the embodiments discussed herein are directed specific implementations of the invention, it will be understood that combinations, sub-sets and variations of the embodiments are within the scope of the invention.

The above-described embodiments of the invention are intended to be examples of the present invention and alterations and modifications may be effected thereto, by those of skill in the art, without departing from the scope of the invention which is defined solely by the claims appended hereto.
WE CLAIM:

1. A system for remotely updating at least one electronic device across a communications link, where said system comprises:

   an update server, operable to transfer an update to the at least one electronic device across the communications link, the update comprising core firmware and auxiliary software;

   a volatile memory to temporarily store the transfer received from the update server;

   a non-volatile re-writable storage unit within said at least one electronic device divided into at least first and second partitions, the first partition storing one of a version of core firmware and auxiliary software and the second of the partitions storing the other of a version of core firmware and auxiliary software; and

   an update client executing on the device and operable:

      (i) to overwrite the version of the auxiliary software stored in one of the first and second partitions with the received updated core firmware stored in the volatile memory and to verify the success of this write;

      (ii) to configure the device to execute the core firmware stored in (i) upon the next reboot of the device;

      (iii) to overwrite the version of the core firmware stored in the other of the first and second partition with the received updated auxiliary software store in the volatile memory and to verify the success of this write; and

      (iv) to reboot the device to execute the updated core firmware and updated auxiliary software.

2. The system as claimed in claim 1 wherein said update client is further operable to inform the update server as to whether the device is available for updating at a given time, the update server being responsive to the information received from the update server to delay updates to the device when the device is not available for updating.

3. The system as claimed in claim 2 wherein the update server can prioritize an update such that the update client will make a device available for the update which would otherwise be unavailable.

4. A method of updating software in a plurality of remote devices connected to a network,
comprising the steps of:

(i) placing an update onto an update server, the update comprising at least a core firmware update;

(ii) identifying the devices connected to the network to be updated;

(iii) transferring the update from the update server to the identified devices through the network, each identified device verifying the reception of the update, requesting retransmission of and receiving any previously incorrectly received portion of the update;

(iv) writing and verifying the core firmware portion of the received update into a partitioned non-volatile re-writable storage unit, the core firmware portion overwriting a partition containing a previously stored version of software while ensuring that a valid copy of the previous version of the core firmware is always present in the storage unit;

(v) identifying the verified updated core firmware partition as being the valid core firmware to be used by the device and identifying the previous version of the core firmware as being unusable; and

(vi) rebooting the device to load and execute the updated software.

5. The method of claim 4 wherein the previous version of the core firmware is copied to overwrite a partition of the storage unit which contained auxiliary software and the updated version of the core firmware is written over the partition which originally contained the previous version of the core firmware.

6. The method of claim 5 wherein the update further includes updated auxiliary software and the auxiliary software is received and verified by the device and wherein, between steps (v) and (vi), the partition containing the unusable previous version of the core firmware is overwritten with the auxiliary software update.

7. The method of claim 4 wherein the update further includes updated auxiliary software and the auxiliary software is received and verified by the device and wherein, between steps (v) and (vi), the partition containing the unusable previous version of the core firmware is overwritten with the auxiliary software update.

8. The method of claim 4 wherein step (ii) also comprises the device informing the network
whether or not it is available to be updated.
300 Determine devices to be updated.

304 Put update on update server for devices to be updated.

308 Determine within radio sectors, which devices are available for update.

312 Instruct available devices to perform update.

316 Commence transmission of update to devices.

320 Retransmit any unreceived portions of update.

324 Upon receipt of update, determine appropriate time to perform update and reboot.

328 When update completed, advise of completion.

332 Any remaining devices to be updated?

Y

N

336 Update complete.

Fig. 7