Multimed Tools Appl
DOI 10.1007/s11042-007-0166-z

Towards building large scale live media streaming framework for a U-city

Eun-Seok Ryu · Chuck Yoo

Abstract This paper proposes camera and media stream management techniques at the middleware level for implementing a U-City (ubiquitous city). The study focuses on overcoming the difficulties associated with developing middleware capable of processing and streaming multimedia data from a large number of cameras by expanding the traditional media processing technology. The content of the study can be classified into two main categories: One is a camera array management technique that involves the middleware-level framework and protocol for managing the camera array. The other is the media stream management technique for effective delivery management and processing of the multimedia streams from the camera array.

Keywords U-City · Multimedia streaming · SLiM · Media framework

1 Introduction

Rapid developments in mobile handset and networking technologies are expediting the advent of the ubiquitous age. At the current rate, the homes and offices of the future will evolve to form a U-City that showcases an entire city with ubiquitous devices and sensors. The U-City is a space where new services are implemented by interlinking the physical and electronic realms of the city. Consequently, the conventional city management frameworks, IT frameworks and services will be integrated to offer customized services to citizens and corporations. Major ubiquitous projects are already under way in a number of cities, beyond the boundaries of offices and schools, thanks to technical research and development efforts [1, 4, 6, 11]. Seoul, Korea has installed multiple cameras in major spots within the city to

E.-S. Ryu (✉) · C. Yoo
236, Department of Computer Science and Engineering,
Korea University, Seoul, South Korea
e-mail: esryu@os.korea.ac.kr

C. Yoo
e-mail: hxy@os.korea.ac.kr

provide multimedia services, and the city also supports substantially the development of a U-City. There are other research and development projects for the U-City taking place, such as the 'One-North' digital media city project in Singapore and the traffic management framework introduced in Valencia, Spain. A key piece to realize the U-City is the network middleware technology for encoding and streaming to various types of terminals the multimedia data acquired by a number of camera arrays and sensors installed in multiple locations throughout the city. Such fundamental technology is very important since it can support diverse services such as video surveillance, traffic control, disaster evacuation and tourist information, as shown in Fig. 1. However, most previous studies have been limited to the scopes of office buildings and school campuses, far short of the mass media management and delivery technique for handling hundreds or thousands of camera arrays installed in various locations within a city. This is due to the fact that the essential camera media processing technology calls for much greater resources, and makes it difficult to process large-scale citywide services even with today's frameworks.

This paper investigates the technical difficulties associated with large-scale media streaming and proposes solutions for overcoming the difficulties at the middleware level. The proposed solutions focus on improving the traditional media streaming framework by streaming high-capacity scalable live media and developing it into practical applications.

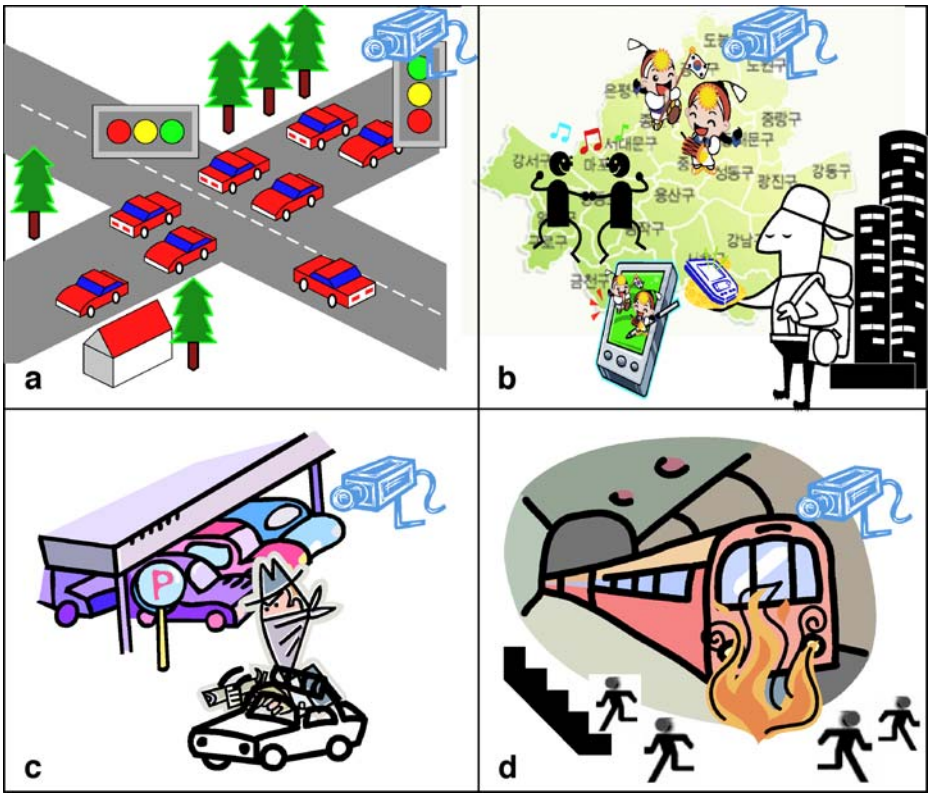


Fig. 1 Several applications of the media streaming technology for the citywide environment **a** traffic control **b** tourist route information **c** video surveillance **d** disaster evacuation

1.1 Problems with building large scale live media streaming framework

There are many problems in streaming large scale live media. In this section we define the problems and discuss ideas for initial solutions.

The conventional small-scale media streaming technique shown in Fig. 2 involves delivering the multimedia data captured by the camera to the server, encoding it and delivering it to the specified clients with an identical capability. Such an approach is difficult to apply in large-scale service environments that contain hundreds or thousands of cameras, and that support diverse user terminals and requires multiple simultaneous user interactions.

The main technical difficulties for applying a traditional streaming framework to a city-wide multimedia delivery framework are as follows.

1. Hotspot problem
2. Camera array management
3. Media streaming management

First, let us consider a framework for controlling the traffic flow and providing multimedia tourist information using cameras installed throughout a city. Since the locations of interest are limited to a few spots, data requests will be concentrated on the multimedia data from those spots. In order to manage such "hotspot" requests, transcoding, scalable encoding and resource management are needed. Moreover, the camera array management module is necessary to handle operation of the numerous cameras that can be added or removed in real-time. In addition, media stream management is required for variable user device support and multimedia services capable of coping with the network changes as well as processing the service request for supporting user interaction. However, simply expanding the traditional framework to resolve such problems will tremendously increase the server load, as indicated with the dotted line in Fig. 2, therefore making it difficult for the camera module to conduct real-time processing. In other words, a bottleneck will be created at the camera server. An obvious solution is simply to increase the number of servers and construct a storage area network (SAN), but there is the need for analyzing the amount of server processing load in detail and systemizing the server roles according to the analyzed server loads. This paper proposes a method for dividing camera management and media stream management and implementing a two-step processing technique, depending on the amount of requests, as shown in Fig. 3. It shows a conceptual division of tasks among servers. A mechanism is also presented for managing the camera arrays and the media stream. In turn, this paper explains a middleware level live media streaming framework that improves the conventional multimedia streaming approach.

From a conceptual point of view, if the number of servers is fixed, the higher the server load, the longer the processing time, as shown in Fig. 3a. Therefore, when there is a high

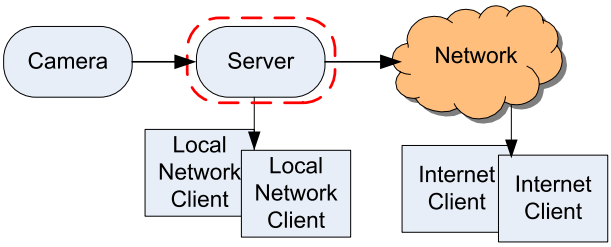


Fig. 2 Conventional media streaming framework model

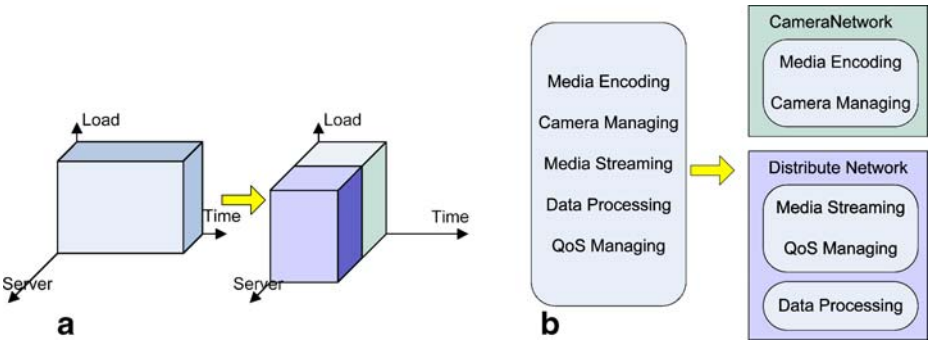


Fig. 3 Conceptual division of tasks among servers. **a** Processing time, **b** task allocation

quantity of multimedia data from multiple cameras, a conventional framework is unable to stream real-time live media. When tasks are split to different servers as shown in the figure on the right of Fig. 3a, the processing time for the task load is reduced. Such an approach is different from simply increasing the number of servers to handle all the tasks, and it facilitates quality of service (QoS) management, including server allocation according to the overall framework resources. Furthermore, when using multimedia data from cameras for different services such as traffic flow control, security monitoring, video tourist information and disaster measure framework, dividing the acquisition, processing and distribution stages for the multimedia data enables easy deployment of different services.

In order to handle the large number of cameras described in this paper, the task allocation on each server is shown in Fig. 3b. It shows that server tasks are classified into two stages. A server must manage multiple camera arrays, which must acquire data from the cameras and perform multimedia encoding using a necessary codec. Moreover, if there is a need to analyze the media data, the server must manage QoS for analysis, processing and delivery, based on which media streaming is provided. These main categories of tasks are classified into two stages according the process flow.

The first stage is the ‘camera network’ and the second stage is the ‘distribution network’. The camera network is the stage responsible for camera array management as well as encoding the multimedia data provided by the cameras. The distribution network plays the role of efficiently distributing the multimedia data acquired by the cameras. Each stage is explained in detail in this paper.

Figure 4 displays the conceptual design of our framework dividing the process tasks into the two stages described above. The Scalable live media server (SLiM server: camera server role) under development manages and controls multiple cameras and encodes multimedia data acquired by the cameras. The A H.264 scalable encoding technique is used in this stage. The streaming server streams the encoded multimedia content according to the QoS policy (in this paper, the QoS policy is for adapting the changes in network bandwidth and the capabilities of user devices). Moreover, at this stage, the distributing server transfers the media to the corresponding data processing server, where it is analyzed, processed and fabricated. For example, in the case of the traffic flow control framework, the multimedia data are recognized and analyzed at this stage to determine the traffic flow control, notifying the traffic light processing framework for providing control commands.

This paper is structured as follows: Section 2 explains the example U-City projects related to this study and introduces the sensor and device management techniques studied by various research organizations. Section 3 describes the framework architecture and the

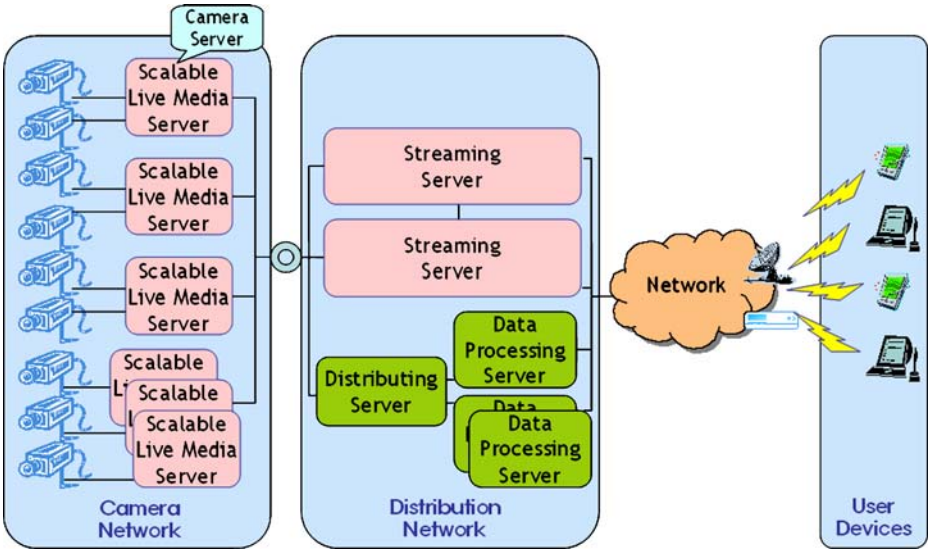


Fig. 4 High level conceptual view of the SLiM framework

research progress. Sections 4 and 5 look into the camera array management technique and the media stream distribution, and control techniques, respectively. Finally, Section 6 provides the conclusion of this paper as well as potential topics for future studies.

2 Related works

2.1 Researches about U-City

Examples of U-City are the U-Seoul project in Korea and the "One-North" digital media city project in Singapore. This section investigates the recent technical trends in these ongoing developments.

The U-Seoul project in Korea involves installing intelligent CCTVs throughout the city for disaster monitoring and crime prevention. Intelligent sidewalk assistance devices will be implemented in Seoul's apartment complexes, which provide position data to the visually handicapped. Locations of children walking between homes and kindergartens will be sent to the parents via SMS (short messaging framework). Multimedia terminals will be installed in each household to allow the residents to search for information such as nearby shopping malls, transportation and cultural events. When a medical emergency occurs, the patient's image or video can be transmitted to a medical facility for "first aid" treatment. Utility usage such as water, electricity and gas will be monitored jointly, and the users will be able to freely access public services such as police using the computer, TV or mobile phone. The city of Seoul has already undertaken detailed designing of these services and some are expected to commence in late 2007.

A more specific example is the TMS (traffic management framework) project in Valencia, Spain [4]. TMS controls the city traffic flow based on the video stream from the cameras installed in various locations in the city. The road traffic video data is acquired from multiple cameras, and the video is encoded with MPEG-4. Each camera is linked to

camera servers that are constructed in a star topology with the core network. The coded streaming data is delivered to the traffic management center through the core network. Upon receiving the data, the traffic management center stores and analyzes the video stream to utilize it as the learning material for the "artificial vision framework." The published video signal is transmitted to each traffic light control device through the stream management server of the core network, to dynamically control the traffic light framework according to the circumstances.

2.2 Middleware for ubiquitous computing

The research regarding middleware applications that support ubiquitous environments include GAIA (http://choices.cs.uiuc.edu/gaia/) from the University of Illinois at Urbana-Champaign, AURA from Carnegie Mellon University and Dey's Context Toolkit from Georgia Institute of Technology. This paper addresses each of these research projects. GAIA [8] is intended for middleware infrastructure relating to human-centric pervasive environments. It is intended to coordinate software entities and heterogeneous networked devices contained within a physical space. GAIA is composed of four major building blocks, the GAIA Kernel, the GAIA Application Framework, the QoS Service Framework, and the Applications. AURA [5] (http://www.cs.cmu.edu/~aura) aims to provide a distraction-free computing environment where people can get services or perform their jobs without interruption from the framework or environment. It views human attention as a scare resource in pervasive environments. So, it takes on board the challenge of minimizing user distractions and framework intrusion. AURA applies two broad concepts to accomplish its goal, proactivity and self-tuning. AURA is composed of five main components, Intelligent networking, Coda, Odyssey, Spectra and Prism. The Context Toolkit [3] (Context Toolkit, http://www.cs.berkeley.edu/~dey/context.html) provides designers with the abstractions they have described—widgets, interpreters, aggregators, services and discoverers, as well as a distributed infrastructure. The Context Toolkit was developed using the Java programming language, but programming language independent mechanisms were used in order to allow the creation and interoperability of widgets, interpreters, aggregators and applications in any language.

3 Framework architecture

The SLiM framework is designed to be a middleware responsible for processing and delivering multimedia data captured by the cameras of the U-City. As explained earlier, the framework consists of the camera network and the distribution network. The camera network manages the camera arrays and encodes the video obtained by the cameras. The distribution network receives the multimedia data from the camera network and transmits it to the application server or streams it to the user device. An example of an application server is a traffic flow control framework. As indicated in Fig. 5, the framework receives the multimedia data from the cameras, processes it and analyzes the number of vehicles on the road. The analysis results are used for applications such as controlling the traffic signal lights. Other applications include tourist information distribution, an example would be delivering real-time video of tourist spots of various locations in the city to the PDAs. In such applications, unlike the traffic flow control framework, the video would be delivered directly to the user device through the streaming server. The distribution network manages QoS for supporting various types of user devices. For example, if the information obtained

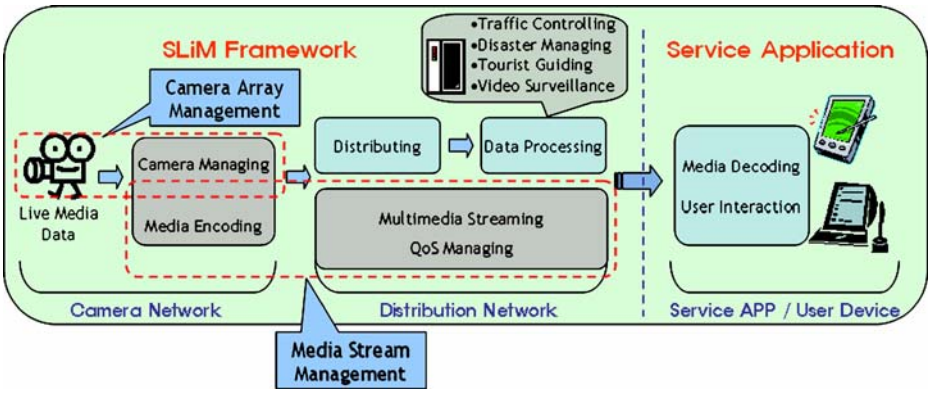


Fig. 5 Conceptual media processing steps in SLiM framework

from the interaction with the user device specifies that QCIF size video will be served at 300 kbp, the information is sent to the SLiM server of the camera network, to transmit a scalable encoded video stream.

As explained in Section 1.1, the SLiM has two stages. The distribution network is again divided into two stages so that the SLiM provides a structure that easily supports multiple users as well as various application servers. Such a structure has the advantage of a single framework being able to support many services. The camera network in Fig. 5 involves the camera server referred to as the SLiM server being responsible for media encoding in addition to camera array managing. The distribution network contains an internal streaming server that manages multimedia streaming and media QoS.

There are two key technologies for the framework as indicated in the overall architecture: camera management for controlling the cameras and media stream management for encoding and transmitting the multimedia data acquired by the cameras. The two areas are indicated with dotted lines in the figure, and their internal structures and processing techniques will be dealt with in detail in Sections 4 and 5.

3.1 Implementing SLiM framework based on previous studies

There were numerous attempts to develop the SLiM framework proposed in this paper. That is to say, it is a newly designed framework based on extending previous studies. We have already implemented the Widget Integration Framework (WIF, http://os.korea.ac.kr/mediateam/WIF.htm) [9] demonstrated in the “active room” ubiquitous environment. The widgets used were the location sensors for transmitting the user location data to the framework to implement various corresponding services. The SLiM is also based on the IMS (interactive media framework) [10] developed previously. The IMS study defines various functions for interactions between the server and the user device in interactive media language (IML), which is an XML-based language, making it possible to offer comprehensive interactive multimedia services at the mobile device level. Applications related to the IMS were also independently implemented on the PDA and demonstrated at international academic conferences, details of which are described in the papers cited in the reference list. In addition, the H.264 SVC technology has been studied and experimented through projects, and it is expected to be applied in the SLiM.

The SLiM is structured for the U-City by implementing each module and integrating the experimented segments as well as those undergoing detailed design processes.

4 Camera array management

Multiple cameras are installed in various locations in a U-City, and a framework is required for methodically managing the cameras. For this purpose, a middleware-level camera management mechanism is introduced in this paper. A camera network is defined in this paper as a group consisting of multiple camera servers (SLiM servers).

The SLiM server has three major functions as exhibited in Fig. 6. The SLiM server performs transcoding and scalable encoding in order to compress the multimedia data captured by the cameras, and contains the camera array module for managing multiple cameras in real time.

The camera array management module is designed to control multiple cameras from the SLiM server. Since the camera widget state information needs to be shared among the SLiM servers, the data can be read and updated when necessary through an external DB server. The information is organized into a camera widget state table and managed by the list updater of the camera array management module. The cameras and the SLiM server have a many-to-one relationship. In other words, a single camera transmits video to a single SLiM server, which performs scalable encoding and converts the camera video into various streams. The information is then delivered to the distribution network. At this stage, the clear distinction of the media encoding technique is the fact that there are independent and separate modules for calculating the QoS parameter according to the capability and network condition of the user device for performing media encoding. The former is located in the distribution network, and the latter in the camera network for this framework.

This section explains the camera array management technique deployed in the camera network.

Figure 6 displays the internal structure of the camera array management module. Its role is managing the camera array, the first of the two major areas of study for the proposed framework. The module is the result of modifying parts of the WIF of a previous research project, and it can interact with other components within the middleware.

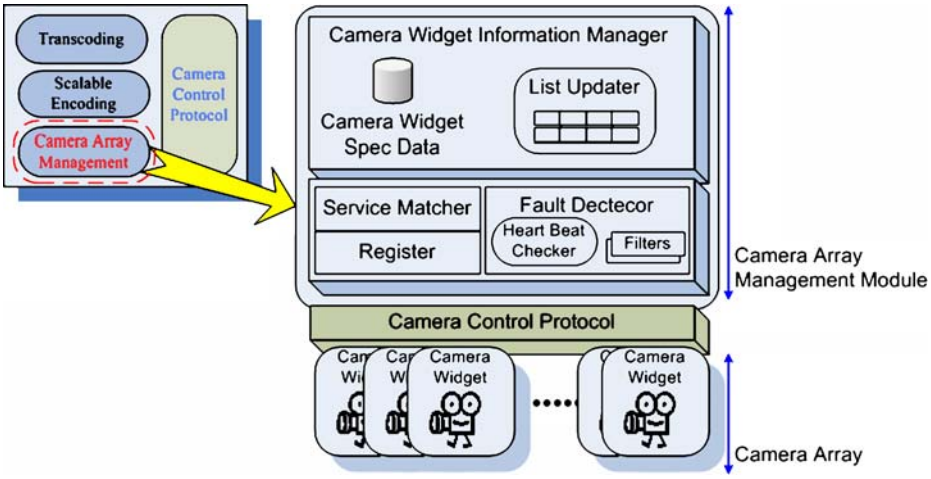


Fig. 6 Internal architecture of SLiM server

The camera array management module has the following advantages.

- Dynamic binding

The camera arrays are linked with widgets for effective wrapping to enable real-time registration and deregistration of cameras and to provide scalability to the overall framework. This advantage can be applied to various areas to improve efficiency.

- Fault tolerant

The fault detector performs regular heart beat checking of the cameras to monitor malfunctioning units. Malfunctioning units are managed by a table to be excluded from the services, making the framework highly reliable.

4.1 Camera widget

We use the concept widget as the context toolkit. Widgets encapsulate information about a single piece of context, for example location or activity. They provide a uniform interface to components or applications that use the context, hiding the details of the underlying context-sensing mechanisms. In this paper, this concept of a widget is used to explain the camera widget. Therefore, the camera widget deals with the context information of cameras.

4.2 Camera widget information manager

The camera widget information manager (CWIM) manages widget state information. It manages state information of widgets received from the widget register or fault-detector with the list updater. It also keeps specification information of several widgets in order to find a suitable widget through the service matcher for services required from the camera discoverer (acts as service). It also exchanges all information of widgets with other modules in the middleware application.

4.2.1 List updater

The list updater (LU) belongs to the CWIM and manages the camera widget state table. This table information is shared by SLiM servers. Location number in Table 1 divides widgets by the camera location. For example, number 0 means that those widgets have relations with the city hall area and number 1 means that those widgets are related to the

Table 1 Example of camera widget state table usage

Camera widget ID		Live information	State information
Location info. no.	Widget no.		
0	001	1	1 (Occupied)
(Area: city hall)	002	1	1 (Occupied)
	001	1	1 (Occupied)
1	002	0	2 (Requested)
(Area–National Bank)	003	0	0 (Available)
2	...		
(...)	...		

national bank area. Within such classification, the widget number is granted to the widget, which is actually installed or is going to be installed. Then, the location info number and the widget number are combined and used as camera widget ID. The LU communicates whether the relevant widget operates, encoded as 1 or 0 to the live information field. The state information expresses the state of the widget [1: Occupied, 2: Requested, 0: available (non-used)]. Value 0 means that the widget is not used or required at all, value 1 means that it is being used and value 2 means it was required for use if the relevant one is linked to the framework. The LU's role is table management. Therefore, other modules can use the requested widget or unavailable widget in services using this table representation.

4.3 Fault detector

The Fault Detector is designed to check the state of the widget through a periodic heartbeat checker. It updates the live information value and the state information value regarding the widget state table in the widget information manager if an error occurs in the sensor or widget. The LU informs camera discoverer of these events. This guarantees high reliability by removing problematic widgets or replacing them with a new one.

4.4 Service matcher

The service matcher finds a suitable widget for services that are required from the service discoverer. For example, the service discoverer in middleware requests a video streaming service from a specific position and understands those requests, finds a suitable camera widget and binds to it. The service matcher references the camera widget state table as in Table 1 and finds an available camera widget, which can service video streaming at the requested position. Detailed process steps are explained in Section 4.6. In this paper, we suppose the service discoverer [7, 13] is located in the outer side of our framework.

4.5 Widget register

The widget register takes charge of registration and deregistration of a widget. The widget helps the widget information manager by passing the state event to the LU when it is registered/deregistered to/from the framework. Though other frameworks receive data only from the pre-registered widget, the widget register can support dynamic environments where widgets are registered/deregistered in real time by accepting a new incoming widget and sending it to the widget information manager. As a result, the widget register makes the camera array management module bind a new incoming camera widget.

4.6 Interaction diagram among components

The components of the camera array management module run through frequent interaction among them. This section explains four cases to show the dynamic binding process using the interaction diagrams below.

4.6.1 Widget registration

The following diagram displays the registration process when a new camera widget is introduced to the framework (Fig. 7).

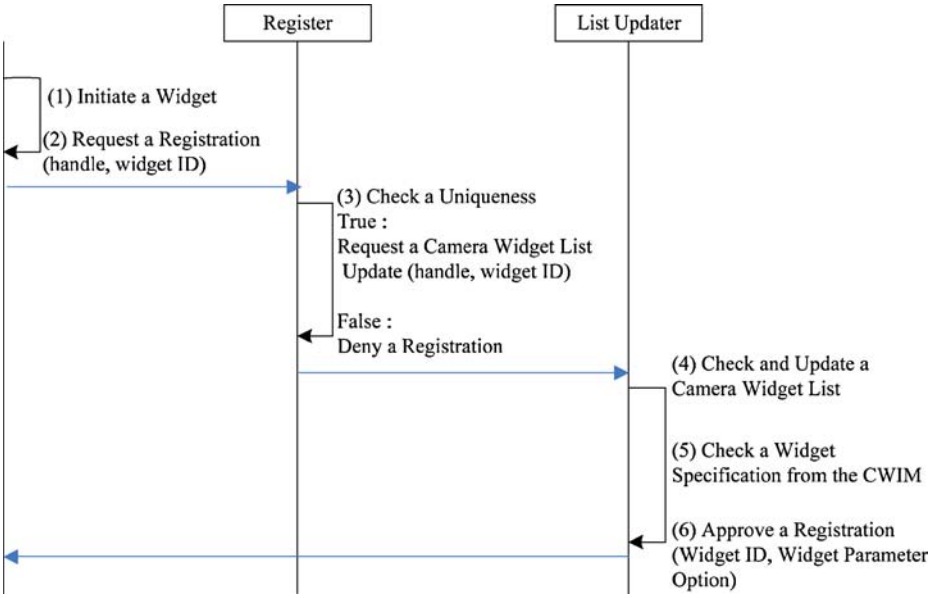


Fig. 7 Widget registration diagram

4.6.2 Widget deregistration

The following diagram displays the process of deregistering a camera from the camera widget when a camera is removed from the framework for various reasons including malfunction (Fig. 8).

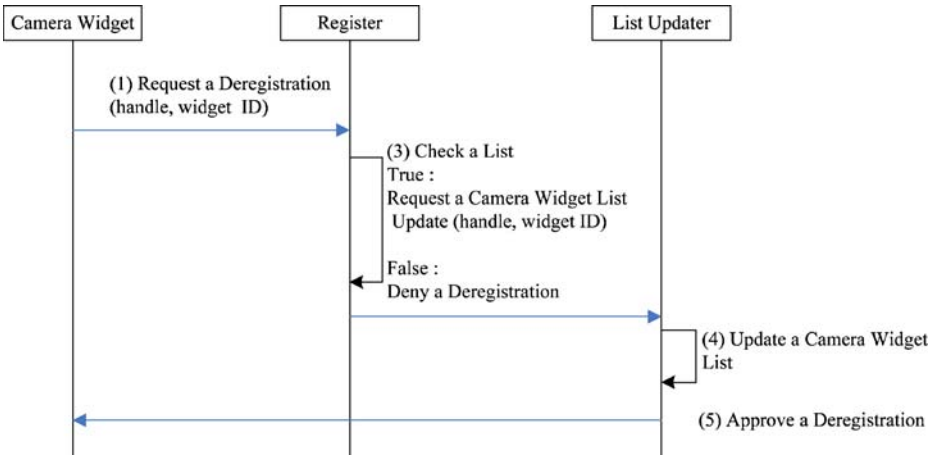


Fig. 8 Widget deregistration diagram

4.6.3 Widget assignment by service request (if the necessary widget is already registered)

The service matcher exerts best effort to accept the multimedia data request from the user device at the middleware level. If the serviceable camera widget is already registered, it is registered in the camera widget state table and served (Fig. 9).

4.6.4 Widget assignment by service request (if the necessary widget is not registered)

The camera discovery manager exerts best effort to accept the multimedia data request from the user device at the middleware level. However, if a serviceable camera widget is not registered, the service pending status is notified to the camera discovery manager through the list updater. When the required camera widget is linked to the framework and registered in the list, it acknowledges the service pending status and becomes serviceable from that point.

There are two major instances for the camera widget being registered during the process. First case is for a newly added camera. An example would be installing a temporary camera during a concert or disaster. There is also the consideration for future usage of registering mobile cameras. Second is a registered camera widget resuming after fault detection. Hence the module provides seamless dynamic binding of the framework even when a camera widget is newly registered during framework operation (Fig. 10).

5 Media stream management

The subject of this section lays the main point on the method delivering the data encoded from a camera of camera network through the streaming server of distribution network Fig. 5.

The most critical element on delivering the media stream is classified into two categories: The first is encoding and distribution of media stream, and the second is resource management to decide the streaming bandwidth suitable to the network situation, controlling the request of media by diverse users. Accordingly, this section describes the way of processing these categories by means of using this framework.

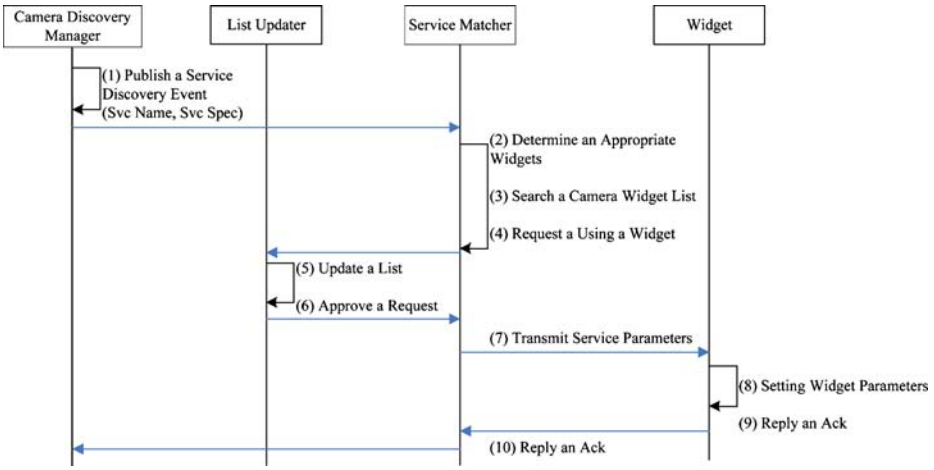


Fig. 9 Widget assigning diagram: case 1

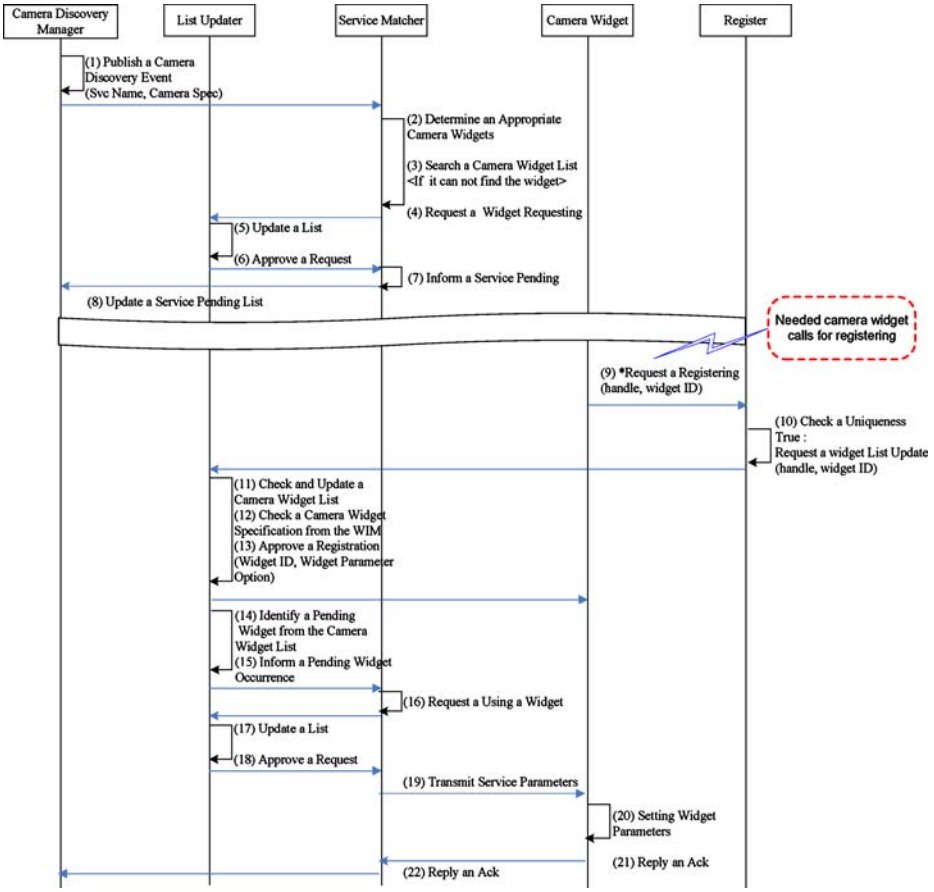


Fig. 10 Widget assigning diagram: case 2

5.1 Multimedia encoding and streaming

This paper uses the technique of H.264 scalable extension for encoding the camera images. Therefore the media encoder of the camera makes scalable media, and the streaming manager streams it to the user as device capability by assisting the QoS manager (Fig. 11). This processing is explained below.

5.1.1 Media encoder—H.264 scalable extension

It is not easy for a framework to simultaneously support the multiple devices of users. One of the problems is that the capability of user's devices is quite diverse. So, conventional multimedia streaming framework has been constantly improved to solve these problems through several ways. The easiest way is to do only multimedia service of format fixed to the client application. However, this way is not suitable to the current environment due to various mobile devices. The next approach is using the transcoding technique. However, it is also hard to support diverse devices in real time because the complexity of transcoding is too high. Therefore real-time transcoding is not yet suitable to this large-scale framework.

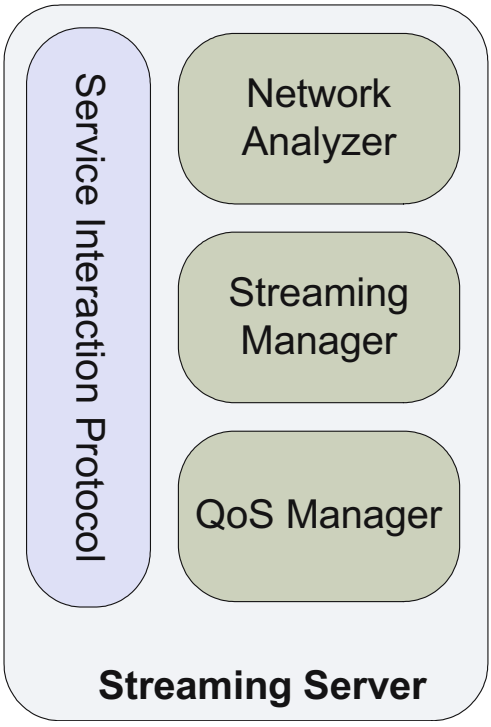


Fig. 11 Internal modules of streaming server

One way to overcome this problem is to use scalable encoding techniques. It has a merit of supporting multiple services with a single source, so that it can handle diverse user devices and also support real-time streaming. As a result, the SLiM uses H.264 scalable extension techniques. Since it is possible for this technology to support spatial, temporal, and SNR scalability simultaneously, it can support the diverse screen size and resolution and also diverse network bandwidth. Further, because the ratio of compression with respect to quality of picture is excellent, it is possible to reduce the bandwidth of the entire framework. We used the latest reference code (JSVM 7: JVT-T203) via a CVS server. In Korea, since the mobile phones are already equipped with a H.264 decoder, it is possible to apply H.264 for the U-City project.

Figure 12 shows the method of supporting diverse user devices by using the H.264 scalable extension. This is processed at the scalable encoder module and the streaming manager module in our framework. First, the camera server of the camera network makes the layered encoding data by using the H.264 scalable encoder. At the same time, the priority with respect to the contents is indexed: the indexing is carried out in order of weight prearranged of major objects pictured by camera, because some problems exist in the way the image recognition is applied into a large-scale framework. For instance, where the image of the security camera is encoded, it has high priority, and this information can be retrieved through camera widget table of camera server. Then, the scalable encoded data is extracted into bitstream corresponding to the capability of user's devices by a streaming manager. In this way, the SLiM can provide differentiated services according to the capability of the devices.

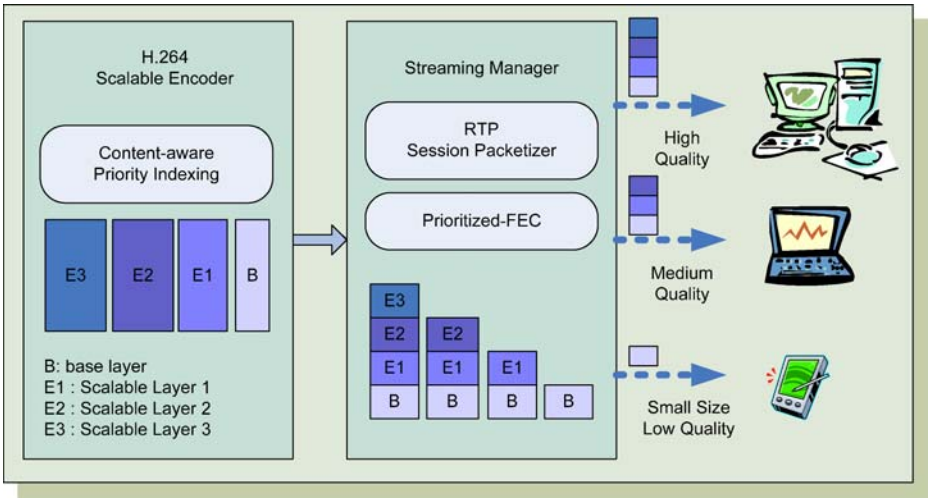


Fig. 12 Scalable encoding and priority-based packetizing scheme

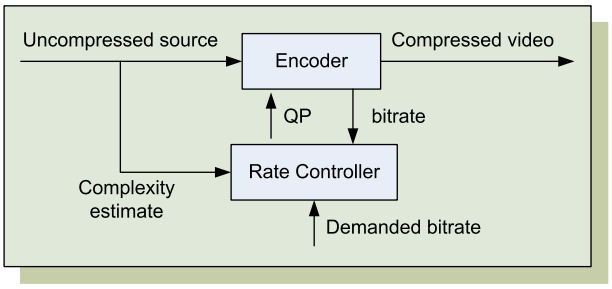
5.1.2 Media streaming

The conventional rate control technique is as shown in Fig. 13. It creates a quantization parameter (QP) for encoding media data by using information of complexity estimation and demanded bitrate in the rate controller. Then it encodes media data as target bitrate by using this QP value for the encoding option. However, in this paper, we designed a new media stream management module as shown in Fig. 14, for adaptive streaming to various user devices. As shown in this figure, the target bitrate information delivered to the media encoder is used as the category information for determining the level of scalable encoding. For example, if all sessions can be served by the base layer encoding, the media encoder does not encode the enhancement layer. In this way, the selective scalable encoding method saves the resources of the complete framework.

The media encoder of the camera server gets the target bitrate information as the encoding option from the QoS manager of the streaming server. The media streaming processes are as follows:

1) User device requests streaming service from user device.
2) QoS manager performs resource management module by using request information and camera information.
3) Media encoder encodes the H.264 scalable layers as resource allocating information.

Fig. 13 Traditional rate control mechanism (CBR)



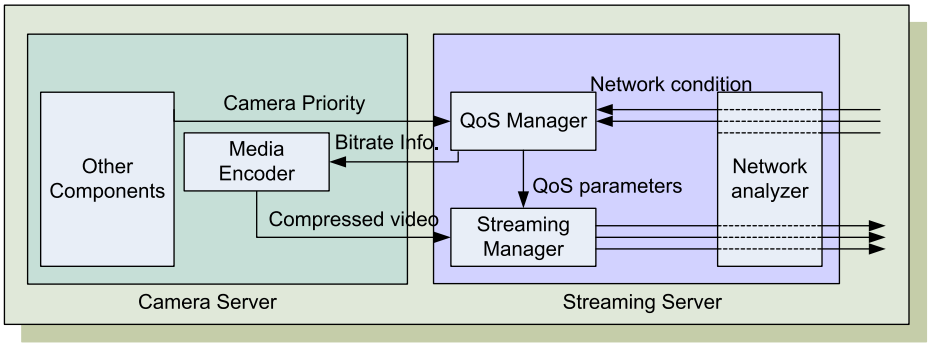


Fig. 14 Media management in SLiM framework

4) Streaming manager extracts bitstream according to capability of user device and information of resource allocating.
5) Streaming manager streams media contents after packetizing.

Firstly, the network analyzer informs the QoS manager of the network status between the user device and the streaming server. Next, based on the camera information that the QoS manager obtains from the camera array manager and the resource management model addressed in Section 5.2.1, the QoS manager determines the transmission rate of the user session. Then, this information is transmitted to the camera server, and adjusts the frame rate, number of layers and filters of the encoding configuration option of the H.264 scalable extension. After encoding these adjusted options, the framework extracts the bitstream from encoded data according to capability information of user devices and streams it to the user.

5.2 QoS manager—resource management

To provide the service differentiation as the priority of session of user and the characteristics of the user device, there must be a component to control the resource of the SLiM [12]. The components of the SLiM have N:N relations: the relationship between the streaming server and user's device is N:N, and the relation of the camera server and the streaming server is also N:N. This section explains the QoS management between components with N:N relation.

QoS managing of the network bandwidth can be performed by interaction of the internal information between the SLiM server and the user device. By quantifiable server resource between the camera server and streaming server, QoS can be supported in resource

Table 2 Notations

Symbol	Meaning
LBWi	Lower bound bandwidth
UBWi	Upper bound bandwidth
Pi	Priority of ^{i}th Session
AddBWi	Additional bandwidth to ^{i}th session
Bwi	Bandwidth of ^{i}th session
MaxBW	Maximum available bandwidth
AvailableBW	Available bandwidth of current state

reservation between servers. In this paper, the priority-based resource management model is implemented for managing the network resource between streaming server and user device. QoS can be applied to sub-frameworks of the SLiM, and as a first step, we apply QoS to network bandwidth allocation in the streaming server, and then QoS is applied to the resource management between servers. Since this module manages network bandwidth with not only the range of resource requested but also with priority order, it may suitably be applied into the traffic control framework which has high priority.

5.2.1 Priority-based resource management model

The resource managing model controls admission through allocating the network bandwidth according to the request of user applications. The algorithm for allocating

Table 3 Algorithm pseudo code

$AvailableBW \leftarrow MaxBW - \sum_{i=1}^{n} LBWi$

if($\sum_{i=1}^{n} UBWi \le MaxBW$)
{
  assign BWi of each channel to $UBWi$;
  return;
}
if($\sum_{i=1}^{n} LBWi > MaxBW$)
{
  can't assign bandwidth;
  return;
}
assign BWi of each channel to $LBWi$;

$AvailableBW \leftarrow MaxBW - \sum_{i=1}^{n} LBWi$;

$i \leftarrow 1$;
while($AvailableBW > 0 \quad or \quad |P| > 0$) // $|P|$ is the size of Priority List
{
  $AddBWi \leftarrow Min(AvailableBW \times \frac{Pi}{\sum_{i=1}^{n} Pi}, UBWi - LBWi)$;
  $BWi \leftarrow LBWi + AddBWi$;
  $AvailableBW \leftarrow AvailableBW - AddBWi$;
  remove Pi from Priority List (P);
  $i \leftarrow i + 1$;
}
return;

Table 4 Simulation example by priority-based resource management model

Index	Session no.	*LBWi*	*UBWi*	*Pi*		Index	Session no.	*LBWi*	*UBWi*	*Pi*	*AddBWi*	*BWi*
1	Session 1	3	5	6		1	Session 1	3	5	6	2	5
2	Session 3	2	6	3	→	2	Session 3	2	6	3	3	5
3	Session 2	5	8	1		3	Session 2	5	8	1	1	6

bandwidth is explained as follows: The user application which requests the data from the streaming server notices the maximum and minimum bandwidth that would be served. At this time, the session of the user device has priority, and this information can be set up by the policy of the QoS manager.

In such way, the resource manager allocates the bandwidth to each session of the user by using the *UBWi* (upper bound bandwidth), *LBWi* (lower bound bandwidth), and *Pi* (priority) information. A point of difference with other models is using the information of 'upper/lower bound bandwidth' for allocating the resource. This has the advantage that it can use capability information of various mobile devices and quality of service information from user applications for resource managing. For instance, the upper-bound information is determined as capability limitation of user device having low performance and lower-bound information is determined for guaranteeing the least service quality to the user application. Firstly, this model calculates the sum of least bandwidth of each session. If this value is exceeds, the maximum available bandwidth is analyzed by the network analyzer, and it cannot allocate the resource to each session. If the sum of maximum bandwidth requested from each session comes less than the maximum available bandwidth, as much as the maximum bandwidth is allowed to be allocated into each session. We must solve the problem in every case except the two examples mentioned above. When the least of bandwidth is allocated into each channel, there remains available bandwidth (AvailableBW). Then, the remained bandwidth is allocated in turn according to the priority order of each session. If the bandwidth allocated this way is greater than the sum of the requested maximum bandwidth of the application, only the requested maximum bandwidth is allocated. The session allocated in this way is deleted from the list, and the algorithm keeps on running repeatedly until the list is empty. The notation for explaining the algorithm and the algorithm itself are shown in the table (Table 2).

The simulation exampled by the present algorithm is shown below, and *MaxBW* is set to 16. The left table shows the information in the order of high priority of the sessions and the bandwidth demanded by each session, and the right table is resultants distributed the

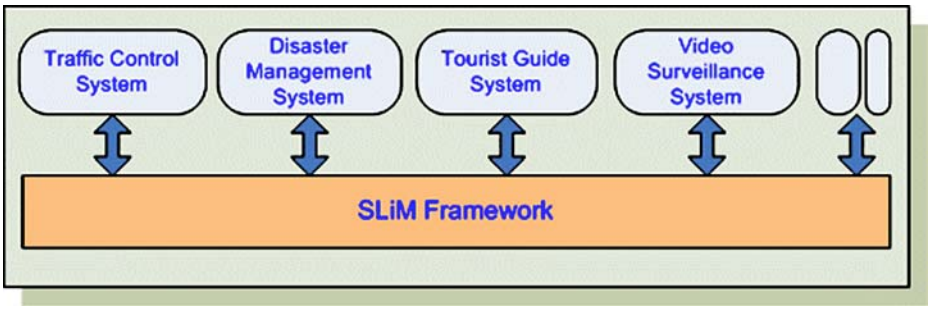


Fig. 15 Several applications of SLiM framework

resource by means of using the algorithm of Table 3. The bandwidth allocated in accordance with each session is 5, 5, and 6 respectively (Table 4).

As mentioned above, the information of the bandwidth determined by this algorithm is transmitted to a media encoder (Table 4).

6 Conclusion and future works

This paper proposes the camera and media stream management techniques in the middleware level required for implementing the U-City. The study focuses on overcoming the difficulties associated with developing middleware capable of processing and streaming multimedia data from a large number of cameras by expanding the traditional media processing technology. Therefore, this paper addresses the integrated framework which can process and distribute the multimedia stream acquired from the camera array [1].

The SLiM framework which can be used in the various fields efficiently provides the large-scale camera media delivering mechanism to the various user devices. It consists of camera network and distribution network. The camera network is the stage responsible for camera array management and encoding the multimedia data provided by the cameras. The distribution network plays the role of efficiently distributing the multimedia data acquired by the camera network. The user device services variable user requests by using the IMS (interactive media framework).

The below Fig. 15 shows the service example using the SLiM framework. The result of this paper can support diverse services such as video surveillance [2], traffic control, disaster evacuation and tourist guiding.

Studies of the SLiM framework are ongoing. This includes techniques for managing various cameras and supporting user devices. It also includes techniques for implementing camera networks and efficient scalable encoding. In the future, the load balancing technique between camera servers will be studied. The large scale live media streaming framework developed in this paper will be used in various parts of the U-City.

References

1. Berkvens W (2005) Media distribution in a pervasive computing environment. IEEE Proceedings of the Conference on Pervasive Computing and Communications, March 2005
2. Desurmont S (2004) A generic flexible and robust approach for intelligent real-time video surveillance systems. Proceedings of the SPIE—Real-time imaging VIII, vol 5297, no 1, Jan. 2004
3. Dey AK, Salber D, Abowd GD (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum Comput Interact J 16(2–4):97–166
4. Esteve M, Palau CE (2006) A flexible video streaming system for urban traffic control. IEEE Multi Med 13(1):78–83
5. Garlan D, Siewiorek D, Smailagic A, Steenkiste P (2002) Project AURA: towards distraction-free pervasive computing. IEEE Pervasive Computing, special issue on Integrated Pervasive Computing Environment 1(2):22–31 (April–June)
6. Hess CK, Roman M, Campbell RH (2002) Building applications for ubiquitous computing environments. International Conference on Pervasive Computing (Pervasive 2002), Zurich, Switzerland, pp 16–29, August 26–28
7. Marti S, Krishnan V (2002) Carmen: a dynamic service discovery architecture. Technical Report, August 2002
8. Román M, Hess CK, Cerqueira R, Ranganathan A, Campbell RH, Nahrstedt K (2002) GAIA: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing, pp 74–83, Oct–Dec 2002

9. Ryu E-S, Hwang J-S, Yoo C (2005) Widget integration framework for context-aware middleware. Lect Notes Comput Sci 3744:161–171 (Oct)
10. Ryu E-S, Yoo C (2004) An approach to interactive media system for mobile devices. Proceedings of the 12th ACM International Conference on Multimedia (MM 2004), Oct 2004
11. Trivedi MM, Mikic I, Kogut G (2000) Distributed video networks for incident detection and management. Proceedings of the IEEE Conference on Intelligent Transportation Systems. IEEE Press, 2000, pp 155–160
12. Yang Z, Nahrstedt K (2005) A bandwidth management system for wireless camera array. Proceedings of the 15th International Workshop on Network and Operating Frameworks Support for Digital Audio and Video (NOSSDAV '05), Stevenson, WA, 2005
13. Zhao W, Schulzrinne H, Guttman E, Bisdikian C, Jerome W (2002) IETF RFC 3421, "select and sort extensions for the service location protocol (SLP). November 2002

Eun-Seok Ryu (esryu@os.korea.ac.kr) is a Ph.D. candidate in the Department of Computer Science and Engineering at the Korea University. His research interests are in the area of multimedia communications and networking, video/audio codec, and wireless mobile system. He received his B.S. and M.S. in computer from Korea University in 1999 and 2001, respectively.

Chuck Yoo (hxy@os.korea.ac.kr) received the B.S. and M.S. degree in electronic engineering from Seoul National University, Seoul, Korea and the M.S. and Ph.D. in computer science in University of Michigan. He worked as a researcher in Sun Microsystems Lab. from 1990 to 1995. He is now a professor in Department of Computer Science and Engineering, Korea University, Seoul, Korea. His research interests include high performance network, multimedia streaming, and operating systems. He served as a member of the organizing committee for NOSSDAV 2001 and JCCI 2006.