MULTIPLE APPLICATION CO-ORDINATION AND DEMAND-BASED APPLICATION RETRIEVAL VIA INBAND FOR SET-TOPS

Abhijit Chatterjee and Navneeth Kannan
General Instrument Corporation

Abstract

The advanced set-top terminals of today are characterized by extensive software content. They provide a multitude of features to the subscriber at home via a variety of applications. The demand for new and enhanced applications is growing as cable operators and subscribers are discovering new facets of usability of the Set-top terminal. Smooth co-ordination between applications and the need to have more applications than a cost-effective memory model can support are two problems that face set-top software designers.

This paper describes one implementation of an Application Manager based on user-input to solve the first problem and a Demand-based application retrieval system to solve the second problem.

The Application Manager and Transient Application Server have turned out to be successful tools around which many applications have been built in the product-line of Advanced Network Systems group in GI.

PART-1: APPLICATION MANAGEMENT IN SET-TOPS

Software Architecture

The advanced set-top terminal of today is characterized by extensive software content. A real time operating system is at the heart of many of the advanced set-tops. The use of a real-time operating system allows flexibility to add applications and at the same time to be capable of addressing any real-time communication needs for a set-top. In a set-top, typically, there are drivers and servers for providing various platform services. Applications are the clients of these services. This view of the architecture is presented in Fig.1

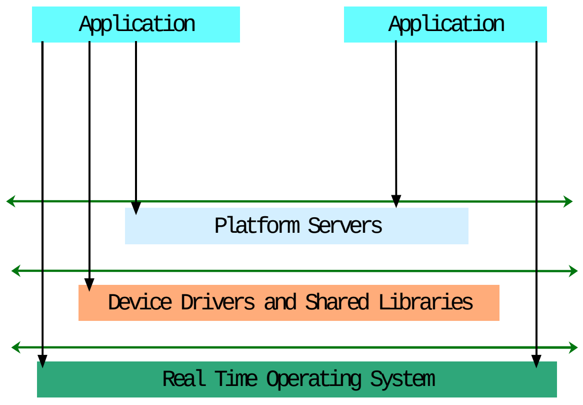
Application
Application
Platform Servers
Device Drivers and Shared Libraries
Real Time Operating System

Fig.1 Traditional Embedded Software Architecture

The above traditional embedded architecture is acceptable only as long as all the applications are written by the same vendor and the applications all have a smooth, predetermined way of transferring control from one to another.

In a typical advanced set-top system, there are different applications provided by potentially different groups. There are also applications written by Independent Software Vendors (ISVs) on the advertised, open platform.

The problem then faced by different application writers immediately becomes one of "Acquisition and Transfer of Control". How is an application supposed to take control, and when and to which other application is it supposed to relinquish control, if it chooses to, are questions that need to be answered.

We can try to make an analogy to the Personal computer world and suggest a solution similar to the Program Manager of the Windows paradigm.

However, the advanced set-top terminal is still characterized by the need to drive down the cost to the cable operator. This often translates to a limited amount of memory that is made available in the terminal. With stringent memory constraints, it is difficult to suggest a solution based on complex Operating Systems. Using a traditional real-time kernel is the best way for reducing the memory requirements.

Token based approach for Application Management

In this paper, we propose a unique way of application management using the concept of tokens.

Tokens and Token Management:

The term token is used to represent the unit of communication exchange between applications. The Token based Application Manager (also called the Token Manager) passes tokens amongst applications. The Manager receives all the keys from the remote control, and passes them as tokens to the current active application. Any unprocessed tokens from that application are also obtained and passed to other applications that may need them.

In the set-top scenario, the IR (InfraRed) Remote Control and the Front panel keys on the terminal are the standard devices using which a user interacts with the set-top to invoke applications.

The approach presented here for application management takes into account the essential nature of applications characterized below:

1. An application that is active usually controls the standard devices for user interaction (the remote control and front panel keys for input and On Screen Display (OSD) for output).
2. Applications are typically invoked upon a special key or because of a specific user selection from a displayed menu.
3. Applications handle a set of keys; interpreting some of them for specific actions; dropping other keys that they don't know how to handle.
4. Applications exit upon some specific user action, whereupon they are not expected to handle any more keys.

Receiving Application:

We define the Receiving Application as an application that is currently active (or in-focus). The Receiving Application is the one that receives all the tokens from the Application Manager.

Token Registration:

Applications that need to participate in this process register with the Application Manager. As a part of the registration, an application would specify the token upon which it would like to become the "in-focus" application. This could be either a direct key or a token generated by a menu program. A direct key is a key on the remote control (like the GUIDE key invoking the electronic program guide) directly invokes an application. While registering, the application specifies whether it wants to see the token before the current in-focus application or after the current in-focus application has had a chance to look at it.

Default Application:

The Application Manager needs a 'default application' to send those tokens that have not been registered as input tokens by any other application. Typically, in a set-top scenario, the application that is active when the viewer is watching TV is the default application (which handles channel surfing, volume control and power-on off conditions)

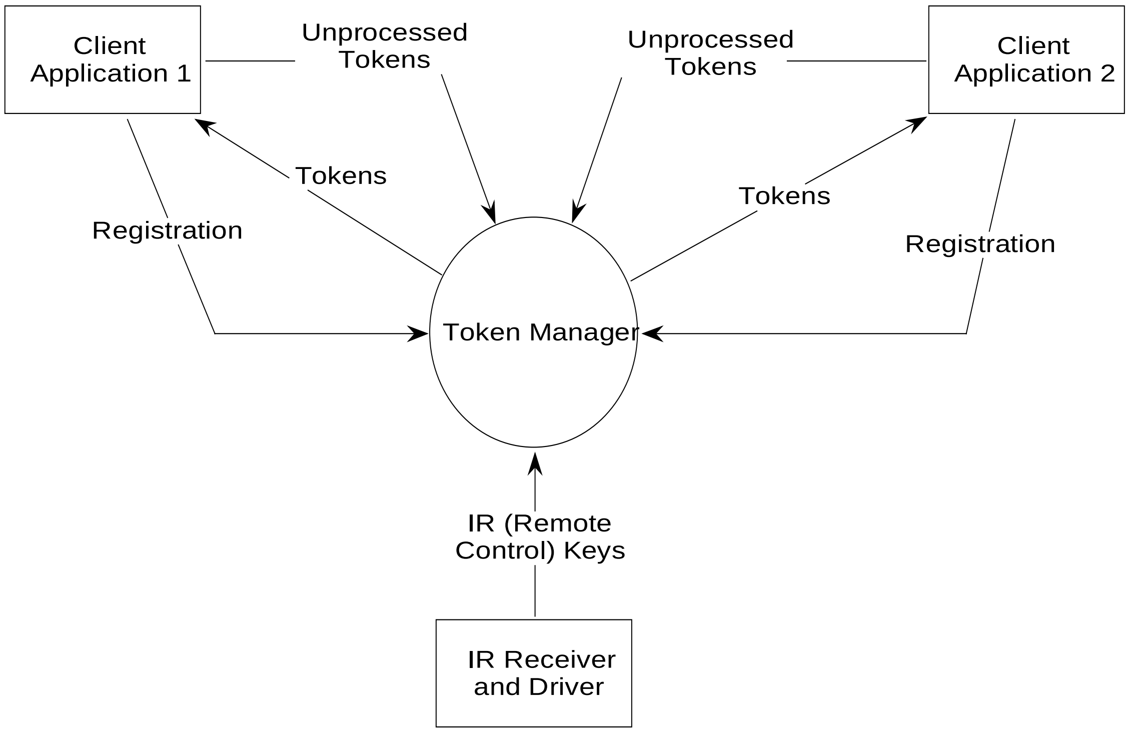

Fig.2: Data Flows amongst Applications and the Application (Token) Manager

Application Management:

The following are the main aspects of the Token based Application Manager.

- Key Forwarding
- Application Flow Control
- Hyper Token Generation
- Application Switching

Key Forwarding

The Application Manager receives all the keys from the hardware. A predefined map establishes the relationship between a key and a token. A menu key pressed on the remote control and a menu key pressed from the front panel of the set-top both map to the same essential token (while retaining source information which may be relevant to the application). Usually, it is the in-focus application, termed the Receiving Application that receives all the tokens.

Application Flow Control:

Flow Control consists of metering out tokens to the applications at the rate at which the applications process the tokens. This ensures that applications don't have to worry about handing over any buffered tokens after their decision to relinquish control. There is just a single buffer for user input and it is maintained by the Application Manager.

Hyper Token Generation:

Sometimes, the user presses a remote control key and keeps it pressed, expecting the same action to be taken multiple times. This is common in channel surfing, browsing through an electronic program guide etc. The Application Manager handles multiple repeat sequences from the key, and directs additional key tokens to the application at a regular interval that can be configured. Applications may choose to turn this feature off, if they cannot handle the held-down key.

Application Switching:

When the Application Manager gets a key, it examines the registry to see if there was any application that had registered for looking at this token before the current in-focus application has had a chance to do so. If it finds one, then it is time for doing an application switch. The current in-focus application is informed that it is no more the in-focus application. It is the responsibility of the in-focus application to give up any substantial or relevant resource (like the On Screen Display, etc.), clear up the display and inform the Application Manager that he is done winding up. The Application Manager then invokes the new application by sending a message to it indicating that it has indeed become the new in-focus application.

More often than not, applications register for a specific token, but do allow the current in-focus application to look at it before processing it. This allows two applications to process the same key, and interpret it differently.

Consider the following example of two applications in the set-top. First, an EPG (Electronic Program Guide) which is activated by the GUIDE token. The other application is the one that handles the Digital Audio features, and also incorporates an electronic guide for the Digital Audio Music Channels. When the Digital Audio Application is the one that is being used by the subscriber, pressing the GUIDE key should present the Music Channel Guide (as opposed to the Video Program guide).

This is accomplished by the two applications by following simple rules. The following sequence of events describes briefly as to how this works.

1. The Electronic Program Guide registers with the Application Manager for the 'GUIDE' key. (It also specifies Post Processing, which would allow the current in-focus application to look at the key before sending it back to the Application Manager)
2. The Digital Audio Application registers with the Application Manager for the 'MUSIC' key.

3. The Tuning Application registers with the Application Manager as the default application.

Scenario 1:

When there is no user interaction, the Tuning Application is the one that is active and is the 'Receiving Application'.

1. The User presses the 'GUIDE' key. Now, since he is in a mode viewing TV, the expected EPG is the Video Channel EPG.
2. The Tuning Application receives the GUIDE key, and realizing that it does not know how to interpret it, passes it back to the Application Manager.
3. The Application Manager examines the registry to see who has registered to post-process the GUIDE key, finds the EPG application and forwards the key to it, also making it the in-focus application.
4. The Application Manager also informs the Tuning Application that it should clear up.
5. The Tuning Application clears up, and the EPG application takes Control.

Scenario 2:

1. The User Presses the Music Key, while watching the EPG application.
2. The EPG application does not handle the MUSIC key, and therefore hands it back to the Application Manager.
3. The Digital Audio Application takes over control.
4. User Presses the GUIDE key now. Application Manager gives it to the in-focus application (which is the Digital Audio Application)
5. The Digital Audio Application (unlike the Tuning Application) knows how to handle this key, and puts up the Music Channel Guide.

In some cases, the current application may relinquish control because of user action (like pressing the EXIT key). In this case, the current in-focus application, upon receiving the EXIT token, relinquishes control by itself, and responds back to the Application Manager to indicate that it not only processed the token, but also wishes to relinquish control.

In certain conditions, asynchronous events could necessitate switching of 'in-focus' status between applications. The following sequence serves as an example of this scenario:

1. The viewer sets up a VCR timer to record an event occurring in the near future and goes back to watching video.
2. A few seconds before the clock reaches the start time of the chosen event, the viewer presses GUIDE key to bring up the Electronic Program Guide and starts surfing through it.
3. The VCR timer indicates that it is time to start recording. The 'Record a Program' application informs the Application Manager that it has to become the 'in-focus' application right now.
4. The Application Manager informs the EPG application that it has to relinquish control due to an external stimulus.
5. The EPG application clears up and informs Application Manager likewise which then makes the 'Record a Program' application the in-focus one.
6. The recording of the event is started.

Conclusion:

In our experience, the concept and use of the token-based application management scheme greatly enhanced the ability of independent application development groups (both within the organization and the Independent Software vendors) to work in unison. The problems associated with integration of multiple applications especially with respect to their smooth co-existence were minimal.

We also have been able to utilize the principles of application management and coordination that we learnt and applied them to more difficult problems like the one described in Part-II of this paper.

PART-II: DEMAND-BASED APPLICATION RETRIEVAL VIA INBAND FOR SET-TOPS

Currently the software for the set-tops is stored either in ROM or programmable Flash memory. The size of the storage area is limited in Set-top terminals and cannot be upgraded easily as demand for newer applications grow. Hence alternate ways for making the Set-top terminal capable of executing multiple applications within the ROM/Flash size constraints had to be identified. The goal was to maximize the usage of the limited storage space and create a notion of a larger 'virtual' executable space. Demand-based Transient Application Retrieval via Inband was designed to meet this requirement.

Overview of Demand-based Transient Application Retrieval

The various applications that are invoked and executed by subscribers can be categorized based on the frequency of usage and expected response time. If an application is not a frequently chosen one or if it does not impose any restriction on the response from the set-top terminal, it is a candidate for demand-based retrieval. Some examples of such applications are Set-top configuration, Favorite Channel setup, Parental Control Password setup etc. These applications can be classified as 'transient' and need not be stored in the ROM/Flash. They could potentially free up some more of the limited storage space for applications that are called upon more frequently or demand faster responses. The 'transient' applications are loaded into the set-top only when demanded and they relinquish all resources once the execution is completed. Essentially, they don't occupy precious storage space permanently.

Instead, the executable code for the 'transient' applications is continually broadcast on one or more dedicated inband channels and the set-top will be able to pick up the code for the particular application in demand and store it in dynamic memory for subsequent execution. Thus, the set-top uses the inband channel as a secondary storage medium. In order to improve the cycle time of the inband carousel, the executable code for the 'transient' applications can be transmitted in a compressed form. In such a case, the set-top inflates the code before execution. Since this process of code acquisition and conditional decompression causes additional latency, a ‘transient’ application should be chosen selectively.

The set-top has a Transient Application Server that controls the acquisition and execution of all ‘transient’ applications. During the start-up sequence of the terminal, the Transient Application Server acts as a ‘proxy’ for all ‘transient’ applications and takes care of registration with Application Manager.

The Server reserves a portion of the dynamic memory as the execution space for transient applications. The size of this dedicated area (called TRansient Application Code Execution area or TRACE) is determined based on size of the largest ‘transient’ application defined for the system. This size also acts as a guideline for determining future candidates for ‘transient’ label as well as in the design of future transient applications. Based on the dynamic memory configuration of the terminal, space could be reserved for more than one transient application thereby improving throughput.

When the subscriber selects to run any ‘transient’ application (by user menu selection or otherwise), the Application Manager switches the ‘in-focus’ application status to the chosen transient application and sends a special API message to inform it of the new status.

The Transient Application Server intercepts the API message and examines its contents to find the target ‘transient’ application name. Thereafter it examines the TRACE area to ascertain if the application is already available. If so, the application starts executing immediately. Otherwise, the Server off-tunes the terminal to the appropriate inband channel and starts acquiring the executable code for the application. Upon completion of acquisition, control is transferred to the ‘transient’ application for execution.

The transient application continues to execute till the user action requires another application to be invoked. After the Application Manager switches the ‘in-focus’ status to the new application, the Transient Application Server instructs the transient application to relinquish all resources.

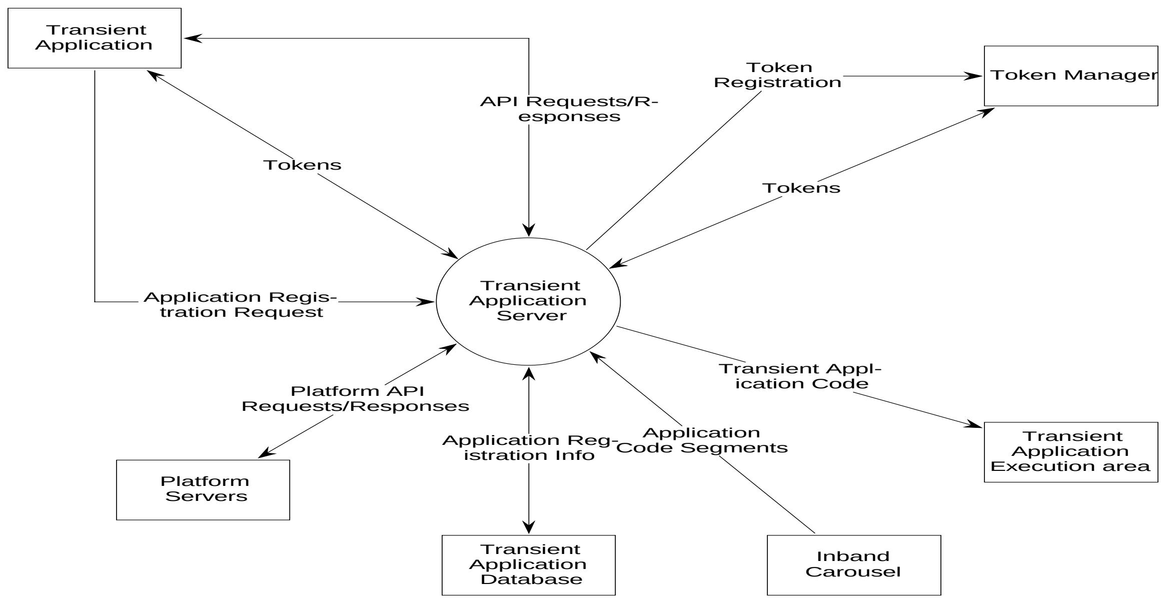
Transient Application
API Requests/Responses
Token Registration
Token Manager
Tokens
Tokens
Application Registration Request
Transient Application Server
Transient Application Code
Platform API Requests/Responses
Application Registration Info
Application Code Segments
Transient Application Execution area
Platform Servers
Transient Application Database
Inband Carousel

Fig 3: Data flow among Transient Application Server, applications and Inband carousel.

Transient Application Management

The primary aspects of interaction between 'transient' applications and the Transient Application Server are as follows:

- Registration with the Server
- API Message handling
- Transfer of Control
- Management of TRACE area

Registration with Server

All the 'transient' applications defined for the system have a small proxy agent defined in the software resident on the set-top terminal's ROM/Flash. During system start-up, these agents provide registration information to the Server regarding the associated application:

- application name
- 'compressed' vs 'uncompressed' status
- dynamic memory requirements
- Execution space requirements
- Inband Channel information

The Transient Application Server saves the registration information for all applications in the Transient Application Database. It also allocates DRAM for run-time requirements at this time so that the transient application is not starved for DRAM when it is loaded off the inband channel for execution.

API Message Handling

The Transient Application Server handles the API interface with the Platform Servers on behalf of the applications. In order to send an API request, the application uses a set of library functions provided by the Server that identifies the requesting application uniquely. The Manager receives all API responses and forwards them to the appropriate application based on unique routing information embedded in the response.

Transfer of Control

Based on user actions, when Application Manager decides to switch the 'in-focus' status to a transient application, it sends the special 'start-up' token meant for the application. The Transient Application Server intercepts the message and performs the following steps:

1. First, it examines the message contents to find the target 'transient' application name and looks up the Registration Database to locate the record for the application.
2. Thereafter it examines the TRACE area to ascertain if the application is already available. If so, the 'start-up' token is forwarded to the application and it starts executing immediately.
3. Otherwise, the Server off-tunes the terminal to the appropriate inband channel and starts acquiring the executable code for the application.
4. Upon completion of acquisition, the 'start-up' token is transferred to the 'transient' application for execution.

Similarly, when the transient application is ready to relinquish control in response to some user action, it sends a special 'exit' token to the Application Manager via the Server. The Application Manager switches the 'in-focus' status and sends a notification to the erstwhile active transient application too. The Server intercepts this message and instructs the transient application to release all system resources acquired during execution.

Management of TRACE area

Based on the dynamic memory configuration of the set-top terminal, the Transient Application Server may reserve a larger area for storage and execution purposes of more than one transient application. This improves the throughput of the terminal since inband acquisition may not be required for all 'transient' application accesses. The TRACE area is managed based on a Least-Recently-Used criterion. For each application currently resident in this area, the Server maintains information about its size and the time at which the application was loaded.

When need arises to accommodate a new transient application in the TRACE area, the Server compares the application executable space requirements with the available free space. If enough space is available, the portion of TRACE area is marked for the new application and inband acquisition begins. On the other hand, if the free space is not enough, the Server looks for application(s) that has not been used recently and is of adequate size to accommodate the new application. The storage space allotted to this application(s) is acquired, gets marked for the new application and inband acquisition begins.

Conclusion

In our opinion, the concept of Demand-based Transient Application Retrieval will enhance the capabilities of the advanced set-top terminals immensely. If applications are categorized appropriately, this implementation will allow the set-top to execute multiple applications within the constraints of a limited storage space without impacting the performance. It will provide cable operators with a mechanism to add on new features without enhancing the hardware configuration of the terminals.

Author Information:

Abhijit Chatterjee
Staff Engineer
Advanced Network Systems,
General Instrument,
2200, Byberry Road,
Hatboro, PA, 19040
(215-957-6749)

Abhijit Chatterjee has been in the software engineering field for 15 years and has been with the Embedded Software Engineering group of GI for nearly 4 years. He got is BSEE from IIT Kharagpur, India.

Navneeth Kannan
Senior Staff Engineer
Advanced Network Systems,
General Instrument,
2200, Byberry Road,
Hatboro, PA, 19040
(215-957-8391)

Navneeth Kannan has been in the software engineering field for 15 years and has been with the Embedded Software Group of GI for the past nearly 5 years. He got his BSEE from Madras University and MSCS from IIT Kanpur India.