International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.1 (2014), pp.89-100
http://dx.doi.org/10.14257/ijmue.2014.9.1.09

A Design of Linear Task Set Model for Multimedia Services in IP Network

Hyoungjin Kim[1], Jaekon Oh[1] and Gyusu Jung [2]*
[1]*Department of IT Applied System Engineering, Chonbuk National University*
[2]*ENCORE Co. Ltd*
kim@jbnu.ac.kr, gsjung@en-core.com

**Corresponding Author*

Abstract

The purpose of this article is to design a multimedia streaming system with a complex structure as the one with a simple and basic work structure and realize system design that can enhance both expandability and portability based on that structure. To solve the problem that it is too complex and has low expandability, this study suggests a system design structure based on the LTS (Linear Task Set) model. Based on the simple, basic structure grounded on LTS structure, the study defines a consistent process and allows free structure alteration in the simple system structure according to the service purpose and demand. To secure independence and simplicity among components, the study defines a port channel object of a singleton structure and realizes design that can allow stream transmission and overall stream monitoring on the system for all ports with the port channel based on the mapping information of a component port. And based on this model, the paper suggests a component system model that can realize multimedia streaming service.

Keywords: *Multimedia Streaming System, Linear Task Set, Component System*

1. Introduction

For multimedia streaming service, comprehensive system technologies including media status, network bandwidths, buffer use rate, disc access time and memory management based on the defined spatio-temporal information are being used and in order to maintain service quality fit to user environments and conditions, comprehensive management of parameters affecting both directly and indirectly service qualities including network status, butter use rates and packet loss rate should be considered [1]. New function of the system also requires considerations on flexibility and adaptability of multimedia for adapting to scalability, portability on heterogeneous or various types of multimedia data. For this, in multimedia system, multimedia framework was developed, its design and establishment were easier and productivity of systems to use multimedia data including video conference, video mail and CCTV was improved.

Infopipes focused on categorizing and expressing tasks operated on distributed multimedia streaming system as component for easier understanding in its abstraction phase [2], while SMART put emphasis on performances of the system through the methods of multiple loop prediction, loss decreases and macroblock mode selection by using MPEG-4 FGS for efficient, scalable and robust streaming video system [3]. Previous literatures focused mostly on multimedia process technologies such as video compression, encoding/decoding and realtime transmission and the system has problems of difficulties for system scalability or

ISSN: 1975-0080 IJMUE

management because its degree of coupling between multimedia data and process technologies is higher and its media synchronization or transmission module depends highly on data process technologies.

In order to solve these problems, this article aims to redesign complex multimedia streaming system in simpler ways for the system with higher portability and scalability based on this structure. Complex multimedia streaming system was designed for consisting of simple task process structure and extending additional functions. The system also allows free structure modification according to the purpose and requirement of the service and the Linear Task Set model for the system maintenance and higher scalability was suggested. LTS consists of part to manage data flow between tasks, part to manage task flow between tasks different in processing time and part to manage buffer according to data flow. So this article suggests basic component model of multimedia streaming system based on this model. And the design will be made with extension by assembling basic components and with powerful portability and scalability to produce new components by adding new algorithms or architectures [5].

2. Suggestion for Linear Task Set (LTS) Model

2.1. Filters and Pipes Model

In this article, the basic model explains a system model with greater portability and productivity for multimedia streaming system as well as excellent scalability of new technologies and maintenance. This model consists of filters and pipes and each filter connects to other in very simple interface. And a filter receiving messages through input pipe, manages a unit of task and transmits the outcomes to next filter through output pipe. A pipe means a basic pattern model which is connected from one filter to other one and transmits output message from one to next one.

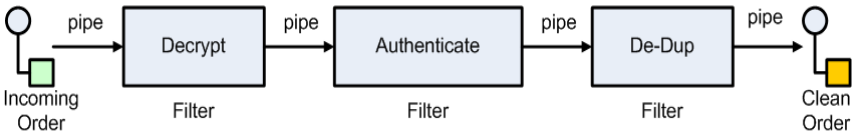


Figure 1. Filters and Pipes Pattern

Figure 1 is a scheme of the "Filters and Pipes" pattern suggested in [4] and if a message about an encrypted order reaches the decrypt, it functions to decrypt the received message and transmits it to next filter. And then, it shows that the authentication filter authenticates data received through pipe and transmits the outcome to next filter. That is, filters and pipes are objectified by dividing into each function on an order of processing, barely depend on each other because of their equal connection via pipes and can add an object with new function between each one by consisting of simple structure.

2.2. A Set of Filters

The task structure of the system should have lower subordination and degree of coupling for higher portability and scalability. The tasks of the system should be independent from each tasks and a task object should be simple in the scope without direct influence of a task to another. A simple task object can focus on free task only and it should be defined with the method to transmit data between tasks with different functions.

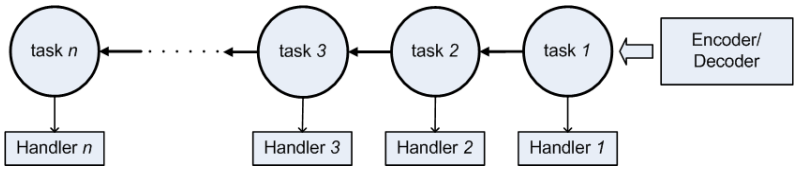


Figure 2. Linear Structure of Task Process

If it is a model for transmitting stream only, tasks of the system are made of one-sided direction and as seen in Figure 2, data passes through tasks from task 1 to task n and functions of tasks are processed. If processing structure of tasks not only transmits data but also receives packet information from other systems and then it makes a model making logs by a task analysis, it consists of two-sided structure.

If it consists of non-linear structure, non-linear graph structure has lower cost of processing time compared to merits of graph and tree structure in scheduling, buffer management and event management for independent tasks. In multimedia streaming system requiring real time transmission, reduction of processing time improves service quality and linear list structure is more efficient in operation time than graph.

The following list is for considerations when designing a system of the linear task set.

①Simple and clear task structure

②Design of independent and non-subordinate tasks

③Definition of component available for portability

④Consideration of portability with simplification task structure

⑤Data transmission between components

⑥Synchronization between components

⑦Rearrangement of components

2.3. Message Block Transmission Flow

LTS should keep consistency of transmission method, processing data transmission between tasks asynchronously in order to process data securing independence between tasks. The flow transmitting message blocks between tasks, as seen in Figure 3, transmits and processes messages and then the messages are sent to next task. This transmission flow is not a consecutive process but a asynchronously simultaneous generation of flow from task 1 to task 3 by three task objects' reception of thread, as seen in Figure 3. Independent tasks are processed in their own queues respectively by calling messages to the tasks asynchronously and dispatch loaded data to the tasks. As a result, tasks are not generated in order but processed almost concurrently by threads.

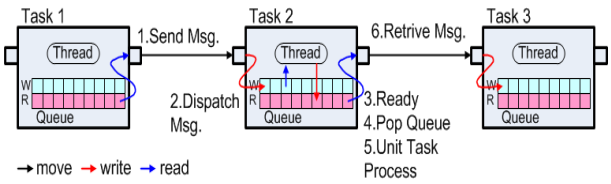


Figure 3. Message Transmission Flow between Tasks

①Transmission of messages from previous component

②Saving received messages in its own message Queue

③Preparation of its own tasks to be processed

④Reading messages in Queue for processing its own tasks

⑤Processing the received tasks

⑥Transmission of the processed messages to next task after finishing the task

3. A Component Model based on Linear Task Set(LTS)

3.1. The Concept of the Component

To make a multimedia streaming system, firstly, there are managing objects to manage data and processing objects to process data from this managing object for the purpose. For the processing objects, there are objects to support data as well as purpose to process genuine data. After processing, there should be objects to transmit them or to control its transmission according to various protocols in order to provide its service. It aims to define a conceptual component not specific one, for every multimedia streaming and to design the system to operate streaming by these components. Basic and conceptual component is defined as seen in Figure 4, in order to satisfy synchronization, portability and scalability for many objects of the system to be implemented.

A component can have more than one port available for transmitting streams with other components with its own purposes and functions. To process streams, the component assigns stream transmission between components to ports focusing on its own functions. Ports have directions and they are divided into input port to receive from outside and output port to send out to outside. As seen filter component in Figure 4, left port of the component is the input port expressed as arrow coming into the component and right port is the out port expressed as arrow going out from the component. Accordingly, source has the output port only while sink (render) showing the final result has the input port only.

Basically, source for multimedia and filter for processing source are representative objects. As seen basic component in Figure 4, source recognizes format cleary for various multimedia sources and has functions of read/write. Filter performs streaming processing functions including encode/decode/convert/extract for various multimedia and general processing of data such as synchronize/compress/detect error/sub-transform can be processed in buffer. And, in order to transmit streams processed in previous component, there is a transmitter component for processing transmission protocols including TCP, UDP and RTP. And final data processed or transmitted through defined components until now are seen to end-users by sink filter (render filters).

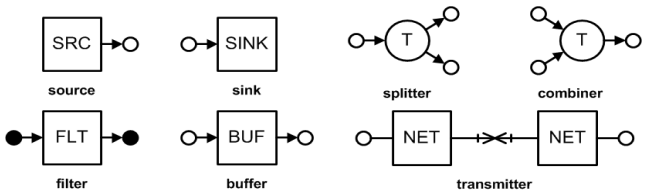


Figure 4. Basic Component

By coupling basic conceptual component objects, new complex component should be made for function of specific purpose. In complex system, in order to maintain simplicity of the component, new complex component is made by reconstructing components with specific purpose as basic component. T component in Figure 5 (a), has an input port and three output ports but internally, it consists of two splitters as seen in Figure 5 (b). Even though it is conceptional, for example, it reminds us of a component separating streams from a source component to three streams of video, left side sound and right side sound.

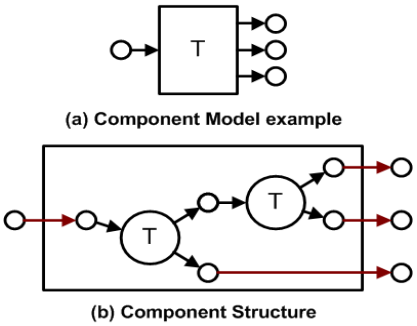


Figure 5. Composition of Complex Component

For data transmission between ports of independent components, direct push from previous component to following component may be inefficient because it may be inevitable for implementation depending on both previous and following components, to push directly. If an independent component should figure out attributes of its own previous and following components or recognize its own processing, it is not consistent with independence and if independence is not able to be guaranteed, it cannot secure speedy redesign of streaming system as well as simplicity for connections between components. For instance, if stream data loaded from source component is unconditionally transmitted to decoding filter, the filter may have to have a large amount of buffer. That is, when transmitting data between components, there are components to request actively or to transmit data at the point of request according to attributes of each component. Source component in waiting is more effective in transmitting data to the requesting filter when receiving the stream data request message.

Therefore, in order to assign data transmission between components to port object and to make roles of data request and transmission between ports clear, as seen in Figure 6, it is decided whether it is active or inactive with port attributes. Active port communicates with port calling messages asynchronously in order to observe and request connected inactive port.

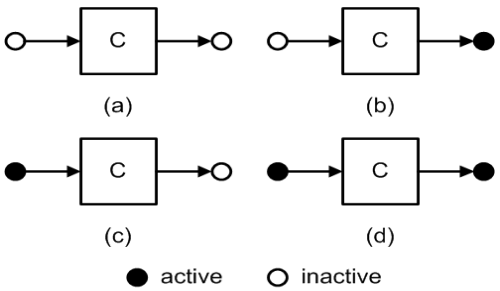
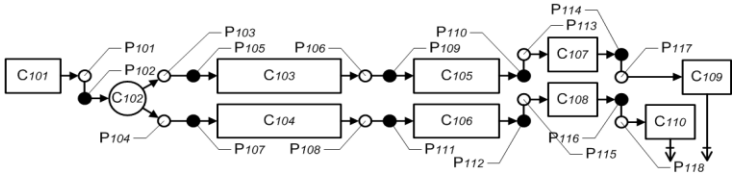


Figure 6. Component Port active/inactive

3.2. Composition of Components

The components consisting of multimedia streaming system, even though the structure of LTS suggested in Chapter 2 aims to process tasks lineally, save component information and manage information for ports to connect components in order to make them clear because relations of the components consisting of the system are complex. Information of the ports can be used as information of connection of the components and clarify the precise point for synchronization of the input port of the components. And information to secure flawlessness of the port should be managed when registering and connecting the components.



(a) Component / Port Identification Number

Comp.Table	
C_{ID}	C_{name}
C101	SRC
C102	TS
C103	Video Decoder
C104	Audio Decoder
C105	Video Controller
C106	Audio Controller
C107	BUF1
C108	BUF2
C109	NET1
C110	NET2
C111	Synchronizer

(b) Component Table

PortTable				
P_{ID}	O_{ID}	I/O	Active	R_{ID}
P101	C101	O	-	P102
P102	C102	I	A	P101
P103	C102	O	-	P105
P104	C102	O	-	P107
P105	C103	I	A	P103
P106	C103	O	-	P109
P107	C104	I	A	P104
P108	C104	O	-	P111
P109	C105	I	A	P106
P110	C105	O	A	P113
P111	C106	I	A	P108
P112	C106	O	A	P115
P113	C107	I	-	P110
P114	C107	O	A	P117
P115	C108	I	-	P112
P116	C108	O	A	P118
P117	C109	I	-	P114
P118	C110	I	-	P116

(c) Port Table

Figure 7. Example of Component Registration Table and Port Registration Table

Figure 7(a) shows a component ID and a port ID in the composed components, corresponding to server model parts respectively. If registering components, as seen in Figure 6 (b), components have its own component ID and component name and are registered in the port table in Figure 7(c) for ports own by each component. In (b), the component of C_{105} is registered in the port table in (c) with the input port P_{109} and the output port P_{110} as owners and it can be available for checking whether there are input and output and whether it is activation port. The activation input port P_{109} of the component is C_{105} connected to the non-activation output port P_{106} of the component C_{103} while the activation input port P_{110} is connected to the non-activation output port P_{113} of the component C_{107}.

The component table consists of Component ID(C_{ID}) and Component Name(C_{NAME}) while the port table is made of Port ID(P_{ID}), Owner ID(O_{ID}), I/O classification, activation classification and Reference ID(R_{ID}). Reference ID represents ID of port of other component connected to the port of the component.

In doing so, composition of components can manage easily the entire system with the conceptual design seen in Figure 7(a) and tries to make stream flow clear.

3.3. Component Port

Component may consist of multiple ports and transmits stream by connecting between ports. Port should be defined as input and output data of its own component and transmission of stream between ports does not involve directly in the components. A port of component is not directly connected to other component and as seen in Figure 7(c), its relations are managed with mapping information. As seen in Figure 8, stream of stream objects between ports are transmitted to other port in order of call through a channel. Conceptionally, a channel has a singleton concept and it monitors stream transmission between the entire components and manages stream flow. And independence between components and portability of port management such as check of validity of port connection should be considered. To do so, it is necessary for an object for the purposes of testing whether port are available for connection at the point of registration of components and connection of their ports and managing stream flow between ports and this object is the port channel object.

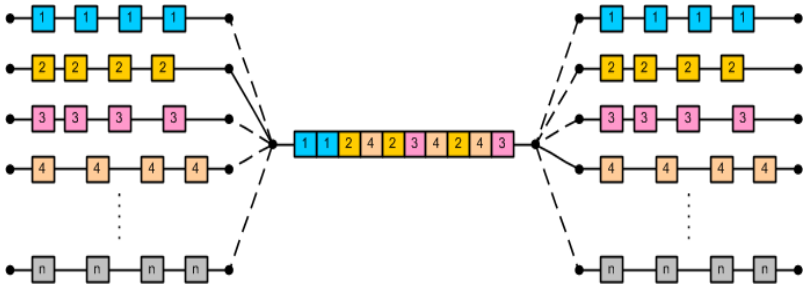


Figure 8. Concept of Channel Transmission between Ports

Although transmission between ports in Figure 7(c) is processed by asynchronous message processing, it is managed by the port channel for the structure with simple and high portability on various streams. To secure consistency for data transmission of streams of various components, it transmits streams between ports with common binary object. Independent ports should implement binary object interface for processing in port channel object and as seen in Figure 9, it transmits the object pointer to the port from the port channel. In Port Map Information of Figure 7(c), it is defined that the output port(P_{108}) of Audio Decoder(C_{104}) is connected to the input port of Synchronizer(C_{106}). P_{111} of Synchronizer calls message to P_{108} and by port channel object, the object pointer of binary object of audio stream are transmitted from P_{108} to P_{111}. The port P_{111} c which received the object pointer completes transmission registering the binary object of the object pointer to the buffer of Synchronizer (C_{106}). The port channel secures clear transmission between ports based on mapping information of streams in Figure 7(c). And then it uses the object pointer only not real stream data.

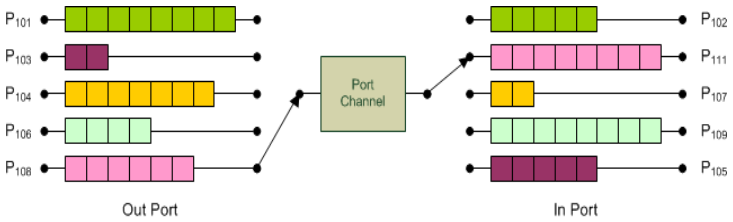


Figure 9. Types of Streaming Transmission between Ports

Unification of the port channels has purpose to implement unification of processing of various stream flows and monitoring processing and by securing independence between components, it considers flexibility and portability of the system. If there is an issue for functions of the port channel, when the internal performance improves, its effects on composing components can be minimized.

3.4. Component System Model

It aims to define conceptual components for every multimedia streaming and to design the system for operating streaming by these components. In order to satisfy synchronization, portability and scalability of a number of objects for the system, it should be able to define extended component based on basic and conceptual components. To link components based on this, it shows that it is available for conceptual system design based on mapping information of port objects.

When designing a system to support various multimedia streams, it is available for two methods, transcoding method to convert to one targeted stream, to transmit it and to convert it to desired format again not to apply one process to various formats and control method of space and time resolutions supported in MPEG-2. As seen in Figure 10, streaming system can be designed by defining conceptual components.

To be available for various formats of audios and audio sources in server transmitting streaming, transcoder to convert formats of sources, converts format to MPEG-2 stream. Converted stream supports spatial scalability through DCT hierarchial quantization by decoder.

QoS mapper extracts frames from combiner and sets temporally and spatially optimized QoS parameter information and priority information based on packet importances and user requirements. Users can choose frame play rate or low error rate in mapper or focus on resolution more than them. Therefore, it sets up priority according to user requirements and packet importances, rearrange them and save them to buffer by their priority. QoS mapper also provides parameter to adjust space and time resolutions to decoder and makes it used as parameter for adjusting temporal and spatial resolution.

Transmission controller extracts packet to be transmitted from buffer, checks its QoS parameter and transmits optimized streaming fit to the requirements. It transmits priority streams based on time stamp and priority information in the packet set in QoS. When expected network is not available for packet reaching in time, it abandons packets with low priority.

In client, output for monitering event received from network component has activation attribute. Received packets are arranged in order of time stamp and saved in butter. It notifies an event starting to transcode when data in buffer reaches both maximum and minimum of buffer amounts set in the system. Buffer monitors both maximum and minimum amounts and continues to call messages making other components refer to them. Transcoder and decoder transmit arranged packets to play buffer and convert packet data to MPEG or image format. Converted data is transmitted to synchronizer through complexer. Synchronizer synchronizes using time stamps of audio and video streams and when inner buffer is empty, input port checks if there is stream in complexer and notifies butter emptiness to the system when there is no stream. Figure 10 shows the composition model for the entire system.

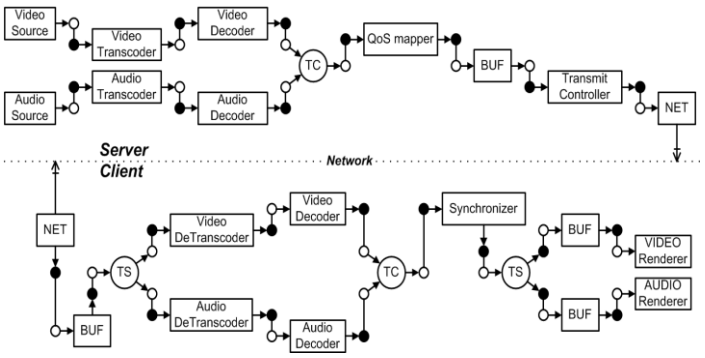


Figure 10. Component System Composition Model

4. Performance Evaluation

Generally, multimedia streaming service is evaluated in its performance by parameters including transmission rate, loss rate and jitter. Transmission rate indicating bandwidth for allocation to the service may be different according to network transmission speed, loss rate, buffer amount of component and processing capability. Client sets up network buffer and source buffer based on basic information video play rate, packet size, packet transmission allocation and pile size. Jitter makes overflow or underflow to buffer of client and it can have huge influences on streaming quality.

When offering stram service to a channel, system load is measured as information for performance parameter and as number of channels increases, evaluation is conducted according to performance criteria.

Table 1. Definitions of performance parameters

Parameter	Explanation
$T_{Response}$	Response Time for Packet Transmission
$T_{Received}$	Time to Receive Packet
T_{sent}	Time to Send Packet
θ	Packet Transmission Rate
S_{sent}	Amount of Sent Packets
$S_{received}$	Amount of Received Packets
δ	Packet Loss Rate

Response time for packet transmission is calculated by deducting time to send from time to receive as seen in the equation (1). Packet loss rate δ is measured by comparison between the entire received stream amount and received file size. The loss rate represents percentage of deduction of received amount from the entire received stream amounts in the equation (3)

$$T_{Response} = T_{Received} - T_{Sent} \quad (1)$$

$$\theta = \frac{\sum S_{sent}}{T_{Response}} \quad (2)$$

$$\delta = \frac{S_{sent} - S_{received}}{S_{sent}} \times 100(\%) \quad (3)$$

Performance evaluation is conducted according to bit rate and frame play rate per second of different audio/video when four streams are transmitted for 300 seconds by each channel. In the condition of 100Mbps Lan, the packet size is 8192bytes, frame size of ethernet is 12KB (jumbo frame), packet loss rate for video is 10-1sec, loss rate that for audio is 10-3sec and packet overdue in time is regarded as loss and abandoned.

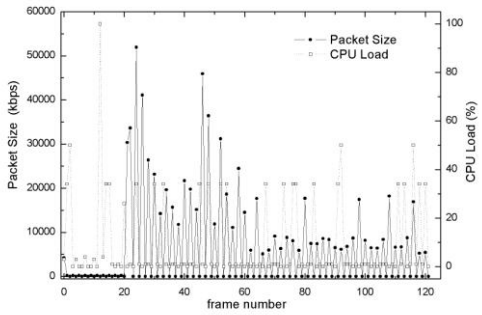


Figure 11. Comparison between Average Packet Size per Channel and CPU Load

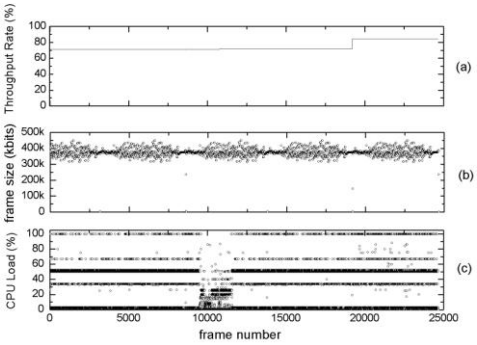


Figure 12. Processing Rate for Channel Stream and its CPU Load

In order to compare performance evaluations, it shows a service offering a basic channel. Figure 11 is the result to measure changes in packet size and CPU load for 120 frames in one channel. Processed stream(S), stream jitter(D), Load time(ω_n) and the entire buffer use rate of the components(β) were measured. For the results, when changing number of channels, CPU load and packet size are parameters to determine performances(loss rate, transmission rate and jitter). For packet size made in one channel, CPU load requires performance of 10~40%. Figure 12 shows 70~80% of processing rate for the channel $C_4$and frame size are in about 400Kbps and CPU load is around 50%.

5. Conclusion

In this article, LTS model with higher scalability was designed by modeling complex operation flow of the system for multimedia streaming service to simpler structure. And it made consistency between each task keep data transmissions. In LTS, loose control message between tasks were transmitted. However, powerful control management has effect to heighten degree of coupling between tasks only. The component was made by extending interface for control and synchronization to tasks and information for composing tasks. Components can be connected to each other and has port attribute to adjust each other's processing as well as data exchanges and is available for collaboration and as seen in Figure 10, it can be analyzed more clearly for position to manage in multimedia system. It also shows excellence through its performance evaluation.

References

[1] S. Bogaerts and D. Leake, "IUCBRF: A Framework For Rapid and Modular Case-Based Reasoning System Development", IUCBRF, Technical Report, vol. 617, **(2004)** August.

[2] A. P. Black, J. Huang, R. Koster, J. Walpole and C. Pu, "Infopipes: An Abstraction for Multimedia Streaming", Multimedia System, vol. 8, **(2002)** January, pp. 406-419.
[3] F. Wu, H. Sun, G. Shen, S. Li, Y. -Q. Zhang, B. Lin and M. -C. Li, "SMART: An Efficient, Scalable and Robust Streaming Video System", EURASIP Journal on Applied Signal Processing, Special issue on Multimedia over IP and Wireless Networks, vol. 2, **(2004)** January, pp. 192-206.
[4] T. D. C Little and A. Ghofoor, "Distributed Multimedia Objects Composition and Communication", IEEE Network Magazine, **(1990)** November, pp. 72-84.
[5] C. K. Soo, "Design and Implementation of Distributed Multimedia Streaming System based on DirectShow Framework", Doctorate Dessertation for Kunsan National University, **(2007)**.
[6] C. -Y. Yu, C. -H. Ke, R. -S. Chen, C. -K. Shieh and N. Chilamkurt, "MyEvalvid_RTP: a Evaluation Framework for More Realistic Simulations of Multimedia Transmission", IJSEIA, vol. 2, no. 2, **(2008)** April, pp. 21-32.
[7] R. Jung and S. W. Kim, "Multimedia Real Time Systems using CBD", IJSEIA, vol. 3, no. 2, **(2009)** April, pp. 69-80.
[8] S. W. Jung, S. Y. Ko, Y. Ji. Nam and D. -W. Seo, "Block Link File System with Fast Writing after Editing for Large-sized Multimedia Files", IJSEIA, vol. 6, no. 2, **(2012)** April, pp. 119-124.
[9] Y. -J. Lin, S. Lee, H. A. Latchman and B. J. Park, "EVGATOR – An Enhanced Visualization Simulator for Multimedia Networking Protocol Analysis", IJSEIA, vol. 3, no. 3, **(2009)** July, pp. 19-32.

Authors

Hyoungjin Kim

He received M.S. and ph.D. degrees in Department of Information and Communication Engineering from Kunsan National University. Currently an associate professor in Department of IT Applied System Engineering, Chonbuk National University.

Research interests: Multimedia Systems, Ubiquitous, Sensor Networks, Computer Vision, Image Processing, Smart Media

Jeakon Oh

He is currently a PhD from the IT Applied System Engineering, Chonbuk National University. LG Hitachi public sales manager is working.

Research interests: Traffic Engineering Systems, Ubiquitous, Sensor Networks, Computer Vision, Image Processing, Smart Media

Kyusoo Jung

He received M.S. and ph.D. degrees in Department of Information and Communication Engineering from Kunsan National University. Currently, He works as Data Architecture Consultant in Encore Ltd.

Research interests: Data Architecture, VLDB, Data Mining, Algorithm, Multimedia Systems