Priority Based Multimedia Streaming Control

Rustam Rakhimov Igorevich [1], Pusik Park [2], Daekyo Shin [3], Dugki Min [4]

[1,4.] DMS Laboratory, Department of Computer Science, Konkuk University, 120, Neungdong-ro, Gwangjin-gu, Seoul 143-701, Republic of Korea,
[1,3,2.] Korea Electronics Technology Institute, 22 Daewangpangyo-ro 712beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do 463-400, Republic of Korea
[1.] rustam@keti.re.kr, [2.] parksik@keti.re.kr, [3.] dkshin@keti.re.kr, [4.] dkmin@konkuk.ac.kr

Abstract: Usage of multiple cameras has become quite popular in various domains. The final output is usually displayed to the human who will watch and make a decision. When the number of cameras increases it is difficult to handle the outputs by single person. To overcome this issue, smart cameras and intelligent control center systems are created. This work is about handling video inputs from multiple cameras, by prioritizing the scene context. In this paper we suggest our architecture for efficiently building priority based multimedia streaming control system.
[Rustam Rakhimov Igorevich, Pusik Park, Daekyo Shin, Dugki Min. **Priority Based Multimedia Streaming Control.** *Life Sci J* 2014;11(9):550-556]. (ISSN:1097-8135). http://www.lifesciencesite.com. 91

Keywords: Priority, Multimedia, Stream, Context, Player, Server, Smart Camera

1. Introduction

Multiple camera usage has broadened its field in scenarios such as sports, security surveillance, smart meeting rooms, smart homes and other kind of N Screen services. Switching between multiple cameras to browse the real-time situation is important issue. Since person can concentrate on one screen to analyze it and react accordingly.

Philosophically the single stream selection among multiple resources can be compared with era of information. Where we have dozens of information out there in internet and one person is incapable of studying all of them all alone. That's why human being started learning how to surf among this huge amount of information flow to get exactly what they want, to satisfy their need. That's how the search systems such as google become important part of our life. On the other hand multiple stream sources in a single place causes similar problem that was caused in information era. To make it clear it is good to consider simple scenario where the security command center browsing dozens of screens. It does not cause big trouble when there are only 50 CCTV's are browsed at real-time, but if one person should track of thousands of camera inputs, it would become impossible to follow all of them. To solve this sort of problem the similar solution with the information era was applied. Context extraction and detection comes as a solution here. The smart algorithms will browse the video streams to detect some events or perform search of specific actions. That way thousands of incoming video streams from CCTV's can be reduced to few screens. Another scenario is where the smart house's security system with the multiple cameras presents on a single display the important view when unexpected situation occurred. The architecture of the scenario with the smart house surveillance system is illustrated in this paper.

The next step after context extraction and detection was not important until recent time, where the number of input streams has been increased tremendously. The importance of priority of the stream becomes next important issue to resolve. More exactly asking the questions such as: What is the priority of the stream? How to assign the priority to the stream? Where and when the priority should be assigned to the stream? These are the questions remains without clear answers and definitions. In this paper we try to give some clue and solutions to these questions by presenting priority based streaming control architecture. Although there are many works have been done on topic of context extraction and detection not much of works are concentrated on the selection of the prioritized contexts.

2. Material and Methods

There are many works have been done related to the topic of selection of the video stream. There are different purposes of switching between multiple streams into one single channel. These purposes can be best described in real scenarios. In some scenarios the best-view selection is important, another cases the hot issue or event occurring views become more important. Inductively it could be generalized as priority based selection, where the priority can be considered as a variable which decides and selects one of the streams to put in the main stream channel.

Selection can be divided into two parts, namely methods on content analysis and camera selection (or view planning). Content analysis also includes ranking mechanism to assign the rank to

each stream from the array of multiple streams. The process of assigning rank comes when the content analysis performed. Camera selection process can be considered as an action performing part of the stream switching process. On this part the decision is made and performed as an action.

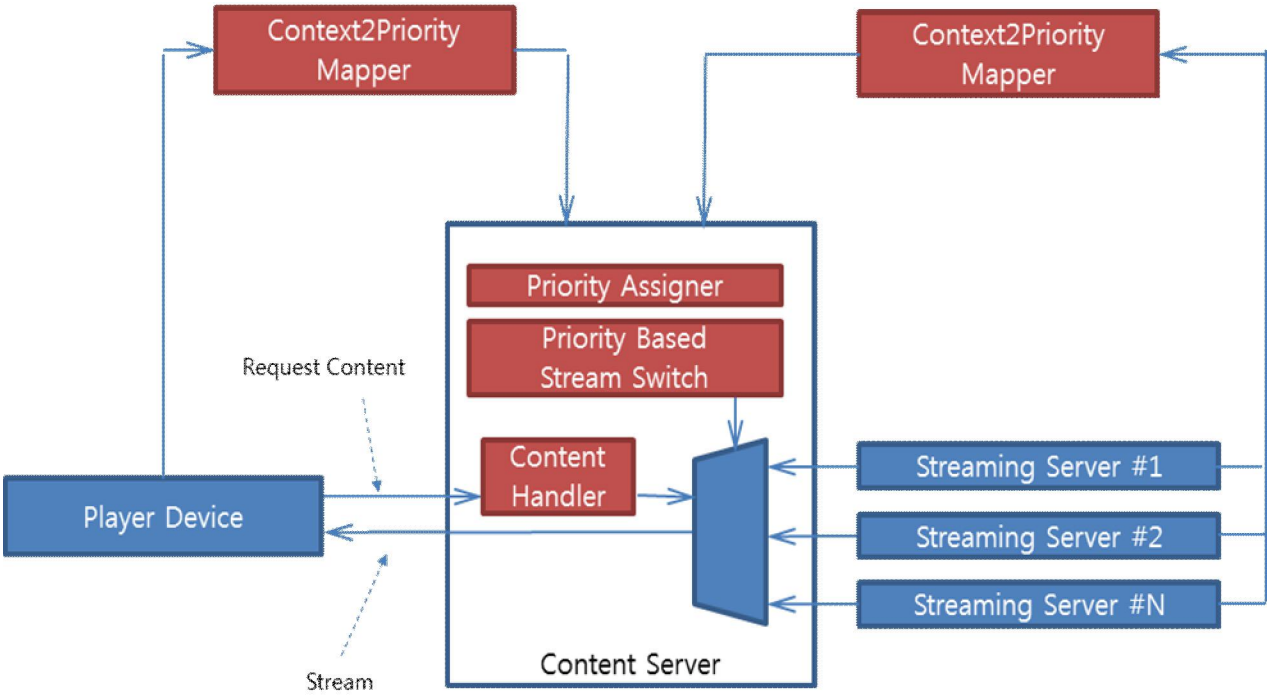

Figure 1. Streaming Through the Content Server

In [1] authors have built security services for context-aware environments, where they mostly focused on security-relevant contexts. Their architecture was designed in a rule based approach where the Authorization service is responsible for retrieving policy definitions, to check particular requests which are granted or denied.

Another content-aware based multi-camera selection technique was presented in [2]. They have done analysis of objects by their features that are extracted from the frame-level. First objects are detected using color-based change, next is by analyzing trajectory information, which is generated by matching multi-frame graphs. As a final step they generate object score by analyzing multiple features such as size and location of the object. This work can be used as a context analyzer module in our suggested architecture. It can be located on server side or could be embedded directly into the smart camera modules.

Camera selection or view planning is another important part of the stream switching process. It considers physical constraints such as the scheduling interval, oriented speed and location of sensors in the network. Switching between channels should be done in an intelligent way to keep main goal of the system without change. For example in a surveillance system, the camera switch should be performed where it follows the surveying object. If next switch of the stream suddenly shows something meaningless or totally different perspective, then it does not satisfy the user of surveillance system.

3. Architectural Overview

This section describes overall architecture of Priority Based Streaming Control and various possible configurations between Player and Server devices. Two kinds of Server devices were described in this work: Content Server and Streaming Server. Streaming Server is mostly describes ordinary source of stream with camera devices, context extraction and context detection modules. Operations related to priority (such as assigning or extracting priority) performed on Content Server side. It's designed that way for the sake of releasing Content Server from the context detection related overloading works. Context2Priority mapper is another important component of the architecture; it is used to generate priority from the given context.

The basic components for design architecture of Priority Based Streaming Control were derived from the general N-Screen service architecture. Primitive architecture for N-Screen

service guideline contains basically three tiers: Player, Streaming Server and Content Server.

Since this work concentrates on streaming technologies, there are two Streaming Models can be given in this scope: By means of the Content Server and Direct Streaming.

In a Streaming through the Content Server configuration (Figure 1) the Content Server plays central role for interconnecting the Player and Streaming Servers. It can define the priority for player and get full control on client's display.

Streaming Server and Player Devices are discovered and saved in a Registry block. Priority Assigner is the key block that is responsible for assigning priority to the Streaming source servers. It can be configured manually before actually the whole system was started.

Priority Based Stream Switcher directly interacts with the Context2Priority Mapper that is logically located outside of the Content Server (by decision of the developer it can be merged with the Content Server Implementation). It is used to collect user's context information to provide more high quality services. Local context trackers can be built-in the player device. Local context trackers can also be started as an independent device.

Context2Priority mapper is one of the important contributions in this work. It receives the context change event from the Streaming Server or Player device. Generated context can be defined in a contextual structure. Context2Priority works based on static or dynamically generated table.

Content Server plays a central role in a process of Player and Streaming Server interactions. If Content Server misses, then player and streaming server should behave in a peer-2-peer notion. In that case control over priority of streaming source depends on Server. Priority Assigner sets up the priority for each discovered input stream. Priority Based Stream Switcher is activated when the priority change event occurs. Priority Based Stream Switcher interconnects input and output streams by reading received priority information.

The Content Server also plays Brokerage server's role, it can forward streaming messages like a gateway. But re-streaming incoming stream to the destination would give a big load to the Content Server. Especially in case when there are multiple streams should be bound with multiple destination player devices.

Simplified streaming server structure is illustrated in a Figure 2. Context Detector receives streamed video from each camera and performs image processing operation. It contains context detection mechanism that will trigger a time based event when something unexpected happened in the image.

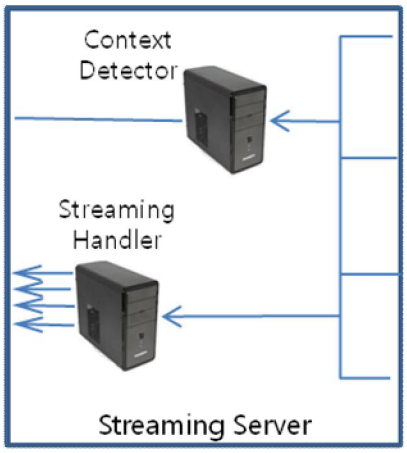

Figure 2. Simplified Streaming Server

Streaming Handler server receives all camera frames and performs multiplexing and encoding into channels, to forward it to Content Server. Streaming Handler does not contain any intelligent operations, but still it gets quite busy and loaded depend on the number of input streams. Streaming Handler can be exchanged with the Multimedia boards with the number of input streams from different hardware interfaces. There are number of existing solutions with embedded boards to perform Streaming Handler's tasks.

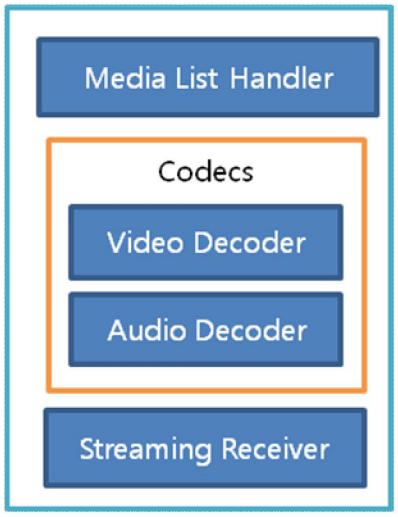

Figure 3. Simplified Player Device

Player device contains three major modules: Media List Handler, Codecs (contains video and audio decoders as sub-blocks) and Streaming Receiver (Figure 3). Those modules are quite common for the Player devices. To fully converge and cooperate in this priority based streaming

network the Player device should be forbidden to ask Streaming Server about content list in a direct manner. To restrict player from this kind of operation two ways are possible:

- Client implementation should not contain the content request directly. This solution could solve the problem easily but then Player loses flexibility in other situations.
- Second approach is less conservative and related to the Streaming Server implementation. Where the server should block any request for the contents unless requester is not proved as a Content Server.

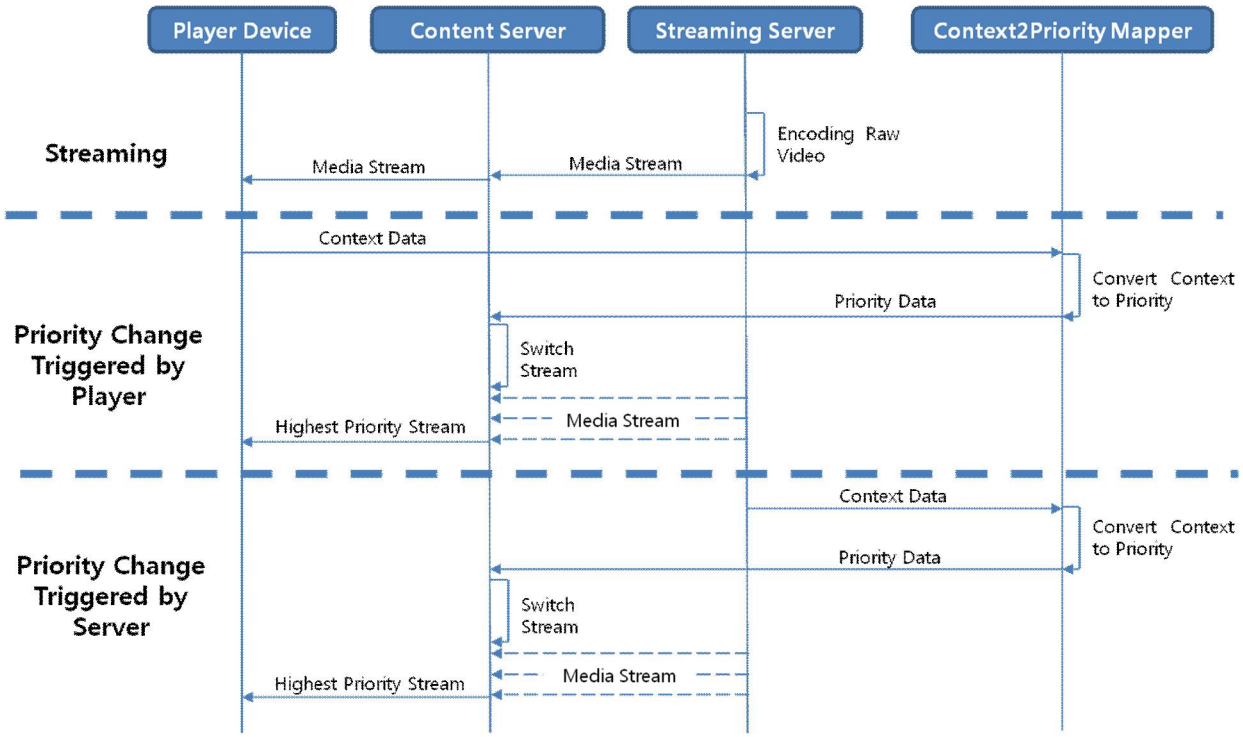

Figure 4. Stepwise Overall Sequence Diagram

There are some requirement lists were drawn from the architecture designing process:

- Streaming (Source) Server creates videos and delivers them to the Content Server. Streaming Server should not have a control over priority
- Priority should be configured by third side server (Content Server)
- Priority is derived from the contexts by the Context2Priority Mapper which gathers contexts.
- Content Server should multiplex videos to the Player based on the priority.

State of low device capability, is the point where priority based multimedia streaming becomes important and required. Low device capability state can be defined by following factors:

- The number of screens in the current context is not enough
- The size or computational power of the device is not enough to provide multi screens on same device

Sequence Diagram

Overall sequence diagram demonstrates how the information flows in a ladder diagram view. Sequence diagram contains four major blocks that are playing most intensive interaction role: Streaming Server, Content Server, Player Device and Context2Priority Mapper.

It is important to note here that not all modules in terms of Content Server and Context2Priority Mapper location are fixed. Depend on the developer or application requirements those could be merged or distributed. This statement also related to the internal sub modules of Content Server, such as Priority Assigner and Priority Based Stream Switcher.

Streaming Sequence Diagram is illustrated in a Figure 4. It is divided into three sections horizontally. Originally it supposed to be three separate sequence diagrams: Streaming, Priority Change Triggered by Player and Priority Change Triggered by Server.

When the context is changed or some context event happened, Player device and Streaming Server generates Context Data packet. Context Data Packet forwarded to Context2Priority Mapper. Context2Priority mapper decides the priority of the context data in its input. Resulted Priority Data goes

to the Priority Assigner block in Content Server. Priority Assigner defines what kind of actions should be performed based on received message. If there is some new change should be done in a real-time streaming pipe, then Priority Based Stream Switch would be activated. It can switch main channel to requested one easily. That way higher priority media becomes an active channel.

4. Priority Layers

It is vital to distinguish events by their importance factors. Layered priority definition is one easy way to represent priorities.

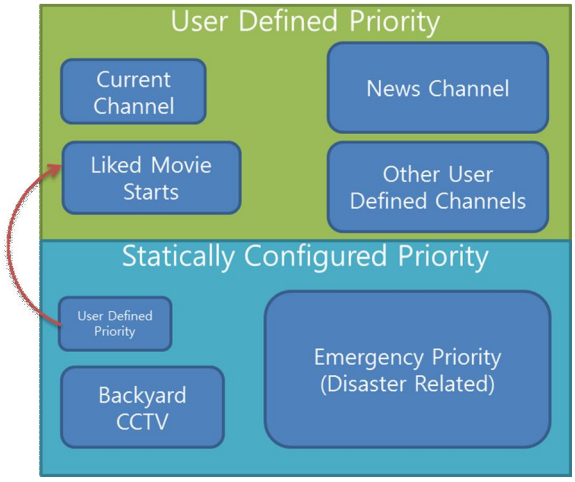

Figure 5. Two layered Priority Factors Example

The layered diagram in Figure 5, illustrates priority definition. There are two classes: Statically Configured Priorities and User Defined Priorities. Statically configured priority contains emergency priority, which contains all kind of disaster related blocks. Since it is configured for specific house, we consider they have Backyard CCTV, it has second highest priority, and it just comes out from the security factors. User defined priority is another static priority block. This priority layer can be switched and defined by the user. For example user might define, the some sport event on some channel as an important, and no matter what is on current channel, when the sport event is started it should become on a main track, because it has been assigned a higher priority than other channels.

Example Scenarios

Example scenarios are the good representation in order to explain suggested idea of Priority based streaming interface. There are two scenario schemes are given in this work. First scenario contains multiple dummy camera devices that are connected to streaming server. Second scenario contains multiple smart camera solutions where each camera is able to handle its images to do some processing and decision making operations.

Both example scenarios considered in a modern house surveillance system with multiple cameras, which are installed all around the house. The main intention of the system is to keep house secure and safe. We are not going deep into the details of how the system is build, what kind of hardware/software platforms are used and what kind of context awareness technologies applied. The main focused issue point in this work is how the cameras input are shown to the user, and all other topics are out of scope of this work. System should decide the priority of the cameras by their given context information. The camera with the highest priority should be streamed into main screen of the User.

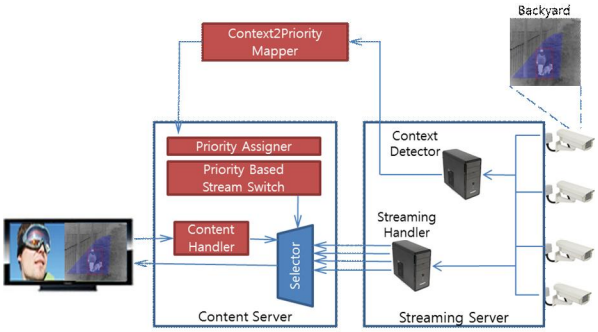

Figure 6. Scenario-I With Dummy Camera

Scenario I: With a Dummy Camera

Multiple dummy camera solutions are connected to Streaming Server, where inside Streaming Server two high computation processor devices are functioning: Context Detector and Streaming Handler (Figure 6).

As it has been mentioned above the Streaming Handler server receives all camera frames and performs multiplexing and encoding into channels, to forward it to Content Server. Streaming Handler does not contain any intelligent operations, but still it gets quite busy when the numbers of inputs are increased.

Context Detector receives streamed video from each camera and performs image processing operation. It contains context detection mechanism that will trigger a time based event when something unexpected happened in the image. Depend on the camera and its location the context detection algorithm can vary. For example for the backyard camera, it should trigger almost any time when somebody appears in the screen, otherwise any motion on backyard should cause the event triggering.

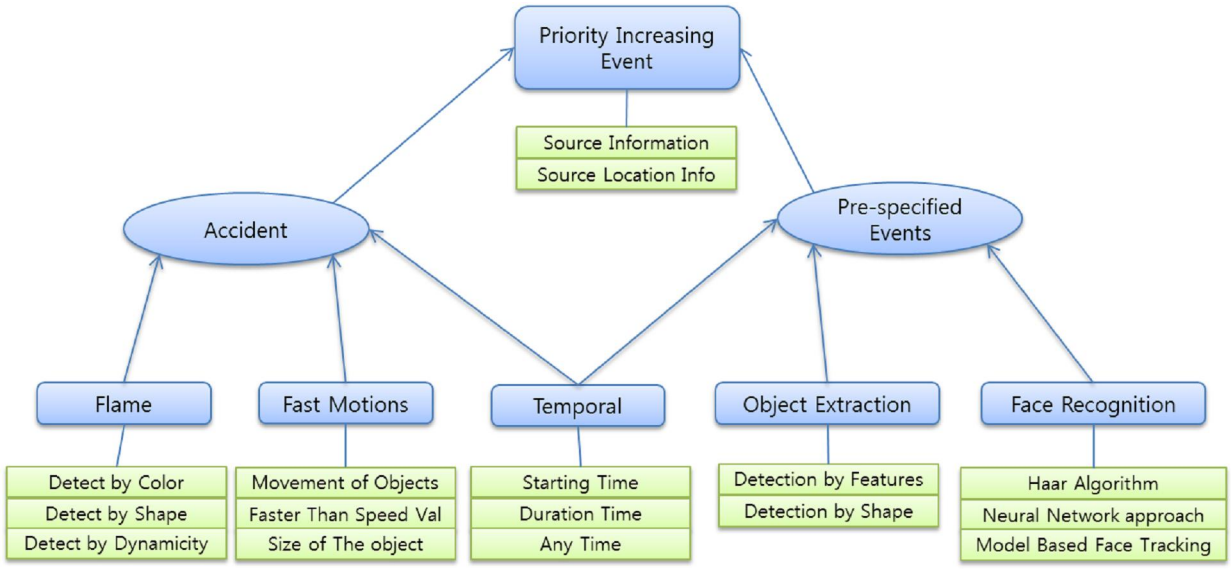

Figure 7. Use Case of Priority Increase Event

Compared to the backyard camera the front door camera shouldn't react on any passing person. It should trigger event when something unusual happens, such as, when somebody doesn't moves away for a long time. It gives some doubts to the Context Detector. More context features are involved for the front door camera processing compared to backyard camera processing.

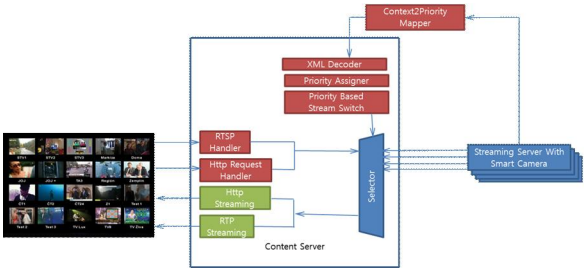

Figure 8. Scenario-II With Use of Smart Cameras

Scenario II: With Use of Smart Camera

In the Scenario II (Figure 8), considers the usage of Smart Camera where the camera is itself contains the set of image processing and context detection algorithms. Main difference with the Scenario I, is that Context Detector server and Stream Handler servers are missing. Stream Handler server is not needed because each Camera would send their own streams to the Selector block located inside the Content Server. Since these are the Smart Camera's each of them contains Context Detector block inside, which means each of them able to inform Context2Priority Mapper about context changes. Since last decision will be made by Priority Based Stream Switch block, it doesn't matter which of those cameras are going to send context event information first.

Scenario with usage of Smart Cameras has huge advantage compared to the dummy camera powered scenario. Distribution is the main power of the second scenario. Since every camera represents context awareness module, the streaming server cannot be single point of failure in that scenario.

Priority Use Case Example

Based on those given example scenarios priority use cases it becomes possible to build semantics of reasons that cause the priority changes. For example in previous scenario examples the priority change event triggered priority increase. Depend on the example scenario the event can trigger different actions, but let's assume for our example it triggers priority increase event.

As it is illustrated Figure 7 the cause of Priority Increase can be triggered by two events: Accident and Pre-Specified Events. As an accident we are considering Flame, or Fast Motions (or non-ordinary motions that are usually occurred when the some emergency and accident situation is happened). Those factors of flame or fast motions can be described and tracked by the predefined features. For example Flame is caused by Color, Shape and Dynamicity of context. Fast motion are mostly described by dynamicity, speed and size of the objects. Of course in a more real examples number of those pictures increased.

On the other hand Pre-Specified Events can be considered as a user defined events where some object detection or face recognition algorithms comes

ahead. Those processes contain their own application specific features, and application specific algorithms. Haar, Neural Network and Model Based Face Tracking algorithms are used for face recognition block. That can cause event when the unknown face is detected in front of our door or somewhere near house.

Even though there are two separated events both of them share common temporal feature. Temporal feature is important, because any context awareness system needs the timeline, to make a decision. Depend on the timeline the detected features and decisions are changes. For example flame detection system can be different depend on the timeline.

Any triggered event of the events triggers chain event towards the head of events semantics. If the triggered event passes the all event blocks it reaches the head of the event block. At the top of this semantics Priority Increase Event is located which is going to cause priority change action. To explain that chain reaction of the events illustrative example is required. For example the system detected too many motions where a lot of active actions are happening. But those motions were caused by kids who are playing out there in the front yard of the house. Fast Motion detection module is going to detect too much movements and fast motions. It will trigger an event of accident. When accident module receives the event from bottom level, it should make a conclusion considering other factor. In this case temporal feature gives that it is daytime and contains information about kids front yard playing schedule (in timing sequence). Using this information the Accident event block understands it is fault alarm and cancels the event chain.

```
<?xml version="1.0"?>
<s:Body>
    <u:assignPriority xmlns:u="urn:schemas-upnp-org:service:serviceType:1">
        <channel_id>2</channel_id>
        <priority>3</priority>
    </u:assignPriority>
</s:Body>
```

Assigning priority was described and enlisted in a xml format. Above the body content of enveloped upnp soap message is given. Current streaming channel always should have highest priority (priority=1). When the user sets some specific channel it becomes highest priority channel. When the event happens the highest prioritized stream channel is changed.

5. Conclusion

In this work the concept of Priority Based Control system was designed which is applicable to many different areas, such as Smart Security Surveillance systems, N-Screen Services and other domains where the multiple multimedia streams should be processed. It is not finished work that's why this paper was constructed in work in progress report style. In a future works implementations and experimental comparative analytical results would be presented by new publications.

Acknowledgements:

This work was supported by the IT R&D program of MOTIE/KEIT. [Project No.: 1004125, C&D Infotainment System Development and Software Eco-System for the Future IT-Convergence Automobile].

Corresponding Author:
Pusik Park
Senior Researcher
Korea Electronics Technology Institute.
Seongnam-si, Gyeonggi-do 463-400, Rep.Korea
E-mail: parksik@keti.re.kr

References

1. Michael J. Covington, Prahlad Fogla, Zhiyuan Zhan, Mustaque Ahamad, "A Context-Aware Security Architecture for Emerging Applications" Computer Security Applications Conference, 2002. Proceedings. 18th Annual. 2002;249-258.
2. Fahad Daniyal, Murtaza Taj, Andrea Cavallaro. "Content and task-based view selection from multiple video streams" Multimedia Tools and Applications, January 2010, Volume 46, Issue 2-3, 235–258.
3. G Thomas Schierl, Yago Sanchez de la Fuente Ralf Globisch, Cornelius Hellge, Thomas Wiegand "Priority-based Media Delivery using SVC with RTP and HTTP streaming". Multimed Tools Appl 2011, 55: 227-246.