Karthik Venkata Krishna Nagabhushana

External Media Announcement Approach For Media Resource Function Processor

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 28.03.2012

Thesis supervisor:

Prof. Jukka Manner

Thesis instructor:

Leena Pitkäranta, Oy LM Ericsson AB

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Karthik Venkata Krishna Nagabhushana

Title: External Media Announcement Approach For Media Resource Function Processor

Date: 28.03.2012 Language: English Number of pages:9+61

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Prof. Jukka Manner

Instructor: Leena Pitkäranta, Oy LM Ericsson AB

Announcement application service is a multimedia service in IMS which plays media (audio/video) announcements to the User Equipment (UE). This thesis studies Media Resource Function Processor (MRFP), a key network component in IMS which is responsible for the announcement application service. This thesis looks into the MRFP architecture and tries to find a feasible and alternative approach to handle media announcements. The factors this thesis concentrates on are: (1) primarily, the limited availability of memory to store the media announcements within the MRFP, and (2) the possibility to reduce the number of Digital Signal Processors (DSPs) reserved for playing announcements.
For a possible solution to the above mentioned research questions, this thesis proposes an external announcement approach to provide the announcement service. This is studied by using a separate server outside the MRFP node to store all the media announcement files. In this thesis, the external announcement approach is studied by using three different protocols - RTSP, HTTP or FTP as the interface between the MRFP node and the external server. The impact on the software architecture of MRFP due to the external announcement approach is studied. Also, a prototype is built to test the performance of the external announcement approach in an emulated Wide Area Network (WAN) environment. One other key area of study made in this thesis is to use the existing limited MRFP memory as a cache. The number of requests made to the external server to fetch the announcement files can be decreased by implementing an effective caching algorithm in the MRFP, which improves the performance. The study made shows that Greedy Dual-Size Popularity (GDSP) algorithm can be a relevant and very effective caching mechanism in MRFP. Through this work, it is demonstrated that an effective announcement service can be achieved in MRFP by using an external server.

Keywords: IMS, MRFP, Announcement Service, External Media Announcements, RTSP, HTTP, FTP, Caching algorithm

Preface

This Master's thesis work was carried out at the Research and Development Center of Oy L M Ericsson AB, Jorvas, Finland. During this thesis work, I have received help from many talented individuals.

I would like to thank my supervisor Prof. Jukka Manner for his encouraging comments and suggestions on the work undertaken. I would like to express my gratitude towards my thesis instructor Leena Pitkäranta at Ericsson for her continuous support during this thesis work. I would like to thank Node Architect at Ericsson, Olli Hynönen for his valuable ideas, guidance and expertise.

Espoo, 28.03.2012

Karthik Venkata Krishna Nagabhushana

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
AS	Application Server
AuC	Authentication Center
CDR	Charging Data Records
CSCF	Call Session Control Function
DSP	Digital Signal Processor
DTMF	Dual-Tone Multi-Frequency
DTP	Data Transfer Process
FIFO	Fist In, First Out
FTP	File Transfer Protocol
GD-S	Greedy Dual-Size
GDSF	Greedy Dual-Size Frequency
GDSP	Greedy Dual-Size Popularity
HLR	Home Location Register
HLRU	History-based Least-Recently Used
HR	(Cache) Hit Ratio
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
I-CSCF	Interrogating-Call Session Control Function
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ISC	IMS Service Control
LDAP	Lightweight Directory Access Protocol
LFU	Least-Frequently Used
LRU	Least-Recently Used
LRV	Lowest Relative Value
MAC	Media Access Control
Mb	IP interface for Multimedia
Megaco	Media Gateway Control Protocol
MGCF	Media Gateway Control Function
MGW	Media GateWay
Mp	Media-processing interface
MPEG	Moving Picture Experts Group
MPM	Multi-Processing Module
Mr	Media-resource interface
MRF	Media Resource Function
MRFC	Media Resource Function Controller

MRFP	Media Resource Function Processor
netem	Network Emulator
O&M	Operations and Maintenance
OSI	Open Systems Interconnection
P-CSCF	Proxy-Call Session Control Function
PSTN	Public Switched Telephone Network
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
S-CSCF	Servicing-Call Session Control Function
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLRU	Segmented Least-Recently Used
tc	Traffic Control
UA	User Agent
UE	User Equipment
URI	Uniform Resource Identifier
VoIP	Voice over IP
WAN	Wide Area Network

1 Introduction

In recent years, using Internet to consume and share multimedia information has increased enormously. IP Multimedia Subsystem (IMS) provides a framework to deliver such information in a reliable and efficient way. IMS defines an architecture and framework converging data, speech and mobile network technology over an IP-based infrastructure. IMS is advantageous both from the end-user and the service provider standpoint. With the IMS architecture it is easier to create and deploy new applications and services.

Media Resource Function (MRF) is the part of the IMS architecture which performs media related functions in the core network. MRF provides functionality to manage and process media streams such as voice, video, text-to-speech and real time transcoding of multimedia data. MRF is mainly involved when a network service like playing announcements and mixing media streams is required by an IMS application. Each MRF in the network can be divided into Media Resource Function Controller (MRFC) and Media Resource Function Processor (MRFP). MRFC is the signaling plane which interfaces which other IMS network elements using Session Initiation Protocol (SIP). MRFC interprets the information from Application Server (AS) and Servicing-Call Session Control Function (S-CSCF) and controls the MRFP using H.248 protocol. MRFP is the media plane which does the media processing based on the instructions from MRFC. Figure 1 is a simple representation of part of IMS architecture showing various interfaces used with MRFC and MRFP [2]. As shown in Figure 1, the communication between AS and S-CSCF is handled by using a SIP-based IMS Service Control (ISC) interface and the communication between S-CSCF and MRFC is handled through the SIP-based Mr interface. The AS invokes SIP messages to MRFC to play network tone and announcements. MRFC controls the MRFP via the H.248 protocol based Mp interface. The processed media by MRFP based on commands from MRFC are received via the RTP-based Mb interface.

Fundamentally, playing announcements involves playing media to user's terminal device. Announcements can be in different forms such as stored static media files (for example, an `.mp3` audio file), media files generated in real-time (for example, a `.mp3` audio file generated in interactive voice response), media streams generated in real-time (for example, a `H.264` video stream on making a video call), multimedia objects, or combinations of the above [1]. Announcements are invoked by IMS application services using Uniform Resource Identifier (URI) which contains the information about the location of the media file to be played. The details on how the network announcement service is handled depends on the MRFP solution. This thesis mainly focuses on the network announcement service in one such MRFP solution [3].

1.1 Problem Statement

Traditionally, the media announcement files that are processed and played by MRFP are stored within the network element (MRF) itself. This approach has two main

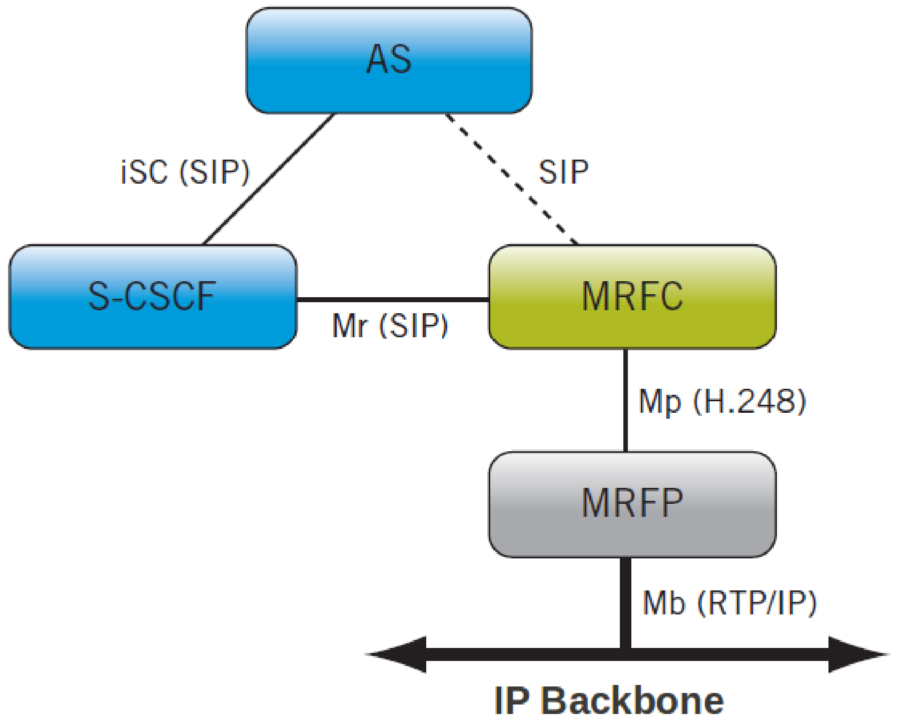

Figure 1: IMS Network Element Topology. ([2], modified)

challenges: 1) It takes huge amount of memory to store all the announcement media files, which makes it unfeasible to store them within the MRFP node. Video announcements, for example can be of several megabytes and storing such files internally is a critical area to optimize in MRFP solutions. 2) Traditional approach also leads to reservation of additional resources in the network element like Digital Signal Processor(s) (DSP) for playing announcements.

In this thesis work, the aim is to study different ways to tackle the above mentioned challenges by configuring and implementing an alternative approach to play the media announcements in MRFP. The main idea behind this approach is to store all the media announcement files externally in a separate media server outside the MRFP. When MRFP receives the command from MRFC to play the media announcement, it will get the URI pointing to the location of the announcement in the media server to be played. MRFP uses the URI and plays the announcement to the required IP address corresponding to the user's terminal device. The external announcement approach is implemented using three different protocols for transaction between MRFP and the media server namely Real Time Streaming Protocol (RTSP), Hypertext Transfer Protocol (HTTP), and File Transfer Protocol (FTP). Playing media announcements can be more efficient if they are stored in the MRFP itself as there would be no external media server transactions. However, this thesis tries to use caching algorithms to make best use of the existing limited memory in MRFP.

1.2 Thesis Outcome

The work shows that the announcement service in MRFP can be made more feasible by using an external server to store all the announcement media files. A prototype built to test the external announcement architecture shows that it is possible to use RTSP, HTTP and FTP as possible interfaces between MRFP and the server. The prototype is tested with audio and video announcements with emulated Wide Area Network environment by introducing variable delays and packet loss between MRFP and the external server. The results from the tests show that the decision on which interface (RTSP, HTTP or FTP) to use depends on the properties (size, format, etc.) of announcement files. Also, various caching algorithms are studied to use the limited memory in MRFP as a cache and the GDSP caching algorithm turns out to be most suitable for MRFP.

1.3 Outline

The document is organized as follows. Section 2 gives a detailed background on the MRFP solution architecture and the existing media announcement approach. Section 3 describes the new announcement approach using an external media server. It also gives a detailed description of the different protocols used to achieve this. Section 4 describes the importance of effective utilization of limited memory in MRFP as cache memory and it also outlines algorithms which can be used to achieve the same. Section 5 describes the prototype implementation to achieve the external media announcement. Section 6 explains the measurements taken from the implemented new approach. It also does an analysis and comparison of measurements (like latency) when different protocols are used for external server transactions. Section 7 concludes the thesis work mentioning the benefits and limitations of external announcement approach and looks into the possible future improvements.

2 MRF Architecture in IMS

This chapter gives an overview on IMS explaining IMS architecture, its benefits and the role of MRF in IMS. The goal of this chapter is also to give a detailed description of the architecture used to play media announcements using MRFP. Essentially this chapter intends to provide all the information required to gain better understanding of future sections of the thesis.

2.1 Overview of IMS

This section gives an introduction to the layered architecture of IMS and also explains the functions of different core network components in IMS.

2.1.1 IMS Architecture

IMS is an internationally recognized standard, first specified by Third Generation Partnership Project (3GPP/3GPP2) [5]. IMS standard provides a generic architecture for Voice Over IP (VOIP) and multimedia services. It fills the gap between two most successful communication paradigms, cellular and Internet technology. It also provides real-time voice and data services over packet-switched network via SIP infrastructure. The IMS architecture supports variety of services enabled based on SIP. The purpose of the IMS architecture is to provide various multimedia services that can be accessed by a user from various devices via IP network or traditional telephony system.

IMS provides a horizontal, layered network architecture which enables common functions to be reused by multiple applications. This also enables inter-operability and roaming, and provides bearer control, charging and security [4]. As shown in Figure 2 the layers in IMS architecture can be classified as follows.

- **Service layer:** The Service layer is essentially the application services layer where all the services live. This layer comprises of IMS Application Servers which host and execute services in IMS environment. IMS specific Application Servers are also called SIP Application Servers as they use SIP as interface protocol. Service layer also defines interfaces based on IMS standard to achieve common functionalities like identity management, billing services, etc [7].

- **Control layer:** The Control layer controls the authentication, routing, and distribution of IMS traffic between Service layer and Connectivity layer. This layer comprises of network components for session set-up, modification and release. The control layer hosts two most significant network elements in IMS namely Home Subscriber Server (HSS) and Call Session Control Function (CSCF). Control layer also provides support functions for provisioning, charging and Operations and Maintenance (O&M). Also, control layer supports the inter-working between different kinds of networks via border gateways.

- **Connectivity layer:** The Connectivity layer is responsible for network access. It allows different IMS devices to connect to the IMS network. This layer comprises of IP backbone along with traditional networks like PSTN. The responsibility of this layer includes IP provisioning and registration of devices with the higher layers.

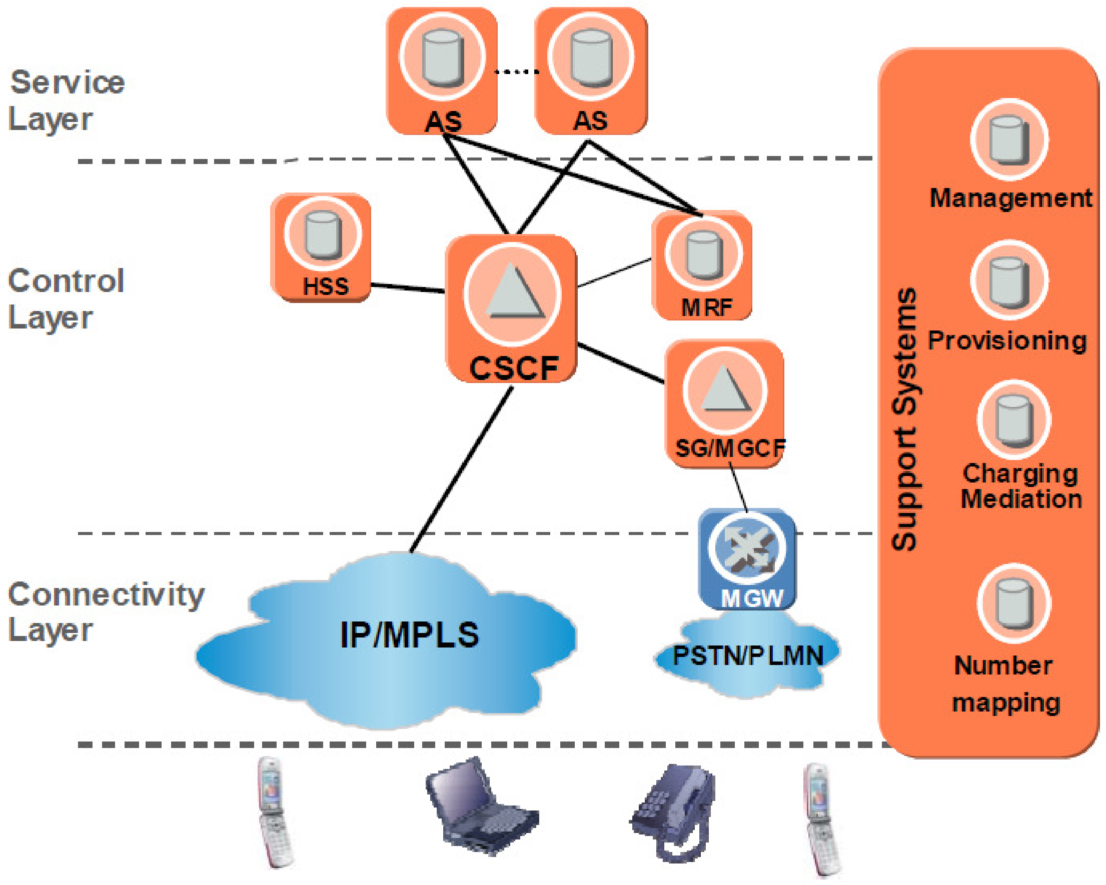

Figure 2: IMS architecture. [4]

2.1.2 IMS Core Network Components

IMS core network comprises of many components. Some of the key components and their functionalities are explained below [6].

- **Application Servers (AS):** AS are the components which host and execute the logic of IMS services. SIP AS processes SIP messages related to an IMS service. AS may host variety of services like voice-mail, announcements, interactive voice response, Push-to-talk over Cellular (PoC), etc. Each AS is capable of hosting multiple services and this has significant advantages with regard to loading of IMS core network nodes [4]. AS invokes requests to MRF for media related services like playing network announcements and this feature is discussed in detail in Section 2.2.

- **Home Subscriber Server (HSS):** HSS is a component in the control plane which is a centralized repository for all the subscriber information. HSS, also called User Profile Server Function (UPSF) is a master database which provides information to all other IMS network entities when required. HSS mainly includes information about subscriber registration (name, address, services, etc), subscriber preferences (call baring/forwarding settings etc), subscriber location and service-specific data. Using these information, HSS performs subscriber authentication and authorization [8]. HSS is similar to Home Location Register (HLR) and Authentication Center (AuC) in GSM. In IMS environment, AS and CSCF are the entities which interact with HSS.

- **Call Session Control Function (CSCF):** CSCF, also known as SIP server is the central component in the control plane which processes SIP signaling messages in the IMS. CSCF which uses SIP as protocol provides various network services like session control services (subscription, registration, routing, roaming, etc), central service based charging, secure authentication using HSS, Quality of Service control, etc. CSCF plays a key role in providing independence between different application services hosted by AS [6]. It is achieved by defining SIP-based ISC interface with standard filters between AS and CSCF as shown in Figure 1. CSCF signaling functions can further be split into three SIP servers:
  - Proxy-CSCF (P-CSCF): This is the first point of contact for the device or User Equipment (UE) and it controls authentication.
  - Interrogating-CSCF (I-CSCF): This is the entry point for all SIP messages.
  - Serving-CSCF (S-CSCF): This SIP server manages all the session control functions in IMS.

- **Media Resource Function (MRF):** MRF is the IMS component which handles all the media related functions like media processing, playing of tones and announcements, etc. MRF is divided into two main parts:
  - Media Resource Function Controller (MRFC): MRFC is the signaling part of MRF and it controls the MRFP. It controls the media stream resources in MRFP as required by the AS. As shown in Figure 1 it uses SIP messages to interpret the media service request from AS and controls MRFP to perform the service using H.248/Megaco messages. IMS specifications gives MRFC the task of generating Charging Data Records (CDR) for billing purposes.
  - Media Resource Function Processor (MRFP): MRFP is the part in the media plane which performs the actual mixing and processing of media streams. MRFP is able to carry out services like playing announcements,

converting text-to-speech, mixing audio for conference calls, voice processing, etc [6]. MRFP performs various tasks like controlling bearer at Mb reference point (Figure 1), providing resources to be controlled by MRFC, mixing media streams, handling media stream processing, etc.

References [9] and [10] give a detailed sequence diagram for an IMS conference call explaining various functions of MRFC and MRFP respectively.

- **Media Gateway Control Function (MGCF):** MGCF is the component which integrates IMS with PSTNs and traditional telephony services. It communicates with CSCF (using SIP) and controls the IMS Media Gateway (MGW) as shown in Figure 2. MGCF does the mapping from non-SIP packetized voice to SIP UA and vice versa.

- **Media Gateway (MGW):** MGW is controlled by MGCF by H.248/Megaco interface. MGW does the conversion between SS7/TDM calls from legacy networks and SIP/RTP traffic in IMS. MGW supports media conversions (transcoding) and functionalities like echo cancellation etc. MGW bridges the gap between TDM and VoIP domains.

2.2 Announcement Service in IMS

Announcements are static or real-time media files delivered/played to the user. Playing announcements is one of the basic network media services in IMS. This section describes the protocols and the messages involved in achieving this service in an IMS environment. Playing announcement involves using server protocols such as SIP, often with markup languages like VoiceXML (VXML) [11] and Media Server Control Markup Language (MSCML).

Consider a scenario of playing announcement after a SIP call has been setup. The exchange of messages between AS and MRF is depicted in Figure 3. To start with, AS sends an INVITE SIP method to MRF requesting the announcement service. The MRF then negotiates Session Description Protocol (SDP) with AS and responds with 200 OK. After MRF receives ACK message from AS, it goes ahead to play the announcement to the requested device. Finally, after playing the announcement MRF issues BYE command to AS and AS acknowledges MRF with 200 OK. The detailed description of offer/answer model of SDP can be found in [13].

MRF gets the announcement service request from AS in the form of SIP URI. The URI defines various parameters like format, location of the announcement media file, etc [12]. The user part of the request URI describes if the request is for the announcement service. This is indicated by ”annc” in the URI. Also, for an announcement service URI, ”play” is a mandatory parameter which specifies the resource to be played. If the request URI has some parameters unsupported by MRF, MRF responds with an appropriate response error code. Examples for announcement request URIs are shown below [1]:

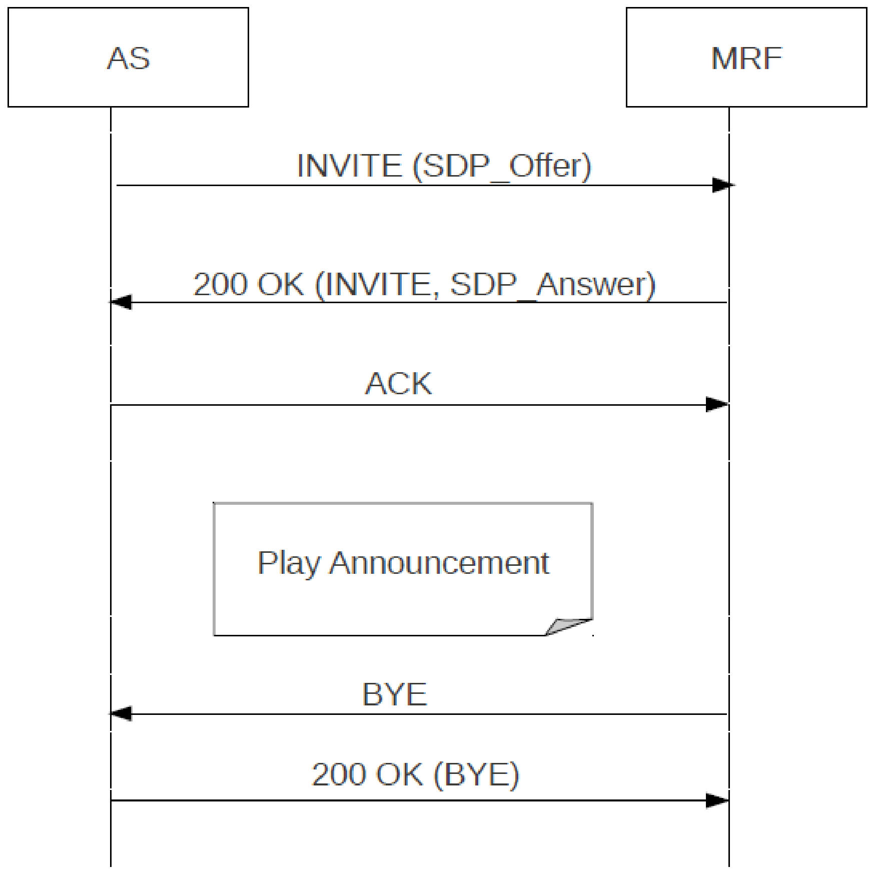

Figure 3: A scenario of invoking announcement service.

sip:annc@ms2.example.net; play=`http://audio.example.net/allcircuitsbusy.g711`

sip:annc@ms2.example.net; play=`file://fileserver.example.net//geminii/yourHoroscope.wav`

A generic scenario in announcement service involves having AS and S-CSCF (serving SIP proxy) in between caller and the MRF (media server). The caller may also invoke the announcement service directly without involving AS [1]. Figure 4 shows the protocol diagram showing different messages exchanged between the caller, serving SIP proxy and media server.

Figure 4 is similar to Figure 3 but shows the proxy between the caller and the media server. The caller first invokes INVITE method to the serving SIP proxy. The SIP proxy determines which audio prompt to play to the caller by SDP negotiation. SIP proxy responds back to caller with 100 TRYING. Proxy also issues INVITE method to the media server with details of the announcement media file in the URI. The media server responds to the INVITE method with 200 OK to proxy. SIP proxy then issues 200 OK method back to the caller. The caller issues an ACK method as acknowledgement which is relayed to media server by SIP proxy, thus establishing

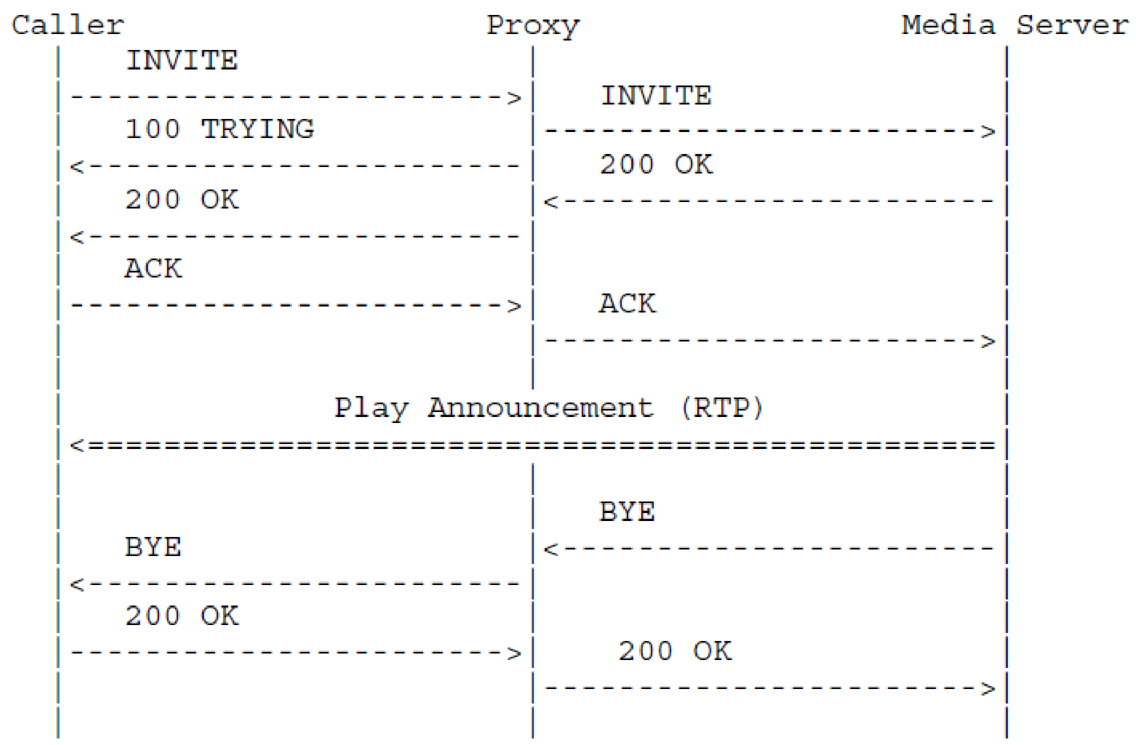

Figure 4: Announcement service protocol diagram (with proxy). [1]

SIP call. Now the media server plays announcement to the caller. On completion of playing the media announcement, the media server issues a BYE method to the proxy which is relayed to the caller. On receiving BYE, the caller responds with 200 OK to the proxy and the proxy forwards the same to the media server thus completing the announcement operation.

2.3 MRF Architecture

In practice, MRFP solution in IMS can be provided as a standalone node or as an integral part of Media Gateway (MGW) node [14]. This section describes MRF (media server) component as part of the IMS solution.

2.3.1 MRFP Components

Figure 5 shows the high-level architecture of the MRFP solution. The key functions of MRFP is explained in Section 2.1.2. The MRFP is designed as an application on top of a transport and connectivity platform [15] as shown in Figure 5. The MRFP node contains components to handle various resources like DSPs, IP ports, etc. The uppermost layer is the MRFP Application which has the fundamental control logic of MRFP. MRFC communicates with MRFP Application via Mp (H.248) interface. MRFP also has components to control the resource handling within MRFP along with O&M. However, the scope of this thesis is limited to only specific MRFP components mainly MRFP Application and DSP devices.

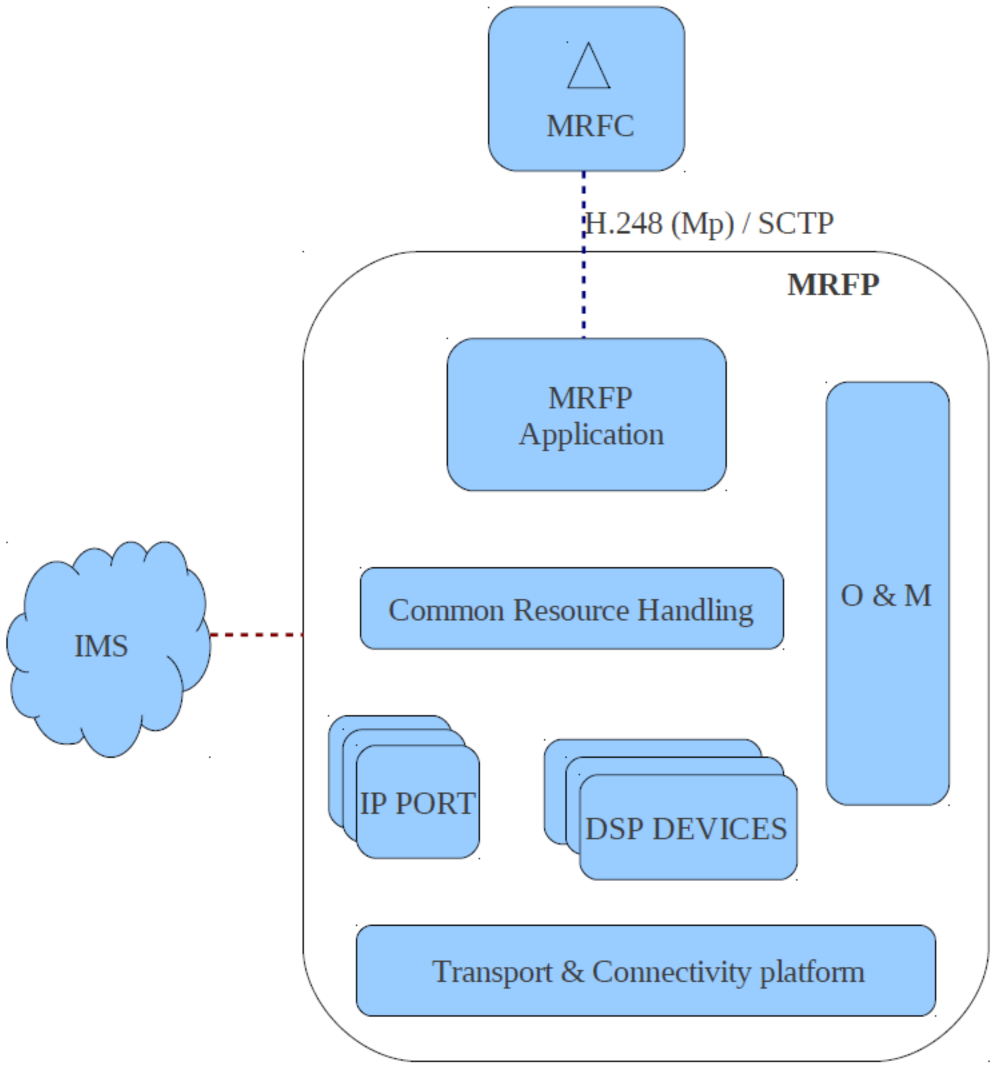

Figure 5: MRFP Components in a High-Level

2.3.2 MRFP Architecture for Playing Announcements

The announcement operation is carried out as explained in Section 2.2. The announcement media files (audio/video) are stored in the MRFP node. MRFC sends the URI to MRFP Application via H.248 interface. This URI contains the details about the location of media announcement file in the MRFP.

Figure 6 shows the different MRFP components involved when a media announcement is played. MRFP Application receives the URI from MRFC and forwards the request to Streamer component. The Streamer is associated with the

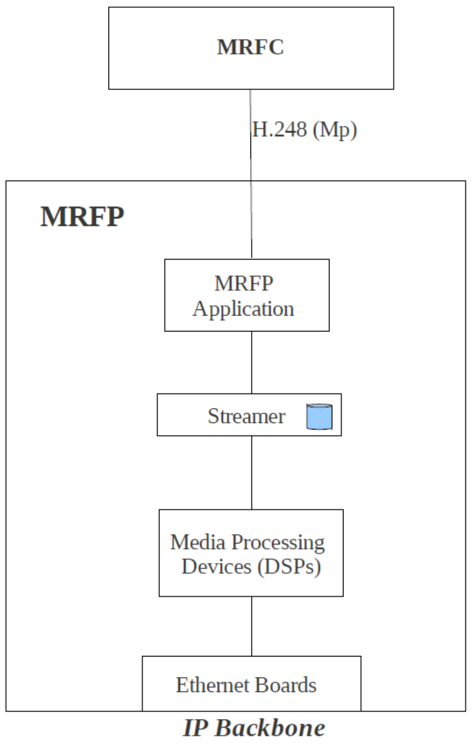

Figure 6: MRFP Architecture for Playing Announcements

MRFP memory to store the media files. The Streamer picks out the requested announcement file and sends the file to the Media Processing Device(s) to perform media processing. The processed announcement media file is then played to an IP address/port on the Ethernet board. The Ethernet boards lie in the edge of MRFP and connects MRFP to the IP backbone.

One of the key challenges this thesis discusses is the lack of memory space for the announcement files in the MRFP. All the announcement files are stored in the MRFP memory which is very limited in size. If the memory is full, a new announcement file can be added only by deleting existing file(s) in the memory. With the wide range of announcement media available, the MRFP architecture is studied to support more feasible file storage capability. Another architectural optimization possibility is regarding reservation of additional resources for playing announcements. The MRFP Application reserves new DSP device(s) to handle the announcement service. Hence there are additional devices reserved each time an announcement service is requested, which may be avoided. This thesis also discusses the MRFP solution to avoid extra reservation of these device(s).

2.4 Summary

Chapter 2 gives an understanding to the announcement service operation in IMS. To start with, this chapter introduced us to the functionalities of various network components in IMS. Also, this chapter explained the steps involved to handle the announcement service in IMS (Section 2.2). This chapter specifically looks into the MRFP architecture to understand the architectural flow to achieve the announcement service and points out the possible areas of improvements to it. The key thing to take away from this chapter is the MRFP software architecture shown in Figure 6 which can be compared against the MRFP software architecture in the external announcement approach described in Chapter 3.

3 External Announcement Approach

For the reasons described in Section 2.3.2, there is a need to find an alternative solution for handling announcement services in IMS. The MRFP architecture described in Section 2.3.2 stores all the media files internally in the MRFP. In the external announcement approach, the media files are stored externally in a media server and are accessed by MRFP using different protocols. This chapter describes in detail the proposed architecture for external announcement approach followed by selected different protocols that may be chosen to achieve this proposition.

3.1 External Announcement Architecture for Playing Announcements

Figure 7 is a modified version of Figure 6 which shows the proposed approach for playing announcements externally. As shown in Figure 7, the media server located external to the MRFP stores all the audio/video announcements of different formats and sizes. This server is accessed by MRFP software components via an interface. The interface may use various protocols agreed between MRFP components and server. For instance, if the server is an RTSP server implementation, RTSP is used as the signaling protocol to communicate between MRFP components and the server.

The MRFP application gets the request URI from MRFC with the details about announcement file location and file format. This request is forwarded to the Streamer component which can have a built-in media player for streaming. The MRFP Application/Streamer then communicates with the external media server to get the announcement file using a suitable protocol as interface. The announcements can be played in two different ways depending on the type of server implementation used:

- Streaming: If the server is a streaming server, the announcement video/audio file is delivered by streaming it to the required destination IP address on the Ethernet board of MRFP. RTSP server is an example for this way of playing announcements. Essentially MRFP Application component establishes connection with the RTSP server using RTSP signaling. The server then streams the audio/video announcement requested to the given destination IP address on the Ethernet board using Real-time Transport Protocol (RTP). This method is explained in detail in Section 3.2

- Playing locally: If the interface between the MRFP and the server is a networking protocol supporting client-server architecture, the requested announcement audio/video file is first downloaded from server to the MRFP. The downloaded file is stored locally in the MRFP memory. The file is then played locally within the MRFP to the destination IP address using RTP. For instance, a HTTP based web server can be used as an external server to store all the announcement files. In this case, the Streamer requested file is first downloaded (HTTP

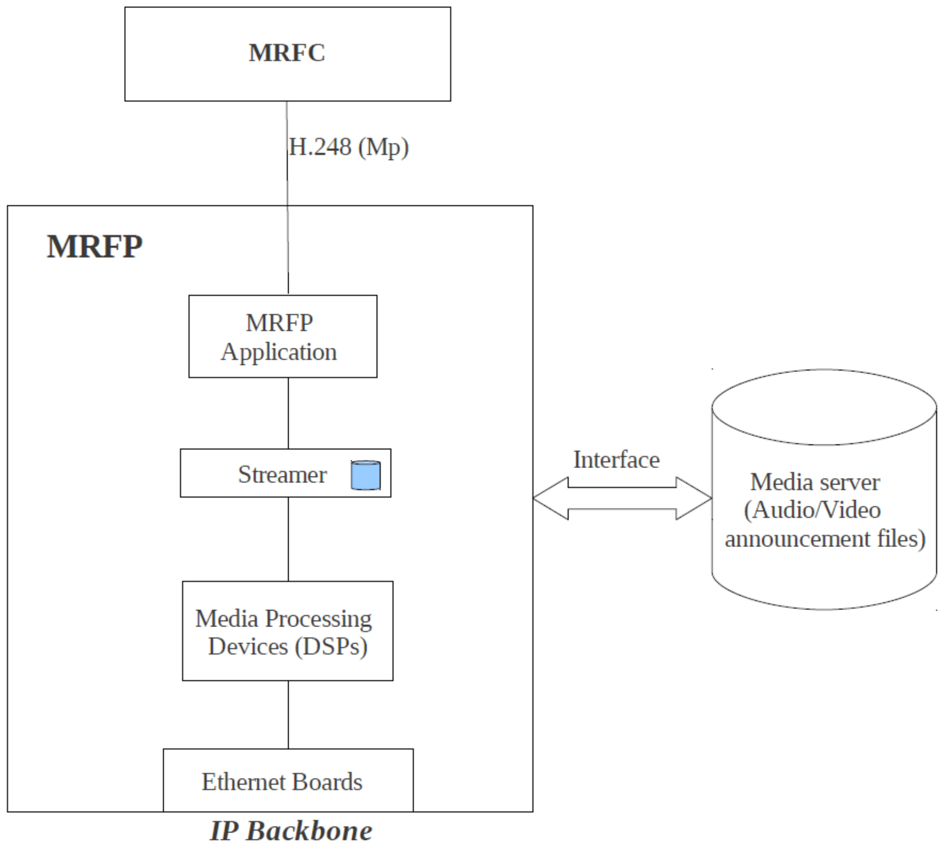

Figure 7: Proposed external announcement approach architecture

GET) from the server using HTTP which is later played to the destination IP address using RTP. This method is explained in detail in Section 3.3.

3.2 External Announcement Approach Using RTSP

This section gives a detailed description on how announcements are played using RTSP as the interface between MRFP and the external server. In this section, the working of RTSP is delineated followed by it's usage to achieve external announcements.

3.2.1 RTSP Overview

RTSP is a widely used protocol to control the media sessions streaming between the media client and the media server. RTSP is an application-level protocol which is used to control delivery of multimedia streams with real-time constraints through the network [16]. RTSP enables controlled on-demand delivery of real-time audio and video. RTSP does not itself deliver streaming data; instead it controls choosing the delivery channels such as UDP, multicast UDP and TCP. RTSP operation does not depend on the underlying protocol used for data transmission. Typically RTP is used as the protocol to deliver streaming data. Figure 8 shows a simplified block diagram with RTSP/RTP between a media client and a media server.

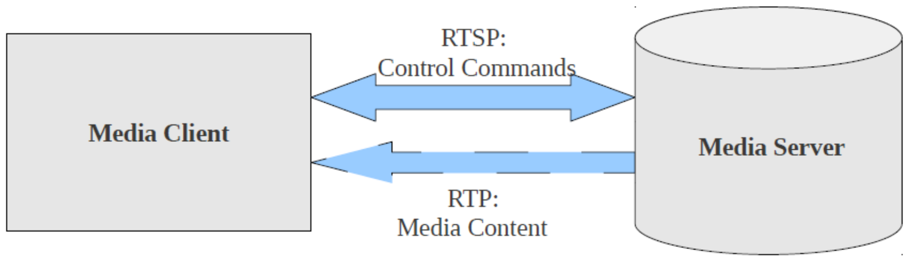

Figure 8: RTSP mechanism

RTSP is a text-based state-oriented protocol, i.e., a RTSP server maintains states of an RTSP session labeled by a session identifier. RTSP is also transport-independent and hence an RTSP session may be established using unreliable or reliable transport protocols. RTSP session involves exchanging RTSP messages between the client and the server over a negotiated transport protocol and a port number. RTSP uses 554 as the default transport layer port number [17]. Some of the properties of RTSP are listed below [16]:

- RTSP is a bi-directional protocol. i.e., both client and server can send requests/responses to each other.
- RTSP is an extendable protocol. i.e., it is easy to add new methods/parameters to RTSP.
- The content presented to the client from the server can be a combination of multiple media streams (for example, audio and video). Each of these media streams can be fetched from different media servers and synchronized. Hence, RTSP supports multi-server capability.
- RTSP is HTTP-friendly. Though HTTP is a stateless protocol, it's existing framework is reused for RTSP.
- RTSP client and server negotiate an appropriate connection-less or connection-oriented transport protocol before any continuous media stream is processed.
- RTSP re-uses HTTP security mechanisms such as HTTP authentication etc. RTSP may also use transport-layer and network-layer security technologies like Transport Layer Security (TLS) [19].
- RTSP provides an effective server control mechanism. If a RTSP client starts a media stream, it must also be capable of stopping the stream. On the other hand, if a server starts a media stream, it must not prevent the client from stopping the stream.

The media content which is streamed by the media server towards the client can be in different forms:

- *Pre-stored media:* These are readily recorded media contents stored in the media server. When client requests the server, media will be delivered as a stream.
- *Live broadcast media:* These are the media contents directly streamed to the user as it is produced. However, there might be delay involved due to editing or buffering of the data.
- *Live interactive media:* In this case, there may be several users interacting with each other and producing their own content. Video conferencing is an example for such media content where streaming is not controlled by one broadcaster.

3.2.2 RTSP Operation

A media stream is identified by an RTSP URL which also indicates the location of the continuous file in the media server. RTSP can operate in either unicast or multicast mode. In unicast mode, the media server streams data to the client which sent the RTSP request. Also, the server uses the port number chosen by the client to transfer data. In multicast mode, the media server streams the content to different RTSP clients the specified by multicast address and port. The scope of this thesis is limited to RTSP operation in unicast mode.

As explained in Section 3.2.1 the RTSP signaling is independent of the actual data transmission from the media server. Therefore RTSP session states are maintained to associate RTSP requests with a media stream. RTSP signaling is established by the exchange of RTSP control messages between client and server. An RTSP message mainly comprises of different RTSP methods.

Following are the basic RTSP methods used in RTSP messages:

- **OPTIONS:** The client issues OPTIONS command to the server stating the protocol version it is using. The server responds with all the available methods that it accepts.
- **DESCRIBE:** The client issues the DESCRIBE method to fetch the low level description of the media object represented by the URL. The server response contains a full media description of the media object in Session Description Protocol (SDP) format [20]. The DESCRIBE method performs the media initialization step in RTSP signaling.
- **SETUP:** The client informs the server about the transport mechanisms it wants to use by issuing the SETUP method. The response to this method contains the transport parameters selected by the server. For example, the client may inform the server that it wants to use RTP over UDP on specific

Table 1: Overview of RTSP methods

Method	**Direction**	**Requirement**
OPTIONS	C –>S	Mandatory
DESCRIBE	C –>S	Recommended
SETUP	C –>S	Mandatory
PLAY	C –>S	Mandatory
PAUSE	C –>S	Recommended
TEARDOWN	C –>S	Mandatory

port numbers for streaming the media object. If the server agrees to client's transport selection, it acknowledges with the confirmation and details about the selected transport mechanism.

- **PLAY:** The PLAY command starts the data transmission from the server to the client over the transport mechanism negotiated in the SETUP phase. The PLAY request contains various parameters like the RTSP session identifier, the URL of the media(s) and an optional range. The URL may represent one or multiple media to be streamed. The range parameter specifies the starting position of the media to be streamed. If no range is specified, the media is streamed from the beginning.
- **PAUSE:** The PAUSE request temporarily halts the media stream. This request may optionally contain range parameter specifying the time position at which the stream should be paused. If no range is specified, the media stream is interrupted immediately on receiving the PAUSE request.
- **TEARDOWN:** The TEARDOWN request terminates the RTSP session and frees all the resources associated with the media stream. The client issues the TEARDOWN command specifying the URL representing the media stream(s) and the RTSP session identifier. The server then acknowledges this TEARDOWN request and stops the media stream(s).

Table 1 gives a summary of the above mentioned commands. It shows if the commands are sent from client (C) to server (S) or vice-versa. It also shows the requirement of the commands in an RTSP server implementation [16].

Figure 9 shows an example of the message exchanges involved between the client and the server during an RTSP session [21] [22]. For simplicity, this section deals with only primary RTSP commands. An elaborate description of RTSP involves various other commands like ANNOUNCE, GET_PARAMETER, SET_PARAMETER, REDIRECT, RECORD, etc which are out of scope of this text.

3.2.3 Dataflow Analysis for External Announcements Using RTSP

Section 3.1 describes the proposed approach to handle announcements using an external media server. This section gives a detailed description of RTSP as a pos-

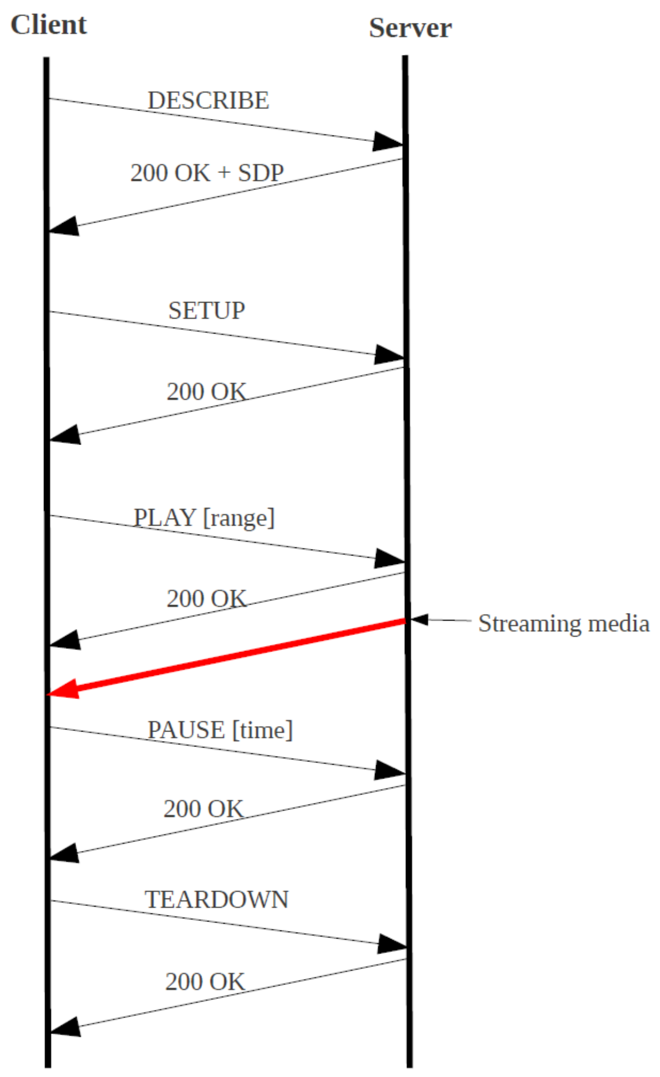

Figure 9: An RTSP flow example. ([23], modified)

sible interface between MRFP and the external media server. Figure 10 shows the dataflow between different components to achieve external media announcements.

As shown in Figure 10, MRFC receives the announcement request from the UE after SIP negotiations between different IMS components [24]. The MRFC forwards the received announcement URI to the MRFP via the H.248 interface. MRFP Application thus gets the details about the media file format and location of the file in the external media server. MRFP Application has the control logic of a RTSP client which uses the URL of the announcement file to establish RTSP session with the external RTSP server. Section 3.2.2 briefly explains the main steps involved to establish an RTSP session. MRFP Application negotiates RTP/UDP as transport mechanism used by the server to stream the announcement. MRFP Application also sends the details about the destination IP/port address of the Ethernet board to which the server should stream the announcement media. When the MRFP

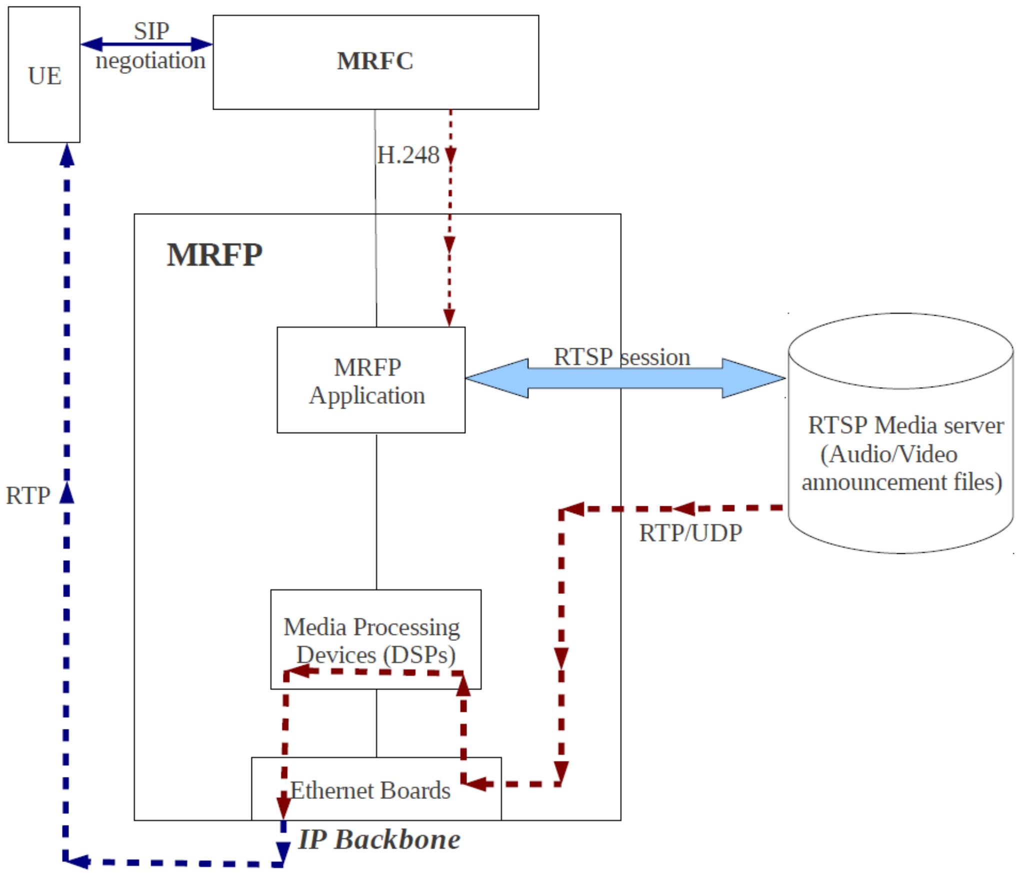

Figure 10: External announcements dataflow using RTSP

Application issues PLAY command to RTSP server, the server starts to stream the announcement media over RTP/UDP transport to the given destination IP/port address.

The RTP/UDP media played to the Ethernet board may need media-transcoding to match the format supported by the UE. Therefore the streamed media is again transcoded according to requirement by the Media Processing Devices (MPDs) as shown in Figure 10. In short, transcoding is a process of converting media file from one format to the other. The transcoded media output from the MPDs is streamed out of the MRFP to the announcement requesting UE via RTP. The external announcement is hence achieved using RTSP as an interface between MRFP and the external media server. More details on the RTSP client and RTSP server configurations/implementation as a prototype is described in Section 5.1.

On comparing the MRFP architecture in Figure 7 with that in Figure 10, it can be noticed that in the latter the Streamer software component is non-existent. This is because the Streamer component can have an in-built media player which is used to play the media files locally. But when RTSP is used, the media is directly streamed via RTP/UDP from media server to the Ethernet board. Additionally, the MRFP memory may not be used to store the announcement files as they are directly streamed to the destination instead of being downloaded. Hence Streamer is not a required component in MRFP node in RTSP based approach.

Therefore, RTSP based approach solves a critical challenge of storing the media announcement files internally in MRFP by storing them in an external RTSP server. Also, since media is directly streamed to Ethernet boards, this approach avoids reserving additional DSPs for playing announcements and gets rid of Streamer component in MRFP. Hence, RTSP based approach addresses both the challenges mentioned in the problem statement of this thesis work in Section 1.1. Section 6.3 gives an analysis of the results collected by configuring/implementing a prototype of this approach.

3.3 External Announcement Approach Using HTTP

The goal of this section is to discuss a possible approach with Hypertext Transfer Protocol (HTTP) as an interface between MRFP and the external media server. This section describes the basic features of HTTP followed the by description on how HTTP is incorporated as an interface to achieve external media announcements.

3.3.1 HTTP Overview

HTTP is the widely used protocol in the Internet today. HTTP is a text-based application-level networking protocol for distributed, interactive multimedia information systems [25] [27]. HTTP is a "request-response" protocol and it is based on client-server architecture. It essentially involves a client which sends requests and a web server which responds to these requests. The client and the server follow the HTTP message structures for request and response respectively. HTTP identifies and locates the resources in the server using URIs. The URIs are included as part of the HTTP requests sent from clients to the servers.

Unlike RTSP, HTTP is a stateless protocol. Each transaction between the client and the server is independent of the previous one. During an HTTP session, the client first establishes a TCP connection to a specific port on the server (typically port 80 [17]). The server then waits on that port for client's request messages. When the server gets the client's message request, it sends back a response to it and closes the TCP connection. Both HTTP request and response follow a similar message structure. A HTTP generic message structure is shown below:

```
<initial-line>
<message-headers>
<empty-line>
[<message-body>]
[<message-trailers>]
```

Listing 1: HTTP message structure

Table 2: HTTP response categories

Status codes	**Response category**
1xx	Informational message only
2xx	Success of some sort
3xx	Client redirection to another URL
4xx	An error at the client end
5xx	An error at the Server end

In the message structure shown in Listing 1, `<message-body>` and `<message-trailers>` are optional parameters. The `<initial-line>` conveys the purpose of the message. `<initial-line>` is different for request and response messages. In a HTTP request message, `<initial-line>` has three parts: the method name, the URI specifying the location of the resource and the version of HTTP being used. An example for an HTTP request `<initial-line>` is shown below:

```
GET /path/to/file/mp3_test.mp3 HTTP/1.1
```

In the above example, `GET` is the method name followed by URI specifying path to the resource `mp3_test.mp3`. The HTTP version used for the transaction is `HTTP/1.1`. Figure 11 shows the interaction between a HTTP client and an HTTP server during `GET` method.

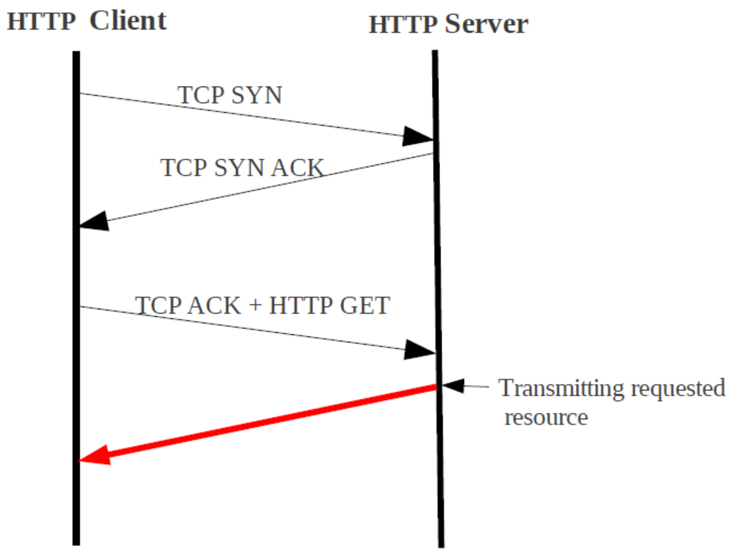

Figure 11: An HTTP flow example. ([18], modified)

In a HTTP response message, `<initial-line>` has three parts as well: the HTTP version, the response status code and the textual phrase describing the status code. An example for a response message `<initial-line>` is shown below in which `HTTP/1.1` is the HTTP version, `200` is the status code and `OK` is the textual description of the response code indicating that the transaction was successful.

```
HTTP/1.1 200 OK
```

The HTTP status code is a three digit number and the first digit represents the category of the server response. Table 2 outlines different response categories based on status codes.

In the HTTP message structure shown in Listing 1, `<message-headers>` are the name-value pairs which provide data about HTTP request or response. In a HTTP response message, `<message-headers>` provides information about the object sent in `<message-body>`. Both `<message-headers>` and `<initial-line>` parameters must end with Carriage Return (`<CR>`) followed by Line Feed (`<LF>`). An example for a HTTP request message header is shown below. It shows a name-value pair with the name `User-Agent` whose value indicates the program and system information making the HTTP request.

```
User-Agent:  Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.18)
      Gecko/20110628 Ubuntu/10.04 (lucid) Firefox/3.6.18\r\n
```

HTTP uses various methods in `<initial-line>` to indicate the action to be performed on the resource identified by the URI. Some of the key methods are described below [27]:

- **GET:** The HTTP `GET` method retrieves the resource indicated by the request URI. There are different variants of HTTP `iGET` like `"partial GET"` and `"conditional GET"`. In `"partial GET"`, only a part of the resource is retrieved as defined by the range header field in HTTP request message. In `"conditional GET"`, the resource is retrieved based on some conditions like `If-Modified-Since`, `If-Unmodified-Since`, etc. defined in the header field of HTTP request message. A sample `GET` request is shown in Listing 2

```
GET /example.html HTTP/1.1
Host: 192.168.1.228
```

Listing 2: HTTP GET method

- **HEAD:** This method is very similar to `GET` method except that the server response to `HEAD` request contains no message-body. `HEAD` method retrieves only the meta-information in the HTTP response headers.

- **POST:** The HTTP `POST` method is used for requesting the server to accept the data included in the `<message-body>` of the request. The server allocates a URI for this data and returns the URI back to the client. The `POST` method is most commonly used to submit HTML form data. Listing 3 shows an example of HTTP `POST` method.

```
POST /example.html HTTP/1.1
Host: 192.168.1.228
Connection: Keep-alive
Content-length: 311
Content-Type: text/html

data=<data_sent_to_server>
```

Listing 3: HTTP POST method

- **OPTIONS:** The `OPTIONS` method is used to request the server for the HTTP methods it supports for the specified URI. This is similar to the `OPTIONS` method in RTSP described in Section 3.2.2
- **DELETE:** The `DELETE` method requests the server to delete the resource specified by the request URI.

HTTP includes various other request methods like `PUT`, `TRACE`, `CONNECT`, `PATCH`, etc [26] which are out of scope of this text.

3.3.2 Dataflow Analysis for External Announcements Using HTTP

The aim of this section is to study HTTP as a possible interface between MRFP and the external media server as described in Section 3.1. Figure 12 shows the dataflow between various components to achieve media announcements using a external HTTP server.

As shown in Figure 12, UE initiates the announcement request to IMS. This request reaches MRFC component after several SIP negotiations between various IMS components [24]. MRFC thus receives the details about the media announcement requested by the user. The MRFC then sends the announcement request to MRFP via the H.248/Megaco interface. The MRFP Application gets the announcement URL and forwards the same to the Streamer component. The Streamer component has the control logic of a HTTP client. The HTTP client sends a HTTP `GET` request to the external HTTP media server using the announcement URL. The server listens to client's request on port number 80. The server responds to the HTTP `GET` by transmitting the media file indicated by the URL. The HTTP `GET` operation is explained in more detail in Section 3.3.1. The `GET` request and response for a µ-law encoded `.wav` audio file compressed at a bitrate of 88.2 kbps is shown below:

HTTP GET Request

```
GET http://192.168.1.228/wav_88_2kbps_ulaw.wav HTTP/1.1\r\n
Host: 192.168.1.228\r\n
Accept-Encoding: identity\r\n
\r\n
```

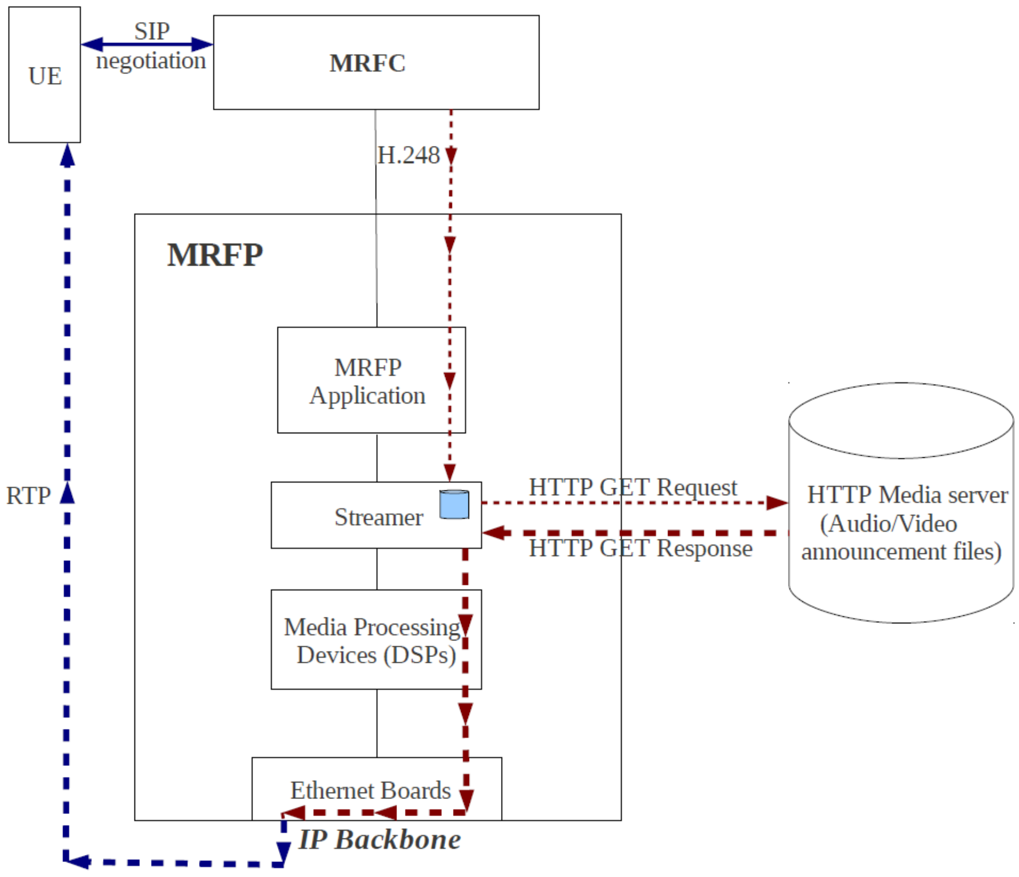

Figure 12: External announcements dataflow using HTTP

HTTP GET Response

```
HTTP/1.1 200 OK\r\n
Date: Tue, 08 Nov 2011 08:48:31 GMT\r\n
Server: Apache/2.0.64 (Unix)\r\n
Last-Modified: Tue, 25 Oct 2011 12:13:03 GMT\r\n
ETag: "c80274-25306-760e51c0"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 152326\r\n
Content-Type: audio/x-wav\r\n
\r\n
```

As shown above, the Streamer component retrieves the `wav_88_2kbps_ulaw.wav` file from the media server. The retrieved file is stored locally in the MRFP memory as shown in Figure 12. The Streamer is responsible to play the media file in the database to the destination IP/port address on the Ethernet board as indicated by the announcement request. The Streamer component has a in-built media player which plays the media locally using RTP. The media played by the media player is transcoded by the MPDs according the user's media format requirements. The transcoded media is then played on the destination IP/port which is streamed out of MRFP to the UE via RTP. The external media announcement is thus achieved by using HTTP as an interface between MRFP and external media server. Section 5.1 gives more details on the implementation/configuration of the client and server used in the prototype. A detailed analysis of the measurements taken by playing different media announcements can be found in Section 6.

3.3.3 Caching Mechanism in MRFP

The MRFP architecture shown in Figure 12 slightly differs from the MRFP architecture for RTSP approach shown in Figure 10. The Streamer is a key component in HTTP based approach. Also, the announcements files played are the ones which are stored in the existing MRFP memory. Since the existing MRFP memory is very limited, it is critical to use this memory as cache to store the announcement files based on appropriate caching algorithm. The caching algorithm takes into account various factors like the announcement file size, access frequency, etc. When an announcement is requested, if the announcement file already exists in cache, the Streamer does not send the HTTP `GET` request to the external server to download the file. Instead, it directly plays the already existing file to the destination IP/port. Therefore, HTTP based approach has an advantage that the MRFP does not contact the external server unless the requested announcement file is not found. This is unlike RTSP based approach where every announcement request involves RTP streaming from the external server to the destination IP/port as described in Section 3.2.3 [28]. Section 4.3 makes a detailed study on different caching algorithms which can be used in MRFP.

3.4 External Announcement Approach Using FTP

The third possible approach is to use File Transfer Protocol (FTP) as an interface between MRFP and the external media server. This section gives an overview on FTP, a brief description of key FTP commands used and explains how FTP is incorporated as an interface to study external announcements.

3.4.1 FTP Overview

FTP is a networking protocol which follows the client-server model to transfer files from one host to the other. FTP uses TCP as the transfer protocol for an efficient, secure and reliable data transfer [29] [30]. FTP uses two TCP connections to perform the operation:

- A control connection on which the client sends FTP request messages and the server replies with FTP response messages.
- A data connection to perform actual data (file) transfer.

The data connection may be intermittent but the control connection persists for the entire FTP session. Typically, FTP control connection uses TCP port number 21. The data connection is established and handled by the Data Transfer Process (DTP) in the client and the server. The data connection may be operated in two modes:

- Active mode: In active mode, the server-DTP initiates the data connection. On the control channel, the client would send the server the IP/port address which it can use to establish the data connection. The server-DTP uses this address to establish the data connection from it's own port number 20. The client ports used in data connections are typically ephemeral ports (port numbers greater than 1023). The steps involved to establish a active data connection is shown in Figure 13. In Figure 13, the `PORT` command is used to specify the client's ephemeral port number to which the server should connect.

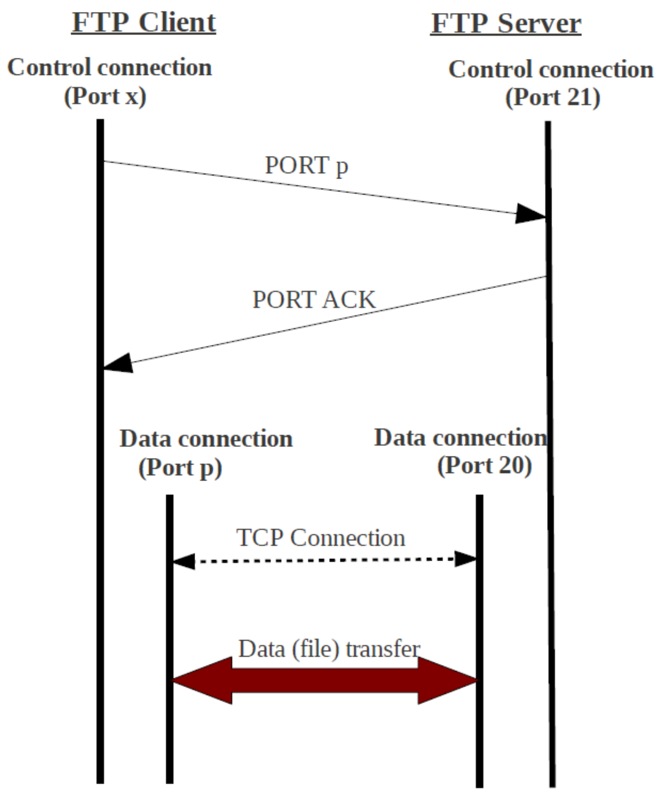

Figure 13: FTP Active data connection. ([32], modified)

- Passive mode: In passive mode, the client-DTP initiates the data connection. On the control channel, the server sends it's IP address and ephemeral port number for data connection. The client-DTP then initiates the data connection from it's own ephemeral port number to the IP/port address sent from the server. An example of a passive data connection is shown in Figure 14. In Figure 14, the `PASV` command is used to indicate the server to use passive mode. The server replies to PASV command with it's ephemeral port number which the client-DTP can use to initiate the data connection [31].

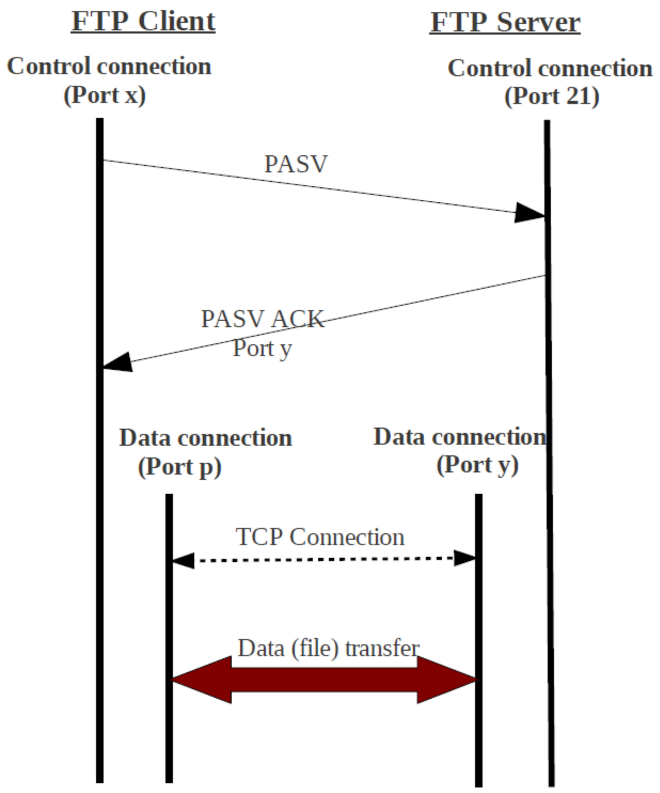

Figure 14: FTP Passive data connection. ([32], modified)

FTP provides a basic authentication mechanism by using the login scheme. When a client wants to access the FTP server, it sends it's `username` and `password` using FTP commands. The combination of `username/password` is verified against server's database to verify if the client has access to the server. There are many security extensions for FTP to provide better authentication, authorization and encryption [33]. On the other hand FTP can be used without any authentication at all by using a general purpose account called "anonymous FTP" [29]. Anonymous FTP is used when the server stores files which can be accessed publicly. Such servers create a special account called `anonymous` with limited access rights. The only operations that can be performed by `anonymous` account is to login, list and retrieve the contents stored in the server. The `anonymous` users are only allowed to retrieve files but they cannot transfer the files to the server.

3.4.2 FTP Requests and Responses

The FTP client can send various commands to the FTP server over the control channel. The server interprets the command, takes appropriate action and responds back with the status code and any other appropriate information. FTP supports various commands performing wide range of operations like:

- **Access control commands:** These commands use the user login and authentication information to handle the access control of resources. `USER`, `PASS`, `CWD`, etc. are some of the examples in this category.
- **Transfer parameter commands:** These commands specify how a data transfer should take place. The transfer parameters have default values and these commands are used only if they need to be changed. Typical examples for commands in this category are `TYPE`, `PASV`, etc.
- **FTP service commands:** These set of commands are used to perform the actual file operations. These commands typically take file pathname as the argument. `RETR`, `DELE`, `STOR`, etc. are commonly used FTP service commands.

Table 3 gives a brief description about the commonly used FTP commands.

Table 3: FTP commands

Command	**Description**
USER	Character string to authenticate the username
PASS	Character string specifying user's authentication password
CWD	Change Working Directory to the specified path
CDUP	Change to parent directory
QUIT	Terminates FTP session
PORT	Allows the client to specify the port number which the server should use for data connection (Figure 13)
PASV	Enables the passive mode FTP data connection (Figure 14)
TYPE	Specifies the format in which data is transferred (binary/ascii)
RETR	Retrieve the specified file from the server
STOR	Store the sent file in the server
REST	Restart file transfer from the where it had stopped
DELE	Delete the specified file from the server
LIST	Lists all the files and directories in the current directory
HELP	Outputs all the commands that the server can understand

All FTP responses for the client's requests are sent over the control channel. The FTP responses are identified by a 3-digit reply code along with some text explaining the response. Each digit of the reply code has a special significance. For example, the first digit indicates whether the response is a success, a failure or incomplete. Table 4 outlines different response categories based on first digit of the reply code.

3.4.3 Dataflow Analysis for External Announcements Using FTP

This section aims to study FTP as a possible interface between MRFP and the external media server to achieve external announcements. The dataflow and the

Table 4: FTP response categories

Status codes	**Response category**
1xx	Positive Preliminary reply - The request has been initiated successfully
2xx	Positive Completion reply - The request has been executed successfully
3xx	Positive Intermediate reply - The request has been accepted but the server is waiting for further action from the user
4xx	Transient Negative Completion reply - The request was not accepted due to a transient error
5xx	Permanent Negative Completion reply - The request was not accepted due to a permanent error

MRFP components involved to achieve such announcement are shown in Figure 15. It can be noted that the architecture in Figure 15 follows the same flow as in the HTTP case (Figure 12) but with FTP used to fetch the announcement media file from the server.

As shown in Figure 15, the UE sends the media announcement request to the IMS which reaches the MRFC component. The MRFC component now has the details about the location and format of the media requested. MRFC forwards the request to the MRFP Application component in MRFP via H.248 interface. The MRFP Application then forwards the required media information to the Streamer component. The Streamer is the key component which stores the media announcements in the MRFP memory. The Streamer has a FTP client built into it which sends FTP requests to the external server to get the requested announcement file. The client uses the anonymous login to access the server. The client uses the passive mode data connection to transfer the announcement file. The client then sends the `RETR` request to fetch the file from the server and stores it in the MRFP memory. The media file being played is first transcoded by the MPDs to meet the media format requirements supported by the UE. Finally, the transcoded media is played to the destination IP/port on the Ethernet board using RTP as shown in the Figure 15.

The FTP based approach has the same drawbacks as HTTP. If the Streamer does not find the media file locally, it is first downloaded from the external server to the MRFP and it is then locally streamed to the known destination IP/port address. Hence, the existing limited memory in MRFP has to be used as a cache to store the announcement files. An advantage from FTP/HTTP based approaches is that the MRFP does not go to the external server at all if the requested announcement file exists locally. An appropriate caching algorithm should be used to make the best use of the limited MRFP memory. A detailed explanation on the possible caching algorithms that can be used is given in Section 4.3. Section 5.1 gives the implementation/configuration details about the clients and servers used to realize this approach. Also, Section 6 gives an analysis of the measurements taken in an

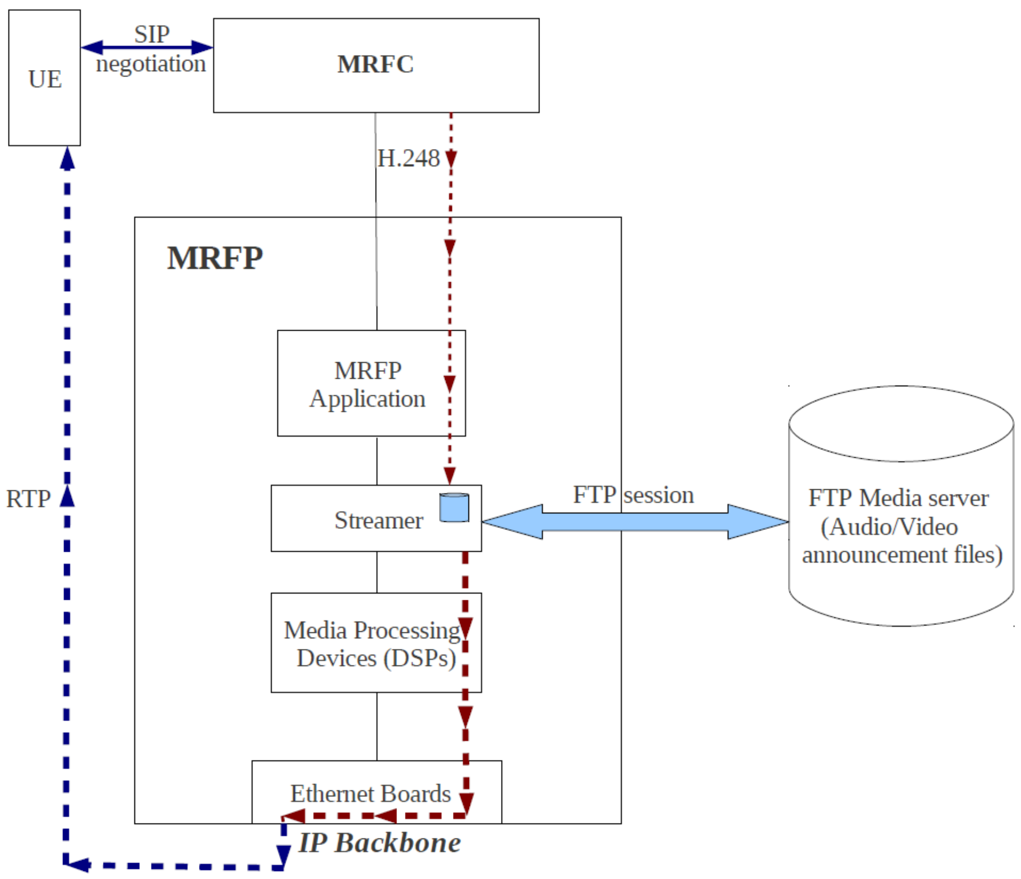

Figure 15: External announcements dataflow using FTP

emulated networking environment involving packet loss, latency, etc.

3.5 Summary

Chapter 3 gives a detailed description on three of the protocols that may be used as an interface between MRFP and the external media server namely RTSP, HTTP and FTP. The MRFP architecture to realize the external media announcement approach using RTSP, HTTP and FTP are shown in Figures 10, 12 and 15 respectively. Therefore this is a key chapter which gives an understanding that it is possible to study external announcements in different ways. Based on the study made in this chapter, a prototype is built which is described in Chapter 5.

4 Effective Utilization of Memory in MRFP

This chapter discusses an important challenge of how to utilize the limited memory in MRFP effectively. Section 3.3.3 briefly discussed about this by explaining the importance of incorporating cache memory mechanism in MRFP. In this section, the need for implementing such algorithms is explained in Section 4.1 followed by Section 4.2 which points out various parameters that needs to be considered in caching. Finally, Section 4.3 proposes the caching algorithms which can be best incorporated in the MRFP.

4.1 MRFP Memory as Cache

One of the prominent research questions in this thesis is to study the possible ways to effectively utilize the limited amount of memory available in MRFP. Using an external server to store the announcement media files does not solve the issue in all cases. When HTTP/FTP is used as an interface to download the announcement file, the downloaded file is first stored in the MRFP memory (by the Streamer component) and then it is played by the media player in the Streamer component to the known destination IP/port address. Therefore, even though there is an external memory capable of storing large number of announcement files, the MRFP dataflow makes it necessary to store the files internally before playing. This brings up the question of how to use this limited MRFP memory in an efficient manner which this section tries to answer. However, if RTSP is used as an interface, there is no such problem to store the announcements in MRFP memory as the announcements are directly streamed to the destination IP/port address. Hence, it is important to note that in this thesis study, the caching mechanism is required only when the interface between MRFP and the external media server is HTTP or FTP.

Figure 16 shows the scenario when the appropriate caching mechanism is required. Since the MRFP memory is used as a cache, if the requested announcement file is readily available in the MRFP, the Streamer component does not download the file from the external server. It directly plays the announcement without involving any HTTP/FTP signaling. There are various parameters that can be used to decide which files to store (or discard) in the cache. Section 4.2 explains such parameters.

4.2 Key Parameters in File Caching

The MRFP supports various audio and video announcement media formats. Also, MRFP handles files with various audio or video compression techniques and of different sizes. The caching issue tackled in this thesis can be directly related to the proxy caching techniques used for caching web objects [34]. The primary intention is to make sure that the announcement media requested by the UE is played at the destination IP/port address with the least latency. Additionally, MRFP caching also reduces the load on busy external servers.

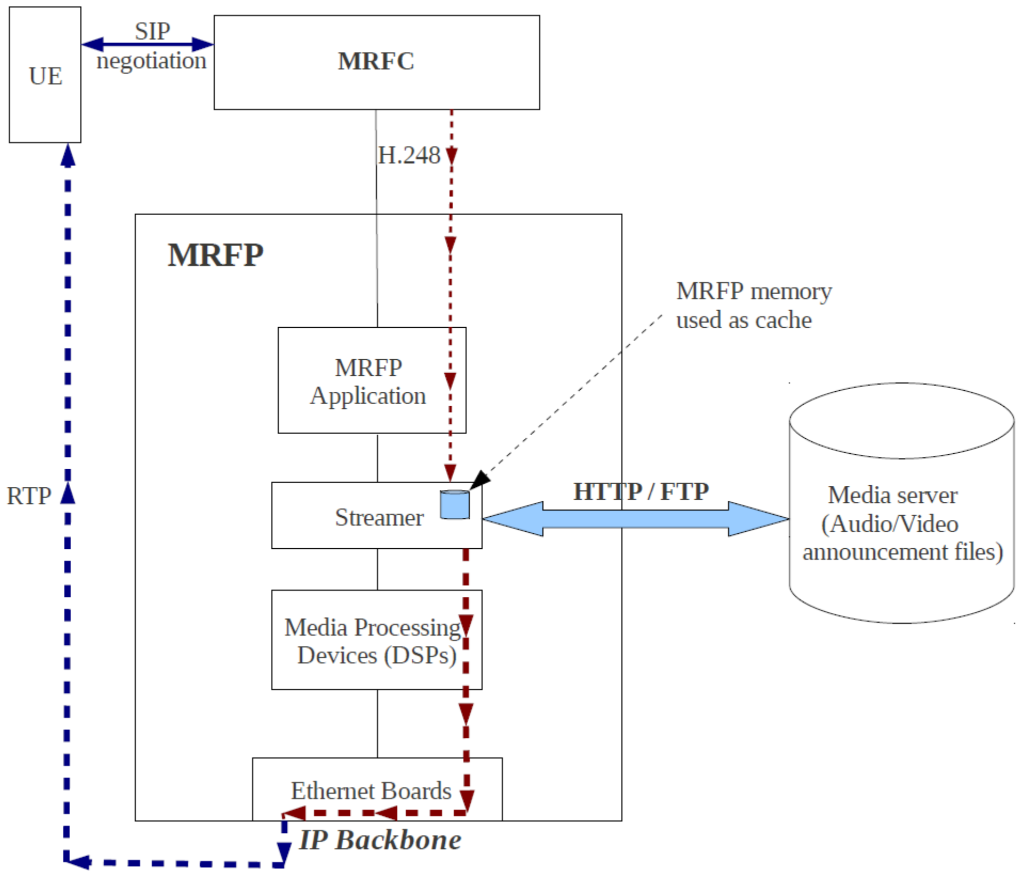

Figure 16: External announcement dataflow requiring caching mechanism

The MRFP caching algorithms are essentially replacement algorithms which are used to decide which file to replace from the cache when it is full. The caching algorithms can be implemented in a relatively transparent manner without any major configuration changes at the client or server end. The MRFP handles files only as a whole and hence fragmenting the file (partial caching [36]) is not supported. The various factors that determine the selection of a MRFP caching algorithm are explained below [35]:

- Announcement file size: The size of the announcement media files may vary from a few kilo bytes (audio) to several megabytes (video). Hence, the MRFP cache can store many small-sized or only a few large-sized announcement files. Depending on the available limited cache memory, the file size is a key parameter to be considered to decide if the file should be stored in the cache [38].

- Retrieval cost: When the requested announcement file is not found in the MRFP cache, there is a cost due to the connection time and the network latency involved to fetch the file from the server. This is the retrieval cost and it is an important parameter utilized to select the caching algorithm. For instance, if the retrieval cost for an announcement file is very high, it is most likely not removed from the cache [40].

- Bandwidth: Another performance parameter measured is the network bandwidth overhead while retrieving an announcement file from the server [41]. The network bandwidth overhead can be reduced if the selected caching mechanism minimizes the number of retrievals from the server. In other words, the network bandwidth is saved every time the requested announcement file is found in the MRFP cache.

- Popularity: This is a significant parameter which indicates the access frequency of the announcement files [39]. The access frequency of each announcement file is calculated over a pre-determined period of time. The access frequency is used to decide the announcement file that should be removed when the cache is full. Therefore, if an announcement has high popularity, it is likely that the caching algorithm would retain it in the cache. However, as Section 4.3 explains, there are factors other than the popularity which can contribute to make that decision. The implementation part of this thesis involves the calculation of the popularity of the announcement files. More on this is described in Section 5.3.

- Last requested time: Each announcement file is associated with the last requested time parameter. It indicates the time when an announcement file was last requested by the UE. Like popularity, the last requested time parameter can also be used to decide the file that should be retained when the cache is full.

Cache hit and cache miss

When the UE requests for an announcement, if the announcement file is already available in the MRFP cache, it is referred as a cache hit. The ratio of the number of cache hits to the total number of requests is defined as cache hit ratio. On the other hand, if the requested announcement file is not found in the cache, it is referred as a cache miss. The ratio of the number of cache misses to the total number of requests is defined as cache miss ratio. Therefore, higher the cache hit ratio, better is the performance of the selected caching algorithm [42].

4.3 Caching Algorithms for MRFP

Based on the parameters described in Section 4.2, there are several algorithms which can be used to implement caching in MRFP. This section carries out a theoretical study of some of the algorithms. The caching algorithms discussed in this section are all replacement algorithms which decide on the announcement file which should be evicted when the cache is full [43].

Least-Recently Used (LRU): LRU algorithm uses the last requested time parameter explained in Section 4.2. LRU evicts the least recently used file from the cache. It is based on the hypothesis that the most recently requested announcements are more likely to be requested again. LRU has many variants. An optimized LRU algorithm is provided in [44].

Least-Frequently Used (LFU): LFU algorithm makes use of the popularity parameter mentioned in Section 4.2. During the eviction process, the replacement algorithm removes the announcement file which has been requested least number of times (least popular). LFU algorithm can also be used in combination with LRU [45].

SIZE: This is another basic algorithm used to decide on files to be stored in the cache based on file size alone. This policy removes the largest file first. But if the files are of the same size, LRU policy is applied [46].

History LRU (HLRU): HLRU is an extended version of LRU algorithm. HLRU examines the history of the number of requests made for each of the announcement files. If $r_1, r_2, ..., r_n$ represent the announcement requests made at the times $t_1, t_2, ..., t_n$, the history function can be mathematically represented as [47]:

$$hist(x, h) = \begin{cases} t_i & \text{if there are exactly } h-1 \text{ references between } t_i \text{ and } t_n \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

The function $hist(x, h)$ gives the time of the past h-th request made to a specific cached announcement file x. Therefore when the cache is full, the HLRU algorithm replaces the announcement file with the highest *hist* value. If there are many cached announcements with the *hist* value 0, the traditional LRU algorithm is followed to decide the file to be evicted. Figure 17 shows the graph of the cache-hit ratio with respect to the cache-size from the experimentation carried out in [47]. Figure 17 shows the curves obtained by using $2, 4$ and 6 as the histories in HLRU and also curve from the traditional LRU. The cache-hit ratio increases with the increase in the histories taken to calculate the *hist* values. Therefore, $HLRU(6)$ gives better cache-hit ratio compared to the other cases.

Segmented LRU (SLRU): SLRU algorithm uses both the frequency and recency parameters when making the replacement decision. In SLRU, the available cache memory is divided into two segments namely protected and unprotected segments [48]. The protected segment is reserved for popular announcement files. When an announcement is requested for the first time, it is fetched and stored in the unprotected segment. When a cache-hit occurs, this file is moved from unprotected segment to the protected segment. When either of protected or unprotected segment is full, LRU policy is used for the eviction process. Therefore, both segments manage the files using LRU algorithm and the files are moved between the segments using LFU policy. By this mechanism, the more popular files are retained the the protected segment of the cache for a longer duration. The key parameter that needs to be specified in this policy is the percentage of cache memory which should be used as protected segment.

Figure 18 shows the variations of the cache-hit ratio with respect to size of the

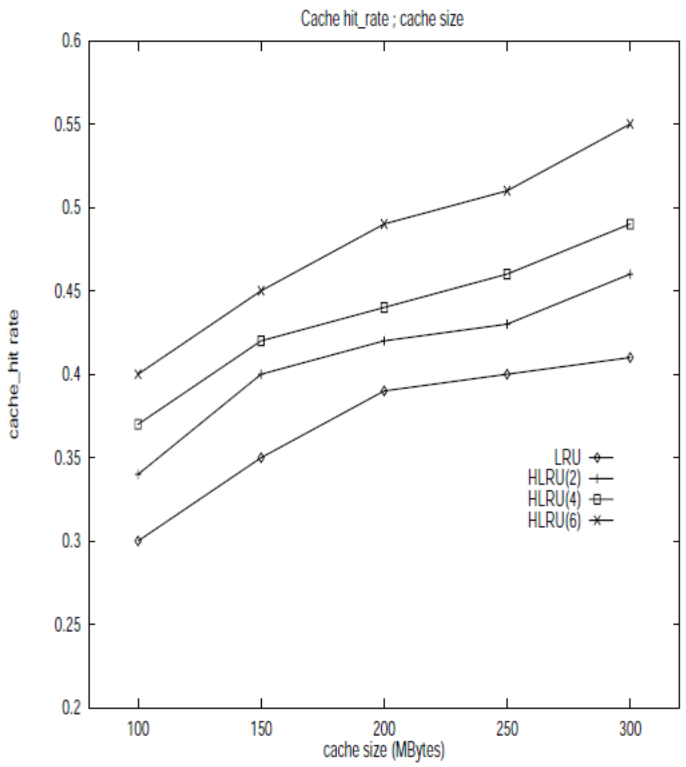

Figure 17: Comparison of cache-hit ratio between HLRU and LRU [47]

protected segment. The curves in Figure 18 are obtained from the simulation carried out in [48]. The only parameter that is varied is the percentage of cache memory which is used as the protected segment. In Figure 18, $SLRU - 10, SLRU - 60$ and $SLRU - 90$ represents that the percentage of cache memory allocated as protected segment are $10\%, 60\%$ and 90% respectively. If the protected segment is too small (10%), the weight given to recency is significantly more than to frequency. Hence, SLRU behaves like LRU with small protected segment sizes. On the other hand if the protected segment is too large (90%), significant weight is given to frequency and SLRU behaves like LFU. Therefore, an appropriate balance should be met between sizes of protected and unprotected segments. The simulation shows that the best result is achieved for $SLRU - 60$ [48].

There are various cache replacement algorithms based on frequency/recency of announcement files' access [49]. Since external announcement involves an external media server placed in the network, parameters like bandwidth and file retrieval cost are key parameters to be considered. The replacement algorithms described so far ($LRU, LFU, HLRU$ and $SLRU$) are based on the parameters like size, popularity (frequency) and recency (last requested time). These algorithms do not consider bandwidth and file retrieval cost explained in Section 4.3. There are various function-based strategies which calculate a value for each file (cache object) based on different parameters. The eviction process is done based on the these values associated with each file (usually the file with the lowest value is evicted) [49]. In this section three

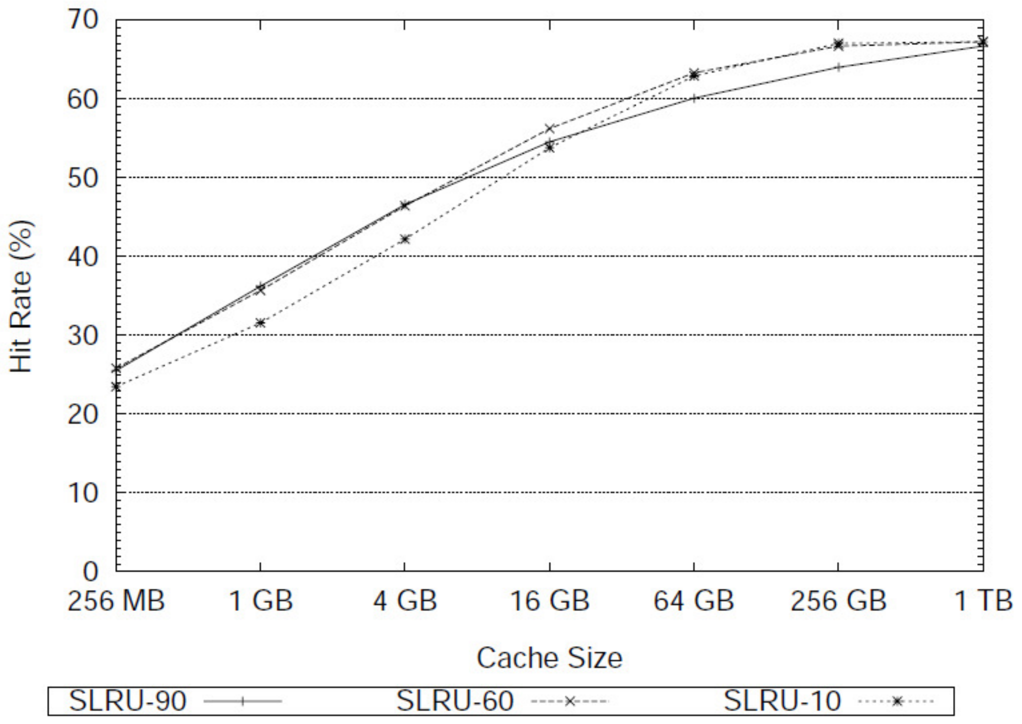

Figure 18: Cache-hit ratio for SLRU policy [48]

such function-based strategies are explained namely Hybrid algorithm (HYB) [50], Lowest Relative Value (LRV) based algorithm [51] and Greedy-Dual Size (GD-S) based policies [52].

Hybrid algorithm (HYB): HYB is a replacement algorithm which uses several factors like file size, popularity (number of past references), available bandwidth to the server and server connection time to calculate value for each file in the cache [49] [50]. The function used to calculate the value of a file is defined as:

$$f(i) = \frac{\left(c_s + \frac{W_b}{b_s}\right)}{s_i} f_i^{W_n} \tag{2}$$

In Equation (2), $f(i)$ gives the value for a file i of size s_i (in bytes), access frequency f_i, available bandwidth to the server b_s and server connection time c_s. W_b and W_n are weighted constants which set the importance of b_s and f_i respectively [50]. This can be verified from the equation as well. As the value of W_b tends towards zero, the importance of the bandwidth to the server is decreased. Similarly as the weight W_n approaches zero, the parameters like s_i become more important than access frequency f_i. The value of $f(i)$ will be large if the file resides in a server which requires large connection time c_s and a low bandwidth link b_s, if the file size s_i is small, and if the file is frequently accessed in the past f_i. Therefore, it is likely that the file with a large value of $f(i)$ is retained in the cache.

Lowest Relative Value (LRV) based algorithm: This policy uses a function to determine the probability that a file in the cache may be accessed again in the future. The value given by the function is the relative value and the file with the Lowest Relative Value (LRV) is removed from the cache when required. The function to find the value (probability P_r) is defined by [49]:

$$P_r(f_i, T_i, s_i) = \begin{cases} P(1, s_i)(1 - D(T_i)) & \text{if } f_i = 1 \\ P(f_i)(1 - D(T_i)) & \text{Otherwise} \end{cases} \tag{3}$$

In the above Equation (3), f_i is the variable which gives the number of times a file i has been previously requested, s_i is the size of the file, T_i is the time when the file was last accessed and $D(t)$ is the time distribution function given by the previous file access times. The probability that a file in the cache will be requested again depends mainly on the time distribution function $D(t)$ of previous file requests. From Equation (3), if the file has been requested only once ($f_i = 1$), the probability that the file will be requested again depends on the size of the file. Figure 19 gives an example curve of a distribution $D(t)$ plotted against logarithmic time scale. Due to the logarithmic time scale, the inter-access time distribution can be shown over a long duration (over 100 days) of time. The study made in [51] uses extensive trace information ([53]) to obtain $D(t)$ and to implement the LRV algorithm.

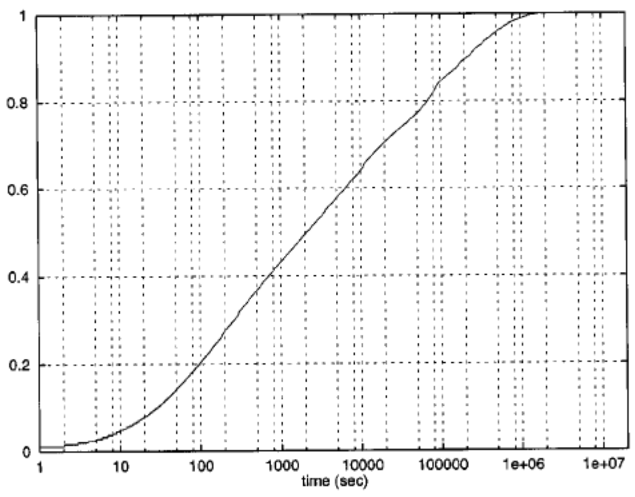

Figure 19: Inter-access time distribution function, D(t) [51]

Figure 20 compares the performance of LRV with other algorithms like LFU, FIFO (Fist In, First Out), LRU, SIZE and RAND (removal algorithm which removes a randomly selected file). Figure 20 shows cache-hit ratios (HR) for different cache sizes. The curves show that the LRV policy gives better performance than rest of the algorithms except SIZE. However, SIZE algorithm shows a poor performance with smaller cache sizes. Therefore, effectively LRV consistently gives better performance when compared with other algorithms [51].

Greedy Dual-Size (GD-S) based approach: Similar to HYB and LRV policies, Greedy Dual-Size (GD-S) algorithm associates each file with a characteristic

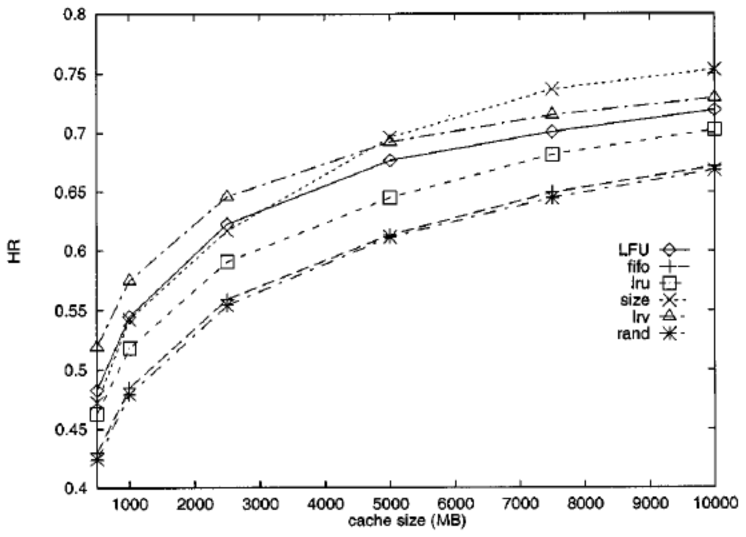

Figure 20: LRV performance (cache-hit ratio, HR) comparison with other algorithms [51]

value H. In the original GD-S algorithm, H is calculated as [52]:

$$H_i = \frac{c_i}{s_i} + L \tag{4}$$

Equation (4) is used to calculate the value H_i for a file i. In the equation, L is called the *inflation* factor and it is calculated whenever there is a cache miss. The GD-S algorithm can be explained in different steps as follows [52]:

- To start with, the *inflation* factor L is initialized to zero. $L \leftarrow 0$
- If a file access gives a cache-hit, it means that the requested file i is already existing in the memory. In this case, H_i is calculated as:

$$H_i \leftarrow L + \frac{c_i}{s_i} \tag{5}$$

- If the file access gives a cache-miss and if there is not enough space to store the file in the cache, GD-S uses H_i values to decide the file which should be removed from the memory. First, a new value of L is calculated as:

$$L \leftarrow min_{i \in M} H_i \tag{6}$$

  From Equation (6), L is assigned to the least value of H among all the files stored in the cache M. The file corresponding to the least H value (now L) is removed from the cache.

- The removed file from the cache is now replaced by the newly downloaded file. The H value for the new file is calculated as:

$$H_i \leftarrow L + \frac{c_i}{s_i} \tag{7}$$

  The L value used in Equation (7) is the one which is calculated from Equation (6).

The implementation and analysis made in [52] show that GD-S outperforms (in cache-hit ratio) other caching policies like LRU, SIZE and LRV. There are many algorithms proposed as extensions of GD-S. Some of the algorithms based on GD-S algorithms are Greedy Dual-Size Frequency (GDSF) [54], Greedy Dual-Size Popularity (GDSP) [40], Greedy Dual* [55], etc. A comparison among GD-S variants is made in [40] and it shows that GDSP outperforms other algorithms and hence this thesis work makes a detailed study on GDSP.

Greedy Dual-Size Popularity (GDSP): GDSP is a modified version of GD-S algorithm. The main idea behind GDSP is to blend the basic GD-S policy with the accurate popularity information of all the files in the cache [40]. One of the drawbacks of GD-S policy is that it does not take the access frequency of files into account. GDSP makes an improvement in this area by using access frequency to calculate a *utility* value for each file. The *utility* for a file i is given by:

$$u(i) = f(i) \times \frac{c_i}{s_i} \tag{8}$$

Equation (8) gives the *utility* value $u(i)$ of i with retrieval cost c_i, size s_i and with access frequency $f(i)$. Therefore, $u(i)$ represents the cost per byte for all accesses of i in a known period of time. GDSP uses $u(i)$ along with the *inflation* factor L and characteristic value H to modify GD-S in an effective way as follows [40]:

- The *inflation* factor L is initialized to zero. $L \leftarrow 0$
- If there is a cache-hit on file request, H is calculated as:

$$H_i \leftarrow L + f(i) \times \frac{c_i}{s_i} \tag{9}$$

$$i.e, H_i \leftarrow L + u(i)$$

- If there is a cache-miss and if there is no space in the cache for the new file, GDSP uses the same method as GD-S to decide the file to be removed from the cache.

$$L \leftarrow min_{i \in M} H_i \tag{10}$$

The file corresponding to the least value of H is removed from the cache to make room for the new one. It can be noted that Equation (10) and Equation (6) are exactly the same. The difference lies in the calculation of H_i.

- The H value for the newly replaced file is calculated as below:

$$H_i \leftarrow L + f(i) \times \frac{c_i}{s_i} \tag{11}$$

The L value used in Equation (11) is obtained from Equation (10).

The implementation and performance evaluation of GDSP is described in [40]. The simulations carried out in [40] use extensive trace information from DEC ([53]) and NLANR ([56]). Cache-hit ratio (HR) is taken as a performance metric to compare GDSP with other algorithms like LRU, LFU and GD-S. Figure 21 shows the cache-hit ratios (HR) of these algorithms as a function of cache-sizes (in logarithmic scale) for both DEC and NLANR traces. Figure 21 shows the version of GD-S and GDSP algorithms with the assumption that the cost c_i of transferring each file is constant. Hence these are represented as $GDS(1)$ and $GDSP(1)$ respectively.

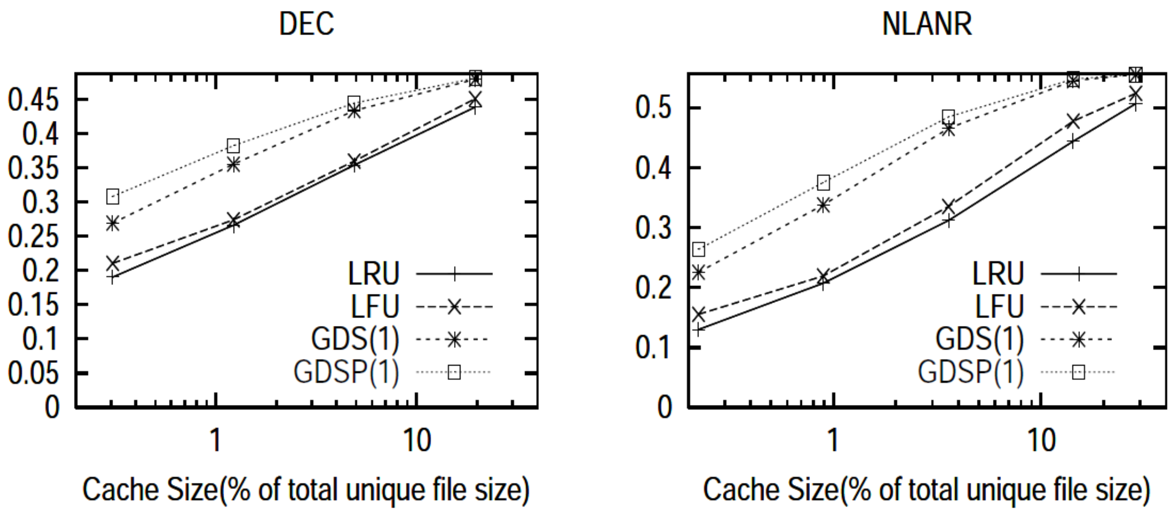

Figure 21: Performance of GDSP under constant cost [40]

With smaller cache sizes, GDSP consistently outperforms GD-S, LFU and LRU policies. However, when the cache sizes are large, the performance of all algorithms tends to converge. Overall, by emphasizing popularity as an important factor, GDSP is evaluated as an effective caching algorithm which can be used in MRFP.

4.4 Summary

Chapter 4 makes a study on using the limited MRFP memory as cache when HTTP/FTP is used as an interface between MRFP and the external media server.

This chapter described in detail about various parameters that affect caching along with explaining various file replacement caching algorithms that are relevant to MRFP [57]. Table 5 summarizes all the algorithms explained in this chapter. It can be noted that the selection of caching algorithm to implement is dependent on the scenario in hand. If the files are of widely varying sizes placed in a network with high latency, GDSP is preferred since it takes all the necessary caching parameters into consideration.

Table 5: Summary of caching algorithms

Algorithm	**Reference parameters**	**Consideration of nonuniformity (size, cost)**	**Implementation complexity (processing time)**
LRU	Last requested time	-	$O(1)$
LFU	Frequency (Popularity)	-	$O(\log_2 n)$
SIZE	-	Size	$O(\log_2 n)$
HLRU	rth access time, Frequency	-	$O(\log_2 n)$
SLRU	Last requested time	-	$O(n)$
HYB	Frequency	Size, latency	$O(\log_2 n)$
LRV	Last requested time, Frequency	Size	$O(n)$
GD-S	Last requested time	Size, latency	$O(\log_2 n)$
GDSP	Last requested time, Frequency	Size, latency	$O(\log_2 n)$

5 Prototyping the External Announcement Approach

Chapter 3 described the architecture proposed to achieve the external announcements in MRFP. This chapter describes a prototype to realize this approach. The prototype includes the use of a network emulator to emulate the properties of a Wide Area Network (WAN) for testing the performance of different external announcement approaches. Section 5.1 outlines the design of the prototype, Section 5.2 explains about the network emulator used to emulate the properties of WAN and Section 5.4 gives an overview of the open-source implementations used as clients and servers in this prototype.

5.1 Prototype Design

The aim of the prototype is to realize the behavior of external announcement approach in a network-like scenario. Figure 22 shows the prototype design to achieve external announcements in MRFP. To start with, MRFP receives the URL for the announcement media file via H.248 protocol. This URL is used by the control logic implemented in MRFP to dispatch the announcement request depending on the type of external server being used. Also, as shown in Figure 22, a network emulator is used between the MRFP and the external server to emulate typical network characteristics like packet loss, latency, packet duplication, packet corruption, etc. The announcement file from the external server is then downloaded/streamed to the known destination IP/port address on the Ethernet board in MRFP. Finally, the RTP stream of the announcement is verified by the audio/video player as shown in the Figure 22.

The implementation done in this thesis is primarily in the control logic of MRFP as explained in Section 5.3. The control logic is generic and it supports RTSP, HTTP and FTP client programs. Additionally, the details about the requested announcement files (like frequency, last requested time, etc.) are recorded which serve as input to the implementation of a suitable caching algorithm. The audio/video player and the servers used in the prototype are open-source implementations. A more detailed description on these is given in Section 5.4

5.2 Network Emulator

As shown in Figure 22, the network emulation is achieved by placing a device between the MRFP and the server to alter the packet-flow and thus simulating the WAN characteristics. The network emulator used in this thesis work is `netem` [58]. `netem` is readily enabled in the Linux kernel distributions 2.6 or higher. In our prototype, the `netem` is run on a $Ubuntu 10.04.3 LTS$ Linux distribution with kernel version $2.6.32 - 33 - generic$. The machine is configured with two Network Interface Cards (NICs) - `eth2` for handling the ingress traffic (IP packets coming into the

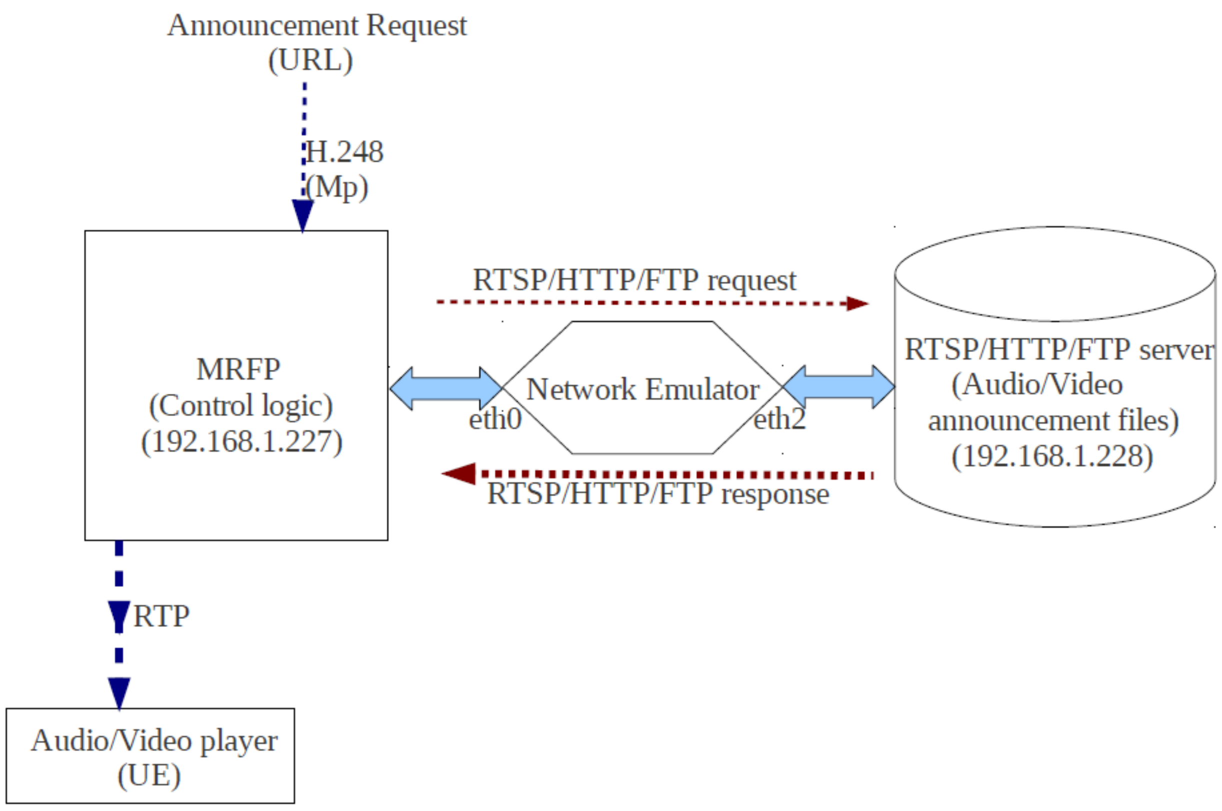

Figure 22: Prototype Design

device) and `eth0` for handling egress traffic (IP packets transmitted out from the device). `netem` can be run in two modes:

- Bridge mode: When the `netem` device is configured to work in bridge mode, it operates in layer two of the OSI (Open Systems Interconnection) model. The two interfaces (for ingress and egress traffic) in the `netem` device function such that the IP packets are forwarded from one to the other. Also, packets are forwarded using a MAC address based forwarding table. Generally the `netem` device is configured in bridge mode if the devices in the network are of the same type. Bridge mode is also easy to configure and is preferred for on a simple network topology. In this thesis work prototype, the `netem` device is configured in bridge mode since it needs to be connected to only two machines (MRFP and external server) of the same type.

- Router mode: This mode operates in layer three of the OSI model. In this mode, the path to forward packets from source to the destination is obtained from the routing table. Configuring `netem` in routing mode is preferred when the network topology is more complex and the devices in the network are heterogeneous. Additionally, routing mode allows us to make more effective and intelligent decisions to send packets.

`netem` uses the Traffic Control (`tc`) tool for packet shaping and packet filtering. Basically, `tc` is used to show/manipulate traffic control settings in the Linux kernel [60]. `tc` is used to affect the packets enqueued in ingress/egress queue [59]. Each of the two interfaces in the `netem` configured device has a queue associated with it which processes the packets in FIFO manner. The queuing disciplines (`qdisc`) make the decision on when to send which packet from the FIFO queues (i.e, how the packets are ordered and sent). Figure 23 shows the block diagram of IP packet handling in the Linux kernel [59].

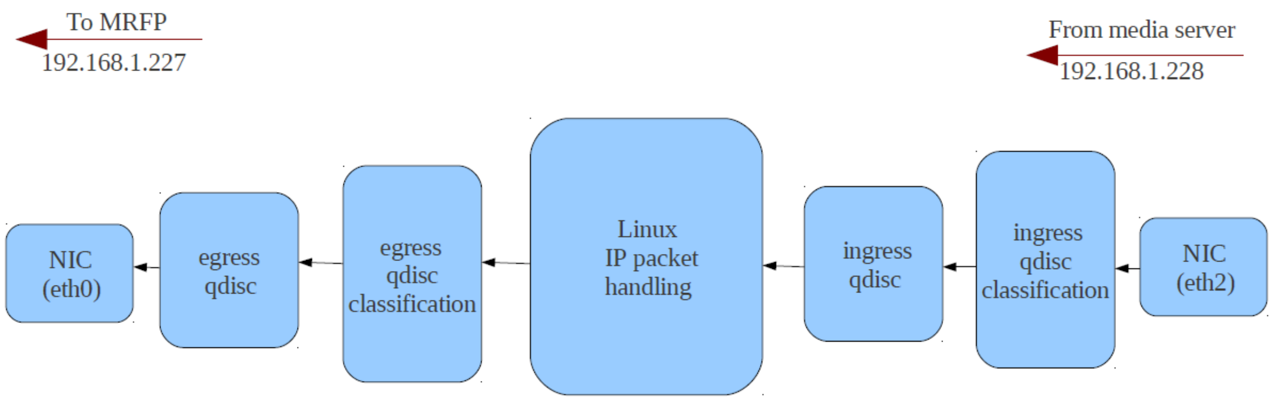

Figure 23: IP packet handling in the Linux kernel [59]

Therefore, the packets entering the `netem` device from one NIC will be processed and sent out from the other NIC. Also as shown in Figure 23, the ingress traffic is generated by downloading/streaming the announcement file from external server to the MRFP. The characteristics of the outbound packets are modified by using `tc` tool to modify the queuing discipline (`qdisc`). `netem` can be configured to perform four basic operations on the IP packets [59]:

- Delay: `netem` can be used to emulate WAN delays by delaying each packet going out of the egress interface. Delays can be introduced in different ways. The simplest way is by specifying a fixed amount of delay on each packet going out of the `netem` device. This is achieved by the following command in the `netem` device:

  ```
  tc qdisc add dev eth0 root netem delay 10ms
  ```

  The above command introduces a fixed delay of `10ms` and it can be verified by `ping` test. When a `ping` is done from MRFP to the server's IP address, the delayed packets can be noticed as below:

  ```
  $ ping -c 3 192.168.1.228
  PING 192.168.1.228 (192.168.1.228) 56(84) bytes of data.
  64 bytes from 192.168.1.228: icmp_seq=1 ttl=64 time=10.8 ms
  64 bytes from 192.168.1.228: icmp_seq=2 ttl=64 time=10.7 ms
  ```

```
64 bytes from 192.168.1.228: icmp_seq=3 ttl=64 time=10.6 ms
--- 192.168.1.228 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 10.671/10.755/10.836/0.108 ms
```

A more practical way of emulating delay is by introducing it in a variable manner. This is achieved by specifying a random delay along with the fixed delay. This method is used in the prototype in this thesis to emulate WAN. An example command is:

```
tc qdisc add dev eth0 root netem delay 10ms 5ms
```

Basically the above command results in delaying the packets from the egress interface with a degree of randomness of `5ms` (i.e.,`10ms`$\pm$`5ms` net delay). A `ping` test on the above command gives the following:

```
$ ping -c 5 192.168.1.228
PING 192.168.1.228 (192.168.1.228) 56(84) bytes of data.
64 bytes from 192.168.1.228: icmp_seq=1 ttl=64 time=5.87 ms
64 bytes from 192.168.1.228: icmp_seq=2 ttl=64 time=7.28 ms
64 bytes from 192.168.1.228: icmp_seq=3 ttl=64 time=13.0 ms
64 bytes from 192.168.1.228: icmp_seq=4 ttl=64 time=6.73 ms
64 bytes from 192.168.1.228: icmp_seq=5 ttl=64 time=12.9 ms
--- 192.168.1.228 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 5.870/9.184/13.074/3.163 ms
```

- Packet loss: The `netem` device can drop some packets before sending it towards MRFP, resulting in packet loss. Packet loss is generally specified in percentage and the `tc` command used to achieve the same is:

```
tc qdisc add dev eth0 root netem loss 20%
```

The packet loss can be confirmed by verifying the sequence number of the arriving packets by performing a `ping` test from MRFP to the external server. The `ping` test results are as below:

```
$ ping -c 5 192.168.1.228
PING 192.168.1.228 (192.168.1.228) 56(84) bytes of data.
64 bytes from 192.168.1.228: icmp_seq=1 ttl=64 time=0.306 ms
64 bytes from 192.168.1.228: icmp_seq=2 ttl=64 time=0.797 ms
64 bytes from 192.168.1.228: icmp_seq=3 ttl=64 time=0.577 ms
64 bytes from 192.168.1.228: icmp_seq=5 ttl=64 time=0.828 ms
--- 192.168.1.228 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.306/0.627/0.828/0.209 ms
```

Clearly, the packets with the sequence number 4 (`seq=4`) is missing thus verifying the packet loss. In the above example, a high packet loss percentage is given (20%) only for the purpose of `ping` test. In the actual prototype settings, the packet loss percentage used are very low (0.1%, 0.5%, etc.).

- Packet duplication: `netem` can be configured to duplicate packets using `tc` command as below:

  ```
  tc qdisc add dev eth0 root netem duplicate 10%
  ```

  With the above command, `netem` duplicates 1 in every 10 packets. The `ping` test verifies the packet duplication as shown below:

  ```
  $ ping -c 5 192.168.1.228
  PING 192.168.1.228 (192.168.1.228) 56(84) bytes of data.
  64 bytes from 192.168.1.228: icmp_seq=1 ttl=64 time=0.427 ms
  64 bytes from 192.168.1.228: icmp_seq=2 ttl=64 time=0.594 ms
  64 bytes from 192.168.1.228: icmp_seq=3 ttl=64 time=0.617 ms
  64 bytes from 192.168.1.228: icmp_seq=4 ttl=64 time=0.537 ms
  64 bytes from 192.168.1.228: icmp_seq=4 ttl=64 time=0.560 ms (DUP!)
  64 bytes from 192.168.1.228: icmp_seq=5 ttl=64 time=0.591 ms
  --- 192.168.1.228 ping statistics ---
  5 packets transmitted, 5 received, +1 duplicates, 0% packet loss, time 4000ms
  rtt min/avg/max/mdev = 0.427/0.554/0.617/0.065 ms
  ```

- Packet corruption: A packet can be corrupted by introducing single bit errors at a random offset in the packet [58]. Such corrupted packets lead to random noise thus emulating the environment of a WAN. The number of packets to be corrupted can be specified as a corruption percentage in the `tc` command as below:

  ```
  tc qdisc add dev eth0 root netem corrupt 1%
  ```

  The corrupted packets are not received. A `ping` test with packet corruption gives a similar output as packet loss.

In the prototype tested in this thesis, the WAN is emulated by using a combination of variable delay and packet loss. Different combinations of delays and packet loss are considered to test the behavior of the external announcement setup. The analysis of the measurements taken from this setup is described in Section 6.

5.3 Implementation

As mentioned earlier in Section 5.1, the implementation done in this thesis is a small part of MRFP control logic. The announcement URL obtained via the H.248 interface is alone stored in a separate file in MRFP. The implementation is done using `python` programming language [61] and it performs the following operations:

- If RTSP is the chosen interface, the MRFP establishes an RTSP session with the external server. The target announcement file is streamed directly to the MRFP specified IP/port address. The RTP stream of the announcement is then played by the media player opened by the MRFP.
- If HTTP/FTP is chosen as the interface, the MRFP uses the stored URL to first check if the requested file is already existing in the MRFP memory. If it exists then it locally streams the file via RTP to the media player. If the file does not exist in cache, then it is fetched from the HTTP/FTP server and stored in the memory before streaming it locally. Also, on each announcement request, a `cache_parameter_file` is updated with parameters like size, frequency and last requested time and this file can be used for implementing a caching algorithm in MRFP. A sample `cache_parameter_file` based on the measurements taken from the prototype is shown in Section 6.3.

5.4 Open-source Implementations

This section gives a brief description of the open-source implementations used to realize the external announcement approach from the prototype. Open-source server implementations are used in each of RTSP, HTTP and FTP scenario which handle and respond to the requests made by the MRFP. Also, the MRFP control logic uses a open-source audio/video player to play the RTP stream of the announcement media from the MRFP as shown in Figure 22.

LIVE555 Media Server: "LIVE555 Media Server" is a open-source complete RTSP server implementation [62]. This is used in the external server when RTSP is the protocol chosen as an interface to achieve external announcements. The announcement files to be streamed are stored in the same directory from which the server is launched. LIVE555 server application runs on port `8554` by default. One important reason to use this RTSP server is that it can stream various types of audio/video files like `.mp3, .mpg, .wav, .264`, etc.

Apache HTTP server (httpd): httpd is a popular, robust HTTP server implementation which also offers wide range of features [63]. httpd constitutes of various Multi-Processing Modules (MPMs) which can be selected and configured as per the requirements. When the MRFP wants to fetch the required media file, it does a `HTTP GET` on the httpd external server. The server responds to `GET` request by transmitting the announcement file.

Pure-ftpd: Pure-ftpd is a free, secure, efficient and standard compliant FTP server [64]. Pure-ftpd server has several advantages:

- Pure-ftpd uses simple command line options for configuration thus getting rid of complex configuration files.
- It allows monitoring and limiting bandwidth usage by each user.
- It supports Lightweight Directory Access Protocol (LDAP) [65] server authentication with secure cryptographic hashes.
- Pure-ftpd allows users to have virtual FTP accounts. This enables users to have different passwords for system and FTP accounts.
- All FTP user's information can be stored in a centralized MySQL database.

VLC media player: VLC (originally VideoLan Client) is an open-source, portable and powerful multimedia player which plays various formats of audio and video [66]. MRFP streams the announcement media to the destination IP/port address via RTP. Therefore, it is important to choose a media player which can reliably be used for RTP streaming. VLC satisfies this requirement very well as it can be used as a server to stream content and as a client to receive streams [67].

For external announcement approach using HTTP/FTP protocols, MRFP uses VLC media player as both client and server. It uses VLC media player as a server to stream the announcements locally. At the target IP/port address MRFP uses VLC as a client to capture the RTP stream. When RTSP is used as a protocol, MRFP uses VLC media player at the target address as a client to capture the RTP packets streamed directly from the external server.

5.5 Summary

A prototype to realize the external announcement approach explained in Chapter 3 is described in this chapter. This chapter describes the implementation along with the tools and softwares used to build the prototype. To sum-up, the prototype shows that the external announcement approach can be practically deployed in the MRFP to achieve the announcement service in IMS.

6 Measurements and Analysis

This chapter describes the parameters measured using the prototype shown in Figure 22. Section 6.1 states the parameter which is measured from the setup, followed by Section 6.2 which gives an account of the media files which are used as input for the measurements. The measurements are made under WAN environment emulated by `netem` using packet loss and variable delay. Sections 6.3 and 6.4 reports the actual measurements made and a comparative analysis on the same, respectively.

6.1 Parameter Measured

The parameter which is measured from the prototype is the latency for the announcement file to start playing from the time the announcement request was made to MRFP. This latency is measured when RTSP, HTTP and FTP are used as interfaces to achieve external announcements. Also, the measurements are taken when various types of audio and video announcements of different sizes are requested. This measurement helps to make a behavioral analysis of the external announcement technique.

6.2 Announcement Media Types

The prototype is tested with both audio and video announcement files. The type of audio files tested are MPEG audio (`.mp3`) and waveform audio (`.wav`, μ law encoded). The video file tested is a MPEG (`.mpg`) audio and video compressed file. These formats are chosen as they are commonly used announcements formats. However, it has also been verified that the setup works for other media formats like MPEG-4 AVC (Advanced Video Coding, `.H.264`) and MPEG Transport Stream (`.ts`). Sample audio/video files of various formats are available from [68] and [69].

6.3 Measurements

In this section, the behavior of the prototype is showcased. The results are collected for three different cases (RTSP, HTTP, FTP) and their behavior is compared. To recall Section 5.4, the prototype uses:

- live555 Media server as RTSP server implementation. The default buffer size used in the server implementation is `50 kilobytes`.
- Apache HTTP server (httpd) as for the case of HTTP.
- Pure-ftpd as the FTP server implementation.

Table 6 gives a list of files tested using the prototype. The performance of the prototype using each of these files is evaluated in this section. For simplicity, the files are hereby referred with the names in the `Representation` column of Table 6. Table 6 also lists the sizes of each file tested which is a key information to be considered and is explained in Section 6.4.

Table 6: Test announcement files

Filename	**Format**	**File size** (in kilobytes)	**Representation**
mp3_test.mp3	`mp3`	492	File1
mpg_audio_video.mpg	MPEG-1 (`mpg`)	707	File2
wav_88_2kbps_ulaw.wav	μ law `wav`	152	File3
test_mp3_large.mp3	`mp3`	5541	File4

Table 7: Latency measurements of File1

Delay (ms) (X, x) (X±x)	**Packet loss (%)**	**Latency (ms) FTP case**	**Latency (ms) HTTP case**	**Latency (ms) RTSP case**
-	-	22.1	22.2	47.6
5, 5	0.1	253	220	84.3
10, 5	1	643	612	119
10, 5	0.5	521	570	106
15, 5	0.5	675	681	149

Table 8: Latency measurements of File2

Delay (ms) (X, x) (X±x)	**Packet loss (%)**	**Latency (ms) FTP case**	**Latency (ms) HTTP case**	**Latency (ms) RTSP case**
-	-	243	238	47.6
5, 5	0.1	274	269	84.3
10, 5	1	738	755	119
10, 5	0.5	690	675	106
15, 5	0.5	978	980	149

Tables 7, 8, 9 and 10 show the results from the prototype when the announcement file accessed was File1, File2, File3 and File4 respectively. Fundamentally, these tables show the latency or the time taken for the announcement file to start playing at the destination since the announcement request was made to MRFP. For each file, five different cases are measured for each protocol. First, a measurement is taken without introducing any delay or packet loss. Followed by this the delay and packet loss are used in four different combinations to measure the latency. It can be noted that the latency measured are listed in milliseconds (ms) in case of Tables 7, 8 and 9 and in seconds (s) in Table 10. The latency measured in Table 10 is in seconds because of the size of the file (File4) being accessed. When HTTP or FTP are used as the interfaces, File4 being a large audio file takes more time to be downloaded, which is measured in seconds.

Table 9: Latency measurements of File3

Delay (ms) (X, x) (X±x)	**Packet loss (%)**	**Latency (ms) FTP case**	**Latency (ms) HTTP case**	**Latency (ms) RTSP case**
-	-	13.6	8.14	47.6
5, 5	0.1	84	53.5	84.3
10, 5	1	233	228	119
10, 5	0.5	240	221	106
15, 5	0.5	319	311	149

Table 10: Latency measurements of File4

Delay (s) (X, x) (X±x)	**Packet loss (%)**	**Latency (s) FTP case**	**Latency (s) HTTP case**	**Latency (s) RTSP case**
-	-	0.13	0.14	0.047
5, 5	0.1	1.17	1.53	0.084
10, 5	1	6.92	7.23	0.119
10, 5	0.5	5.31	5.86	0.106
15, 5	0.5	7.17	7.78	0.149

Cache Parameter File

A `cache_parameter_file` is maintained in MRFP which is updated for every announcement request. This file is used as a primary input to implement caching in MRFP. The `cache_parameter_file` has information about the announcement file-name, frequency of each file request, size of the file and the last requested date/time stored in the following format:

The `cache_parameter_file` is updated upon every announcement request in the following format:

```
File_Name     frequency     file_size     Last_Requested_Time
```

A sample `cache_parameter_file` is shown below based on the measurements from the prototype:

```
mp3_test.mp3        4       492147       2011-12-27  14:52:12.328807
test_mp3_large.mp3       5       5541781       2011-12-27  14:55:28.056783
mpg_audio_video.mpg      6       706564       2011-12-27  14:54:52.369125
wav_88_2kbps_ulaw.wav       3       152326       2011-12-27  14:58:32.419514
```

6.4 Analysis

The following observations can be made from the tabulated results in Tables 7, 8, 9 and 10:

- The latency is measured for files of varying sizes from a few kilobytes to many megabytes. It can be observed that as the file size increases the performance in case of HTTP is better than FTP case. However, the performance between the two cases do not show any significant difference when announcements of smaller size (File1, File2 and File3) are considered.

- From signaling point of view, RTSP involves more signaling and therefore takes more time to set up an RTSP session when compared to a HTTP or FTP connection. Hence, if the file sizes are relatively smaller (File1, File2 and File3), the resulting latency to download the file and start playing the announcement is lower than the corresponding RTSP case (Tables 7, 8 and 9). Hence, the file size is a key parameter which can make a difference in the latency behavior.

- It can be noted that in case of RTSP, the announcement is played directly to the destination without reaching MRFP control logic. The latency is independent of the size of the announcement files as there is no download involved in this case. Therefore, the RTSP case results show the same latency values for all the files. Hence, for larger files (File4), the results shown for RTSP case is significantly better than the HTTP or FTP cases. This can be verified from Table 10. In a “worst-case scenario” with a high packet loss of `0.5%` and a variable delay of `15`$\pm$`5 ms`, the latency in downloading the files via HTTP or FTP is over `7` seconds when compared to RTSP which starts playing the announcements in `149 ms`. Therefore, the real-time nature of RTSP gives the results shown in Table 10.

- RTSP case suffers a drawback when audio/video announcement quality is considered. In a WAN environment with large delay and packet loss, even though the announcement play is started earlier than the HTTP/FTP cases, the RTP packet stream from the external server is affected by the network environment and may lead to poor media quality. The audio/video quality analysis is out of scope of this thesis work.

- The latency measured is independent of the announcement being an audio or video file. The prototype follows the same procedure in both cases. The Streamer component in MRFP is capable of locally streaming both audio and video types.

- In cases of HTTP and FTP, using an effective caching algorithm in MRFP would altogether reduce the external server requests which significantly increases the performance.

7 Conclusions and Future Work

The concluding remarks on this thesis work is given in Section 7.1. The possible work that can be done as a continuity to this thesis is pondered in Section 7.2.

7.1 Conclusions

The work done in this thesis studies the behavior of the announcement application in IMS when a server is placed outside the MRFP node in the core network. A prototype to play the announcements externally has been built which involves implementing a part of the MRFP control logic to communicate with the server to download/stream the announcements. In this thesis, the servers used to store the announcements are based on three different protocols namely RTSP, HTTP and FTP. The main idea is to find a suitable protocol which can be used as an interface between the MRFP and the server. The details on how the external announcement is achieved using RTSP, HTTP and FTP is explained in Sections 3.2.3, 3.3.2 and 3.4.3 respectively. Therefore, it can be concluded that it is possible to achieve external announcements in MRFP with the architecture proposed in Chapter 3.

Conventionally in IMS, the MRFP handles the announcement service without any external server. MRFP node uses it's limited memory to store the announcements. As the number of announcement files increases, the existing files in the node has to be flushed (deleted) out to make room for the new ones. This becomes inconvenient and this thesis studies ways to overcome such drawbacks.

Another important study area in this thesis is on how to utilize the existing MRFP memory as cache. This is advantageous when HTTP or FTP are used as interfaces between MRFP and the server. Basically, when a file is downloaded from the server, the caching algorithm in MRFP decides if it should be stored in the cache. If the cache is full, then the algorithm decides which file in the cache is replaced by the new file. Chapter 4 makes a study of various caching techniques like LRU, LFU, SIZE, HLRU, SLRU, HYB, LRV, GD-s and GDSP based on various caching parameters like size, frequency, last requested time, network latency etc. A comparison of various caching techniques studied is shown in Table 5 from which it can be concluded that implementing a caching algorithm based on GDSP would increase the performance when MRFP memory is used as cache.

The results from the prototype of the external announcement approaches are shown in Section 6.3. `netem` tool is used as a network emulator to emulate WAN characteristics. It has to be noted that the conclusions are made by emulating only network delay and packet loss. The reliability of the prototype is not tested under other WAN characteristics like packet duplication, packet corruption, etc. The latency (time taken for the announcement to start playing since it was requested to MRFP) is measured for four files of different sizes. Each file is tested under three protocol interfaces: RTSP, HTTP and FTP. The results show that the protocol that may be used as an interface depends on the properties of the announcements that the MRFP handles. It can be concluded that:

- If majority of the files handled by MRFP are of smaller size (few kilobytes), it is advantages to use HTTP/FTP as an interface to the external server.
- If majority of the announcements handled are of larger size (many megabytes), using RTSP gives better results. HTTP/FTP is inconvenient as the latency to download the announcement is large.
- Using RTSP method reduces the overhead of reserving additional media resources when an announcement is fetched (Section 3.2.3). This is because the file is directly streamed to the destination without going through extra components in MRFP, unlike HTTP/FTP case.
- When HTTP/FTP is used as an interface, the study made on caching algorithms (Section 4.3) prove GDSP to be an efficient technique that may be used in MRFP. GDSP takes into account the parameters like frequency, last requested time, network loss/delay and file size, all of which are available parameters.

7.2 Future Work

This thesis work can be extended and enhanced to further study the external announcement techniques. It would be interesting to study the performance of the proposed architecture in a real WAN environment instead of emulated one. This would result in introducing additional network characteristics like random packet duplication, packet corruption, etc along with variable delay and packet loss. The prototype can also be tested with different open-source server implementations to study the impact on it's performance. Also, one potential area for further research is to analyze the impact of introducing enhanced secure communication between MRFP and the external server.

Finally, one key area to be studied further is the caching mechanism in MRFP. The study made in this thesis shows that with certain limitations, GDSP can be effectively used as a caching algorithm. However, there are also various other caching models which can be further analyzed and compared. When using the caching algorithms listed in Table 5, the entire media object is downloaded and stored as a static object. It would be interesting to analyze the impact of segment-based caching techniques as well. In segment-based caching, the media objects are not stored as a whole; instead the cache only stores the popular segments of the media (for example, first 5 seconds of a audio file) [70]. It is important to make a elaborate study on various caching algorithms before incorporating one in the MRFP.

References

[1] E. Burger, Ed., J. Van Dyke, A. Spitzer, "Basic Network Media Services with SIP," RFC 4240 (Standard), Internet Engineering Task Force, December 2005. [Online]. Available: http://tools.ietf.org/html/rfc4240

[2] Dialogic, "The Architecture and Benefits of IMS," Fetched 15 September 2011. [Online]. Available: http://www5.dialogic.com/products/docs/whitepapers/11297-ims-arch-benefits-wp.pdf

[3] Ericsson, "IMS Control and Media," Fetched 15 September 2011. [Online]. Available: http://www.ericsson.com/ourportfolio/products/ims-control-and-media

[4] Ericsson, "The value of using the IMS architecture," Fetched 26 September 2011. [Online]. Available: http://www.techabulary.com/i/ims/ims_ip_multimedia_subsystem.pdf

[5] 3GPP TS 23.002, "Network Architecture," [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23002.htm

[6] Martin Koukal, Robert Bestak, "Architecture of IP Multimedia Subsystem," *48th International Symposium ELMAR-2006,* Zadar, Croatia.

[7] Metaswitch Networks, "IMS Architecture," Fetched 27 September 2011. [Online]. Available: http://www.metaswitch.com/sbc-session-border-controller/ims-architecture.aspx

[8] Mohammed A Qadeer, Afaq H. Khan, Juned A. Ansari, Sariya Waheed, "IMS Network Architecture," *2009 International Conference on Future Computer and Communication,* Kuala Lumpur, Malaysia.

[9] Eventhelix, "MRFC-AS Interfaces (IMS Conference Call)," Fetched 29 September 2011. [Online]. Available: http://eventhelix.com/ims/conference/mrfc-as-sequence-diagram.pdf

[10] Eventhelix, "MRFP Interfaces (IMS Conference Call)," Fetched 29 September 2011. [Online]. Available: http://eventhelix.com/ims/conference/mrfp-sequence-diagram.pdf

[11] Burnett, D., Hunt, A., McGlashan, S., Porter, B., Lucas, B., Ferrans, J., Rehor, K., Carter, J., Danielsen, P., and S.Tryphonas, "Voice Extensible Markup Language (VoiceXML) Version 2.0," W3C REC REC-voicexml20-20040316, March 2004.

[12] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986 (Standard), Internet Engineering Task Force, January 2005. [Online]. Available: http://tools.ietf.org/html/rfc3986

[13] J. Rosenberg, H. Schulzrinne, "An Offer/Answer Model with the Session Description Protocol (SDP)," RFC 3264 (Standard), Internet Engineering Task Force, June 2002. [Online]. Available: http://tools.ietf.org/html/rfc3264

[14] P. May, "Application Services in an IP Multimedia Subsystem (IMS) Network," *Data Connection Limited,* whitepaper, 2005. [Online] Available: http://www.tmia.org/white_papers/DCL_WhitePaper_IMS.pdf

[15] Jonas Reinius, "Cello-An ATM transport and control platform," *Ericsson,* [Online] Available: http://www.ericsson.com/ericsson/corpinfo/publications/review/1999_02/files/1999021.pdf

[16] H. Schulzrinne, R. Lanphier, A. Rao, "Real Time Streaming Protocol (RTSP)," RFC 2326 (Standard), Internet Engineering Task Force, April 2002. [Online]. Available: http://tools.ietf.org/html/rfc2326

[17] Internet Assigned Numbers Authority, "Service Name and Transport Protocol Port Number Registry," [Online]. Available: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml

[18] Eventhelix, "HTTP Sequence Diagram," Fetched 23 November 2011. [Online]. Available: http://eventhelix.com/RealtimeMantra/Networking/http_sequence_diagram.pdf

[19] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Standard), Internet Engineering Task Force, August 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246

[20] M. Handley, V. Jacobson, "SDP: Session Description Protocol," RFC 2327 (Standard), Internet Engineering Task Force, April 1998. [Online]. Available: http://tools.ietf.org/html/rfc2327

[21] Yan Liu, Guo-Hui Zhong, Yu Liu, Hua-Qiang He, Fu-Rong Wang, "The Research of Streaming Media Mutual Digest Authentication Model Based on RTSP," *Proceedings of the 2008 International Conference on Wavelet Analysis and Pattern Recognition,* Hong Kong.

[22] Yong-Ju Lee, Ok-Gee Min, Hag-Young Kim, "Performance Evaluation Technique of the RTSP based Streaming Server," *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05),* Daejeon, Korea.

[23] Yong Liu, Bin Du, Shuling Wang, Haibo Yang, Xingwei Wang, "Design and Implementation of Performance Testing Utility for RTSP Streaming Media Server," *2010 First International Conference on Pervasive Computing, Signal Processing and Applications (PCSPA).*

[24] J. Soininen, Ed., Nokia, "Transition Scenarios for 3GPP Networks," RFC 3574 (Standard), Internet Engineering Task Force, August 2003. [Online]. Available: http://tools.ietf.org/pdf/rfc3574.pdf

[25] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0," RFC 1945 (Standard), Internet Engineering Task Force, May 1996. [Online]. Available: http://tools.ietf.org/html/rfc1945

[26] T. Berners-Lee, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2068 (Standard), Internet Engineering Task Force, January 1997. [Online]. Available: http://tools.ietf.org/html/rfc2068

[27] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Standard), Internet Engineering Task Force, June 1999. [Online]. Available: http://tools.ietf.org/html/rfc2616

[28] Henning Schulzrinne, "A comprehensive multimedia control architecture for the Internet," *Proceedings of the IEEE 7th International Conference on Network and Operating System Support for Digital Audio and Video, p 65-76, 1997.*

[29] P. Deutsch, A. Emtage, A. Marine, "How to Use Anonymous FTP," RFC 1635 (Standard), Internet Engineering Task Force, May 1994. [Online]. Available: http://tools.ietf.org/html/rfc1635

[30] M. Allman, S. Ostermann, "FTP Security Considerations," RFC 2577 (Standard), Internet Engineering Task Force, May 1999. [Online]. Available: http://tools.ietf.org/html/rfc2577

[31] Bindshell, "Manipulating FTP Clients Using The PASV Command," March 2007. [Online]. Available: http://www.bindshell.net/papers/ftppasv/ftp-client-pasv-manipulation.pdf

[32] Eventhelix, "FTP Session Sequence Diagram," Fetched 23 November 2011. [Online]. Available: http://www.eventhelix.com/realtimemantra/networking/FTP_Port_21.pdf

[33] M. Horowitz, S. Lunt, "FTP Security Extensions," RFC 2228 (Standard), Internet Engineering Task Force, October 1997. [Online]. Available: http://tools.ietf.org/html/rfc2228

[34] I. Cooper, I. Melve, G. Tomlinson, "Internet Web Replication and Caching Taxonomy," RFC 3040 (Standard), Internet Engineering Task Force, January 2001. [Online]. Available: http://tools.ietf.org/html/rfc3040

[35] Wei-Kuo Liao, Pang-Hsin Shih, "Architecture of Proxy Partial Caching Using HTTP for Supporting Interactive Video and Cache Consistency," *Proceedings*

of the Eleventh International Conference on Computer Communications and Networks, p 216-221, 2002.

[36] Benowitz, E., Ercegovac, M., Fallah, F, "Reducing the Latency of Division Operations with Partial Caching," *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, Volume 2, p 1598-1602, 2002.*

[37] Daniel Zeng, Fei-Yue Wang, Mingkuan Liu, "Efficient Web Content Delivery Using Proxy Caching Techniques," *IEEE TRANSACTIONS On Systems, Man, AND Cybernetics - Part C: Applications and Reviews, Volume 34, Issue 3, p 270-280, 2004.*

[38] Zhourong Miao, Antonio Ortega, "Scalable Proxy Caching of Video Under Storage Constraints," *IEEE Journal on Selected Areas in Communications, Volume 20, Issue 7, p 1315-1327, 2002.*

[39] Wei Liu, Chun Tung Chou, Zongkai Yang, Xu Du, "Popularity-wise Proxy Caching for Interactive Streaming Media," *29th Annual IEEE International Conference on Local Computer Networks, p 250-257, 2004.*

[40] Shudong Jin, Bestavros, A., "Popularity-aware greedy dual-size Web proxy caching algorithms," *Proceedings of the 20th International Conference on Distributed Computing Systems, p 254-261, 2000.*

[41] Anja Feldmann, Ramon Caceres, Fred Douglis, Gideon Glass, Michael Rabinovich, "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments," *Proceedings of the IEEE INFOCOM, p 107-116, 1999.*

[42] Daniela Rosu , Arun Iyengar , Daniel Dias, "Hint-based Acceleration of Web Proxy Caches," *Proceedings of the 19th IEEE International Performance, Computing, and Communications Conference, 2000.*

[43] Junho Shim, Peter Scheuermann, Radek Vingralek, "Proxy Cache Algorithms: Design, Implementation, and Performance," *IEEE Transactions on Knowledge and Data Engineering, Volume 11, Issue 4, p 549-562, 1999.*

[44] Predrag Jelenkovic, Ana Radovanovic, "Optimizing the LRU Algorithm for Web Caching," *The 18th International Teletraffic Congress, Berlin, 2003.*

[45] Zhan-sheng Li, Da-wei Liu, Hui-juan Bi, "CRFP: A Novel Adaptive Replacement Policy Combined the LRU and LFU Policies," *IEEE 8th International Conference on Computer and Information Technology Workshops, p 72-79, 2008.*

[46] Williams, S., Abrams, M., Standridge, C. R., Abdulla, G., Fox, E. A., "Removal Policies in Network Caches for World-Wide Web Documents," *Proceedings on the Applications, technologies, architectures, and protocols for computer communications, ACM SIGCOMM, 1996.*

[47] Vakali, A, "LRU-based algorithms for Web cache replacement," *International Conference on Electronic Commerce and Web Technologies, 2000.*

[48] Martin Arlitt, Rich Friedrich, Tai Jin, "Performance Evaluation of Web Proxy Cache Replacement Policies," *Performance Evaluation - Special issue on modelling techniques and tools for performance evaluation, Volume 39 Issue 1-4,p 193-206, 2000.*

[49] S. Podlipnig, L. Böszörményi, "A survey of Web cache replacement strategies," *ACM Computing Surveys (CSUR), Volume 35 Issue 4, 2003.*

[50] Roland P. Wooster, Marc Abrams, "Proxy caching that estimates page load delays," *Journal on Computer Networks and ISDN Systems, Volume 29 Issue 8-13, 1997.*

[51] Luigi Rizzo, Lorenzo Vicisano, "Replacement Policies for a Proxy Cache," *IEEE/ACM Transactions on Networking, Volume 8, Issue 2, 2000.*

[52] Pei Cao, Sandy Irani, "Cost-Aware WWW Proxy Caching Algorithms," *Proceedings of the USENIX Symposium on Internet Technology and Systems, 1997.*

[53] Digital Equipment Corporation, "Digital's Web Proxy Traces," Fetched 30 November 2011. [Online]. Available: http://apotheca.hpl.hp.com/ftp/pub/compaq/traces/proxy/webtraces.html

[54] Ludmila Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy," *HP Technical Report, 1998.*

[55] Shudong Jin, Azer Bestavros, "GreedyDual* Web Caching Algorithm – Exploiting the Two Sources of Temporal Locality in Web Request Streams," *Proceedings of the 5th International Web Caching And Content Delivery Workshop, p 174-183, 2000.*

[56] National Laboratory for Applied Network Research, "Web cache traces," Fetched 30 November 2011. [Online]. Available: ftp://ircache.nlanr.net/Traces/

[57] Hyokyung Bahn, Kern Koh, Noh, S.H., Lyul, S.M., "Efficient replacement of nonuniform objects in Web caches," *IEEE Computer Society Computer Journal, Volume 35 Issue 6, p 65-73, 2002.*

[58] The Linux Foundation, "netem," Fetched 08 December 2011. [Online]. Available: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[59] Ariane Keller, "Manual tc Packet Filtering and netem," Fetched 08 December 2011. [Online]. Available: http://tcn.hypert.net/tcmanual.pdf

[60] Linux Manual Pages, "tc(8)," Fetched 09 December 2011. [Online]. Available: http://lartc.org/manpages/tc.txt

[61] "Python Programming Language, version 2.6.5," [Online]. Available: http://docs.python.org/release/2.6.5/

[62] LIVE555, "LIVE555 Media Server, version 0.66," [Online]. Available: http://www.live555.com/mediaServer/

[63] Apache Web Server, "Apache web server, version 2.0.64," [Online]. Available: http://httpd.apache.org/docs/2.0/

[64] Pure-FTPd, "Pure-FTPd server, version 1.0.32," [Online]. Available: http://www.pureftpd.org/project/pure-ftpd

[65] K. Zeilenga, Ed., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map," RFC 4510 (Standard), Internet Engineering Task Force, June 2006. [Online]. Available: http://tools.ietf.org/html/rfc4510

[66] VideoLan Organization, "VLC media player, version 1.1.9," [Online]. Available: http://download.videolan.org/pub/vlc/1.1.9/

[67] VideoLan Organization, "Overview of the VideoLAN streaming solution," [Online]. Available: http://www.videolan.org/vlc/streaming.html

[68] Mplayer, "Mplayer sample files," Fetched 02 December 2011. [Online]. Available: http://samples.mplayerhq.hu/

[69] Audio Samples, "Audio file format specifications and samples," Fetched 02 December 2011. [Online]. Available: http://www-mmsp.ece.mcgill.ca/documents/AudioFormats/index.html

[70] Songqing Chen, Haining Wang, Bo Shen, Susie Wee, Xiaodong Zhang, "Segment-based Proxy Caching for Internet Streaming Media Delivery," *IEEE Journal on Multimedia, p 59-67, 2005.*