SET-TOP BOXES

Digital set-top boxes (STBs) are electronic devices used to connect a consumer's television set to a broadcast entertainment network, commonly a cable network or a satellite network. The original purpose of an STB was to tune to a specific broadcast frequency (channel or band), then to convert the encoded broadcast signal into a form suitable for use with a normal channel (usually channel 3 or 4) on a television set. During the 1990s, STBs evolved from these simple converter boxes into much more sophisticated consumer electronic devices capable of providing supplementary interactive television (ITV) functions. As STB features have evolved, the way STBs are constructed has also evolved—from hard-wired logic implementations to contemporary implementations using multiple processors, distributed software, and leading-edge data network technology. This article explains this evolution from the original fixed-function converter to today's set-top computers. After providing a background for STBs in this section, the next section introduces interactive TV as a driving force behind the evolution of STBs. The remaining sections describe contemporary STB architectures and STB software.

BASIC FUNCTIONALITY

A normal TV station broadcasts its programs on a preassigned channel. A consumer can receive the broadcast by selecting the corresponding channel on the TV set, causing it to receive the broadcast signal on the same band over which it is being broadcast. Cable TV companies broadcast several different bands (whereas a TV station only broadcasts on a single band). Rather than broadcasting the signal through the air, cable technology uses fiber optic or coaxial cable. The cable is capable of simultaneously carrying multiple bands—usually analog cable TV cable carries about 50 TV channels. Each subscriber to the cable TV company has a coaxial cable connection from the cable company's local broadcast station (called the cable *headend*) to the subscriber's location. The STB is located at the subscriber's location, and connects to the coaxial cable from the cable company. These cable networks are *one-way networks,* meaning that information can be sent from the headend to an STB, but not from an STB to the headend. A single headend can simultaneously transmit information to many STBs, that is, the headend *broadcasts* information to the STBs. The essential function of a STB is to translate the analog signal broadcast by the headend into one that can be received by a normal TV set on channel 3 or 4.

Signal conversion once required that subscribers obtain an STB if they wanted to receive a cable signal on a channel higher than the usual broadcast channels. TV manufacturers eventually began to design integral tuners so that they could perform the required band translation, essentially eliminating this requirement.

Cable companies may also offer different levels of service to subscribers. If subscribers contract for basic service, then they are authorized to receive only a subset of all the channels broadcast by the headend. If a subscriber contracts for premium service, then the subscriber is authorized to receive the premium channels and all normal channels. The cable broadcast technology may transmit all channels on the coaxial cable to all subscribers; the basic channels are usually broadcast as normal signals, but premium channels and programs are encrypted prior to broadcasting them. A "cable-ready" TV set can be used to receive basic service, but an STB is required to receive premium services. When a subscriber contracts for any premium service, the STB assures the service supplier that the subscriber has contracted for the premium service before it is delivered to the TV set. There are a variety of techniques for distributing premium services, though the most widely used approach is to incorporate a decryption mechanism in the STB. (In some cases, the decryption mechanism is installed in a network interface.) Such a mechanism is called a *conditional access* mechanism, since it allows the cable company to select an access policy based on contractual agreements between the subscriber and the supplier. The second basic function of the STB is to decrypt encrypted broadcast signals using a secure conditional access mechanism.

The conditional access mechanism must be difficult to copy or simulate, since it is the only mechanism to enforce subscriber authentication for premium services. Conditional access mechanisms are a critical component of STBs. In contemporary STBs, the conditional access mechanism is implemented in hardware logic, or in a secure microprocessor that is separate from other microprocessors in the STB.

EMBEDDED SYSTEMS: THE MOVE TOWARD SOFTWARE

In the late 1970s, it became cost-effective to implement various functions in electronic devices using *stored logic,* that is, functions previously implemented in hardware logic could be implemented at lower cost—and with more flexibility—by embedding a microcomputer chip controlled by software in the device. In the computer domain, embedded microprocessors are commonly used to implement disk controllers, serial controllers, and other controllers. The trend toward embedded systems stimulated microcomputer technology, and has become a fundamental implementation technique for a broad spectrum of embedded applications ranging from gasoline pumps to microwave ovens. This embedded systems approach was also cost-effective for implementing STBs; by the early 1990s computer-based STBs were commonplace. STB technology first became a computer-oriented technology by virtue of cost-effective implementation, though the evolution in the network also further encouraged the use of computer technology.

THE NETWORK

Today, cable companies broadcast information using traditional analog signaling technology in conjunction with newer digital signaling technology. Satellite systems use digital signaling technology exclusively. Digital signaling technology is more efficient than analog technology, allowing the headend to broadcast more simultaneously available channels (over 100) than is possible with analog technology. Part of the increased capacity is due to more efficient use of the signaling mechanism, and the remainder of the efficiency comes from the ability to *compress* digital signals before broadcasting them. Digital signals can also have higher fidelity than analog signals. It is also possible to enhance a digital signal—giving better clarity and resolution, special effects, and so forth. Because of these features, and because the public demands it, broadcast networks are rapidly evolving to digital signaling.

However, there are already hundreds of millions of TV sets in subscribers' homes, and all of them have been built to receive only analog signals; an STB must be used with a cable or satellite feed to convert the digital signal into an analog signal that can be received on channel 3 or 4 in the TV set. Also note that conditional access is still required in digital signaling.

The Moving Picture Experts Group (MPEG) introduced a network protocol for broadcasting audio and video streams on a digital network. The second version, MPEG-2, is widely used in contemporary digital networks, including cable and satellite TV systems. The MPEG-2 protocol addresses two significant functions: (1) It compresses the audio/video signal into a compact digital representation, and (2) it establishes a network protocol for transmitting information over a subcommunication network—cable, TCP/IP, or any other data network. After the headend encodes the audio/video into a stream of MPEG-2 packets, it broadcasts them to the STBs; the STB receives the MPEG-2 packet stream, decodes each packet, optionally decrypts the information (if it was encrypted by the headend), then converts the stream into an analog signal suitable for reception on channel 3 of a conventional TV set.

Digital signaling enables one-way entertainment networks to be used for very general types of data transmission—not just digitally encoded audio and video streams. Today, digital cable TV networks are also used to broadcast a spectrum of data, including newspaper articles, TV program guides, stock prices, and even commonly used web pages. These *push technologies* predict the type of information that will be needed at an STB, then they broadcast that information over a data channel. Each STB can receive the desired data whenever they are broadcast by the headend. (Analog signaling can also be used for push technology. In this approach, the headend transmits data at the same time the normal audio/video streams are being broadcast. There are several techniques for accomplishing this, but it is sufficient to note that during the time a TV display does a horizontal retrace after it draws a line on the screen, the network is available for data transmission. In the United States, close captioning uses this technique.)

One-way networks can be exploited for a broad class of application domains, by predicting the nature of information that will be attractive to subscribers, then scheduling the broadcast of that information over data bands. However, two-way networks provide a more general mechanism for information distribution, since they allow the data receiver to send requests for specific information to the headend source. It is natural for the broadcast networks to evolve to two-way network technology. The cable industry is now distributing various types of cable modems that allow the normal cable network to be used for two-way transmission. (Two-way transmission applies only to cable networks, not satellite networks.) With a cable modem, a conventional computer can be attached to the two-way network to send and receive information. The resulting network is an *asymmetric* network: information can be transmitted from the headend to the STB at a very high rate, but information transmitted *upstream* (on a back channel) from an STB to the headend is transmitted at a much lower rate. The STB is an essential component in two-way cable networks, since it takes over the role of the subscriber's communication mechanism for interaction between the subscriber and the headend services.

INTERACTIVE TELEVISION

Asymmetric two-way network communication makes it possible to transmit information from the subscriber to the headend facility, that is, the configuration supports interactive television (ITV). In an ITV configuration, the subscriber can issue commands to the headend facility to customize the type of information to be broadcast on demand (or "nearly on demand"). For example, in ITV the subscriber can place an order for a particular pay-per-view program, purchase products on a home shopping network, play interactive games, and so on. The asymmetric nature of web browsing fits well with this type of network configuration; various manufacturers support Internet web browsing using the home television, a two-way network, and an STB with web browser software.

ITV has emerged as a significant commercial market, drawing the attention of the entertainment industry, the cable industry, the computer hardware industry, and software producers. The STB is the client workstation of the ITV distributed systems.

STB ARCHITECTURE

STBs are specialized client computers in an asymmetric, two-way network. The headend server broadcasts information to all client STBs at a very high rate. Each STB accepts one or more channels of information at a time, possibly storing the information for subsequent use, or rendering an audio/video stream as it is received. Given this general context for the operation of an STB, it is now possible to consider how one is designed to be used with an analog/digital cable network.

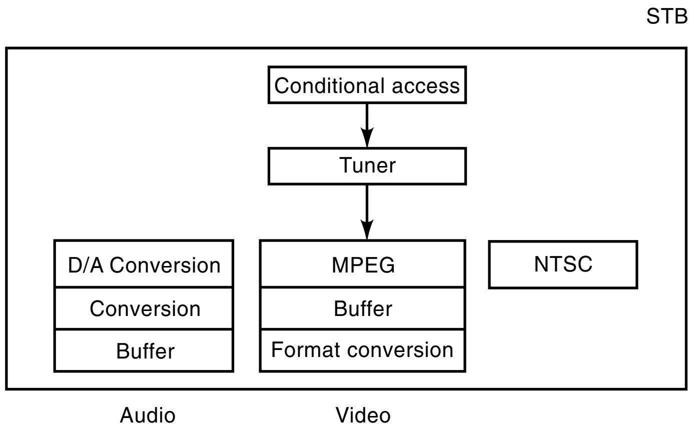

Figure 1. An analog/digital STB.

Figure 1 is a block diagram representing the components of a simple analog/digital STB (1). The conditional access mechanism is used to authenticate the STB so it can decypt an encrypted signal. If the STB is authorized to receive the signal, the tuner distinguishes between analog and digital signals, routing analog signals directly to the NTSC (ordinary analog cable signal) output from the STB to the TV set. The digital bands, containing MPEG streams, are routed to other components for further processing within the STB. The video information is delivered to the MPEG video decompression hardware and the audio information is delivered to an audio converter. Since the audio and video streams can get out of sync in the conversion/decompression step, the results are buffered after being converted, then resynchronized before being converted into signals to be amplified or displayed on the TV screen.

The analog/digital STB can be enhanced to support ITV (and other advanced) features. An electronic program guide (EPG) is a popular ITV example, though it has also been implemented in one-way broadcast networks. The headend periodically transmits the program guide contents on a reserved digital band. An STB that is expecting the guide—either because it requested it, or because it had received information previously that the guide would be broadcast on a given band at a given time—stores the data in its local memory. The subscriber then uses the remote control to query the STB regarding programming, and to select a channel on the basis of that programming.

Figure 2 represents a more complex STB, in which advanced functions such as handling the remote control input device, are supported. This STB includes the conditional access, tuner, analog NTSC component, and the audio and video converters from the simple STB configuration. In addition, it contains a component to read the remote control, to perform advanced functions (such as displaying the program guide), and to transmit interactive requests via the upstream data channel to the headend.

The Advanced Functions box in Fig. 2 represents functions commonly implemented in a conventional computer: displaying menus, properties, and selections on the computer display (TV screen), soliciting input from a pointing/selection device (the remote control), scheduling reception of pushed data, and performing various other processing tasks. These functions could possibly be implemented in hardware logic, though it is much more cost-effective to implement them—and even more extensions—using an embedded system approach.

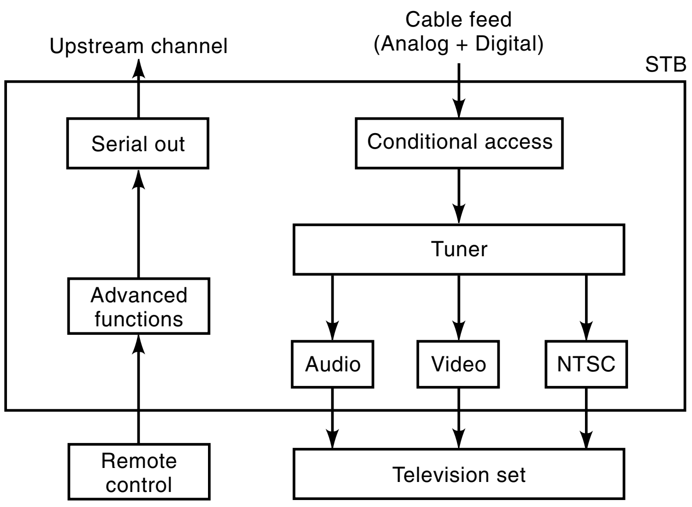

Figure 2. Logical parts of an STB.

Figure 3 shows the hardware configuration for a contemporary set-top box. The CPU and memory are used to implement the advanced functions illustrated in Fig. 2 by constructing software to execute on the embedded computer to control the other hardware components in the STB. The input devices for the STB are the subscriber's remote control and perhaps an optional keyboard; the output device is the TV set. On the network side of the STB are the upstream and downstream communication facilities—either a combination of downstream cable/satellite with a two-way telephone link or a two-way cable link. Figure 3 also shows a graphics function to provide computer output on the TV screen (e.g., to present the program guide, menus, or other set-top control information); in the future, the graphics function will also be used to produce custom special effects for each subscriber.

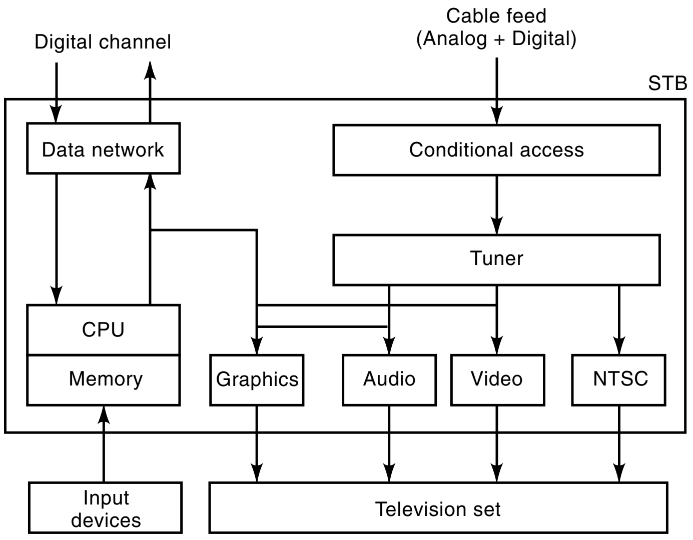

Figure 3. A modern STB architecture.

STB SOFTWARE

Like other software systems, STB software can be divided into application software, middleware, and system software. Application software is designed to accomplish a specific task such as implementing an electronic program guide. Middleware is a term that has been coined in the last five years to identify a common set of functions that applies to different applications from similar domains. For example, ITV and web browsers may have a common set of software functions to decode and encode network packets. Middleware takes advantage of the fact that some software can be reused across multiple applications and multiple domains. Web software is today's most visible example of middleware.

System software provides a common set of facilities for use by all middleware and, hence, by all applications in all domains. It includes various tools, window systems, file managers, along with an operating system. The operating system is responsible for managing the STB's hardware resources and for providing software abstractions of the hardware to the middleware.

Application software is written to use an *application programming interface* (API) created by the middleware and/or the system software. The nature of the API dictates much about the ultimate capability of the applications, and also about the style in which application software will be constructed. For example, if the middleware provides a particular menu system, then all applications will use that menu system for the human–computer interface (meaning that they will all have a consistent "look and feel" with regard to menu operations).

Trends in Software for Embedded Systems

Embedded systems for controllers and onboard computers have been a viable technology for over 15 years. The challenge in these programming environments has been along two dimensions: (1) making the code fit into a limited amount of memory, and (2) making the code execute in a limited amount of time.

High-Level Languages. When the amount of software in the embedded system was small, software for embedded computers was typically written in assembly language. This allowed the programmer to be highly aware of the effect of the source code on the amount of memory being used by the program, and the expected amount of time to execute the code. Unfortunately, this style of code development was very time consuming, error prone, and expensive to develop. The resulting code was also very difficult to maintain or modify to incorporate new functionality. However, success using this approach stimulated the idea of incorporating increasing amounts of functionality into the software. As the functionality requirements increased, the programming time increased at a much faster rate. Assembly language programming became impractical in the face of growing functionality requirements.

In mainstream computer science, high-level programming languages have completely displaced assembly languages. High-level languages allow programmers to work at a much more abstract level than do assembly languages; with high-level languages, programmers can devote more of their energy to designing innovative algorithms and solutions than is possible using assembly language. Before high-level languages could dominate, it was necessary for the language translation (compiler) technology to become efficient enough that the space and performance losses due to the use of the abstraction were outweighed by the increased efficiency at the algorithm level (and in the time saved on programming itself).

Single-Threaded Applications. The original software for an embedded system was written as a single program to be executed by the CPU in the embedded computer. That is, the requirements for the software could be identified, then a single program would be written to satisfy all the requirements. As requirement sets began to grow, the complexity of the control flow in the software became at least, if not more, complex than the requirements. For example, if code modules f_1, f_2, . . ., f_n, were designed to meet requirements r_1, r_2, . . ., r_n, then a main program needed to be written to call f_i whenever appropriate. In the case where there were timing dependencies on the execution of the f_i, the situation could worsen to the point that any particularly function, f_i might have to be decomposed into subfunctions $f_{i,1}$, $f_{i,2}$, . . ., $f_{i,m}$, then to have $f_{i,j}$ called at just the right time. The main program is responsible for implementing this coordination; thus by its nature it is fragile, making it difficult to maintain or change.

Programmers soon realized that this could be handled much more effectively, that is, greatly simplifying the construction and maintenance of the main program, by changing the *single thread of execution* into multiple concurrent threads of execution—*multithreaded execution.* Each of the f_i could be written as a separate program, being executed by a logical machine, using interrupts and synchronization events to start and stop the execution of the subfunctions, $f_{i,j}$. Then, a scheduling entity could simply run each $f_{i,j}$ when it was logically ready to run. This solution was also being widely used in the mainstream software technology in the 1970s, so it was a natural evolutionary change in embedded system software.

In a multithreaded environment, the programmer focuses only on implementing f_i as a set of subfunctions, $f_{i,1}$, $f_{i,2}$, . . ., $f_{i,m}$, each to be executed by a single thread in its own address space.

Time and Space Needs. Multithreaded/multiaddress space technology abstracts the memory space and execution time from the programmer. Experienced C programmers are still able to construct their code so that they can determine space requirements, but control on execution time is lost with the abstraction. (It was also true that the growing complexity made it essentially impossible to construct solutions that met timing constraints in assembly language.) This led embedded application programmers to begin using real-time operating systems, to ensure that the various subfunctions are executed prior to some deadline established by the system requirements. From the programmer's point of view, this requires that the function specification identify the *frequency* at which a subfunction should run, the *time to execute* the subfunction, and a *deadline* by which the subfunction must be completed—*hard real-time* software.

Tailoring the Application Programming Environment for STBs

STBs contain their own embedded computing environment, which must cooperatively implement a distributed computation (with the asymmetric, two-way network and the head-end). Because of cost factors, an STB must be configured without a disk drive and only with a minimum of RAM. These hardware constraints encourage a software environment in which STB function-specific applications can be loaded over the network only when needed.

Encapsulated application technology has emerged as a commercially viable way to produce applications for STBs (as well as other classes of "network computers"). The principle for this style of programming is that the hardware environment is a distributed environment made up of client and server machines. Server construction is accepted as being a software-intensive task, meaning that the construction of the software can be difficult, and the resource requirements to execute server code can be significant. Clients are lightweight entities that can cooperatively execute software by downloading an encapsulated application—called an *applet*—which has been specially designed to conduct the interaction between the client and the server, with a separate interaction between itself and the client environment.

A client application and a server application communicate to jointly perform some work. For example, the client application might be a user interface for browsing a program guide database on a server. The server-specific part of the application—the applet—will have been written by the developer of the server application. The two pieces of software are explicitly designed to communicate with one another over the network. Next, the applet is also designed to interact with the client application through a procedure-call interface (much simpler than the network interface between the server application and the applet). Now, when the user wants to use the server application, the server downloads the applet into the client application. When the user queries the STB application, the STB code passes the query to the applet, which then interacts with the server application to carry out the query.

The applet-based software environment is a key technology for allowing STBs to be configured with modest resources, yet be able to operate in a fully distributed computing environment. It depends on there being a "standard" interface between the applet and the client application, such as the one established in the Java approach to encapsulated applications.

Java applets are portable, object-oriented programs; the Java language explicitly limits the ability to reference arbitrary objects within a program, a concession to help ensure secure operation of programs written in Java. Java programs are translated into a pseudo code language ("bytecodes") rather than into a native machine language. This means that compiled Java programs cannot be executed directly on client hardware, but that they must be interpreted by another package that has been implemented on the target hardware; this interpreter is called the *Java Virtual Machine.* Any machine that contains the Java Virtual Machine can be given a copy of a compiled Java program/applet, and it can then interpret the program.

A Java Virtual Machine can be implemented in any environment, for instance, as an ordinary operating system process or as a part of a web browser. Web browsers such as Netscape Navigator support embedded applications by incorporating a Java Virtual Machine in the browser. As a consequence, when the browser contacts a server to read information, the server can provide specialized functionality by downloading a copy of a Java applet into the STB's browser. The web browser then uses the Java Virtual Machine to execute the program.

Today, STB application software is written to run in a multithreaded operating system environment (2–6). Though early software-based STBs used the then popular approaches to embedded systems, today the cable industry has formed the OpenCable consortium to guide the design and organization of analog/digital STBs (7).

THE FUTURE OF STBs

STBs have evolved from simple band converters into network computers in the home. In this rapid evolution, computer technology has quickly become part of the foundation of STB technology. As consumers increasingly take advantage of the high-bandwidth cable/satellite network connection into their home, STBs will increasingly resemble the home's personal computer. Since STBs are expected to make heavy use of browser interfaces in the next five years, the STB can be used for ITV and web browsing as well as for a broad spectrum of other computing tasks. For many households, the STB is likely to be the consumer electronic computer of tomorrow.

BIBLIOGRAPHY

1. B. Furht et al., Design issues for interactive television systems, *IEEE Comput.,* **28** (5): 25–39, 1995.
2. The PowerTV white paper [Online], 1998. Available www: http://www.powertv.com/product/completewhite.html
3. M. Nelson, M. Linton, and S. Owicki, A highly available, scalable ITV system, *Proc. 15th ACM Symp. Operating Syst. Principles,* ACM, 1995, pp. 54–67.
4. R. W. Brown and J. W. Callahan, Software architecture for broadband CATV interactive systems, Time Warner Cable, May, 1995. Also [Online]. Available www: http://www.pathfinder.com/corp/twcable/index.html
5. R. W. Brown, Pegasus set-top terminal, Time Warner Cable, March, 1997. Also [Online], 1997. Available www: http://www.pathfinder.com/corp/twcable/index.html
6. G. J. Nutt, *Operating System: A Modern Perspective,* Reading, MA: Addison-Wesley, 1997.
7. Anonymous, *OpenCable Functional Specification,* Louisville, CO: CableLabs, 1998.

GARY J. NUTT
University of Colorado

SHANNON'S LIMIT. See MODULATION ANALYSIS FORMULA.