# An END-TO-END QoS ARCHITECTURE FOR ALL-IP 4G MOBILE NETWORKS

by

Ali Akkaya

B.S. in Computer Engineering, Bogazici University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2008

# ACKNOWLEDGEMENTS

I would like to acknowledge the enthusiastic supervision of Assist. Prof. Tuna Tuğcu. His guidance, insightful criticisms, and patient encouragement contributed very much for the realization of this thesis.

I am very grateful to my family, especially my mother for the patience, encouragement, and all the support she has given me during this long time of studies.

Finally, I would like to gratefully acknowledge all of my friends and my colleagues for their patience and moral support.

# ABSTRACT

## An END-TO-END QoS ARCHITECTURE FOR ALL-IP 4G MOBILE NETWORKS

Fourth Generation (4G) Networks can be defined as the convergence of wired and wireless network technologies and formation of all-IP based integrated system with premium quality and high security [1]. As an integrated system, providing end-to-end Quality of Service (QoS) similar to legacy networks has been a challenging issue in such a network.

In this thesis, an all-IP next generation wireless network architecture that supports end-to-end QoS is presented. A new packet based Medium Access Control (MAC) layer, which runs on top of Frequency Hopping Orthogonal Frequency Division Multiple Access (FH-OFDMA) based physical layer is designed and implemented. The proposed MAC layer is based on one of the promising proposals to IEEE 802.20 ([2]) by Qualcomm. Multi Protocol Label Switching (MPLS) and DiffServ support of MPLS is utilized at the backbone of the architecture to provide end-to-end QoS. By integrating Internet Protocol (IP) Layer QoS parameters to proposed packet-based air interface, end-to-end QoS including the air interface is provided.

Two new nodes, Wireless Access Router (WAR) and Mobile Node (MN), which implement the new MAC layer, are introduced. WAR is integrated to the backbone via standard Label Edge Router (LER) of MPLS architecture. The architecture is implemented with OPNET Modeler[3] simulation tool to analyze the QoS perception of MNs within the architecture.

# ÖZET

## TAMAMI IP TABANLI 4. NESİL AĞLARDA UÇTAN UCA SERVİS KALİTESİ MİMARİSİ

Dördüncü Nesil Ağlar, yüksek kalitede ve güvenli, tamamı IP tabanlı entegre bir sistem oluşturmak için sabit ve kablosuz ağ teknolojilerinin birlikte kullanılması olarak tanımlanabilir. Entegre sistem olan bu ağlarda, kendisinden önceki sistemlerdekine benzer uçtan uca servis kalitesini sağlamak zorlu bir konu olarak karşımıza çıkmaktadır.

Bu tezde, uçtan uca servis kalitesi sağlayan, tamamı IP tabanlı yeni nesil kablosuz ağ mimarisi sunulmaktadır. 'Frekans Değişimli Ortogonal Frekans Bölümlü Çoklama' tabanlı fiziksel katman üzerinde çalışan paket tabanlı 'Ortam Erişim Kontrol' katmanı dizayn edilmiştir. Önerilen 'Ortam Erişim Kontrol' katmanında , IEEE 802.20 spesifikasyonuna Qualcomm tarafından teklif edilen yapı temel alınmıştır. Uçtan uca servis kalitesini sağlamak için, mimarinin omurgasında 'Çoklu Protokol Etiket Anahtarmala' ve 'Çoklu Protokol Etiket Anahtarmala'nın 'DiffServ' desteği kullanılmıştır. IP katmanının servis kalite parametreleri, önerilen paket tabanlı radyo aryüzüne entegre edilerek, radyo arayüzünü de içeren uçtan uca servis kalitesi sağlanmıştır.

Yeni 'Ortam Erişim Kontrol' katmanını kullanan, 'Kablosuz Erişim Yönlendiricisi' ve 'Mobil Ekipman' olarak adlandırılan iki yeni ekipman tanıtılmıştır. 'Kablosuz Erişim Yönlendiricisi', omurgaya 'Çoklu Protokol Etiket Anahtarmala' mimarisinde bulunan standard 'Etiket Köşe Yönlendiricisi' ile entegre edilmiştir. Mimari, 'OPNET Modeler' similasyon araci ile 'Mobil Ekipman'ların hissettikleri servis kalitesini analiz etmek için simüle edilmiştir.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| 1G | First Generation |
| 2G | Second Generation |
| 3G | Third Generation |
| 4G | Fourth Generation |
| ADSL | Asymmetric Digital Subscriber Lines |
| ATM | Asynchronous Transfer Mode |
| BA | Behavior Aggregate |
| BWA | Broadband Wireless Access |
| CDMA | Code Division Multiple Access |
| DAB | Digital Audio Broadcast |
| DHMM | Dynamic Hierarchical Mobile MPLS |
| DiffServ | Differentiated Services |
| DLAC | DownLink Assignment Channel |
| DSCP | DiffServ Code Point |
| DVB | Digital Video Broadcast |
| FEC | Forward Equivalence Class |
| FFT | Fast Fourier Transform |
| FH-OFDM | Frequency Hopping Orthogonal Frequency Division Multiplexing |
| Flash-OFDM | Fast Low-latency Access with Seamless Handoff OFDM |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile communications |
| HDSL | High-bit-rate Digital Subscriber Lines |
| HIPERLAN | HIgh PErformance Radio Local Area Network |
| HIPERMAN | HIgh PErformance Radio Metropolitan Area Network |
| ITU | International Telecommunications Union |
| LAN | Local Area Network |
| ICI | InterCarrier Interference |
| IETF | Internet Engineering Task Force |

| | |
|---|---|
| IFFT | Inverse Fast Fourier Transform |
| InstServ | Integrated Services |
| IP | Internet Protocol |
| ISI | InterSymbol Interference |
| ITRAS | IP-Triggered Resource Allocation Strategy |
| LDP | Label Distribution Protocol |
| LER | Label Edge Router |
| LSP | Label Switched Path |
| LSR | Label Switch Router |
| MAC | Medium Access Control |
| MAN | Metropolitan Area Network |
| MBWA | Mobile Broadband Wireless Access |
| MIMO-OFDM | Multiple-Input, Multiple-Output OFDM |
| MN | Mobile Node |
| MPLS | Multi Protocol Label Switching |
| OA | Ordered Aggregate |
| OSI | Open System Interconnect |
| PHB | Per Hop Behaviour |
| PSC | PHB Scheduling Class |
| PSK | Phase Shift Keying |
| PVC | Permanent Virtual Connection |
| QAM | Quadrature Amplitude Modulation |
| QoS | Quality of Service |
| RAC | Random Access Channel |
| RFC | Request For Comments |
| SLA | Service Level Agreements |
| TC | Traffic Channels |
| TDMA | Time Division Multiple Access |
| ToS | Type of Service |
| ULAC | UpLink Assignment Channel |
| ULRC | UpLink Request Channel |

| | |
|---|---|
| UMTS | Universal Mobile Telecommunications System |
| VOFDM | Vector OFDM |
| VPN | Virtual Private Network |
| WAR | Wireless Access Router |
| WiMAX | Worldwide Interoperability for Microwave Access |
| WOFDM | Wideband OFDM |

# 1. INTRODUCTION

The first fully digital mobile communication systems have been introduced in early 1990s. After many years of discussions and field trials, Global System for Mobile communications (GSM) has been standardized in 1991. In parallel, Code Division Multiple Access (CDMA) based standards have also been introduced. These first digital mobile communication systems are referred to as Second Generation (2G) systems whereas First Generation (1G) is used for early analog systems. 2G networks utilize the same circuit switched call control mechanisms as wired telephony systems. The required resources are allocated all over the network during call establishment. When data communication need over mobile networks emerged, 2G systems have been augmented with packet switched domain for data communications in addition to the circuit switched domain for voice communications. The packet switched add-on is called 2.5G, not officially but for marketing purposes only. In 2.5G networks, it is possible to configure the ratio of radio resources for packet switched domain, but it does not change dynamically according to user and channel conditions.

Although 2.5G networks enable data communication, the data rates are too low to provide satisfactory services to end users. With this motivation International Telecommunications Union (ITU) introduced a family of standards referred as Third Generation (3G) networks. 3G systems offer users a wider range of advanced services while achieving greater network capacity through improved spectral efficiency. Higher data rates are possible for packet switched connections thanks to this effective use of air interface. In 3G systems, allocation of radio resources to circuit switched and packet switched domains is done dynamically, but the voice is still carried over circuit switched connections, which leads to problems in satisfying QoS needs of medium and large populations.

There is no current standard or formal definition of 4G (also known as Beyond-3G) Communication Systems, but the term is used to describe next step in wireless communication. The expectation is the convergence of wired and wireless technologies

and formation of all-IP based integrated system with premium quality and high security [1]. It is important for 4G systems to offer all types of services at an affordable cost. Some key features of 4G networks can be stated as follows [4, 5]:

- *High usability: anytime, anywhere, and with any technology:* The users of 4G networks are able to access all services anytime and anywhere, independent of the underlying technology used to access the service. Integrated terminals are able to access different services through different wireless networks.
- *Support for multimedia services at low transmission cost:* 4G systems do not concentrate only on voice communication, but data and multimedia services are also provided with affordable costs. To provide multimedia services, reliable and high data rate services are supported.
- *Personalization:* Since all services are accessible by all users anytime and anywhere, personalization is important to meet user demands. It should be possible for end users to customize the offered services.
- *Integrated services:* Users can use multiple services from different service providers at the same time.

IEEE 802.20 [2], Mobile Broadband Wireless Access (MBWA) Working Group, has been established with the mission of developing the specification for an efficient packet based air interface that is optimized for the transport of IP based services. In all-IP 4G networks, IP packets are expected to traverse an access network and backbone network without any protocol conversion ([6]), hence IEEE 802.20 is a good candidate as air interface of converged all-IP 4G networks. The scope of IEEE 802.20 working group is specification of physical and medium access control layers of an air interface for interoperable mobile broadband wireless access systems that;

- operate in licensed bands below 3.5 GHz
- is optimized for IP-data transport, with peak data rates per user in excess of 1 Mbps.
- support various vehicular mobility classes up to 250 Km/h in a MAN environment
- target spectral efficiencies and sustained user data rates and numbers of active

users that are all significantly higher than achieved by existing mobile systems.

Among the proposals to IEEE 802.20, Flash-OFDM (Fast Low-latency Access with Seamless Handoff OFDM) ([7]), which is an air interface technology designed for the delivery of advanced Internet services in the mobile environment, is promising. Flash-OFDM is a system based on OFDM, and also specifies higher protocol layers. It extends packet switched domain to air interface to fully implement the all-IP idea.

IEEE 802.20 targets to standardize mobile broadband wireless access focusing on IP based services, which is only one aspect of 4G Networks. 4G Networks combine several fix and mobile networks to provide service to end users independent of the underlying technology. IEEE 802.20 specification complements rather than competes with 3G standards, since 4G is the convergence of these standards over IP based core networks and technologies. As a wireless access standard, integration architectures to other networks is not in the scope of IEEE 802.20. 4G networks integrate several access and core networks and define architectures for enabling technologies like mobility and QoS to provide access network independent service to end users, which is still an important research area.

In this thesis, an end-to-end QoS architecture for all-IP 4G mobile networks is proposed. The architecture includes a new Medium Access Control (MAC) sublayer for Frequency Hopping Orthogonal Frequency Division Multiplexing (FH-OFDM) air interface utilizing Multi Protocol Label Switching (MPLS) based core network, and integrated QoS for air interface and core network based on Differentiated Services (DiffServ). The architecture aims to address QoS needs for a converged network containing wireless access network and IP based core network. It defines mechanisms to ensure strict SLAs with strict QoS requirements.

The first contribution of the thesis is the proposed packet based MAC layer, which is based on one of the promising proposals to IEEE 802.20 ([2]) by Qualcomm. The MAC layer includes QoS mechanisms that fit in the overall architecture. The other contribution is the integrated QoS architecture that enables end-to-end QoS

including the air interface. An IP packet traversing the network from one node to the other is exposed to same treatment both in core network and access network. The architecture is dependent on the QoS mechanisms implemented in access network, but can be adapted to access networks having similar QoS mechanisms.

The rest of the thesis is organized as follows: Section 2 explains basic concepts used during the thesis like OFDM and MPLS, and the results of the literature survey are provided stating how our architecture differs from the other studies. Section 3 describes the proposed architecture. In Section 4, we provide the simulation results based on the implementation of the architecture with OPNET Modeler ([3]) simulation tool, and in Section 5 brief concluding remarks are summarized and future research directions are presented.

# 2. BACKGROUND INFORMATION

## 2.1. Orthogonal Frequency Division Multiplexing

### 2.1.1. OFDM History

OFDM is a multicarrier transmission technology, where a datastream is transmitted over multiple lower rate subcarriers. The reason OFDM draws great attention is the robustness against frequency selective fading and narrowband interference. In single carrier systems, a single interferer may fail the entire link, but in multicarrier systems only small portion of the subcarriers is affected, which can be recovered by error correction mechanisms. The concept of OFDM has been introduced around 1960s, and a U.S. patent has been issued in 1970 [8]. In the 1960s, OFDM technique has been used in several military systems; in the 1980s it has been studied for high speed modems, digital mobile communications, and high density recording; and in the 1990s it was used for High-bit-rate Digital Subscriber Lines (HDSL), Asymmetric Digital Subscriber Lines (ADSL) as well as for Digital Audio Broadcasting (DAB) and Digital Video Broadcasting (DVB) services [9]. There are several standardization processes completed and on going under different working groups in which OFDM plays vital role as enabling technology for air interface. These working groups include Wireless LAN radio interfaces IEEE 802.11a, IEEE 802.11g [10] and HIPERLAN/2 [11], Wireless MAN / Fixed Broadband Wireless Access (BWA) standards IEEE 802.16 (or WiMAX) [12] and HIPERMAN [13], and Mobile Broadband Wireless Access (MBWA) standards IEEE 802.20 [2] and IEEE 802.16e (Mobile WiMAX) [12].

### 2.1.2. OFDM Basics

The main idea behind OFDM is to split a radio signal into multiple smaller signal sets and modulate each onto a different lower rate subcarrier. In a classical parallel data system, total frequency band is divided into non-overlapping subchannels, which are modulated separately, then multiplexed. The channels do not overlap, hence inter-

channel interference is eliminated. However, this results in inefficient use of available spectrum. In OFDM, subcarriers do overlap to increase the spectral efficiency, and orthogonality between subcarriers ensures that crosstalk between subcarriers is reduced. Figure 2.1 [9] illustrates the difference between non-overlapping and overlapping multicarrier modulation techniques. As seen, overlapping technique has great spectral efficiency compared to non-overlapping technique.

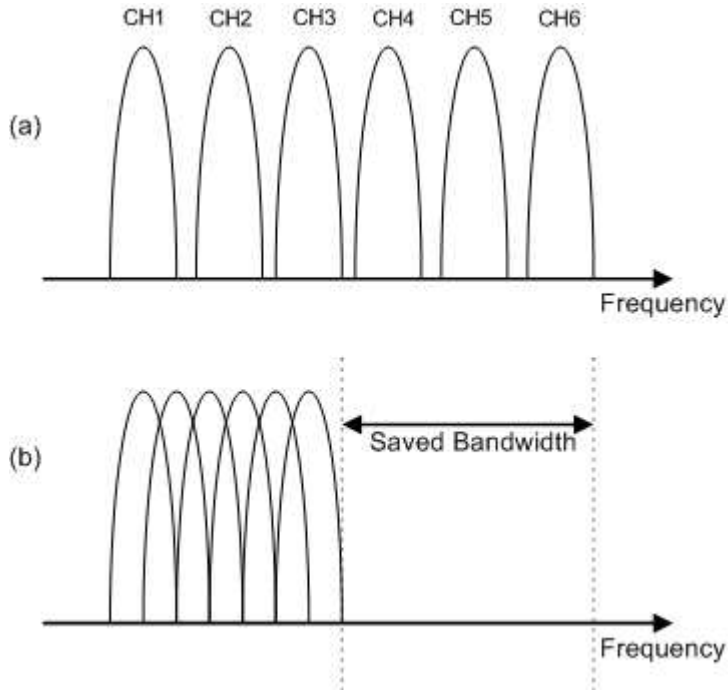


Figure 2.1. Concept of OFDM Signal: (a) Conventional multicarrier technique, and (b) orthogonal multicarrier modulation technique [9].

The subcarrier frequencies are chosen orthogonal to allow the subcarriers' spectra to overlap, thus increasing spectral efficiency. The guard bands, which are used in FDM systems to prevent interference, are no longer required since orthogonal subcarriers do overlap but they do not interfere with each other. The frequency domain representation in Figure 2.2 shows orthogonal nature of subcarriers used in OFDM systems. When a subcarrier is at peak, all the neighboring spectra are zero, so that there is no intercarrier interference (ICI); this is due to the orthogonality principle. For the subcarriers to be orthogonal, each subcarrier should have exactly an integer number of cycles in the symbol duration, and the number of cycles between adjacent subcarriers differ by exactly one.

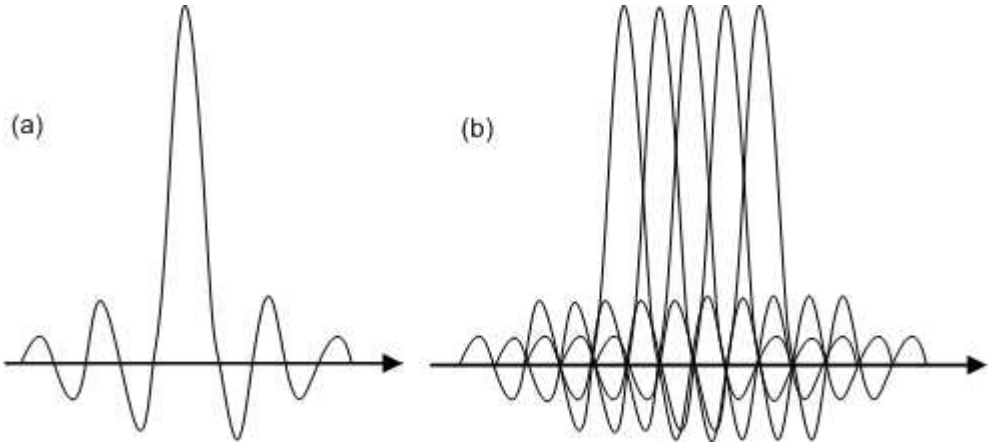


Figure 2.2. OFDM Spectrum [9].

In OFDM, the symbol duration increases for lower rate parallel subcarriers, so the amount of dispersion caused by multipath delay spread is reduced. Intersymbol interference (ISI) is eliminated almost completely by introducing a guard time for every OFDM symbol. To maintain orthogonality in case of multipath delay, OFDM symbol is cyclically extended in the guard time to avoid ICI. It is possible to transmit high volumes of data via OFDM by using various frequency modulation methods like Phase Shift Keying (PSK), or Quadrature Amplitude Modulation (QAM). For instance, IEEE 802.11a uses the binary phase shift keying approach to achieve 6- to 9-Mbps data rates; quadrature phase shift keying for 12- to 18-Mbps rates; and quadrature amplitude modulation for 24- to 54-Mbps rates. Higher data rates are more susceptible to interference and require more digital signal processing power; therefore, thay are not suitable for all situations [14].

OFDM has the following advantages [9]:

- OFDM minimizes multipath effect.
- OFDM is more resilient to narrowband interference since fewer subcarriers are affected.
- It is possible to adapt the data rate per subcarrier according to the subcarrier characteristics.

- It is possible to create single-frequency networks, avoiding frequency planning.

On the other hand, OFDM has some disadvantages:,

- Sensitivity to frequency offset and phase noise.
- Large peak-to-average power ratio.

OFDM based wireless systems play an important role in next generation wireless communication [15]. The robustness against frequency selective fading is very important for high-speed data transmission. After its use in high-rate WLAN and DVB systems, the know how of OFDM systems has increased through several research and development activities. Combination of OFDM and CDMA results in systems that are robust against frequency selective fading and provide high scalability in adapting different data transmission rates.

### 2.1.3. Implementation of OFDM

OFDM systems are implemented using a combination of Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) blocks. These mathematical operations are widely used for transforming data from time domain to frequency domain, and vice versa. These transforms are used to map data onto orthogonal subcarriers.

An OFDM system treats the source symbols at the transmitter as though they are in the frequency-domain. To bring these symbols to time domain, they are fed to IFFT block which takes in N symbols with period of T seconds, where N is the number of subcarriers in the system. The output of IFFT is the summation of all N sinusoids. Thus, the IFFT block provides a simple way to modulate data onto N orthogonal subcarriers. The block of N output samples from IFFT make up a single OFDM symbol. The length of the OFDM symbol is N.T, where T is the IFFT input symbol period mentioned above. The time domain signal that results from IFFT is transmitted across the channel. At the receiver, an FFT block is used to process the received signal and bring it into the frequency domain [16]. The steps of OFDM

implementation are defined in Figure 2.3.

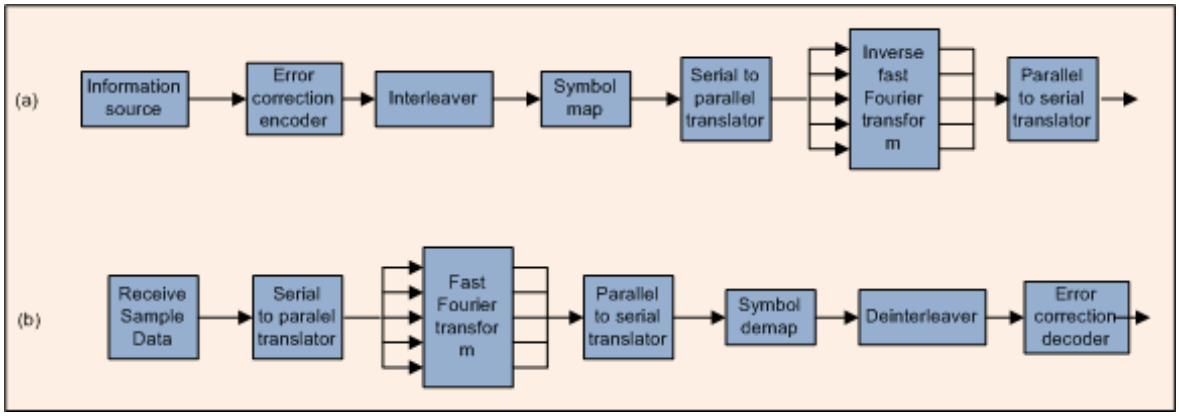


Figure 2.3. OFDM Implementation [16].

### 2.1.4. Guard Time and Cyclic Extensions

One of the reasons why OFDM is used is the efficient way it deals with multipath delay spread. OFDM divides the datastream into $N$ subcarriers; hence the transmission rate is reduced $N$ times. For instance, to transmit at 1 Mbps, a system should send one bit in one microsecond. If the transmission is delayed by multipath effect, it would overlap with the next signal, causing interference. However, if the system transmits 1,000 bits over 1,000 subchannels, each bit can be transmitted per millisecond still reaching 1 Mbps on aggregate. In this case, if delay of slightly more than microsecond occurs it will overlap with one thousandth of the next bit's transmission period, hence it will create almost no interference.

To eliminate intersymbol interference almost completely, a guard time is introduced for each OFDM symbol. The guard time is chosen to be larger than the expected delay, so that multipath components of one symbol cannot interfere with the next symbol. It is possible to send no signal at guard time, but this would cause ICI since the subcarriers would not be orthogonal. This effect is shown in Figure 2.4, where a subcarrier 1 and a delayed subcarrier 2 are illustrated. When the receiver tries to demodulate the first signal, it will encounter ICI, because during the FFT interval, there is no integer number of cycles difference between subcarriers 1 and 2. Due to the same reason, there is crosstalk from first to second subcarrier. To eliminate ICI, the OFDM symbol

is cyclically extended in the guard time. This cyclic extension ensures that there is integer number of cycles within FFT intervals, hence multipath signals with delays smaller than guard time do not cause ICI.

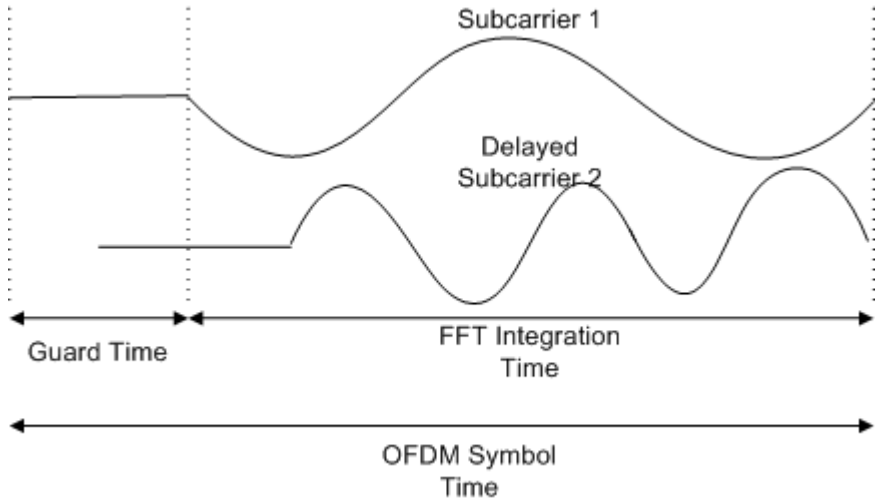


Figure 2.4. Cyclic Extension [9].

### 2.1.5. OFDM Variations

Various types of OFDM are briefly explained below [14]:

- **Vector OFDM**

  VOFDM is developed by Broadcom and Cisco Systems. This system uses multipath signal reflections to increase bandwidth and range. VOFDM is most often used in fixed-wireless metropolitan area networks (MANs).
- **Wideband OFDM**

  In WOFDM, additional frequency space between the orthogonal channels is introduced to reduce interference and permit tolerance for problems like jitter. It is used by wireless internet service providers in MANs. OFDM Forum promotes WOFDM as the version to be accepted as the standard version.
- **Flash-OFDM**

  Flash-OFDM is created by Flarion [7]. It utilizes fast frequency hopping spread spectrum technology, which repeatedly switches frequencies during a radio transmission. This system transmits a signal across a much wider frequency band than

is required.

- **MIMO-OFDM**

  Multiple-input, multiple-output OFDM was developed by Iospan Wireless. The idea behind MIMO-OFDM is to use OFDM to break up a signal and transmit the pieces simultaneously via multiple antennas. The receiver reassembles the transmitted pieces.

### 2.1.6. Frequency Hopping OFDM

OFDM can be combined with frequency hopping as a new spread spectrum technology, realizing the benefits of frequency diversity and interference averaging. The idea is to change the user's set of subchannels after each time period to minimize the losses due to frequency selective fading. The air link resource in FH-OFDM can be viewed in a time-frequency plane, as illustrated in Figure 2.5, where a small box represents a particular tone in a given OFDM symbol. Thus, each column represents all the tones in a given OFDM symbol, and each row represents a given tone over successive OFDM symbols. Frequency hopping is specified with frequency hopping sequences, each of which indicates the corresponding frequency of the hopping sequence at a given OFDM symbol [17].

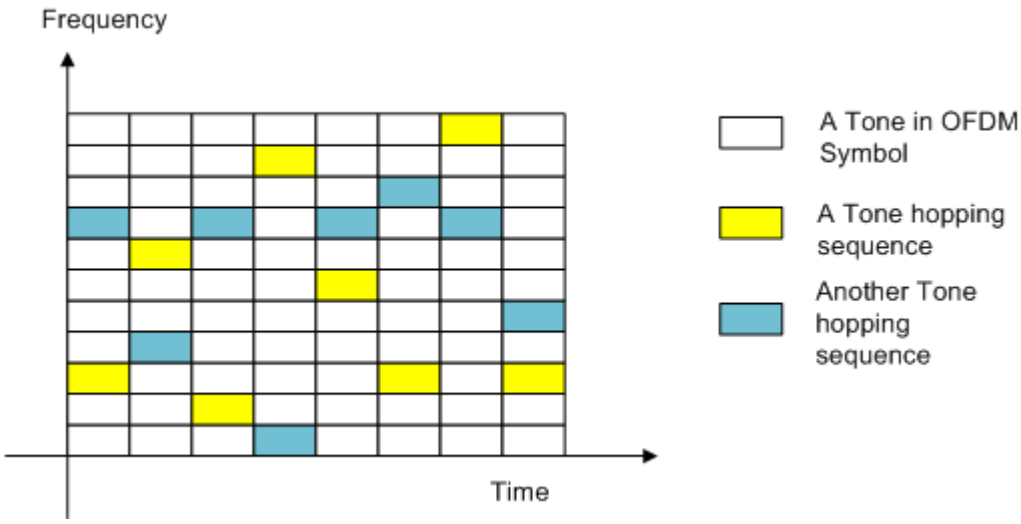


Figure 2.5. Frequency Hopping OFDM

Frequency Hopping OFDM combines the advantages of Time Division Multiple

Access (TDMA) and CDMA systems. As in TDMA, users are orthogonal to one another and it is also a spread spectrum system as CDMA. It also avoids the drawbacks of each, including the need for TDMA frequency planning and equalization, and multiple access interference. The advantages of TDMA/FDMA, CDMA, and Frequency Hopping OFDM are summerized in Figure 2.6.

| | TDMA/FDMA | CDMA | Frequency Hopping OFDM |
|---|---|---|---|
| In-cell interference | Users are orthogonal | Users non orthogonal | Users are orthogonal |
| Out-of-cell interference | Must design for worst case - no frequency reuse | Users see average interference | Users see average interference |

Figure 2.6. FH-OFDM Advantages

## 2.2. Multi Protocol Label Switching

### 2.2.1. Introduction

MPLS is a packet forwarding mechanism for packet switched networks. MPLS operates between Layer 2 and Layer 3 defined in the Open System Interconnect (OSI) Model. The "'Multi Protocol"' term in the name comes from the applicability to any Layer 3 protocol, although currently nearly all usage is with Internet Protocol. MPLS was originally developed to improve packet forwarding speed, but after hardware implementations of packet forwarding algorithms, MPLS offers little or no improvement in this area. However, flexibility of MPLS enabled new application areas to archive QoS and next generation Virtual Private Network (VPN) services, which are difficult to implement or operationally impossible with traditional IP networks.

The idea of *IP Switching* is first proposed by a group of engineers from a company named Ipsilon Networks [18]. The proposed model is defined to work only over Asynchronous Transfer Mode (ATM), hence did not achieve market dominance. Cisco Systems has come up with *Tag Switching* concept not restricted to ATM, which was

renamed to *Label Switching* when it was standardized by Internet Engineering Task Force (IETF) [19].

### 2.2.2. MPLS Terminology

The following terms are important to understand MPLS networks. These terms help to familiarize with the fundamental concepts of MPLS based forwarding.

- **Label**: A fixed length tag that is used for MPLS forwarding. The generic label format is shown in Figure 2.7. It is possible to embed the label in the header of data link layer or in the shim. It contains 32 bits. First 20 bits are the label value and the additional 12 bits are made up of the following:

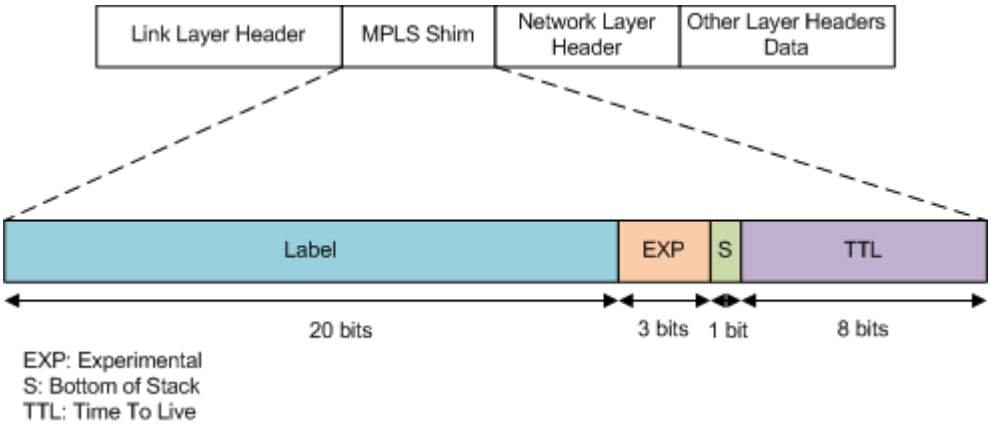


Figure 2.7. MPLS Shim

  - *EXP:* Reserved for experimental use, but nearly all MPLS implementations use these EXP bits to hold a QoS indicator.
  - *S:* It is the bottom of stack bit. If more than one label is attached to a packet, this bit marks the label at the bottom.
  - *TTL:* Time To Live bits are generally direct copy of the IP TTL header. As in normal IP operation, it is decremented at every hop to prevent creation of routing loops.
- **Forward Equivalence Class (FEC)**: A group of packets that require same handling by MPLS network. Unlike the conventional IP forwarding, the assignment of a packet to a FEC is done only once in MPLS networks.

- **Label Binding**: An association of an FEC to a label.
- **Label Stack**: Multiple labels in the same packet to support hierarchical routing
- **Label Switch Router (LSR)**: Any device that can route a packet based on MPLS label.
- **Label Edge Router (LER)**: An LSR that can add label to a packet at the ingress side and remove the label at the egress side.
- **Label Switch Path (LSP)**: The path that a labeled packet follows throughout the MPLS network. The LSP is set up prior to data transmission, and is unidirectional. For the return traffic, another LSP is created.
- **Label Distribution Protocol (LDP)**: The protocol used to distribute the labels between LSRs.

### 2.2.3. MPLS Operation

In MPLS, the assignment of a packet to a FEC is done once when the packet enters the MPLS network. The FEC that the packet belongs to is encoded to construct the label. This assignment is done by LER when the packet enters the MPLS network. At subsequent hops, Layer 3 header is not analyzed for routing; instead MPLS label is used as an index to a table that specifies the next hop and the label. The label is replaced and the packet is sent to next hop.

In an MPLS network, before any traffic starts, labels should be created and distributed. To do this, LDP is used. LERs initiates the distribution of labels and FEC bindings over LDP. Upon receiving the bindings, routers store them at an information base, and update if new bindings are received. Once the distribution is done, LSPs are created. The ingress LER uses its information base to find the next hop and request the label for the specific FEC. The LSRs on the LSP use only the label to find the next hop. Once the packet receives the egress LER, the label is removed and the packet is sent to destination.

Figure 2.8 and Figure 2.9 illustrate an example of label distribution and packet forwarding in an MPLS network. First label distribution is shown in Figure 2.8, where

LER2 sends Label *3* to be used for *192.168* Network and LER3 sends Label *8* to be used for *10.10* Network to LSR1. LSR1 stores this information and sends Label *2* for *192.168* Network and Label *6* for *10.10* Network to LER1. LER1 also stores this mapping.

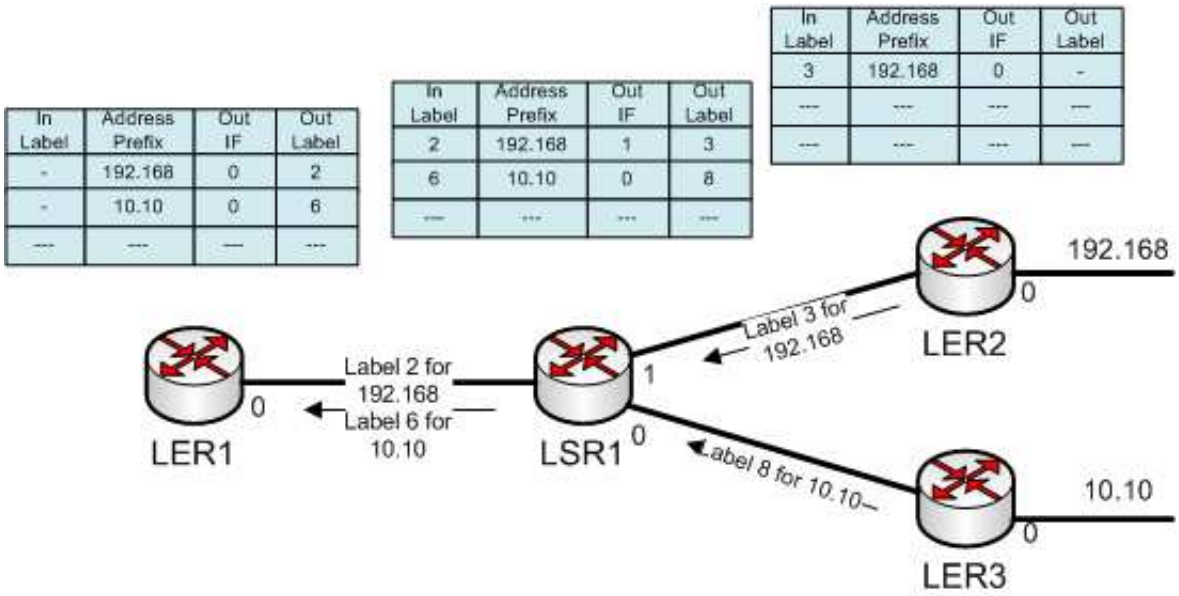


Figure 2.8. MPLS Label Distribution

In Figure 2.9, a packet to *192.168.1.1* arrives at LER1. LER1 assigns Label *2* and sends the packet to LSR1. LSR1 checks the table, replaces the label with Label *3* and sends the packet over interface *1* to LER2. LER2 removes the label and forwards the packet.

### 2.2.4. Advantages and Application Areas

RFC 3031 MPLS Architecture [19] states a number of advantages of MPLS over conventional network layer forwarding:

- MPLS forwarding can be done by switches, which are capable of doing label lookup and replacement, but are either not capable of analyzing the network layer headers, or are not capable of analyzing the network layer headers at adequate speed.
- Since a packet is assigned to a FEC when it enters the network, the ingress router

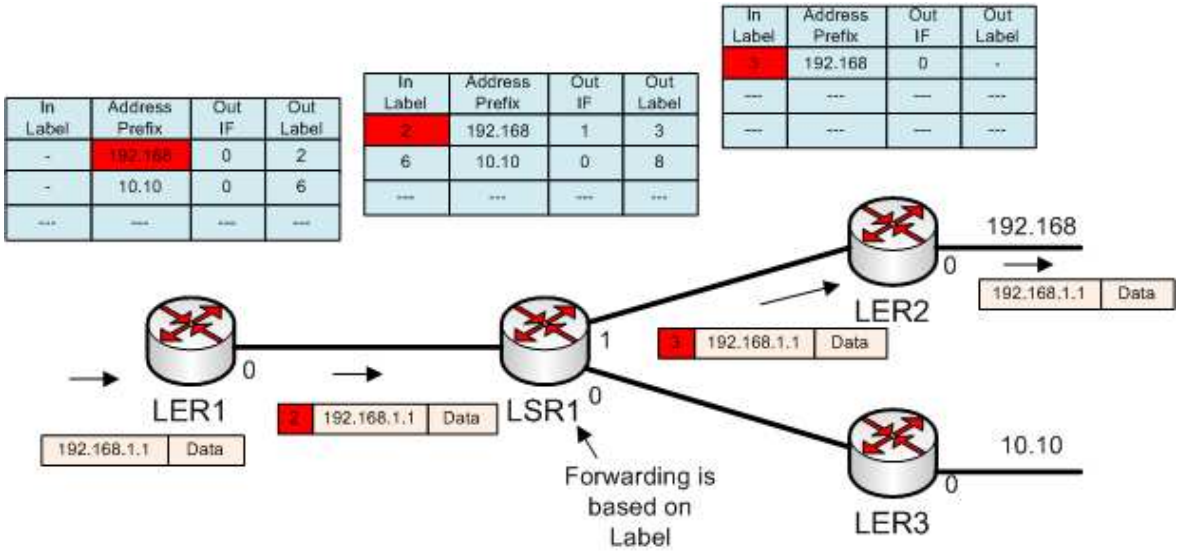


Figure 2.9. MPLS Packet Forwarding

may use, in determining the assignment, any information it has about the packet, even if that information cannot be gleaned from the network layer header. For example, packets arriving on different ports may be assigned to different FECs. Conventional forwarding, on the other hand, can only consider information which travels with the packet in the packet header.

- A packet that enters the network at a particular router can be labeled differently than the same packet entering the network at a different router, and as a result forwarding decisions that depend on the ingress router can be easily made. This cannot be done with conventional forwarding, since the identity of a packet's ingress router does not travel with the packet.
- The considerations that determine how a packet is assigned to a FEC can become ever more and more complicated, without any impact at all on the routers that merely forward labeled packets.
- Sometimes it is desirable to force a packet to follow a particular route which is explicitly chosen at or before the time the packet enters the network, rather than being chosen by the normal dynamic routing algorithm as the packet travels through the network. This may be done as a matter of policy, or to support traffic engineering. In conventional forwarding, this requires the packet to carry an encoding of its route along with it ("source routing"). In MPLS, a label can

be used to represent the route, so that the identity of the explicit route need not be carried with the packet.

The advantages listed above have led new application areas of MPLS other than the initial motivation of improved packet forwarding speed. These application areas can be listed as:

- Quality of Service
- Traffic Engineering
- Virtual Private Networks

Quality of Service and Traffic Engineering are explained in the following sections, since they are within the scope of this thesis.

### 2.2.5. DiffServ Support of MPLS

In Differentiated Services architecture [20] all IP packets that require same DiffServ treatment are called a Behavior Aggregate (BA). At the ingress node, packets are classified and marked with a DiffServ Code Point (DSCP) corresponding to their BA, and DiffServ defines how to use DSCP bits and mechanisms to provide different quality of services in a network. It is a class based mechanism for traffic management. DiffServ does not define what type of traffic should be given higher priority, it only defines a framework to make classification possible. At each node, DSCP is used to select Per Hop Behavior (PHB) that determines the scheduling treatment and drop probability.

MPLS Support of DiffServ [21] defines the architecture to support DiffServ over an MPLS network. It includes mechanisms such that MPLS network nodes are aware of PHB and act accordingly. The solution uses two types of LSPs:

- **EXP-Inferred-PSC LSPs (E-LSP)**

  In E-LSP, a single LSP can support one or more Ordered Aggregates (OA), set

of BAs which share an ordering constraint. For a given FEC, E-LSP can support up to eight BAs. The EXP field of MPLS header is used to determine the PHB of the packet. It is called as "EXP-inferred-PSC LSPs" (E-LSP), since the PHB Scheduling Class (PSC) of the packet depends on the EXP field. The mapping from EXP field to PHB is explicitly signaled at label setup or can be configured prior to label setup. Figure 2.10 shows the mapping between IP headers with DiffServ and MPLS shim headers for E-LSP. Five Tuple in the figure includes source and destination IP addresses, source and destination protocol ports, and a protocol that can be used to determine the FEC of packet in IP header.

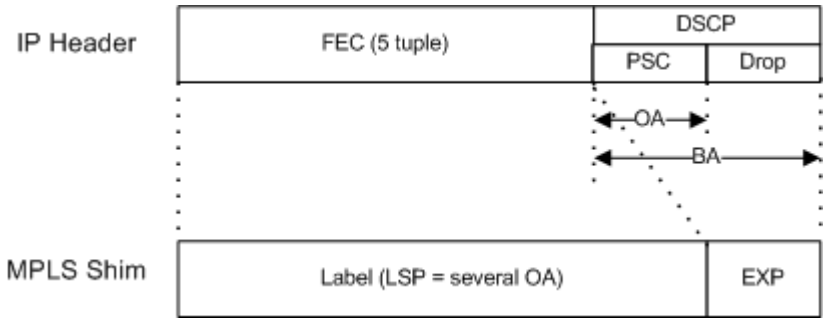


Figure 2.10. IP and MPLS headers Mapping for E-LSP [22].

- **Label-Only-Inferred-PSC LSPs (L-LSP)**

  It is also possible to establish a LSP for each FEC, OA pair. In that case the PSC is signaled at label establishment. LSR can then determine the PHB according to the label of the packet. Drop precedence is determined from EXP bits. It is called as "Label-Only-Inferred-PSC LSPs" (L-LSP) since PSC value can be determined only from label without other fields. Figure 2.11 shows the mapping between IP headers with DiffServ and MPLS shim headers for L-LSP. Five Tuple in the figure includes source and destination IP addresses, source and destination protocol ports, and a protocol that can be used to determine the FEC of packet in IP header.

### 2.2.6. Traffic Engineering with MPLS

When network growth and expansion is considered, two kind of engineering is utilized, network engineering and traffic engineering. Network engineering is deals with

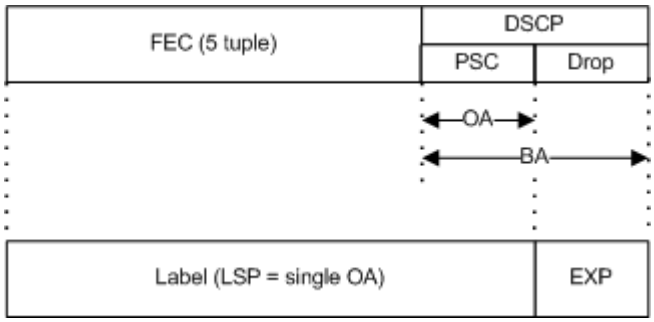


Figure 2.11. IP and MPLS headers Mapping for L-LSP [22].

manipulating your network to suit the traffic. According to the predictions, network is designed and constructed with appropriate circuits and networking devices. It is generally a long term planning activity. Traffic engineering is manipulating the traffic to fit the network. It is the activity of moving traffic around so that congested traffic is moved to unused links.

One of the areas that MPLS is used is traffic engineering. MPLS allows data streams from a particular ingress LER to an egress LER to be identified, which provides mechanism to control traffic flow from one end to other. Suppose in Figure 2.12 there are two traffics of 50 Mbps, one from Node A to Node C, and one from Node B to Node D and bandwidth of each link is 75 Mbps. With normal destination based forwarding, all traffic would be routed over path R1-R2-R3, which is the shortest path shown as Path 1. Even if that link is congested and R1-R3-R4-R5 link is idle, congested traffic would be dropped. With MPLS in place different LSPs can be defined for Path 1 and Path 2. Traffic from node A to Node C can be mapped to Path 1 and traffic from node B to node D can be mapped to Path 2, which results in better utilization of system resources.

## 2.3. Mobile IP

The Internet Protocol is originally designed for fixed networks without any consideration of mobility. Mobile IP [23, 24] is an extension to IP that is designed to allow mobile nodes to roam while maintaining a permanent IP address. To enable mobility, Mobile IP allows mobile nodes to have two IP address, permanent address that is as-

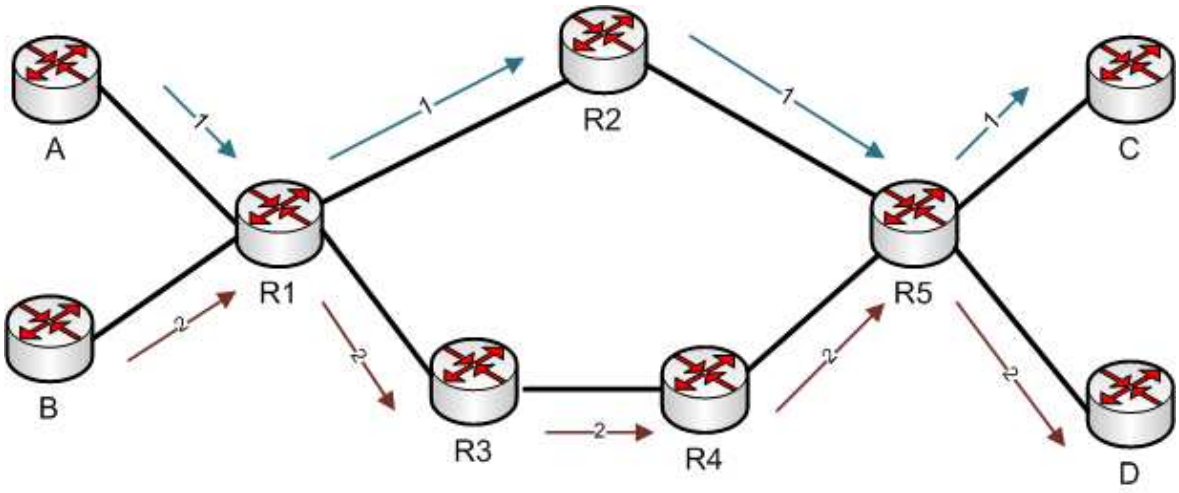


Figure 2.12. MPLS Traffic Engineering

signed at home network and temporary care-of address that shows the current location of mobile node. The binding between these two addresses are transparently maintained by specialized routers called mobility agents. There are two types of mobility agents;

- **Home Agent (HA)** Home Agent maintains the mobility binding of the mobile node and tunnels datagrams for delivery to the mobile node when mobile node is not at home network.
- **Foreign Agent (FA)** Foreign Agent maintains information about mobile nodes visiting its network. maintains the mobility binding of the mobile node and tunnels datagrams for delivery to the mobile node when mobile node is not at home network.

Mobile node is required to register its current location to Home Agent. Home Agent then keeps a record for each Mobile Node that it is supporting. When mobile node is not at home network, Home Agent redirects all traffic to the mobile node to its registered location via tunneling.

Figure 2.13 shows a typical Mobile IP operation. When mobile node is at home network, it does not use Mobile IP features. Suppose mobile node and correspondent node starts communicating when mobile node is at home network. If mobile node

changes location and travels to Subnet A, it registers it location as Subnet A to HA. Afterwords, HA redirects all traffic coming to the mobile node to FA of Subnet A via packet encapsulation. If mobile node again changes location and travels to Subnet B, it again registers it location to HA, which then redirects all traffic to FA of Subnet B.

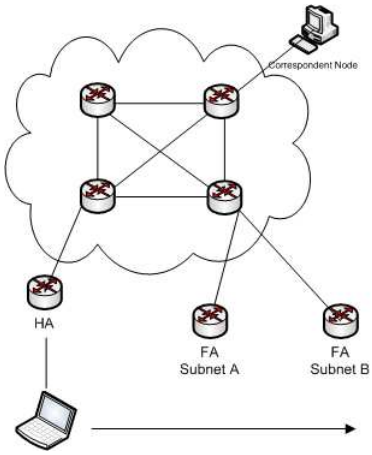


Figure 2.13. Mobile IP

## 2.4. Literature Review

Mobile communication has always been a hot research topic. Until the realization of 3G networks, several standardization work and numerous research studies has been done. Now, as the next generation network, 4G gains important attention with ongoing standardization work and research projects. In this section we will summarize previous studies that are related to our work, and state how our architecture differs from the other studies.

IEEE 802.20 [2], Mobile Broadband Wireless Access (MBWA) Working Group, is established on 11 December 2002 with the mission of developing the specification for an efficient packet based air interface that is optimized for the transport of IP based services. The goal is to enable worldwide deployment of affordable, ubiquitous, always-on and inter operable multi-vendor mobile broadband wireless access networks that meet the needs of business and residential end user markets. The standardization process is still ongoing, with several air interface proposals by different vendors. Among these proposals the one by Qualcomm and Kyocera is of importance regarding the thesis.

Qualcomm has involved in the technology with the acquisition of Flarion Technologies, creator of Flash-OFDM [7]. Flash-OFDM , is an air interface technology designed for the delivery of advanced Internet services in the mobile environment. As its name suggests, the technology is based on the OFDM air link, a wireless access method that combines the attributes of its two predecessors TDMA and CDMA to address the unique demands posed by mobile users of broadband data and packetized voice applications. As a commercial technology, the implementation details of Flash-OFDM is not public but the MAC layer in this thesis is in line with available information about physical layer, extended with QoS mechanisms that fit in our overall architecture.

OFDM based air interfaces are expected to have vital role in 4G networks. In [14], OFDM is stated to be well suited for mobility applications in cellular networks, and explains various types of OFDM. [25] mentions that orthogonal frequency division multiplexing (OFDM) technology in 4G wireless systems may avoid inter carrier interference and increase system resistance to channel impairments. The importance of OFDM based air interface design is also mentioned in [26, 27].

Use of MPLS to cope with shortcomings of traditional IP-based networks is also another research topic. Besides the fix network studies, there are studies related to MPLS use in wireless networks. An MPLS based transport architecture for 3G radio access networks has been proposed in [28]. In [29], implementation of MPLS based approach for 3G core networks is presented. It is argued that the approach presented can reduce the transmission overhead, improve the forwarding efficiency, load balancing, service resilience, and quality of service. There are also studies to issue mobility management in next generation networks which use MPLS based approaches. An integration framework between UMTS-GPRS system and IEEE 802.11 WLAN systems utilizing MPLS technology is presented and benefits of using MPLS in handover and mobility operations are discussed in [30]. In [31], a new scheme for micro-mobility management in 3G all-IP networks is presented. The proposed scheme, based on MPLS and Mobile IP, is shown to reduce handoff delay and packet delay loss during handoff. In [32], another Mobile MPLS protocol named dynamic hierarchical mobile MPLS (DHMM) is proposed for next generation all-IP wireless networks. The protocol

deals with scalability problem of Mobile IP on micro mobility, and QoS provisioning of multimedia applications of mobile users. Since we do not focus on mobility in our architecture and use standard MPLS nodes and concepts, the concepts introduced in these works on MPLS mobility can be integrated in our architecture to provide mobility management.

In [33], an end-to-end QoS architecture with MPLS-based core is described. The architecture focuses on providing end-to-end QoS between two customer LANs connected over MPLS based core network. To provide QoS in core network, MPLS support of DiffServ is used. For LAN to core QoS interworking, it describes two approaches. The one defined by IETF [21] is for interworking models between purely DiffServ based networks. DSCP value is carried over MPLS network, and at the other end same DSCP value is used again. The one defined by MPLS Forum [34] is for interworking between network that are not purely DiffServ based. It uses MPLS Permanent Virtual Connection (PVC) UNI, which defines various QoS attributes of the traffic, such as traffic type, QoS markings that define packet assignment to a specific queue, bandwidth, delay and jitter related reservations. The architecture proposed in this thesis introduces a MAC layer for air interface with DiffServ support. The approached defined by IETF mentioned above is used for interworking between MPLS based core network and access network.

In [17], a wireless system that enables broadband mobile access is proposed. It uses OFDM-based air interface, and cross layer optimization of physical, MAC and link layers. The air interface part of our work is based on these concepts, but concentrating on QoS support of air interface and interworking of access network and core network regarding QoS.

In [6], a model of hierarchical cellular system that combines OFDMA and FH-OFDMA is presented. In the model, there are microcells that use OFDMA and over these microcells there are macrocells that use FH-OFDMA. Low mobility users use microcells that has fine granularity, and high mobility users use macrocells that is robust to channel fading and interference. Mobile Terminal switches cells according to its

traffic type and mobility. They also propose a QoS architecture called IP-triggered resource allocation strategy (ITRAS), which uses layer three QoS mechanisms Integrated Services (IntServ) and DiffServ on wireless channel allocation. To involve IntServ in radio resource management, dedicated channel allocation is used. For DiffServ shared channels are used. Regarding the QoS support, we follow a similar approach to include layer three QoS mechanisms on radio resource allocation, but our resource allocation scheme is also packet based, which is integrated to the QoS mechanisms at core network to provide end-to-end QoS.

# 3. END-TO-END QoS ARCHITECTURE

To address the end-to-end QoS issues in all-IP networks, an architecture is presented. The architecture consists of two networks, an access network for air interface and core network for backbone connectivity. The core network utilizes MPLS technology with LER and LSR nodes. The access network utilizes the new air interface described in Section 3.1. Two new nodes that implement the air interface are introduced, Mobile Node (MN) and Wireless Access Router (WAR). To provide end-to-end QoS, DiffServ support for MPLS is utilized at core network, which is integrated with the DiffServ support in the new air interface as described in Section 3.3. The architecture is depicted in Figure 3.1. In this architecture Wireless Access Router (WAR) plays an important role. It routes the IP traffic received from the network to the appropriate device over the air interface, and vice versa. Since WAR has complete knowledge of both IP traffic and wireless channel environment, it can optimize the air interface usage considering the QoS requirements. In this architecture voice is carried over packet switched network just like other services; it is considered as a data service provided by the network.

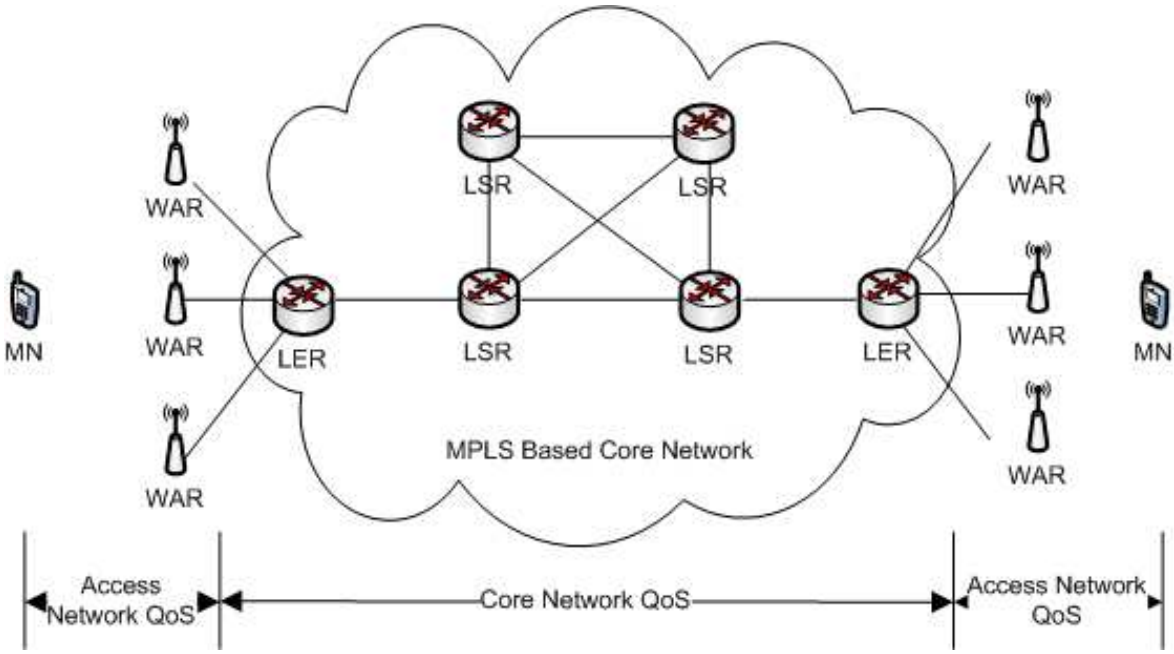


Figure 3.1. Architecture

## 3.1. Air Interface

The air interface described in this section consists of the physical layer and MAC layer as shown in Figure 3.1. Over the MAC layer, Internet Protocol runs. Hence, WAR can route IP packets directly to the core network without further protocol conversions.

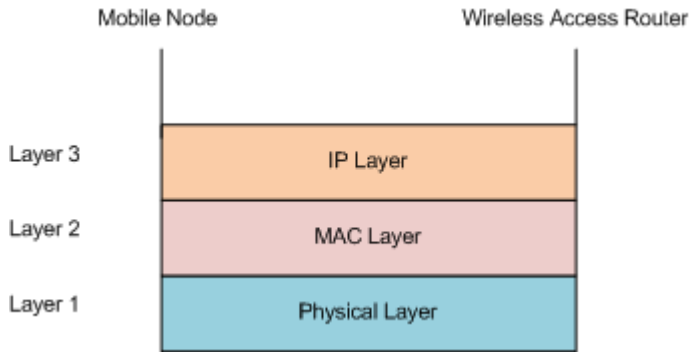


Figure 3.2. Air Interface

### 3.1.1. Physical Layer

FH-OFDM was described in Section 2.1.6. By assigning different time slots of each logical channel in FH-OFDM to different MNs, FH-OFDM can be effectively used as multiple access mechanism, called FH-OFDMA. The logical channels provided by FH-OFDM are used by the MAC layer to construct control and traffic channels as described in Section 3.1.2.

### 3.1.2. MAC Layer

3.1.2.1. Control and Traffic Channels. The aim of the MAC layer is to use "N" subchannels of physical layer in an efficient way to enable concurrent communication of WAR with multiple MNs. To archive this, subchannels are classified as control channels and traffic channels as illustrated in Figure 3.3 and in the time domain, each subchannel is divided into certain number of time slots. The air interface is packet based. Hence, there is no circuit setup for voice calls or any other service. All services including voice are handled as data packets.

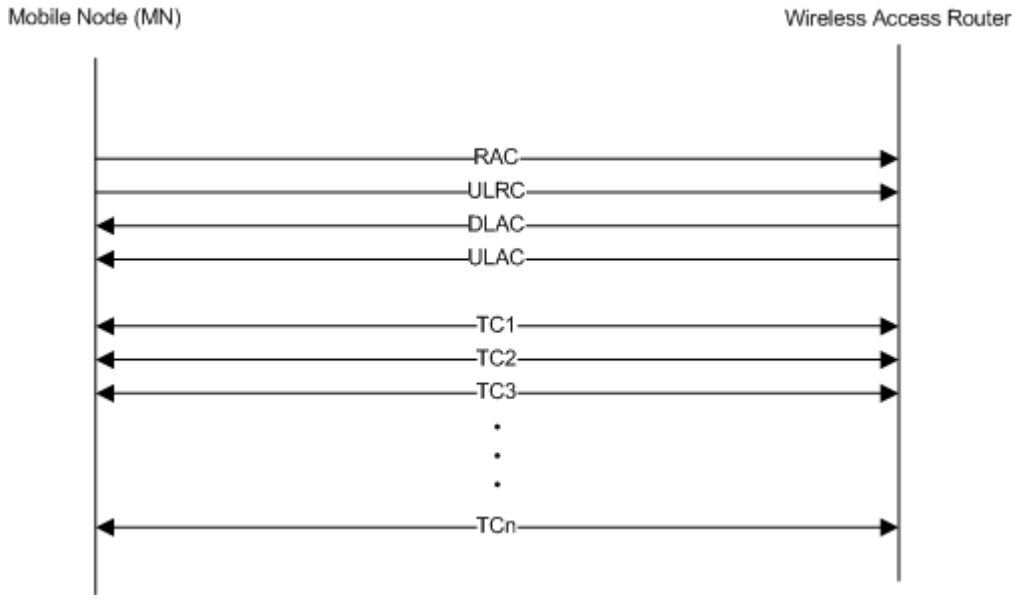


Figure 3.3. Control and Traffic Channels

The following channels are defined:

- **Random Access Channel (RAC)**

  RAC is bidirectional channel from MN to WAR. An MN sends its registration request to the WAR over this channel. This is the only channel that is randomly accessed by MN; therefore collisions may occur. All other channels are under the WAR's control, and there is no collisions for other channels. The response to registration request is sent over DLAC back to MN. If MN does not receive the registration response in a certain time frame, it utilizes exponential back-off algorithm to repeat the registration request.

| Header | Source MAC Address | Destination MAC Address |
|---|---|---|

Figure 3.4. RAC Packet Structure

Figure 3.4 shows the packet structure that is used on RAC. Header field can take one value as defined below:

  - ***Header: REG_REQ***
    - Source MAC Address: The MAC address of MN.
    - Destination MAC Address: The MAC address of WAR that MN wants

to register to.

- **Downlink Assignment Channel (DLAC)**

  DLAC is a bidirectional channel from WAR to MN. Besides downlink assignments, WAR sends broadcast announcements and registration responses over this channel.

| Header | Source MAC Address | Destination MAC Address | Packet Count | Channel | Slot |
|---|---|---|---|---|---|

Figure 3.5. DLAC Packet Structure

Figure 3.5 shows the packet structure that is used on DLAC. There are three header values defined for each functionality:

  - ***Header: DLAC_BROADCAST***

    This is for announcing the MAC address of WAR to MNs

    * Source MAC Address: The MAC address of WAR.
    * Destination MAC Address: Not used.
    * Packet Count: Not used.
    * Channel: Not used.
    * Slot: Not used.

  - ***Header: DLAC_REG_RESP***

    This is registration response for MN. It confirms that MN can use the air interface over specified WAR,

    * Source MAC Address: The MAC address of WAR.
    * Destination MAC Address: The MAC address of MN.
    * Packet Count: Not used.
    * Channel: Uplink Request Channel that MN will use to send its uplink requests.
    * Slot: Time slot that MN will use to send its uplink requests.

  - ***Header: DLAC_DLA***

    This is downlink assignment packet for previously registered MNs.

    * Source MAC Address: The MAC address of WAR.
    * Destination MAC Address: The MAC address of MN.
    * Packet Count: Number of packets that will be sent over specified channel

and slot.

  * Channel: Traffic Channel that packets for MN will be sent over.
  * Slot: Time slot that packets for MN will be sent over.

- **Uplink Request Channel (ULRC)**

  Upon registering to WAR, each MN is assigned a logical channel to send uplink requests. These requests are sent over bidirectional ULRC. This enables WAR to have complete information about transmission requirements of MNs. Then, WAR can perform necessary allocations according to current state.

| Header | Source MAC Address | Destination MAC Address | Packet Count | QoS |
|---|---|---|---|---|

Figure 3.6. ULRC Packet Structure

Figure 3.6 shows the packet structure that is used on ULRC. There is only one header value defined:

  – ***Header: ULRC_ULR***
    * Source MAC Address: The MAC address of MN.
    * Destination MAC Address: The MAC address of WAR that MN is register to.
    * Packet Count: Number of packets that MN wishing to send.
    * QoS: QoS requirement of the packets to be send.

- **Uplink Assignment Channel (ULAC)**

  MNs send uplink requests to WAR, which decides on allocation of uplink traffic channels to MNs. This assignment is done within scheduler module of WAR considering the service level agreements and QoS requirements of MNs.

| Header | Source MAC Address | Destination MAC Address | Packet Count | Channel | Slot |
|---|---|---|---|---|---|

Figure 3.7. ULAC Packet Structure

Figure 3.7 shows the packet structure that is used on ULAS. There is only one header value defined:

  – ***Header: ULAC_ULA***

    * Source MAC Address: The MAC address of WAR.
    * Destination MAC Address: The MAC address of MN.
    * Packet Count: Number of packets that uplink is reserved for MN.
    * Channel: Traffic Channel that MN can send packets over.
    * Slot: Time slot that MN can send packets over.

- **Traffic Channels (TC)**

  Traffic channels are used to carry user traffic. These channels are unidirectional, and utilized for both uplink and downlink.

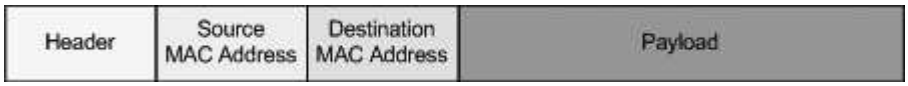


Figure 3.8. TC Packet Structure

Figure 3.8 shows the packet structure that is used on Traffic Channels. There is only one header value defined:

  - ***Header: TC_DATA***
    * Source MAC Address: The MAC address of source node.
    * Destination MAC Address: The MAC address of destination node.
    * Payload: Fixed length payload that carries segmented IP packets.

3.1.2.2. Mobile Node. Mobile Node is the mobile equipment that can attach to wireless access network through a WAR and that can roam between WARs. Figure 3.9 shows the flow diagram of MN.

- After initialization procedures, MN tries to register to the WAR with the strongest signal.
- The registration trials continue until a successful registration occurs.
- After successfully registering, MN starts processing uplink and downlink packets. It listens to downlink assignment channel for downlink packets addressed to itself. When a packet segment is received, it is stored until all packet segments reach MN. When the last segment is received, full packet is constructed and passed to upper layer in protocol stack. When new packets are received from upper layer to

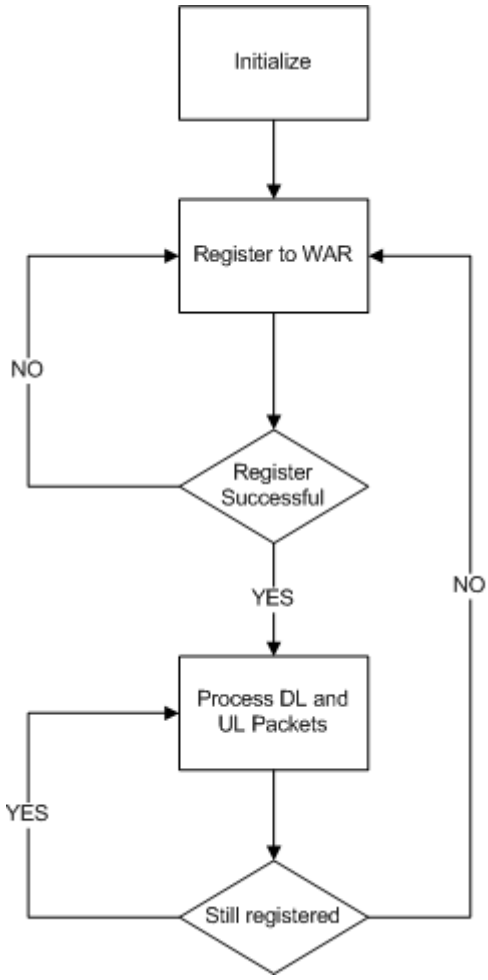


Figure 3.9. Mobile Node Flow Diagram

be transmitted over air interface, they are segmented and uplink for transmitting these segments are requested over uplink request channel. WAR assigns uplink to MN according to scheduling mechanisms as explained in Section 3.1.2.4 and informs MN via uplink assignment channel. MN then sends the segments over the traffic channel and time slot assigned to it.

- While registered to WAR, MN listens to broadcast messages from WAR to be sure that it is still registered. If not, it starts registration process again.

3.1.2.3. Wireless Access Router. Wireless Access Router is the node that bridges the access network to core network. It controls air interface resources, routes IP traffic from core network to access network and vice versa. Figure 3.10 shows the flow diagram of WAR.

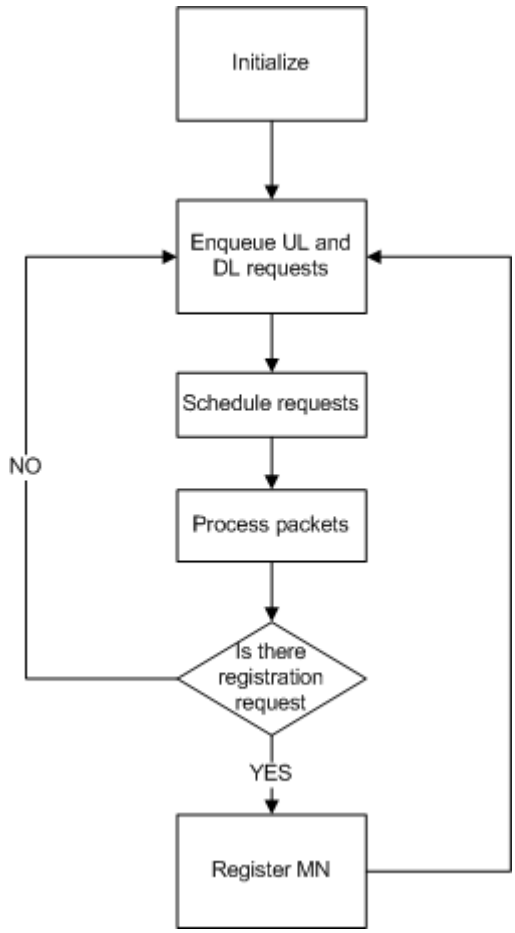


Figure 3.10. Wireless Access Router Flow Diagram

- After initialization procedures, WAR starts to enqueue uplink and downlink requests it receives to feed into scheduler module.
- The scheduler schedules the requests according to SLA and QoS requirements of MNs as explained in Section 3.1.2.4.
- Downlink packets are transmitted from assigned traffic channel during assigned time slot. Packet segments arriving to WAR are stored and after last segment arrives, whole packet is constructed, which is then passed to the upper layer.
- If there is a registration request, the new MN is registered.

3.1.2.4. Scheduler. Based on the assignments of traffic channels and packets enqueued to be transmitted, scheduler decides the new assignment of traffic channels for both uplink and downlink. There are two parameters that affects the channel assignment at scheduler:

- **Service Level Agreement:** It is important for network operators to provide different service levels to its customers and bill the users according to their service level. We define three service levels:
  - Platinum
  - Gold
  - Silver

  Platinum users have precedence over Gold users, and Gold users have precedence over Silver users.
- **Type of Service (ToS):** Another parameter used as input for scheduling is Type of Service. We define three service types:
  - Streaming
  - Standard
  - Background

  Streaming traffic has precedence over Standard traffic and Standard traffic has precedence over Background traffic.

To refer to a specific traffic type throughout the document SLA/QoS notation is used. For instance, to indicate Standard traffic generated by a user with Gold SLA level, Gold/Standard is used. To support these two parameters, we define the mapping of DSCP values in IP header as shown in Table 3.1. Whenever a packet is received, DSCP value is mapped to SLA and ToS values, which is then used throughout the scheduling process. We provide two scheduler modules, and investigate the effects in the simulation environment in Section 4.

- **Priority Scheduling**

  Priority Scheduling is a common concept used in operating systems to schedule process to execute [35]. In our application, the basic idea is to assign a priority to each packet, and then allow the packet with the highest priority to be transmitted next. Downlink and uplink requests are evaluated by the scheduler to assign priorities based on the SLA and ToS values. Scheduler also manages the traffic channel allocation states. With these inputs, whenever traffic channels are available and there are packets to be transmitted, downlink and uplink assign-

ments are done accordingly. This flow is illustrated in Figure 3.11 for downlink assignments and in Figure 3.12 for uplink assignments. In priority scheduling, the queues are checked in priority order starting from highest priority first. If there is a request in that queue, it is processed first. If there is no request, the queue with the next priority value is checked. This enables the requests with higher priority to have precedence over lower priority requests.

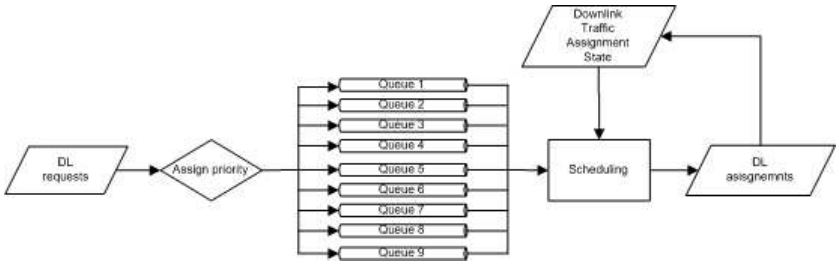


Figure 3.11. Scheduling Downlink Assignments

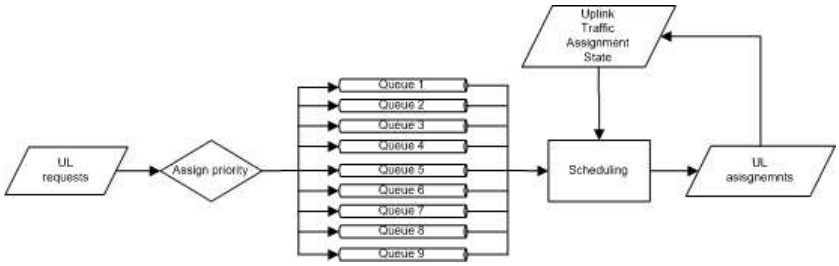


Figure 3.12. Scheduling Uplink Assignments

- **Priority Scheduling with Reservation**

  The main problem with priority pcheduling is starvation. If there are many high priority requests, lower priority requests may never be assigned traffic channels. To prevent this behavior, a reservation mechanism is introduced. Configurable amount of the bandwidth is reserved for specific SLA and ToS value. For instance, 70% of the bandwidth will be allocated to Platinum users, 20% to Gold users, and 10% to Silver users. In this scenario, if 70% of the traffic channels is allocated to Platinum user traffic, then even if there are more request from Platinum users, requests from Gold users are considered next. If 20% is also filled by Gold user traffic, then Silver user traffic is scheduled. It is also possible to reserve bandwidth for different ToS values under each SLA value. If there are no lower priority traffic,

Table 3.1. DSCP Mappings

| | | |
|---|---|---|
| Platinum | Streaming | AF41 |
| Platinum | Standard | AF42 |
| Platinum | Background | AF43 |
| Gold | Streaming | AF31 |
| Gold | Standard | AF32 |
| Gold | Background | AF33 |
| Silver | Streaming | AF21 |
| Silver | Standard | AF22 |
| Silver | Background | AF23 |

then higher priority traffic is allowed to use also the remaining bandwidth until lower priority traffic arrives and requests bandwidth. The flow is the same as in Figure 3.11 and Figure 3.12 but the logic in "Scheduling" process is as described above. This prevents low priority traffic requests from starving.

## 3.2. Core Network

For the core network of the architecture, IP/MPLS technology described in Section 2.2 is used. It is a trend that the integrated voice, video and data is transported in the converged IP/MPLS core network [36], which also perfectly fits in the overall architecture defined in this thesis. IP/MPLS technology provides necessary tools with DiffServ support and Traffic Engineering capabilities.

A LER node is the entrance point of the IP traffic to core network. After the packets are transferred to WAR over the air interface, they are forwarded to the LER to which the WAR is connected to. Each WAR is connected to a LER, but a LER can be connected to several WARs. Hence, there is a one-to-many relationship between a LER and the WARs. When the packet is received by the LER, it is assigned a label before forwarding to the next LSR. After the packet is assigned a label, the next hop is determined according to the LSP by table lookup. Layer 3 header is not considered in

the decision for the next hop. In MPLS, routing is done only when the packet enters the MPLS domain. After that, it is switching rather than routing. When the packet is received by the egress LER, the label is removed from the packet, and the packet is routed to the destination.

### 3.2.1. QoS in Core Network

As a standardized technology, designed from the start to be complementary to IP, MPLS is widely used in telecommunication networks. With its capabilities, it plays an important role in our architecture to meet strict QoS requirements in the core network.

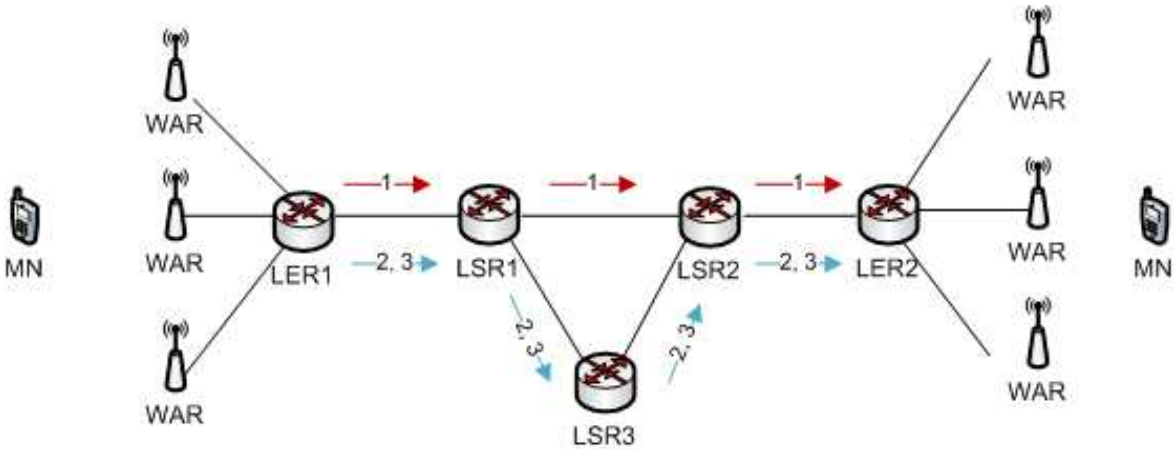


Figure 3.13. QoS in Core Network

In Figure 3.13, a sample scenario is shown for the core network. With DiffServ aware MPLS traffic engineering, it is possible to define different LSPs for different traffic types at for the backbone traffic. Considering the traffic from LER1 to LER2, there are three traffic types regarding the SLA value, Platinum, Gold, and Silver. Platinum traffic is mapped on to LSP shown as '1', and Gold and Silver traffic to LSP shown as '2,3'. These LSPs can be dynamically created based on the traffic requirements of relevant traffic type. It is also possible to define redundant paths to be resilient to link or node failures. DiffServ support of MPLS makes sure that, in case of congestion, packets are scheduled according to their PSC. Hence Streaming, Standard and Background traffics have different priorities in scheduling.

### 3.3. Overall Quality of Service

All-IP based 4G networks will be multi-service IP networks, in which even voice is transported over IP. Such a multi-service network must provide different QoS assurance for different types of services. The challenge lies in the fact that services require strict SLA with strict QoS requirements. The SLAs define the service quality experienced by the traffic transiting the network and are expressed in terms of latency, jitter, bandwidth guarantees, resilience in the face of failure, and downtime [36]. To meet these requirements, we can classify these requirements according to two conditions [37]:

- Different scheduling, queuing, and drop behavior based on the application type
- Bandwidth guarantees on a per-application basis

In this section, we describe how these two conditions are met from one end to the other, crossing the wireless access network and the core network.

In wireless access network, the scheduler at the MAC layer is where these issues are addressed. Priority Scheduler provides different scheduling, queuing, and drop behavior based on the application type, but can not provide bandwidth guarantees on a per-application basis. Priority Scheduling with Reservation extends Priority Scheduler to provide that functionality as well. It is possible to reserve bandwidth guarantees on a per-application basis. One point to remark here is the possibility of failure on providing guaranteed bandwidth in case there are more bandwidth requests for specific SLA and ToS value than available. To prevent this, WAR may limit the number of MNs that can be registered for each SLA value according to the available bandwidth provided by the air interface.

For the core network, DiffServ support of MPLS provides different scheduling, queuing, and drop behavior based on the application type. But DiffServ alone can not provide guaranteed bandwidth. In case the traffic flows from a path with insufficient resources to meet performance characteristics like jitter or latency, it is not possible to

meet the SLA. This problem may be solved by over-dimensioning the network resources, but this decreases the resource utilization dramatically, and is not resilient to link or node failures. By setting up LSPs along links with available resources, MPLS traffic engineering (MPLS-TE) can ensure that bandwidth is always available for a particular flow and avoids congestion both in the steady state and in case of failures. LSPs do not have to follow the shortest path if the resources are not sufficient along that path, which also increases the resource utilization. Another benefit of MPLS is link protection and fast reroute functionalities that provide resilience in case of failure. MPLS DiffServ-TE makes MPLS-TE aware of ToS, which allows resource reservation according to ToS and provide fault tolerance functionality of MPLS for each ToS value. MPLS DiffServ-TE meets the above two requirements for a strict SLA by combining DiffServ and traffic engineering functionality.

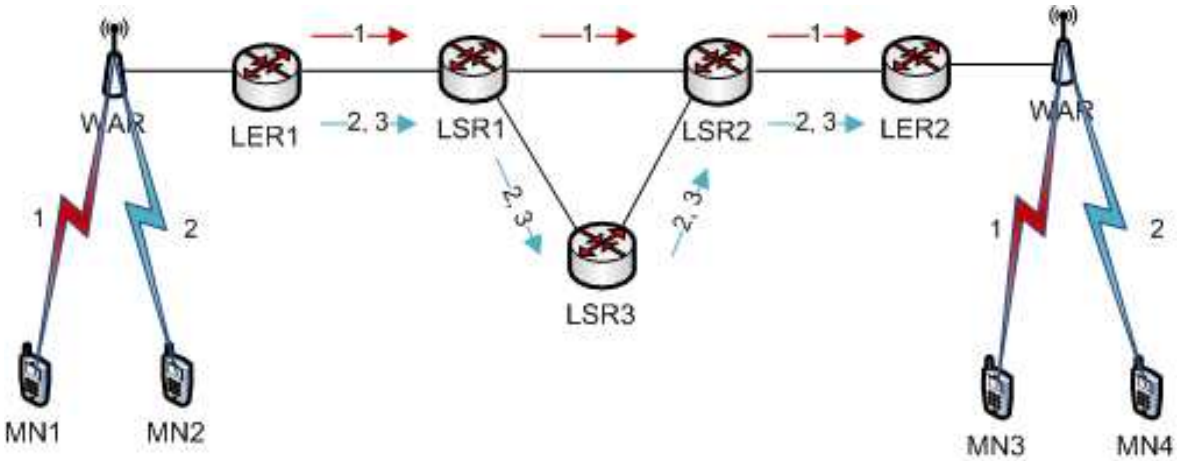


Figure 3.14. Overall QoS Architecture

Both wireless access network and core network use QoS mechanisms are based on DSCP value of IP packets, and can provide required QoS functionality. By jointly configuring access and core network, network operator can provide end-to-end QoS to the users. In Figure 3.14 let's suppose that MN1 is a Platinum user, MN2 is a Silver user, and both are using applications that generate Streaming and Standard traffic. Platinum traffic has precedence over Silver at air interface, and similarly Streaming traffic has precedence over Standard traffic. The scheduling at WAR is done accordingly. After the traffics enter the MPLS domain, Platinum traffic is mapped to LSP 1 and Silver traffic is mapped to LSP 2-3. Along the path, if congestion occurs on the link

between LSR1 and LSR3, Silver/Streaming traffic has precedence over Silver/Standard traffic and scheduling is done accordingly at the LSR nodes.

### 3.4. Mobility

To provide mobility in the network, Mobile IP is utilized. Being an extension to Internet Protocol, Mobile IP enables mobility at Layer 3 level. With Mobile IP, Mobile Nodes preserve IP connections when they are away from their home network. Each MN in the network has a Home Agent defined which is one of the WARs. Each WAR also implements Foreign Agent functionality so that when a Mobile Node enters its serving area, MN can register with its Home Agent and stay connected.

MN registers its current location to its Home Agent WAR. Home Agent WAR then keeps a record for each MN that it is serving. When MN is not at home network, Home Agent WAR redirects all traffic to the MN to its registered Foreign Agent WAR via tunneling.

# 4. SIMULATION

## 4.1. Simulation Environment

To analyze the architecture and the performance of the MAC layer developed, we have implemented a model of the architecture with OPNET Modeler[3] version 11.5. OPNET Modeler has a wireless module with MPLS support that can be used to model wireless communication. The wireless module enables development of custom nodes, which we have used to implement the new MAC layer proposed and create our nodes for WAR and MN that utilize the MAC layer. The process model for the MAC layer is shown in Figure 4.1. The source code is available in Appendix A. Underlying physical layer is implemented with OPNET wireless model radio receiver and transmitter using multiple streams at each receiver and transmitter. Each stream is used as a subchannel by the MAC layer.

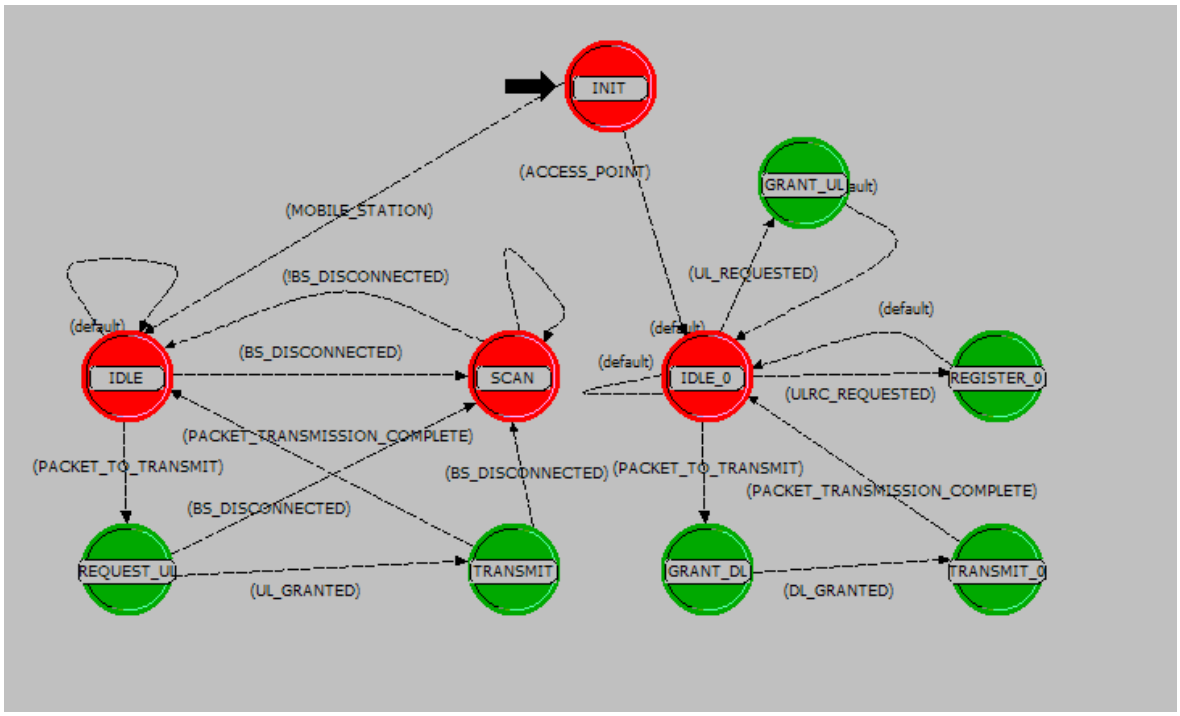


Figure 4.1. MAC Process Model

The OPNET Modeler project we implemented is shown in Figure 4.2 (Mobile Subnet) and Figure 4.3 (Top Level). As the test scenario, we have five WARs, three

Table 4.1. Node Coordinates

| **Node** | **X position (km)** | **Y position (km)** |
|---|---|---|
| WAR1 | 170 | 160 |
| WAR2 | 173 | 160 |
| WAR3 | 176 | 160 |
| WAR4 | 179 | 160 |
| WAR5 | 182 | 160 |
| MN0 | 171 | 161 |
| MN1 | 169.5 | 160.75 |
| MN2 | 170 | 160.75 |
| MN3 | 170.5 | 160.75 |
| MN4 | 172.5 | 160.75 |
| MN5 | 173 | 160.75 |
| MN6 | 173.5 | 160.75 |
| MN7 | 175.5 | 160.75 |
| MN8 | 176 | 160.75 |
| MN9 | 176.5 | 160.75 |
| MN10 | 178.5 | 160.75 |
| MN11 | 179 | 160.75 |
| MN12 | 179.5 | 160.75 |
| MN13 | 182 | 160.75 |
| MN14 | 182.5 | 160.75 |

of which are connected to the backbone over one LER, and two are connected to the backbone over another LER. First WAR has four MNs registered while second, third and fourth have three MNs each and the last one has two MNs. All MNs are connected to the backbone over a LER and they communicate with a remote media server. The coordinates of the MNs and WARs are listed in Table 4.1. Other parameters used during simulation are listed in Table 4.2.

There are four traffic patterns that we use during the simulations as listed in Table 4.3, Table 4.4, Table 4.5, and Table 4.6. There are three LSPs defined. As stated

Table 4.2. Simulation Parameters

| **Parameter Name** | **Parameter Value** |
|---|---|
| MN0 Speed | 320 km/h |
| Cell Radius | 2,000 m |
| Number of Traffic Channels | 2 |

in Section 3.1.2.4, there are three SLA levels; Platinum, Gold, and Silver. Platinum traffic from LER12 to LER8 is mapped to LSP with path {LER12, LSR9, LSR2, LSR7, LSR17, LSR6, LER8}. Gold and Silver traffic from LER12 is mapped to LSP with path {LER12, LSR9, LSR4, LSR7, LSR13, LSR6, LER8}. All traffic from LER11 is mapped to LSP with path {LER11, LSR9, LSR4, LSR7, LSR13, LSR6, LER7}.

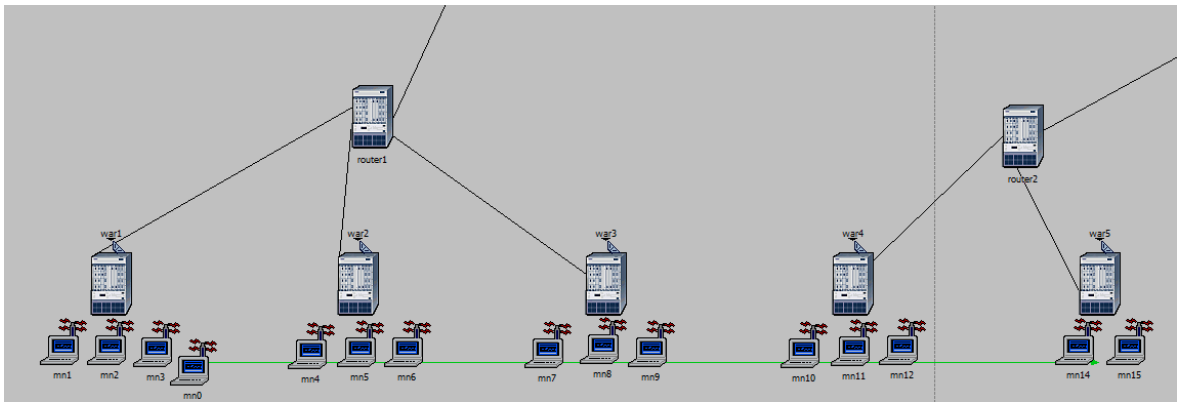


Figure 4.2. OPNET Modeler Project (Mobile Subnet)

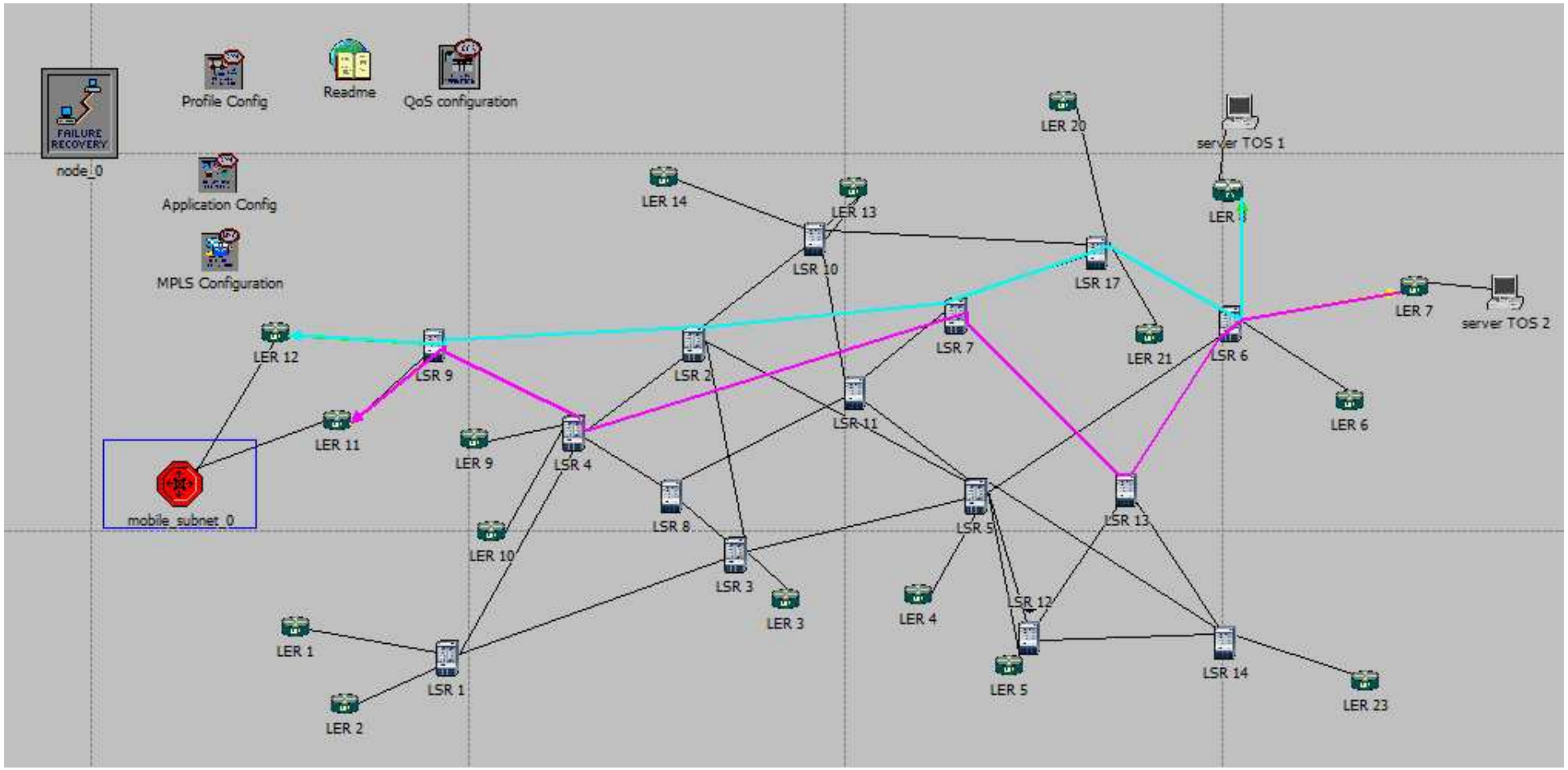


Figure 4.3. OPNET Modeler Project (Top Level)

Table 4.3. Mobile Node Traffic Pattern 1

| **Mobile Node** | **SLA** | **ToS** | **Start Time (s)** | **End Time (s)** | **Application Type** | **Bandwidth (B/s)** | **Mobility** |
|---|---|---|---|---|---|---|---|
| MN0 | Platinum | Streaming | 140 | 160 | Video | 75,000 | NO |
| MN1 | Gold | Standard | 130 | 180 | Video | 75,000 | NO |
| MN1 | Gold | Background | 120 | 190 | Voice | 1,650 | NO |
| MN2 | Silver | Streaming | 110 | 200 | Video | 75,000 | NO |
| MN3 | Silver | Standard | 100 | End of simulation | Video | 75,000 | NO |

Table 4.4. Mobile Node Traffic Pattern 2

| **Mobile Node** | **SLA** | **ToS** | **Start Time (s)** | **End Time (s)** | **Application Type** | **Bandwidth (B/s)** | **Mobility** |
|---|---|---|---|---|---|---|---|
| MN0 | Platinum | Streaming | 140 | 160 | Video | 75,000 | NO |
| MN1 | Platinum | Streaming | 130 | 180 | Video | 75,000 | NO |
| MN2 | Platinum | Streaming | 110 | 200 | Video | 75,000 | NO |
| MN3 | Platinum | Streaming | 100 | End of simulation | Video | 75,000 | NO |

Table 4.5. Mobile Node Traffic Pattern 3

| **Mobile Node** | **SLA** | **ToS** | **Start Time (s)** | **End Time (s)** | **Application Type** | **Bandwidth (B/s)** | **Mobility** |
|---|---|---|---|---|---|---|---|
| MN0 | Platinum | Streaming | 130 | 250 | Video | 75,000 | YES |
| MN1 | Gold | Standard | 120 | End of simulation | Video | 75,000 | NO |
| MN1 | Gold | Background | 115 | End of simulation | Voice | 1,650 | NO |
| MN2 | Silver | Streaming | 110 | End of simulation | Video | 75,000 | NO |
| MN3 | Silver | Standard | 105 | End of simulation | Video | 75,000 | NO |
| MN4 | Gold | Standard | 120 | End of simulation | Video | 75,000 | NO |
| MN5 | Silver | Streaming | 110 | End of simulation | Video | 75,000 | NO |
| MN6 | Silver | Standard | 105 | End of simulation | Video | 75,000 | NO |
| MN7 | Gold | Standard | 120 | End of simulation | Video | 75,000 | NO |
| MN8 | Silver | Streaming | 110 | End of simulation | Video | 75,000 | NO |
| MN9 | Silver | Standard | 105 | End of simulation | Video | 75,000 | NO |
| MN10 | Gold | Standard | 120 | End of simulation | Video | 75,000 | NO |
| MN10 | Gold | Background | 115 | End of simulation | Voice | 1,650 | NO |
| MN11 | Silver | Streaming | 110 | End of simulation | Video | 75,000 | NO |
| MN12 | Silver | Standard | 105 | End of simulation | Video | 75,000 | NO |
| MN13 | Silver | Streaming | 110 | End of simulation | Video | 75,000 | NO |
| MN14 | Silver | Standard | 105 | End of simulation | Video | 75,000 | NO |

Table 4.6. Mobile Node Traffic Pattern 4

| **Mobile Node** | **SLA** | **ToS** | **Start Time (s)** | **End Time (s)** | **Application Type** | **Bandwidth (B/s)** | **Mobility** |
|---|---|---|---|---|---|---|---|
| MN0 | Platinum | Streaming | 130 | 250 | Video | 75,000 | YES |
| MN1 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN2 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN3 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN4 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN5 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN6 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN7 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN8 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN9 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN10 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN11 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN12 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN13 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |
| MN14 | Platinum | Streaming | 130 | 250 | Video | 75,000 | NO |

## 4.2. Simulation Results

We have performed tests to analyze the behavior of the MAC layer in the case of sufficient and insufficient wireless resources.

### 4.2.1. Basic Scenarios Without Mobility

4.2.1.1. No Bandwidth Problem at the Air Interface and the Core Network. This is a reference scenario in which there is no bandwidth problem at the air interface and the core network. Available air bandwidth is 8.2 MBps. Only MN0, MN1, MN2, and MN3 generate traffic according to the first pattern shown in Table 4.3. In Figure 4.4 received flows, in Figure 4.5 delay values, and in Figure 4.6 queue sizes in MAC Scheduler module are shown respectively. Following are the key events to mention in the figures:

- *At $100^{th}$ sec,* MN3 Silver/Standard flow starts.
- *At $110^{th}$, $120^{th}$, $130^{th}$, and $140^{th}$ sec,* MN2 Silver/Streaming, MN1 Gold/Background, MN1 Gold/Standard, and MN0 Platinum/Streaming flows start, respectively. Since there is no bandwidth problem, other flows and delay values are not affected.
- *At $160^{th}$, $180^{th}$, $190^{th}$, and $200^{th}$ sec,* MN0 Platinum/Streaming, MN1 Gold/Standard, MN1 Gold/Background, and MN2 Silver/Streaming flows end, respectively.
- MN2 Silver/Standard flow continues until the end of simulation.

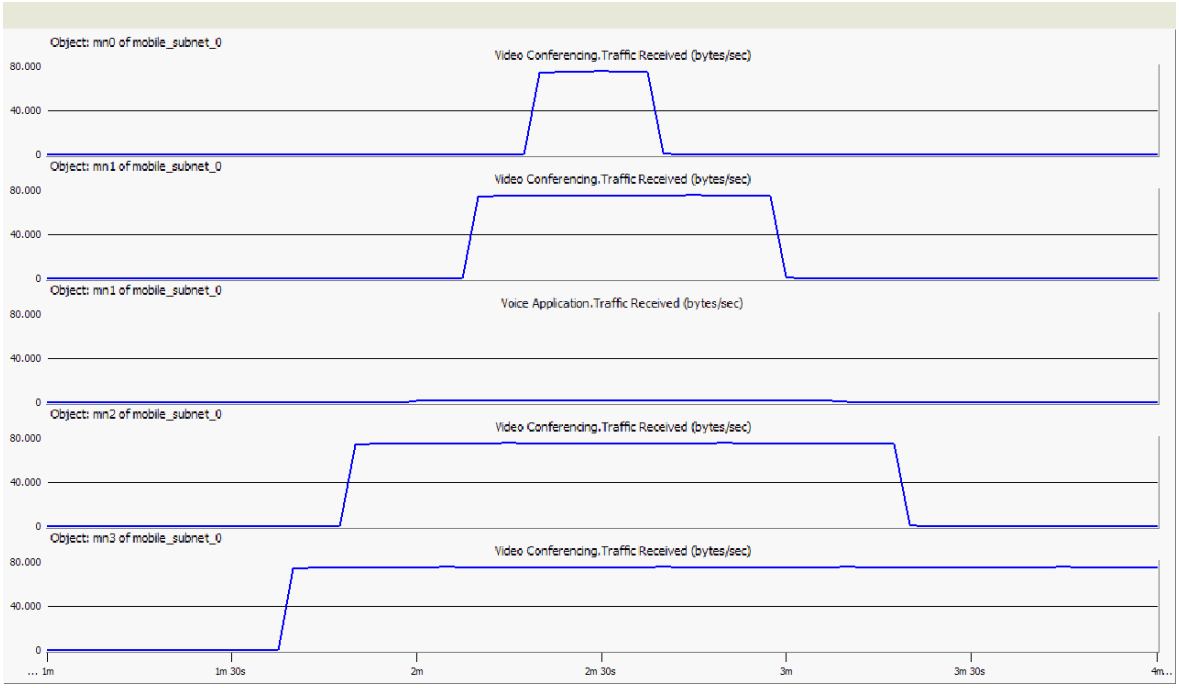


Figure 4.4. Received Flow

In Figure 4.4 received flow is shown. Since there is no bandwidth problem, all nodes can receive the expected flow. When a higher priority flow starts, other flows are not affected.

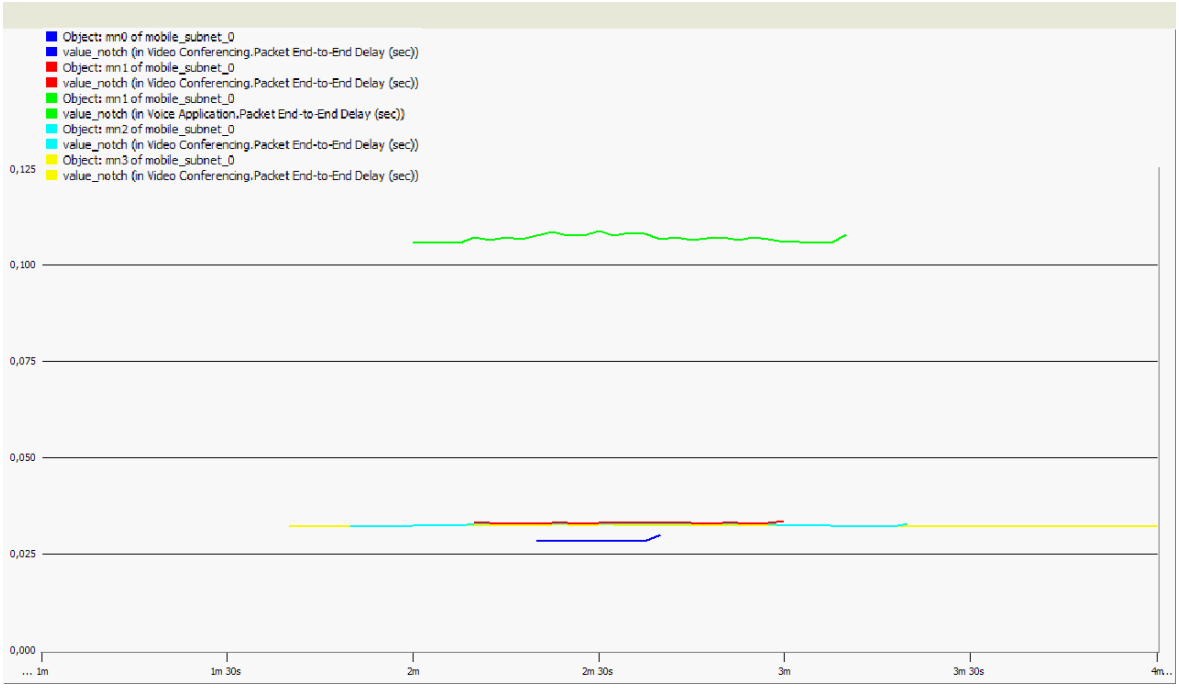


Figure 4.5. Delay

In Figure 4.5, delay values are shown for each flow. For MN3 Silver/Standard flow, MN2 Silver/Streaming, and MN1 Gold/Standard flows, the delay values are around 0.032 sec, each shown for the received flow period. For MN0 Platinum/Streaming flow, the delay value is around 0.027 sec. MN1 Gold/Background flow carries voice, and delay value for this flow is around 0.105 sec.

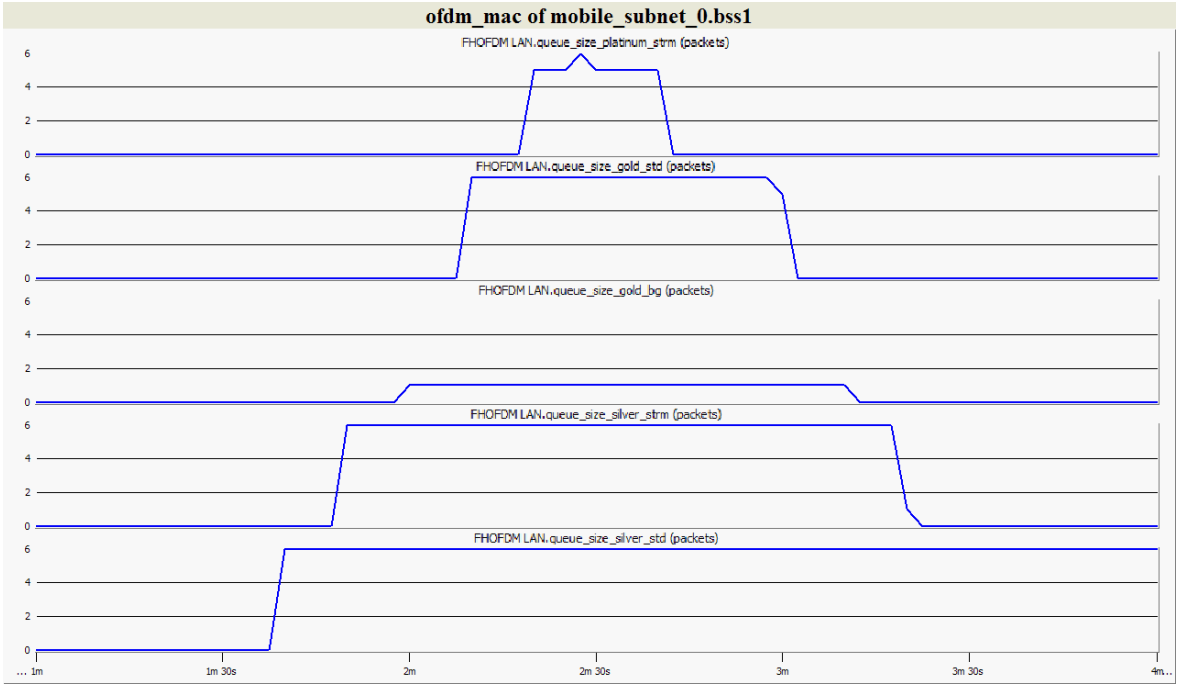


Figure 4.6. MAC Scheduler Queue

In Figure 4.6, queue sizes in MAC Scheduler module are shown. For each flow type, there is a different queue with different priority level at the MAC layer implementation. Since there is no congestion at the air interface, these queues have small number of packets throughout the simulation. When a node starts to receive flow, the packets are queued at the related queue, and sent to the mobile node afterwords. For instance, at $140^{th}$ sec, MN0 starts receiving the flow, until the first downlink scheduling is performed. The queue size goes to five, and stays around that value until the received flow ends. The case is similar for the other flows.

### 4.2.2. Low Bandwidth at the Air Interface (Priority Scheduler)

In this scenario, the bandwidth for the air interface is 2.4 MBps. There is no bandwidth problem for the core network. Priority Scheduler algorithm is used for MAC scheduler, and only MN0, MN1, MN2, and MN3 generate flows according to the first pattern as shown in Table 4.3. In Figure 4.7 received flow, in Figure 4.8 delay values, and in Figure 4.9 queue sizes in MAC Scheduler module are shown respectively. Following are the key events to mention in the figures:

- *At 100$^{th}$ sec,* MN3 Silver/Standard flow starts.
- *At 110$^{th}$ and 120$^{th}$ sec,* MN2 Silver/Standard and MN1 Gold/Background flows start, respectively. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 130$^{th}$ sec,* MN1 Gold/Standard flow starts. At this point, all available air resources are allocated; hence congestion starts. Silver/Standard flow is affected. Same effect is observed in delay and queue size, both increasing.
- *At 140$^{th}$ sec,* MN0 Platinum/Streaming flow starts. Since this is the flow with highest priority, more flows are affected. Silver/Standard flow starts to starve, and Silver/Streaming flow is also affected. For delay graphics, we observe a gap for starving flows, since no flow can be received during this period. The queue sizes also increase dramatically.
- *At 160$^{th}$ sec,* MN0 Platinum/Streaming flow ends. Since there is available bandwidth, firstly Silver/Streaming flow receives the buffered flow, then Silver/Standard flow proceeds similarly. Delay values and queue sizes decrease in parallel.

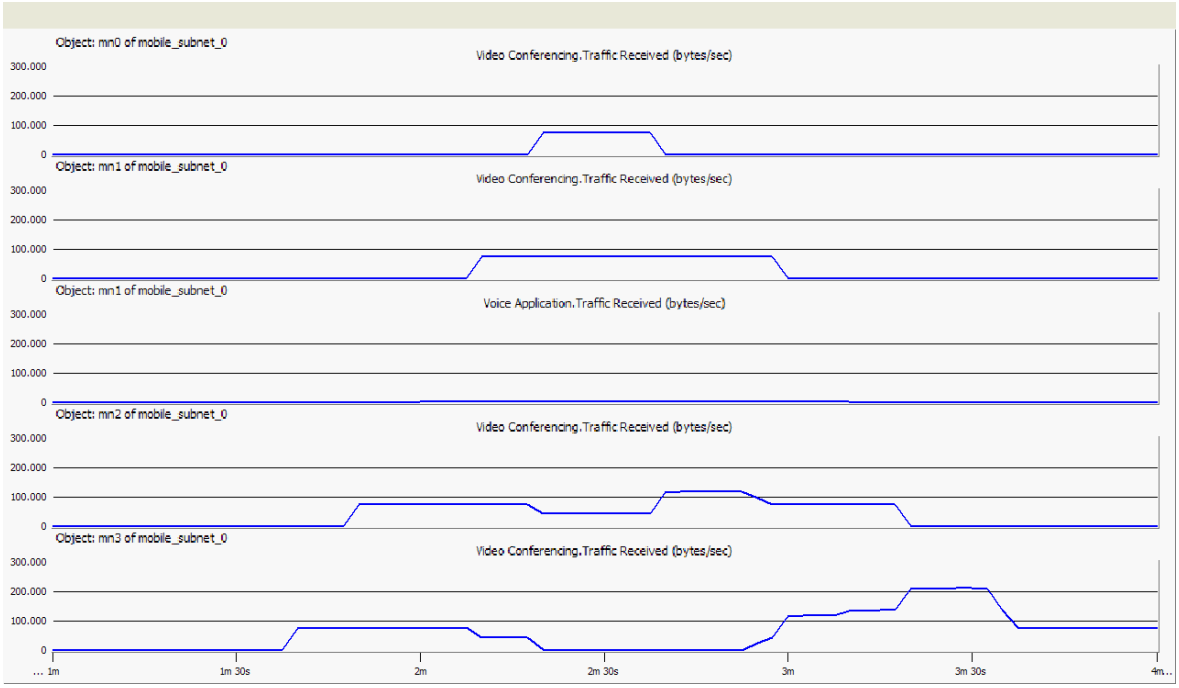


Figure 4.7. Received Flow

In Figure 4.7, received flows are shown. When MN1 Gold/Standard flow starts at $130^{th}$ second, the air interface is congested; hence the lowest priority flow, MN3 Silver Standard, decreases. At $140^{th}$ second, MN0 Platinum/Streaming flow starts as the highest priority flow. Since reservation is not used for this case, MN3 Silver/Standard flow starts to starve and does not receive any flow. MN2 Silver/Streaming flow is also affected, and received flow for this node decreases. When MN0 Platinum/Streaming flow ends at $160^{th}$ second, MN2 Silver/Streaming node starts receiving the flow since there is available bandwidth. The packets for this node are queued at the MAC layer. When there is available bandwidth, these packets are sent to MN2, which explains the peak at MN2 graph. At $180^{th}$ second, MN1 Gold/Standard flow also ends. The available bandwidth is used to send the queued packets to MN3 Silver/Standard flow, which creates the high increase in MN3 graph. After MN2 Silver/Streaming flow also ends at $200^{th}$ second, MN3 receives all the queues packets, and the received flow increases up to 200,000 bps. The queue is emptied after around $215^{th}$ second, and the received flow for MN3 goes down to 75,000 bps.

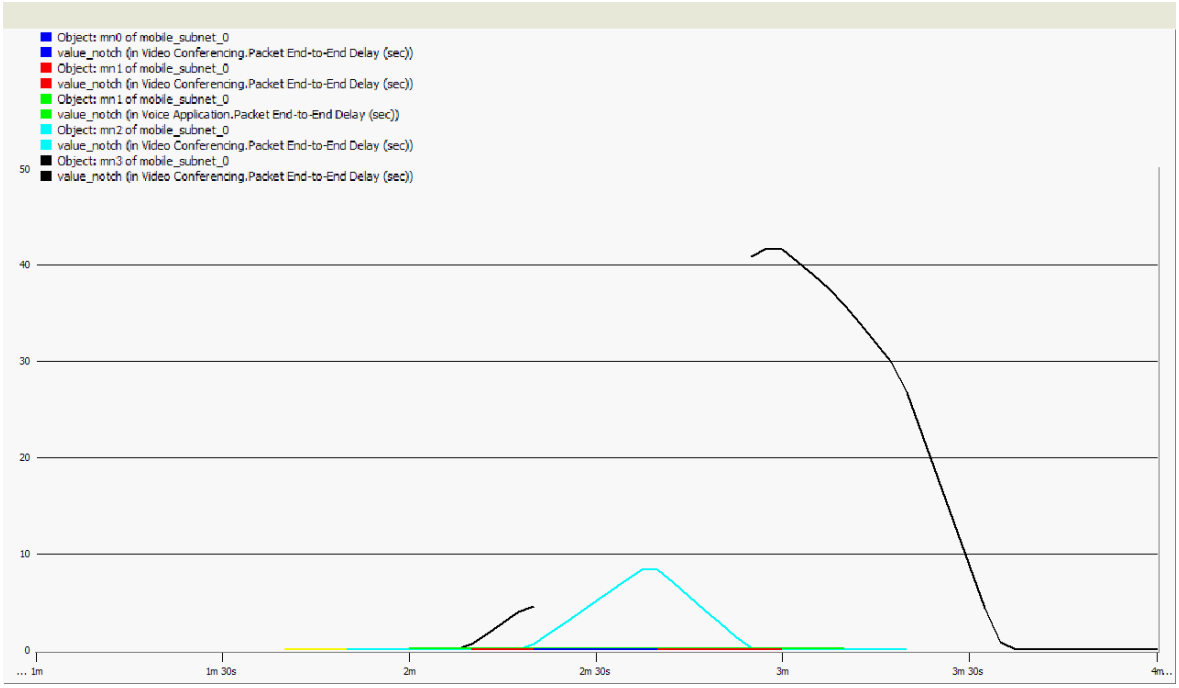


Figure 4.8. Delay

In Figure 4.8, delay values are shown for each flow. As shown in Figure 4.7, MN0 and MN1 are not affected by congestion and receive all flow, hence their delay values are below 0.1 second. When MN1 Gold/Standard flow starts at $130^{th}$ second, the delay value for MN3 increases up to five seconds until the start of MN0 Platinum/Streaming flow at $140^{th}$ second in parallel to the air interface congestion. Then, MN3 can not receive any flow and delay value is not available until it starts receiving the flow at around $170^{th}$ second. At $140^{th}$ second, the delay value for MN2 also starts to increase up to nine, and after $160^{th}$ second, starts to decrease since MN0 flow ends and available bandwidth is used to send queued packets to MN2.

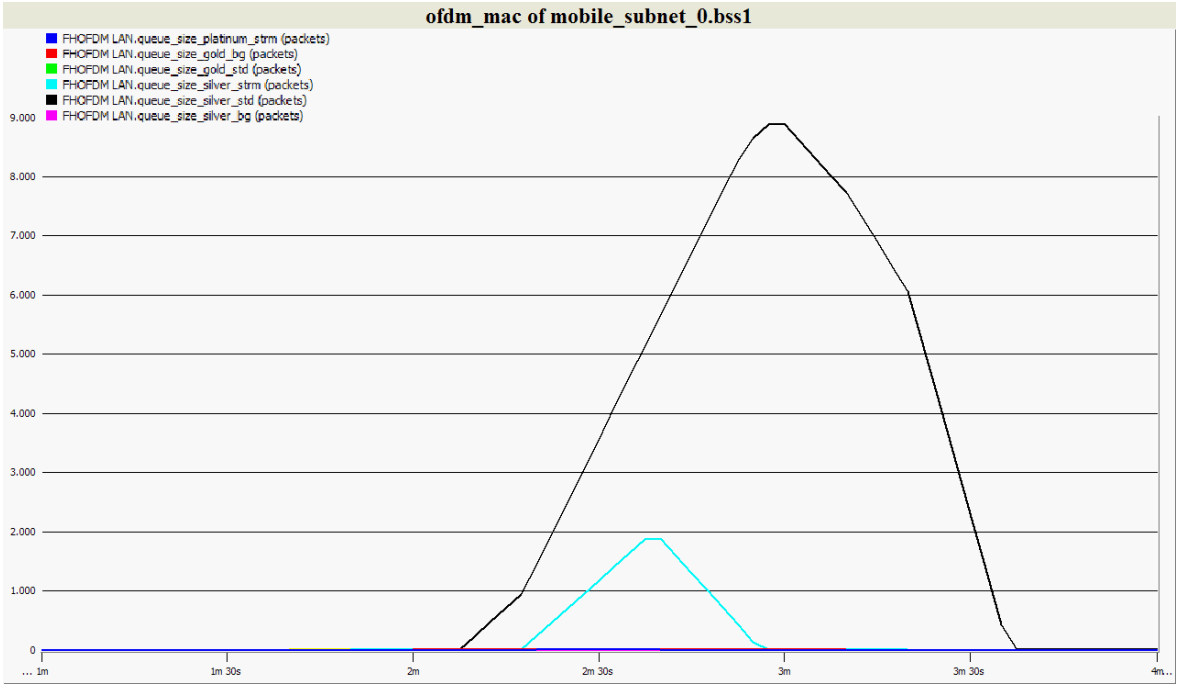


Figure 4.9. MAC Scheduler Queue

In Figure 4.9, queue sizes in MAC Scheduler module are shown. Queue sizes show very similar pattern to the delay values, as expected. When MN1 Gold/Standard flow starts at $130^{th}$ second, the queue size for MN3 starts to increase up to 9,000 in parallel to the air interface congestion. With the start of MN0 Platinum/Streaming flow at $140^{th}$ second, the queue size for MN3 also increases up to 1,900 until $160^{th}$ second when the MN0 flow ends. Packets in the queue are sent to MN2 and queue size decreases since there is available bandwidth. Similarly, after other flows end, packets are delivered to MN3 and the queue size for MN3 decreases.

### 4.2.3. Low Bandwidth at the Air Interface (Priority Scheduler with Reservation)

In this scenario, the bandwidth for the air interface is again 2.4 MBps. There is no bandwidth problem for the core network. Priority Scheduler with Reservation algorithm is used for MAC scheduler with reserved bandwidth for Platinum, Gold and Silver are 70%, 20%, and 10% respectively. Only MN0, MN1, MN2, and MN3 generate flows according to the first pattern as shown in Table 4.3. In Figure 4.10 received flow, in Figure 4.11 delay values, and in Figure 4.12 queue sizes in MAC Scheduler module are shown respectively.

- *At 100$^{th}$ sec,* MN3 Silver/Standard flow starts.
- *At 110$^{th}$ and 120$^{th}$ sec,* MN2 Silver/Standard and MN1 Gold/Background flows start, respectively. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 130$^{th}$ sec,* MN1 Gold/Standard flow starts. At this point, all available air resources are allocated; hence congestion starts. Silver/Standard flow is affected. Same effect is observed in delay and queue size, both increasing.
- *At 140$^{th}$ sec,* MN0 Platinum/Streaming flow starts. Since this is the flow with highest priority, more flows are affected. However, there is no starvation since reservation is applied. Silver Background and Silver/Standard flows are affected. The delay end queue size values also increase.
- *At 160$^{th}$, 180$^{th}$, and 190$^{th}$ sec,* MN0 Platinum/Streaming, MN1 Gold/Standard, and MN1 Gold/Background flows end, respectively. After a flow ends, there is available bandwidth that is used by other flows and they receive the buffered packets. Delay values and queue sizes decrease in parallel.

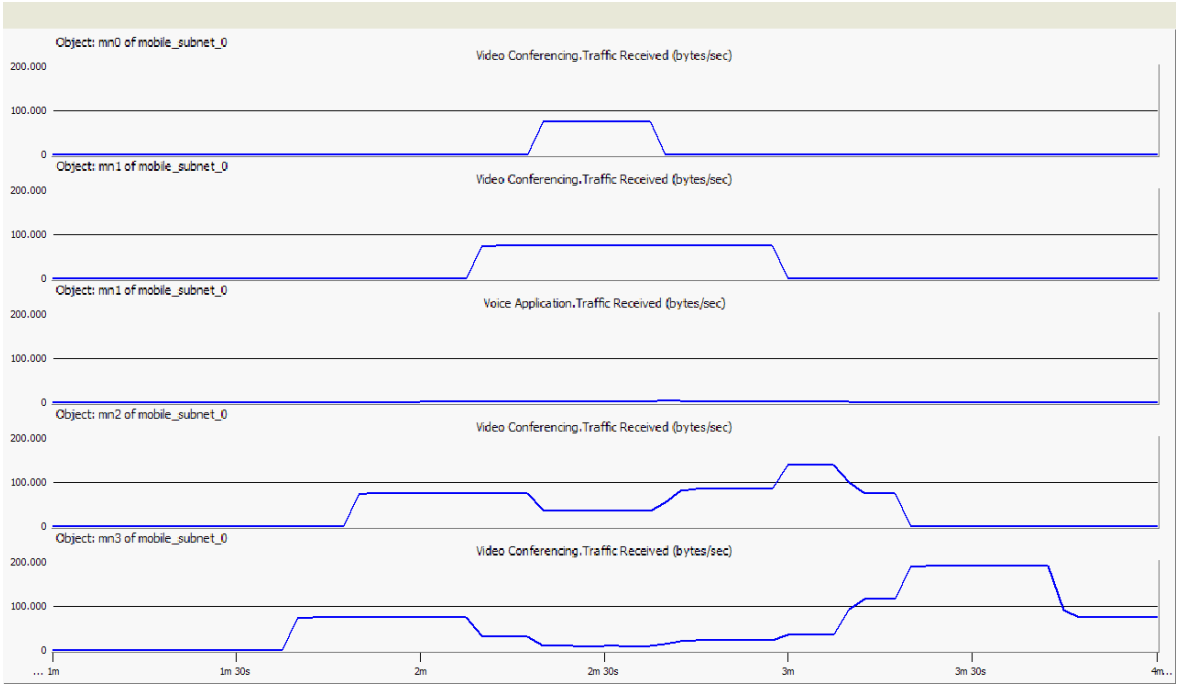


Figure 4.10. Received Flow

In Figure 4.10, received flows are shown. When MN1 Gold/Standard flow starts at $130^{th}$ second, the air interface is congested; hence the lowest priority flow, MN3 Silver Standard, decreases. At $140^{th}$ second, MN0 Platinum/Streaming flow starts as the highest priority flow. Since reservation is used for this case, MN3 Silver/Standard flow does not starve and continues to receive flow according to the reserved bandwidth. MN2 Silver/Streaming and MN1 Gold/Background flows are also affected, and received flow for this nodes decreases. When MN0 Platinum/Streaming flow ends at $160^{th}$ second, the flow received by MN1, MN2 and MN3 start to increase since there is available bandwidth. The packets for this nodes are queued at the MAC layer. We do not observe a peak like the previous case since the available bandwidth is distributed between two nodes. At $180^{th}$ second, MN1 Gold/Standard flow also ends. The available bandwidth is used to send the queued packets to MN0 Silver/Standard flow, which again increases the flows received by MN2 and MN3. After $200^{th}$ second, MN2 receives all queued packets hence received flow rate goes back to 75,000 bps. After that flow also finishes, MN3 starts to receive all queued packets, and after $225^{th}$ second received flow rate for MN3 goes back to 75,000 bps.

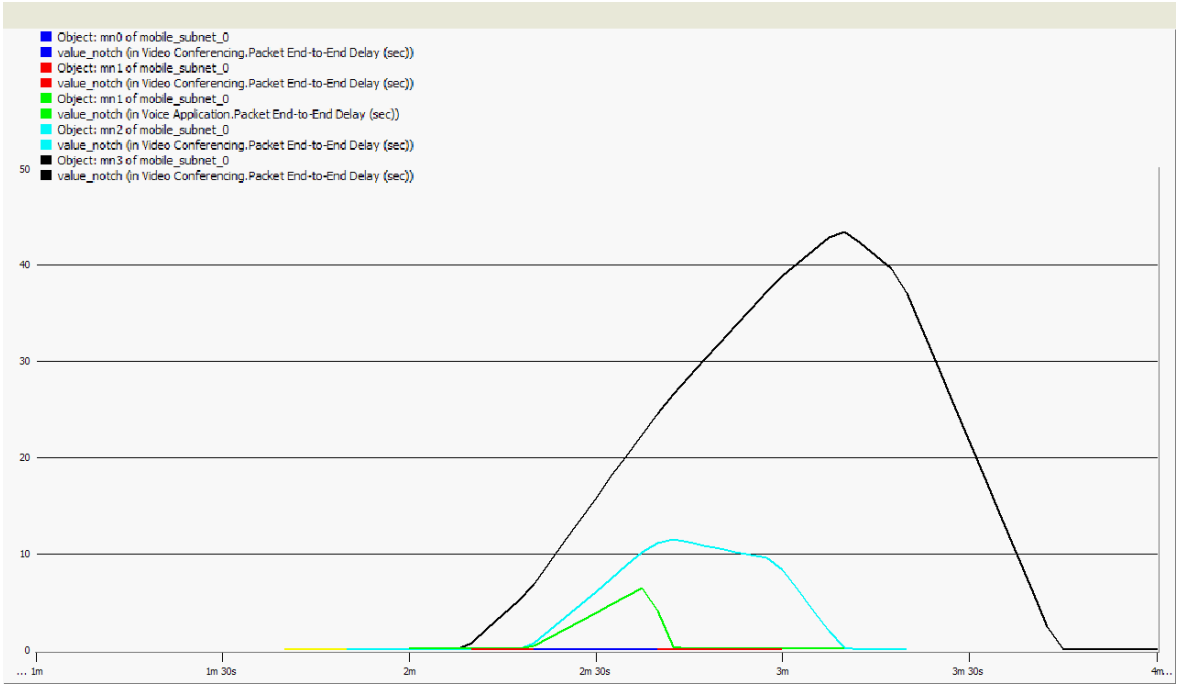


Figure 4.11. Delay

In Figure 4.11, delay values are shown for each flow. As shown in Figure 4.10, MN0 Platinum/Streaming and MN1 Gold/Standard flows are not affected by congestion and receive all flow, hence their delay values are below 0.1 second. When MN1 Gold/Standard flow starts at $130^{th}$ second, the delay value for MN3 starts to increase in parallel to the air interface congestion. Unlike the previous scenario, MN3 does not starve and continues to receive flow due to reservation. The delay value goes up to 42 seconds until other flows end and available bandwidth is used for MN3. At $140^{th}$ second, the delay values for MN1 Gold/Background and MN2 also start to increase. After $160^{th}$ second, delay value for MN1 starts to decrease since MN0 flow ends.

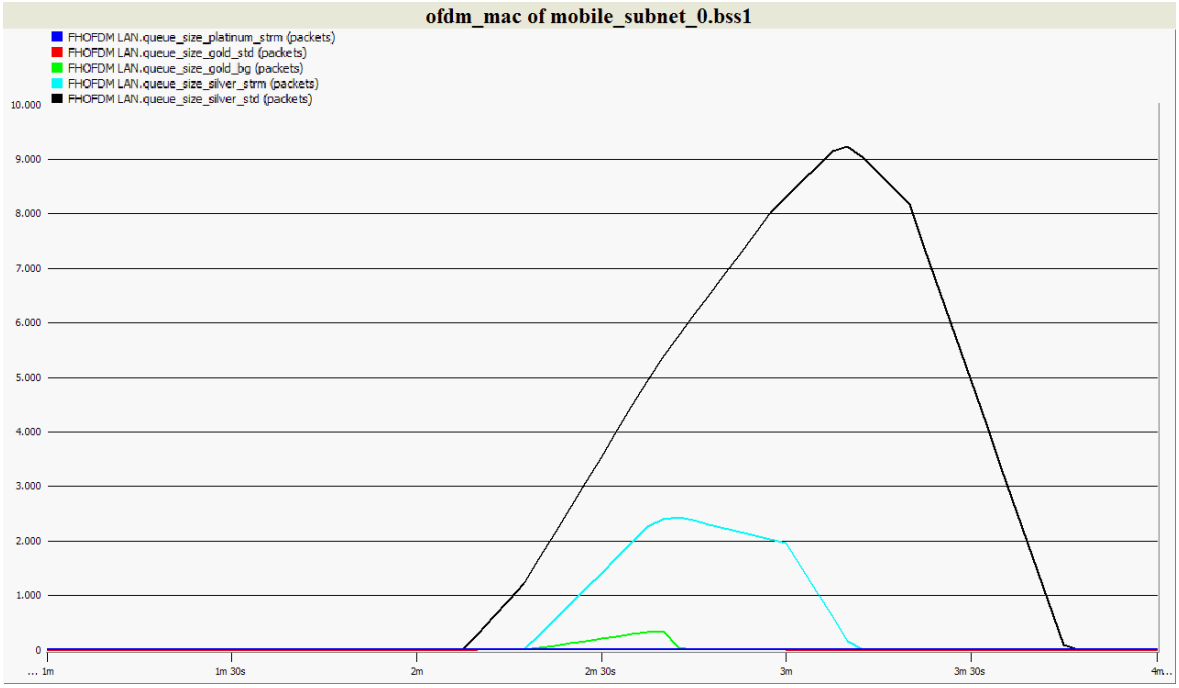


Figure 4.12. MAC Scheduler Queue

In Figure 4.12, queue sizes in MAC Scheduler module are shown. Queue sizes show very similar pattern to the delay values, as expected. When MN1 Gold/Standard flow starts at $130^{th}$ second, the queue size for MN3 flow starts to increase up to 9,100 in parallel to the air interface congestion. With the start of MN0 Platinum/Streaming flow at $140^{th}$ second, queue size for MN1 and MN3 also increases until $160^{th}$ second when the MN0 flow ends. Then packets in the queue are sent to MN1 and MN2 and queue size decreases since there is available bandwidth. Similarly, after other flows end, packets are delivered to MN3 and the queue size for MN3 decreases.

### 4.2.4. Low Bandwidth at the Core Network (DiffServ Effect)

In this scenario, there is no bandwidth problem for the air interface. To analyze the effect of DiffServ at the core network, the bandwidth between LER 12 and LSR 9 shown in Figure 4.3 is set to 2.048 Mbps. Only MN0, MN1, MN2, and MN3 generate flows according to the first pattern as shown in Table 4.3. In Figure 4.13 received flow and in Figure 4.13 delay values are shown respectively.

- *At 100$^{th}$ sec,* MN3 Silver/Standard flow starts.
- *At 110$^{th}$, 120$^{th}$, and 130$^{th}$ sec,* MN2 Silver/Standard, MN1 Gold/Background, and MN1 Gold/Standard flows start, respectively. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 140$^{th}$ sec,* MN0 Platinum/Streaming flow starts. This affects Silver flows. Since there is no queue limit at routers, we observe similar pattern for the received flows.
- *At 160$^{th}$, 180$^{th}$, and 190$^{th}$ sec,* MN0 Platinum/Streaming, MN1 Gold/Background, and MN1 Gold/Standard flows end, respectively. After a flow ends, there is available bandwidth that is used by other flows, and they receive the buffered packets. Delay values decrease in parallel.

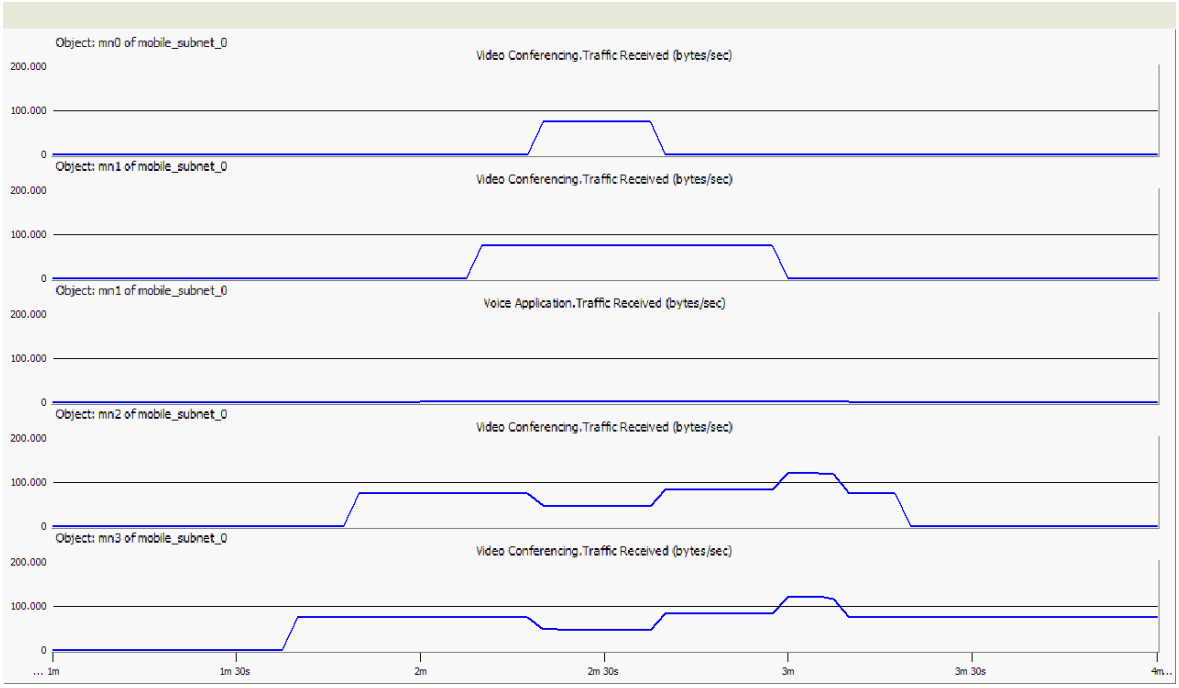


Figure 4.13. Received Flow

In Figure 4.13, received flows are shown. After start of MN0 Platinum/Streaming flow at $140^{th}$ second, the backbone link is congested. This affects MN2 and MN3 flows and they decrease. When MN0 flow ends at $160^{th}$ second, the available bandwidth is used to deliver packets to MN2 and MN3, and their received flow increases. After MN1 flow also ends at $180^{th}$ second, received flows increase up to 120,000 bps as the queued packets are delivered over the available bandwidth. At around $200^{th}$ second, flow rate for the nodes go back to 75,000 bps again.

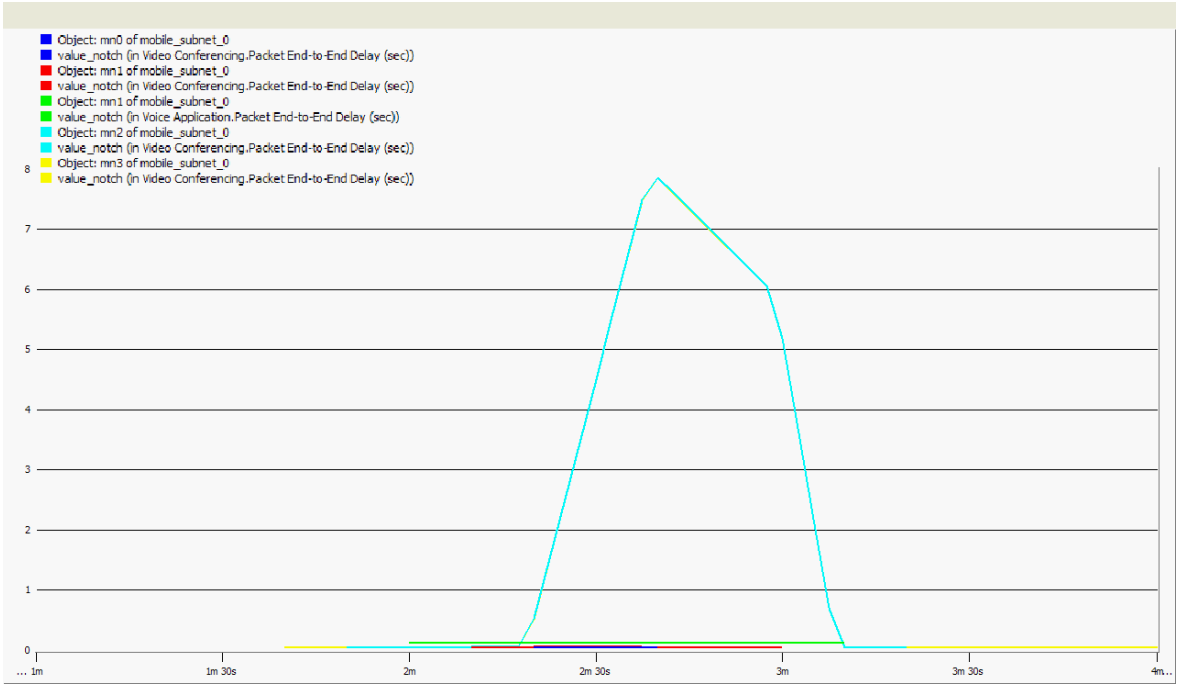


Figure 4.14. Delay

In Figure 4.14, delay values are shown for each flow. As shown in Figure 4.13, MN0 and MN1 flows are not affected by congestion and receive all flow, hence their delay values are below 0.1 second. When MN0 flow starts at $140^{th}$ second, in parallel to the backbone link congestion, the delay values for MN2 and MN3 start to increase up to 8 seconds. When MN0 flow ends at $160^{th}$ second, the delay values start to decrease and $190^{th}$ second they go down to initial values.

### 4.2.5. Low Bandwidth at the Core Network (Traffic Engineering Effect)

In this scenario, there is no bandwidth problem for the air interface. To analyze the effect of MPLS DiffServ-TE at the core network, the bandwidth between LER 9 and LSR 4 shown in Figure 4.3 is set to 1.544 Mbps. Only MN0, MN1, MN2, and MN3 generate flows according to the first pattern as shown in Table 4.3. In Figure 4.15 received flow and in Figure 4.16 delay values are shown respectively.

- *At 100$^{th}$ sec,* MN3 Silver/Standard flow starts.
- *At 110$^{th}$, and 120$^{th}$ sec,* MN2 Silver/Standard and MN1 Gold/Background flows start, respectively. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 130$^{th}$ sec,* MN1 Gold/Standard flow starts. Since Gold and Silver flows are mapped to same LSP which is congested, this affects Silver flows.
- *At 140$^{th}$ sec,* MN0 Platinum/Streaming flow starts. Since Platinum flow is mapped to another LSP which does not have congestion, this does not affect other flows.
- *At 160$^{th}$, 180$^{th}$, and 190$^{th}$ sec,* MN0 Platinum/Streaming, MN1 Gold/Background, and MN1 Gold/Standard flows end, respectively. After Gold flows end, there is available bandwidth that is used by Silver flows and they receive the buffered packets. Delay values decrease in parallel.

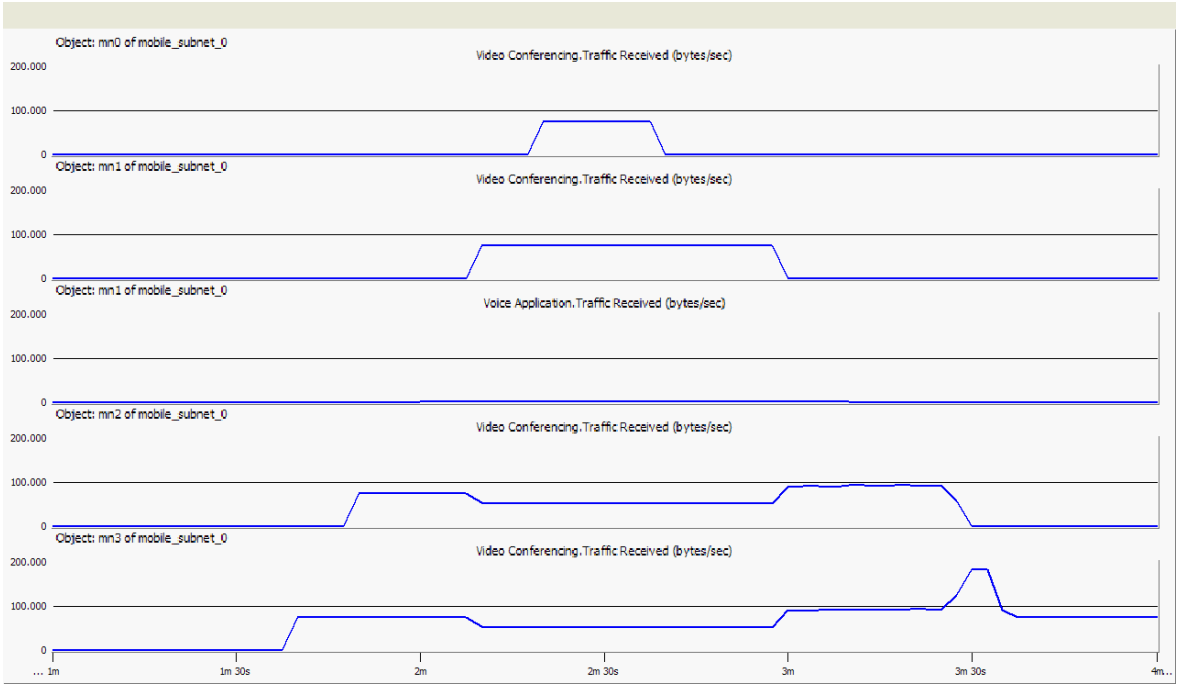


Figure 4.15. Received Flow

In Figure 4.15, received flows are shown. After start of MN1 Gold/Standard flow at $130^{th}$ second, the backbone link is congested. This affects MN2 and MN3 flows and they decrease. When MN0 flow starts $140^{th}$ second, other nodes are not affected since Platinum flow is mapped to another LSP which is not congested. After MN1 flow ends at $180^{th}$ second, received flows increase up to 100,000 bps as the queued packets are delivered over the available bandwidth. At around $200^{th}$ second, MN2 flow also ends and all bandwidth is used to deliver packets to MN3 which results in a sudden increase at received flow for MN3. After all the queued packets are delivered, flow rate goes down to 75,000 bps.

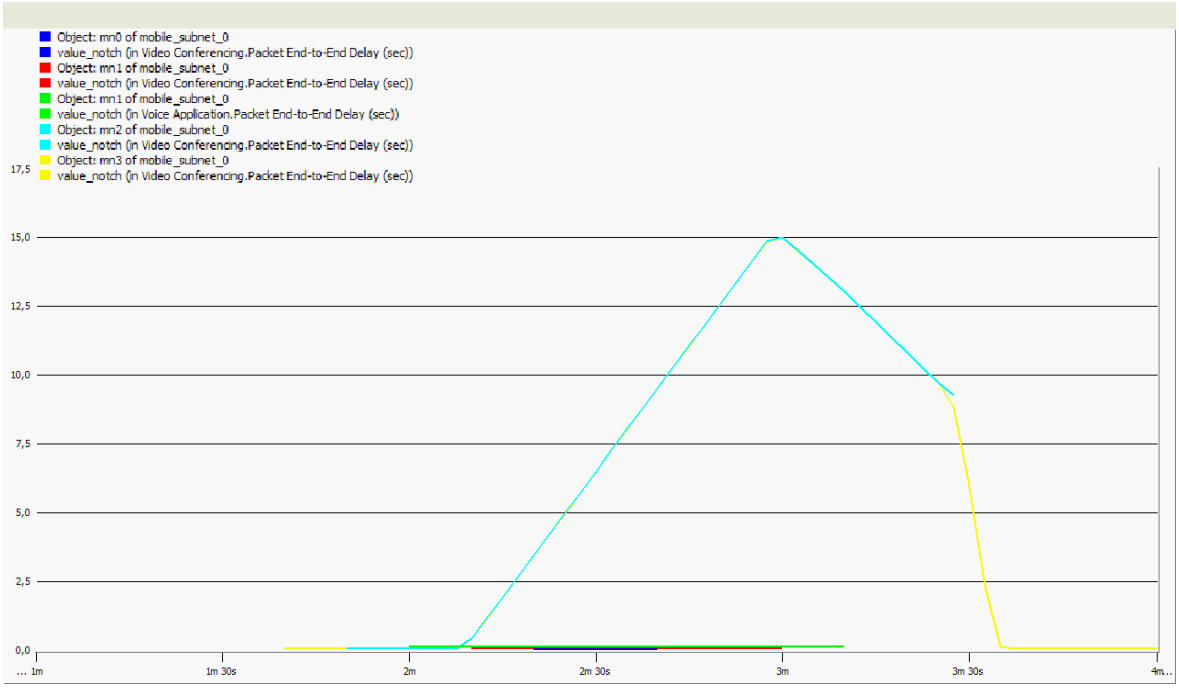


Figure 4.16. Delay

In Figure 4.16, delay values are shown for each flow. As shown in Figure 4.15, MN0 and MN1 flows are not affected by congestion and receive all flow, hence their delay values are below 0.1 second. When MN1 flow starts at $130^{th}$ second, in parallel to the backbone link congestion, the delay values for MN2 and MN3 start to increase up to 15 seconds. The start of MN0 flow at $140^{th}$ second does not affect other delay values since Platinum flow is carried over another LSP. When MN1 flow ends at $180^{th}$ second, the delay values start to decrease and $200^{th}$ second MN2 flow also ends. After that, the delay value for MN3 suddenly decreases down to initial level.

### 4.2.6. No QoS at the Air Interface

This scenario depicts the default behaviour of the system without our proposed scheme. Therefore, we use it as a reference to show the improvement provided by our proposed QoS mechanism. Only MN0, MN1, MN2, and MN3 generate flows according to the second pattern as shown in Table 4.4. In Figure 4.17 received flow and in Figure 4.18 delay values are shown respectively.

- *At 100$^{th}$ sec,* MN3 flow starts.
- *At 110$^{th}$ sec,* MN2 flows. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 130$^{th}$ sec,* MN1 flow starts. At this point, all available air resources are allocated; hence congestion starts. Since all flow have the same QoS values all get the same air bandwidth.
- *At 140$^{th}$ sec,* MN0 flow starts. Again since all flows have same QoS value, available bandwidth is distributed among users
- *At 160$^{th}$ and 190$^{th}$ sec,* MN0 and MN1 flows end, respectively. After flows end, there is available bandwidth that is used by other flows and they receive the buffered packets. Delay values decrease in parallel.

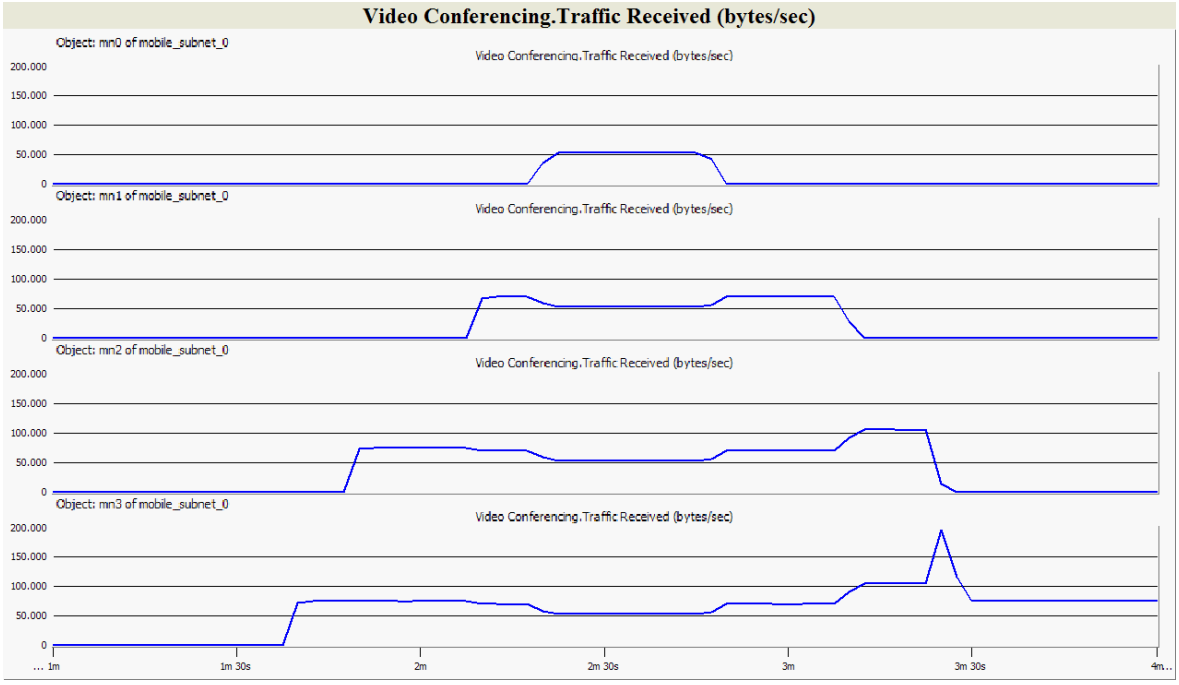


Figure 4.17. Received Flow

In Figure 4.17, received flows are shown. When MN1 flow starts at $130^{th}$ second, the air interface is congested. Since all the flows have the same priority, available air bandwidth is divided between the nodes. All three node receive around 65,000 bps flow. At $140^{th}$ second, MN0 flow starts. Again all available bandwidth is divided between nodes and each gets around 50,000 bps. When MN0 flow ends at $160^{th}$ second, other three nodes again receive flow with rate 65,000 bps. When MN1 flow ends at $180^{th}$ second, MN2 and MN3 shares the bandwidth with around 105,000 bps rate each. After MN2 flow also ends at $200^{th}$ second, MN3 gets all the flow and it is observed as the peak increase at the graph. After all queued packets are delivered, flow rate for MN3 goes down to initial level.

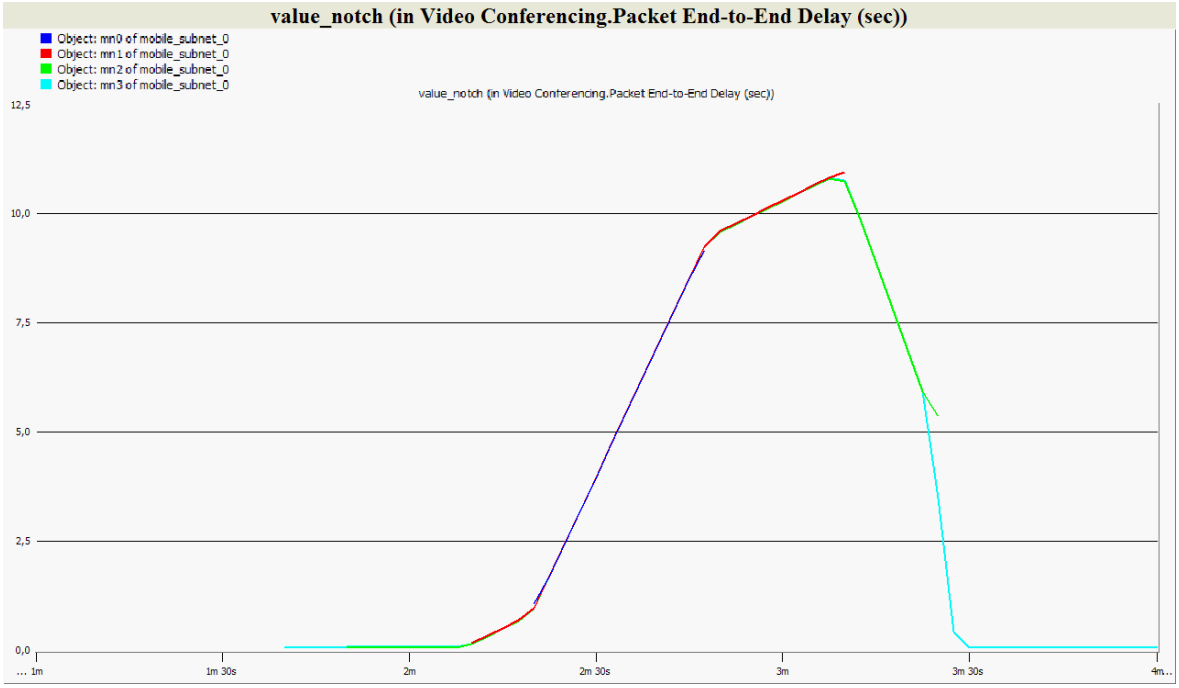


Figure 4.18. Delay

In Figure 4.18, delay values are shown. As expected, all nodes experience the same delay values since all have the same priority. After MN1 flow starts at $130^{th}$ second, the delay values for all nodes start to increase. At $140^{th}$ second, MN0 flow starts. After that, the delay values increase with a higher rate. After MN0 flow ends, the increase rate decreases and after MN1 flow also ends, the delay values for MN2 and MN3 start to decrease. After MN2 flow also ends at $200^{th}$ second, the delay value for MN3 starts to decrease dramatically and goes back to initial values.

### 4.2.7. Scenarios With Mobility

4.2.7.1. No Bandwidth Problem at the Air Interface and the Core Network. This is a reference scenario in which there is no bandwidth problem at the air interface and the core network. All mobile nodes generate flows according to the third pattern as shown in Table 4.5. MN0 moves along the path along WAR1, WAR2, WAR3, WAR4, and WAR5 with a speed of 320 km/h. For each cell, received flow and delay values are shown in the figures below. Following are the key events to mention in the figures:

- *At 105$^{th}$ sec,* MN3, MN6, MN9, MN12, and MN14 Silver/Standard flow starts.
- *At 110$^{th}$ sec,* MN2, MN5, MN8, MN11, and MN13 Silver/Streaming flows start. Since there is no bandwidth problem, other flows and delay values are not affected.
- *At 115$^{th}$ sec,* MN1 and MN10 Gold/Background flows start. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 120$^{th}$ sec,* MN1, MN4, MN7, and MN10 Gold/Standard flows start, respectively. Since the bandwidth is enough for the generated flows, other flows are not affected.
- *At 130$^{th}$ sec,* MN0 Platinum/Streaming flow starts, and this node also moves along the other cells. It first handoffs to cell2 at around 140$^{th}$ sec, then to cell3 at around 170$^{th}$ sec, to cell4 at around 200$^{th}$ sec, and finally to cell5 at around 230$^{th}$ second. At each handoff there is a slight effect observed at received flow, and delay value for MN0 increases since its flow first arrives to its HA, and then redirected to MN0.

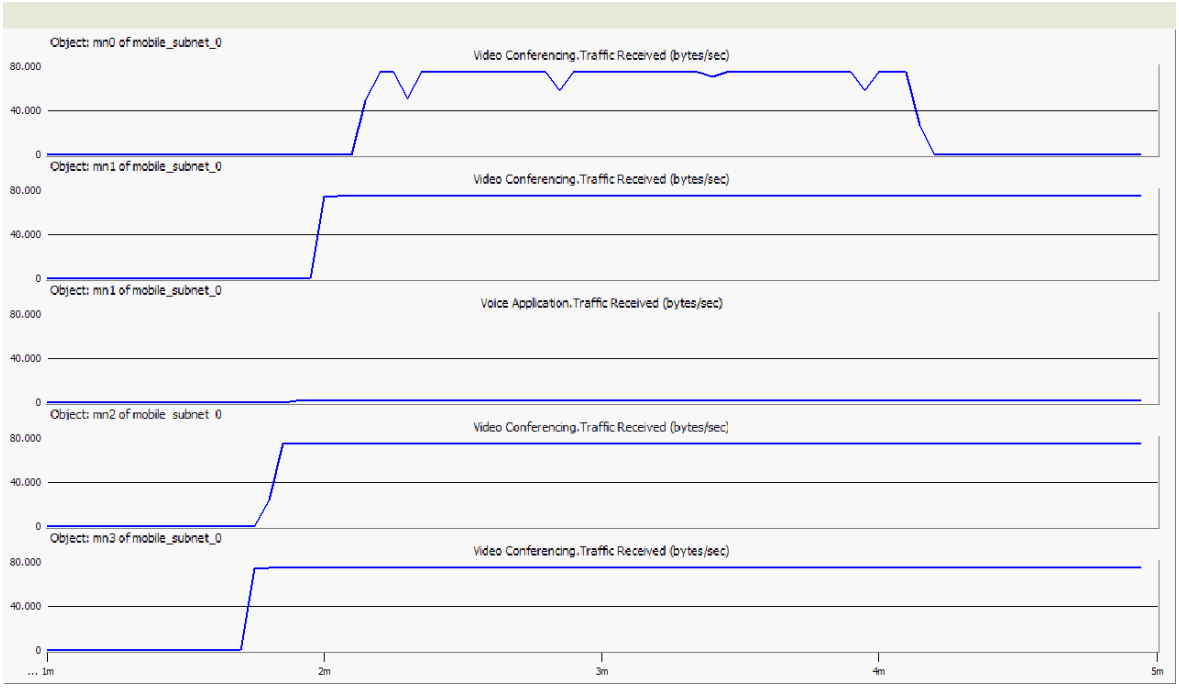


Figure 4.19. Aggregate Data Rate in Cell 1

In Figure 4.19, received flows for MN0, MN1, MN2, and MN3 are shown. Since there is no bandwidth problem, all nodes receive expected flow. MN0 handoffs from cell1 to cell2 at around $140^{th}$ second. It registers to HA over cell2, but during handoffs, until registration is completed, HA can not deliver packets to MN0 over air since MN0 is under cell2. At this period, these packets are dropped and effect is observed at graph.

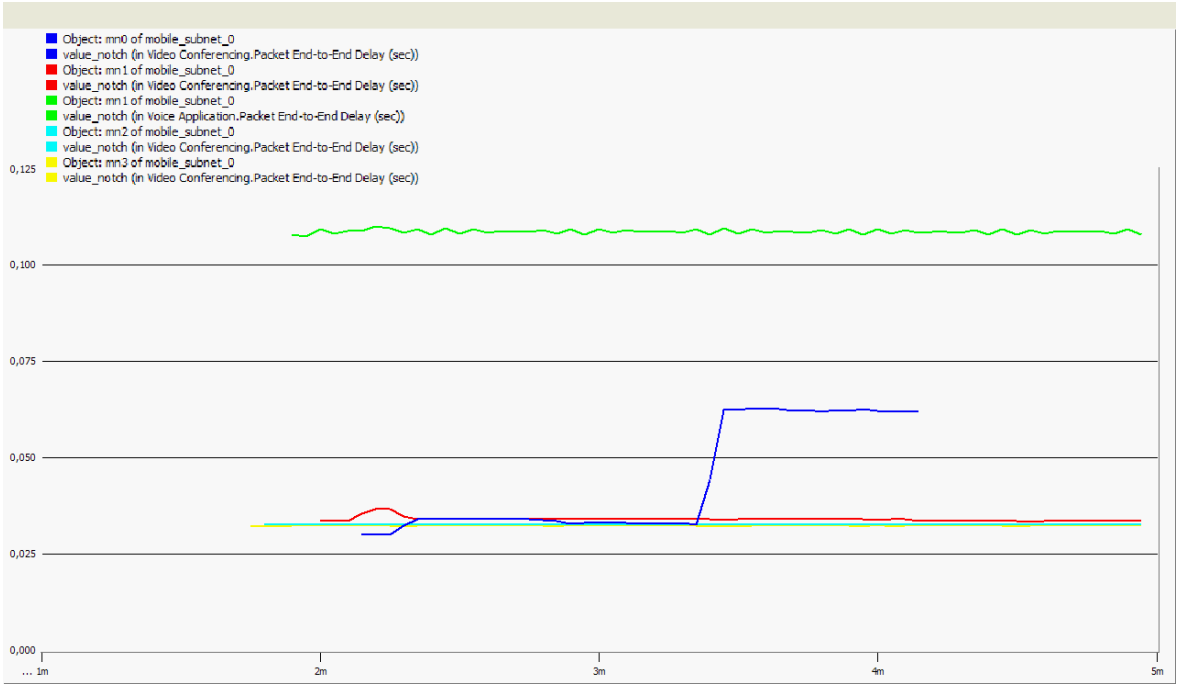


Figure 4.20. Delay in Cell 1

In Figure 4.20, delay values for MN0, MN1, MN2, and MN3 are shown. When MN0 handoffs to cell2 at around $140^{th}$ second, delay value for MN0 increases from 0.03 to 0.034. This is due to the tunneled flow from HA to FA. This creates additional delay time for the flow.

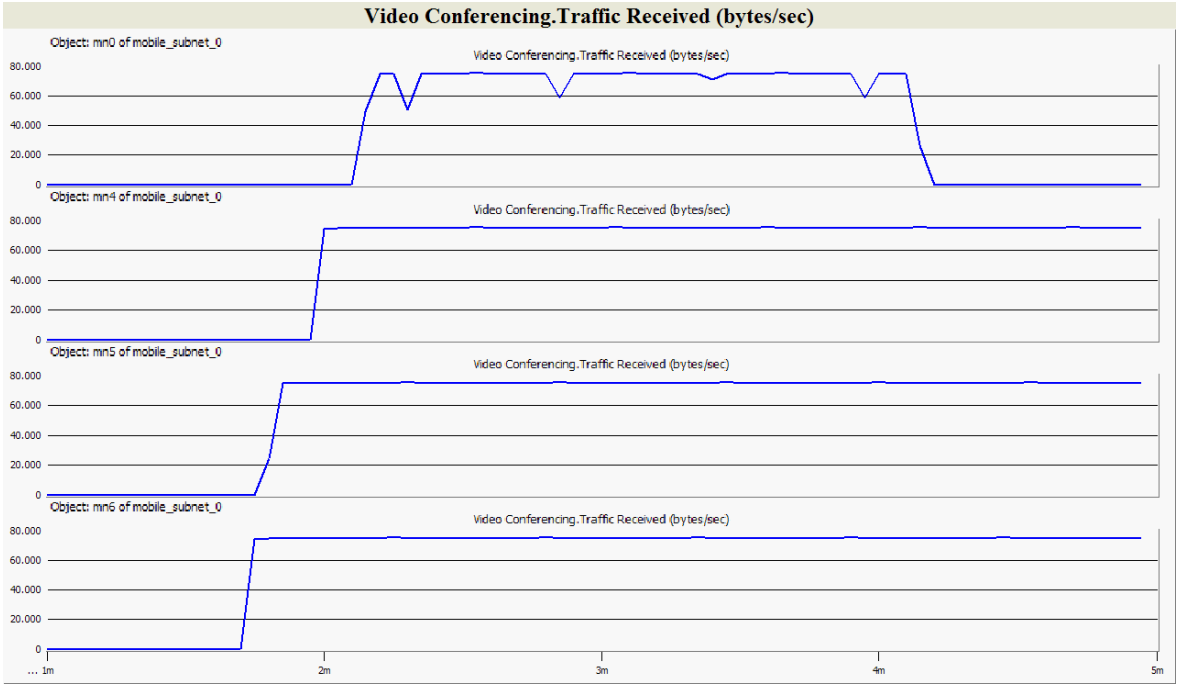


Figure 4.21. Aggregate Data Rate in Cell 2

In Figure 4.21, received flows for MN0, MN4, MN5, and MN6 are shown. Since there is no bandwidth problem, all nodes receive expected flow. MN0 handoffs from cell2 to cell3 at around $170^{th}$ second. It registers to HA over cell3, but during handoffs, until registration is completed, HA delivers the packets to cell2. At this period, these packets are dropped and effect is observed at graph.

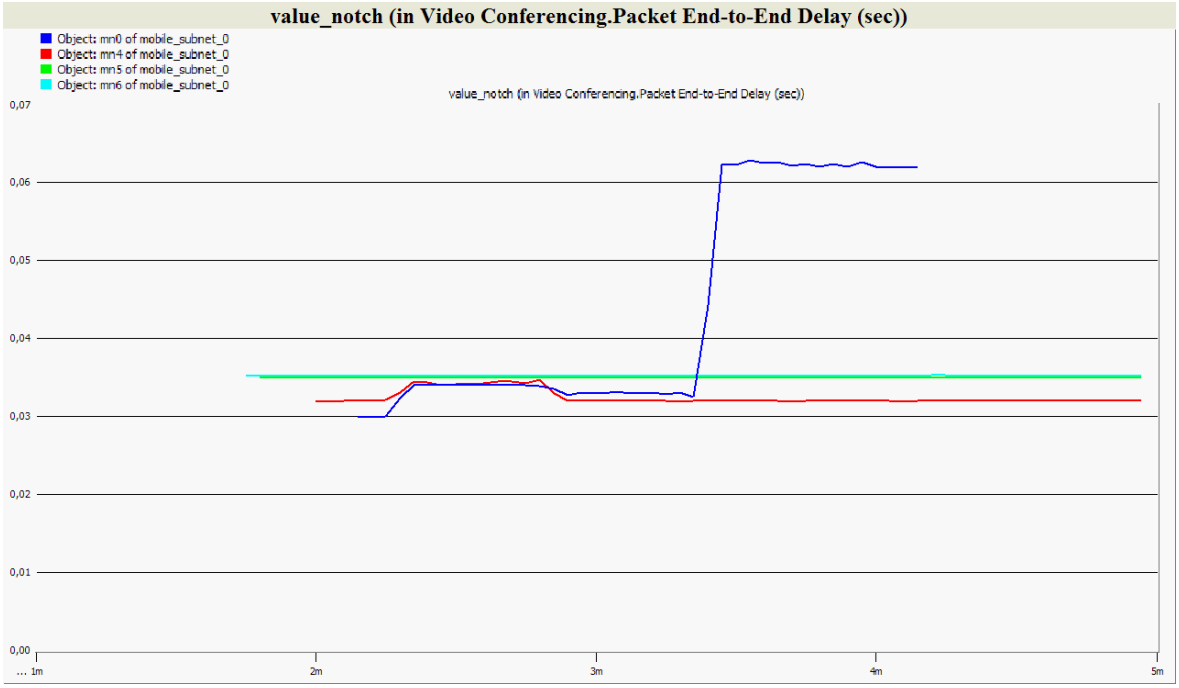


Figure 4.22. Delay in Cell 2

In Figure 4.22, delay values for MN0, MN4, MN5, and MN6 are shown. When MN0 handoffs to cell3 from cell2 at around $170^{th}$ second, we do not observe change in delay value for MN0. This is since the delay between cell1, cell2 and cell1, cell3 is the same. Instead of cell2, HA starts tunneling the flow to cell3.

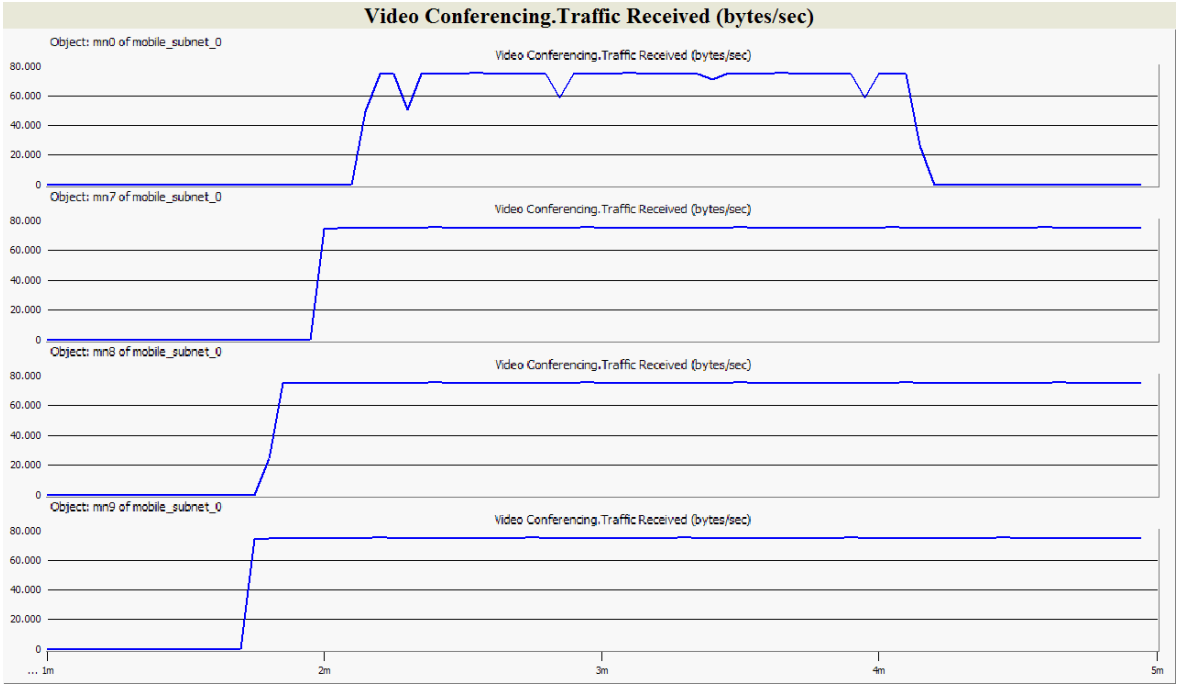


Figure 4.23. Aggregate Data Rate in Cell 3

In Figure 4.23, received flows for MN0, MN7, MN8, and MN9 are shown. Since there is no bandwidth problem, all nodes receive expected flow. MN0 handoffs from cell3 to cell4 at around $200^{th}$ second. It registers to HA over cell4, but during handoffs, until registration is completed, HA delivers the packets to cell3. At this period, these packets are dropped and effect is observed at graph.

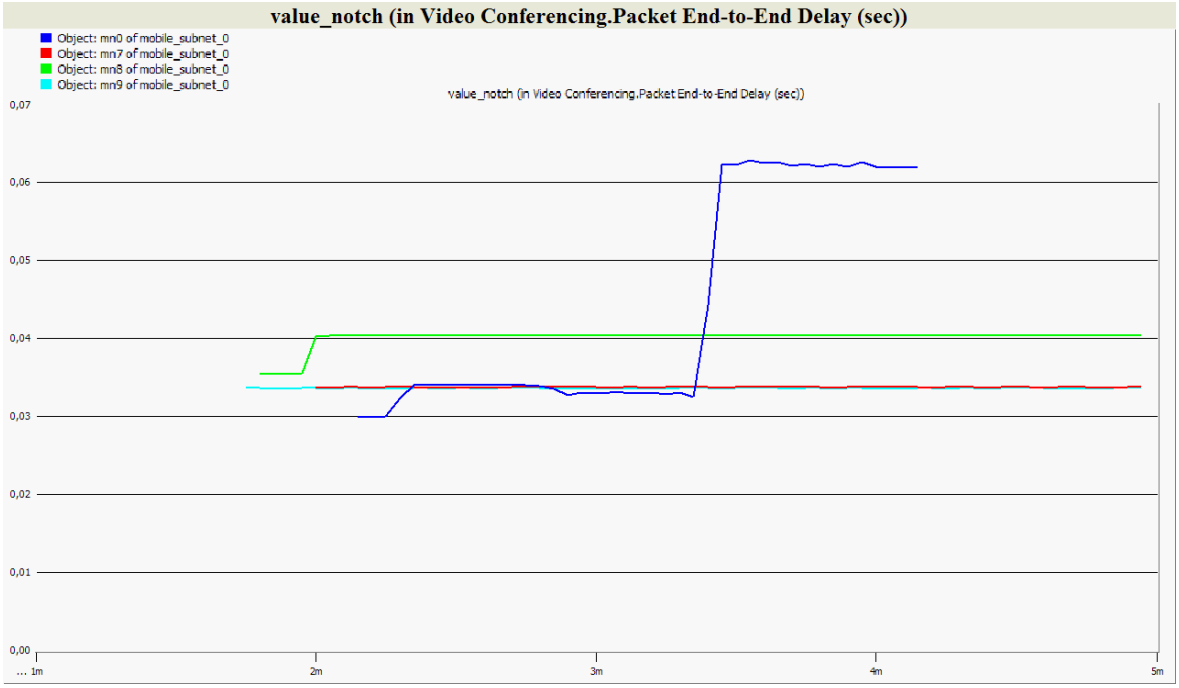


Figure 4.24. Delay in Cell 3

In Figure 4.24, delay values for MN0, MN7, MN8, and MN9 are shown. When MN0 handoffs to cell4 from cell3 at around $200^{th}$ second, delay value for MN0 increases from 0.034 to 0.062. This is due to the tunneled flow from HA to FA now goes over several other nodes at the backbone. WAR4 is connected to backbone over LER11 but WAR1 is connected over LER12.

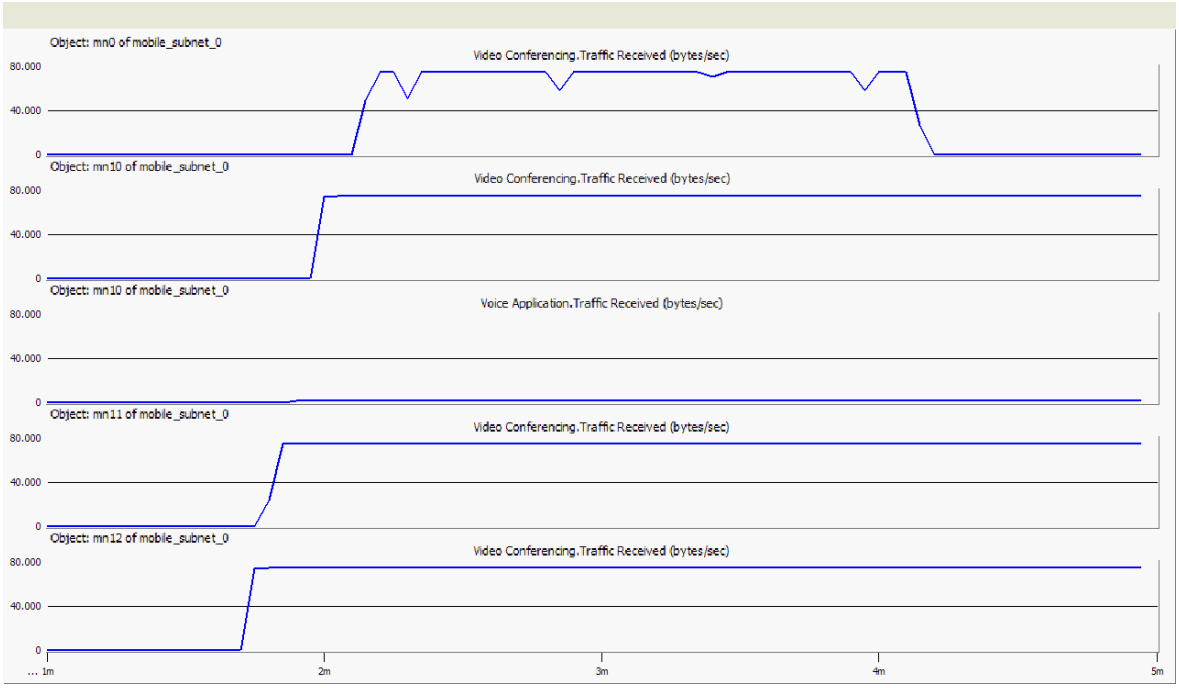


Figure 4.25. Aggregate Data Rate in Cell 4

In Figure 4.25, received flows for MN0, MN10, MN11, and MN12 are shown. Since there is no bandwidth problem, all nodes receive expected flow. MN0 handoffs from cell4 to cell5 at around $230^{th}$ second. It registers to HA over cell5, but during handoffs, until registration is completed, HA delivers the packets to cell4. At this period, these packets are dropped and effect is observed at graph.

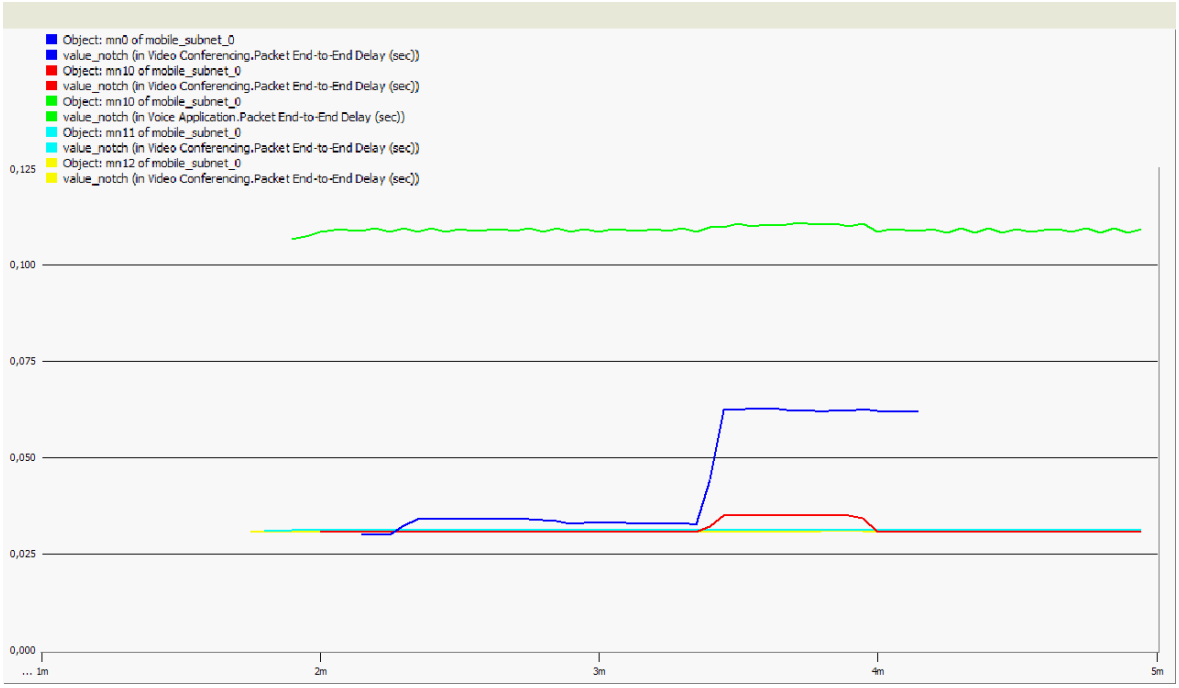


Figure 4.26. Delay in Cell 4

In Figure 4.26, delay values for MN0, MN10, MN11, and MN12 are shown. When MN0 handoffs to cell5 from cell4 at around $230^{th}$ second, we do not observe change in delay value for MN0. This is since the delay between cell1, cell4 and cell1, cell5 is the same. Instead of cell4, HA starts tunneling the flow to cell5.

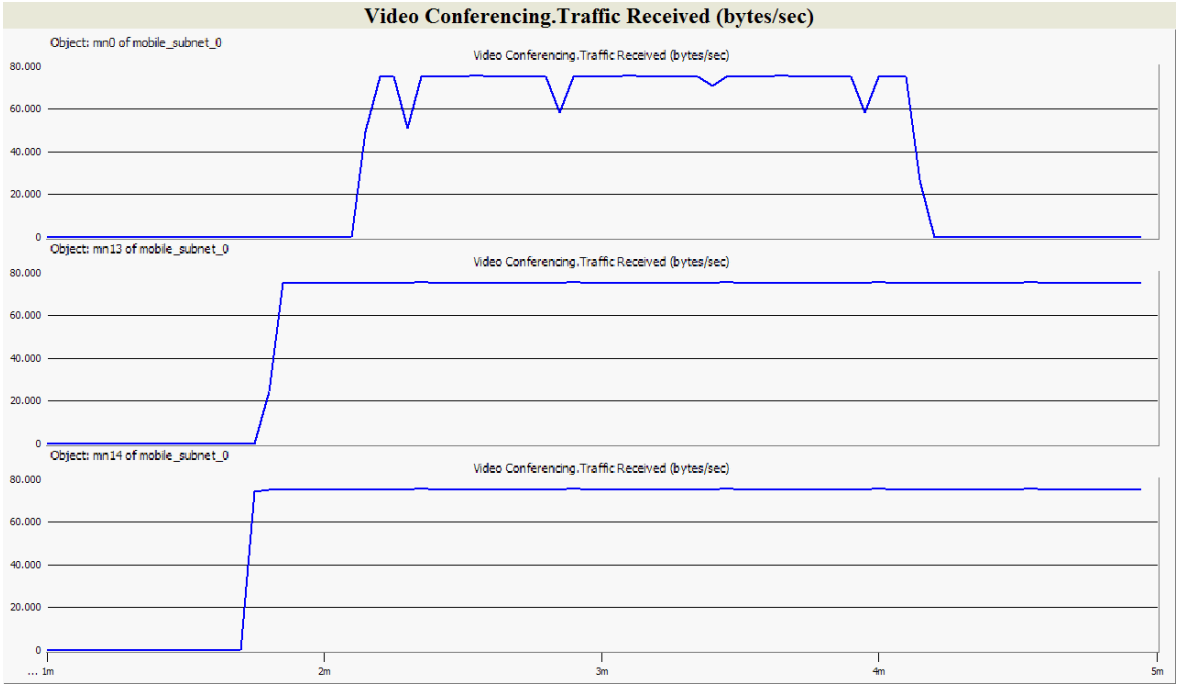


Figure 4.27. Aggregate Data Rate in Cell 5

In Figure 4.27, received flows for MN0, MN13, and MN14 are shown. Since there is no bandwidth problem, all nodes receive expected flow. MN0 handoffs from cell4 to cell5 at around $230^{th}$ second. It registers to HA over cell5, but during handoffs, until registration is completed, HA delivers the packets to cell4. At this period, these packets are dropped and effect is observed at graph.

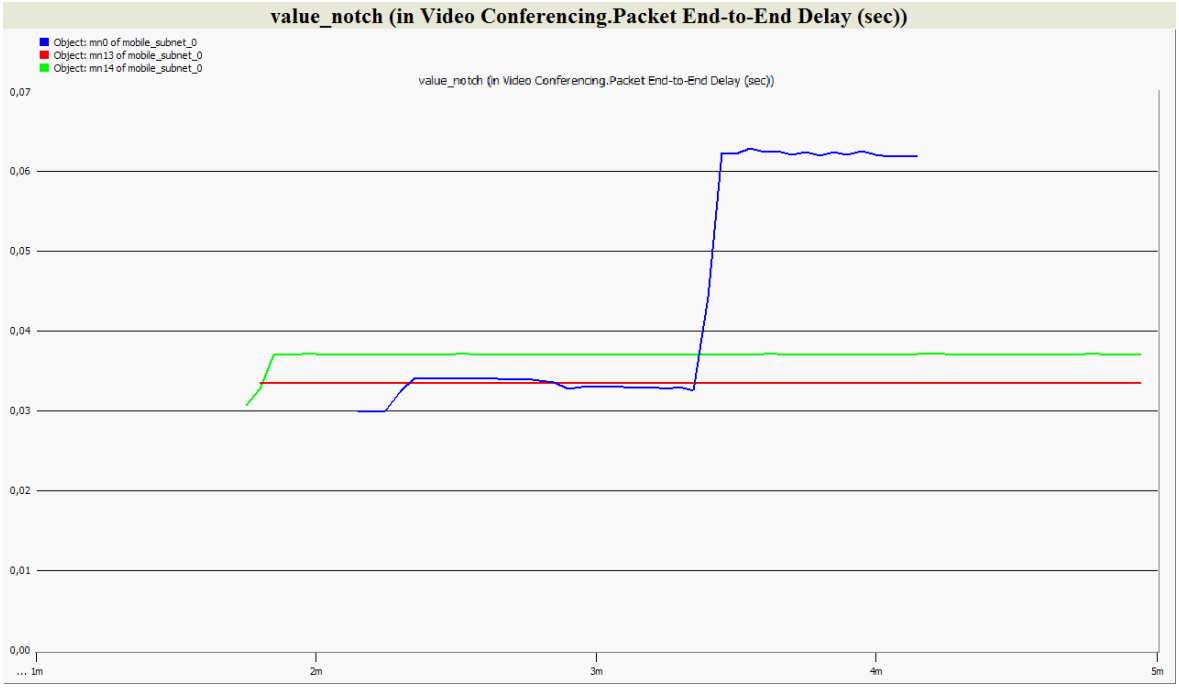


Figure 4.28. Delay in Cell 5

In Figure 4.28, delay values for MN0, MN13, and MN14 are shown. When MN0 handoffs to cell5 from cell4 at around $230^{th}$ second, we do not observe change in delay value for MN0. This is since the delay between cell1, cell4 and cell1, cell5 is the same. Instead of cell4, HA starts tunneling the flow to cell5.

### 4.2.8. Low Bandwidth at the Air Interface (Priority Scheduler)

In this scenario, the bandwidth for the air interface is 2.4 MBps. There is no bandwidth problem for the core network. Priority Scheduler algorithm is used for MAC scheduler. All mobile nodes generate flows according to the third pattern as shown in Table 4.5. For each cell received flow and delay values are shown at following figures. Following are the key events to mention in the figures:

- *At 130$^{th}$ sec,* MN0 flow starts, and MN3 starts starving and MN2 flow is effected.
- *At 145$^{th}$ sec,* MN0 handovers to cell2, MN2 and MN3 starts to receive flow since now there is available bandwidth, but MN5 and MN6 are affected. Since MN0 has the highest priority it still receives all flow.
- *At 170$^{th}$ sec,* MN0 handovers to cell3, MN5 and MN6 starts to receive flow since now there is available bandwidth, but MN8 and MN9 are affected.
- *At 195$^{th}$ sec,* MN0 handovers to cell4, MN8 and MN9 starts to receive flow since now there is available bandwidth, but MN11 and MN12 are affected.
- *At 220$^{th}$ sec,* MN0 handovers to cell5, since there is two nodes there only one node is affected.

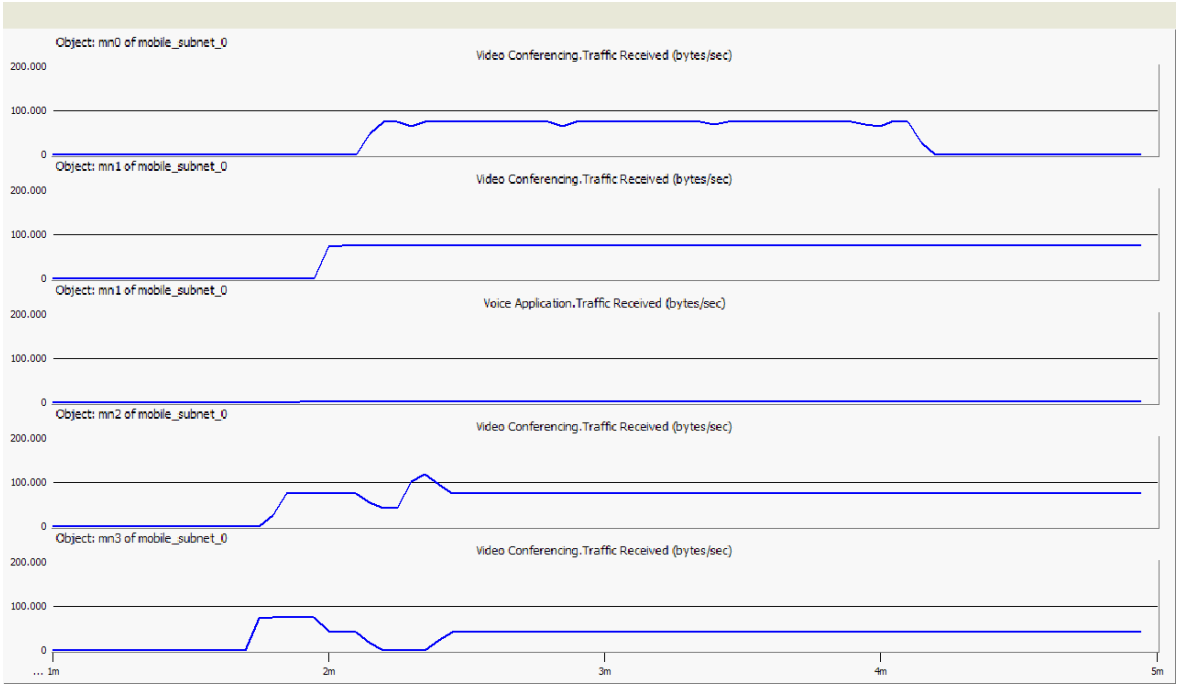


Figure 4.29. Aggregate Data Rate in Cell 1

In Figure 4.29, received flows for MN0, MN1, MN2, and MN3 are shown. At $120^{th}$ second, MN1 Gold/Standard flow starts which affects MN3 flow. When at $130^{th}$ second, MN0 flow starts, MN3 starts starving and can not receive any flow. MN2 is also affected. MN0 handoffs from cell1 to cell2 at around $140^{th}$ second. After that, with the available bandwidth, first MN2 receives all queued packets which results in the dramatic increase at the graph.

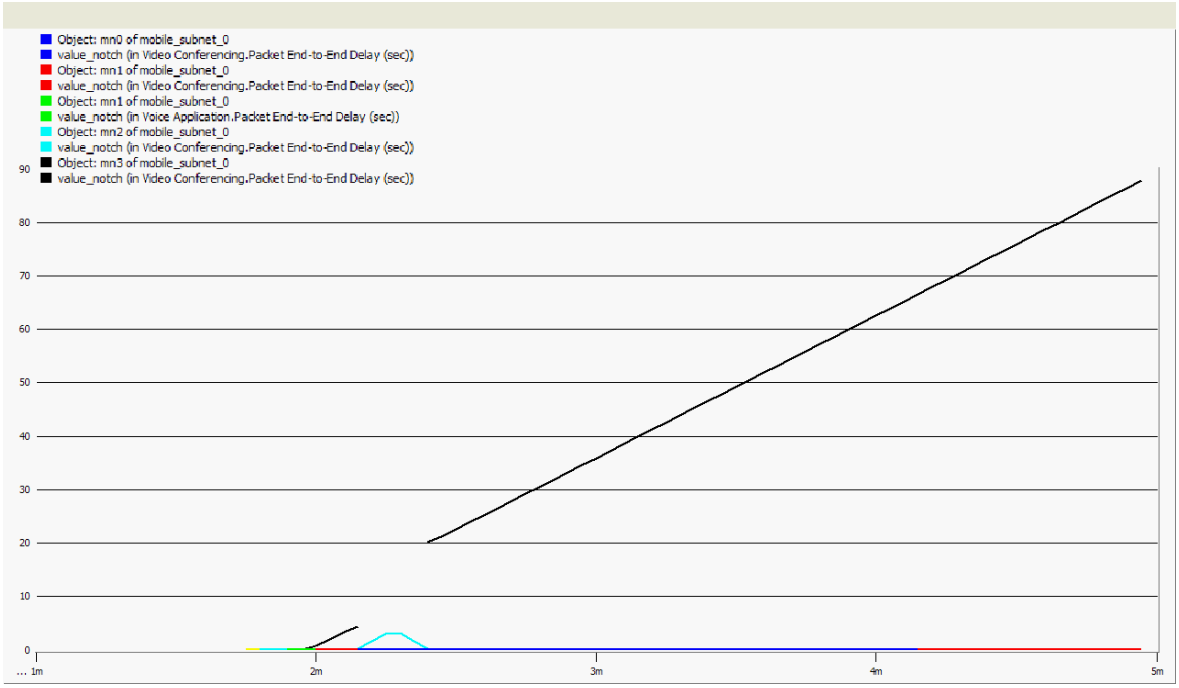


Figure 4.30. Delay in Cell 1

In Figure 4.30, delay values for MN0, MN1, MN2, and MN3 are shown. At $120^{th}$ second, MN1 Gold/Standard flow starts and delay for MN3 starts to increase. When at $130^{th}$ second, MN0 flow starts, MN3 can not receive any flow, hence no delay values are shown. For MN2, the delay values start to increase until MN0 handoffs to cell2 from cell1 at around $140^{th}$ second. When MN3 starts to receive flow after MN0 handoffs, its delay values also continue to increase since there is still not enough bandwidth.

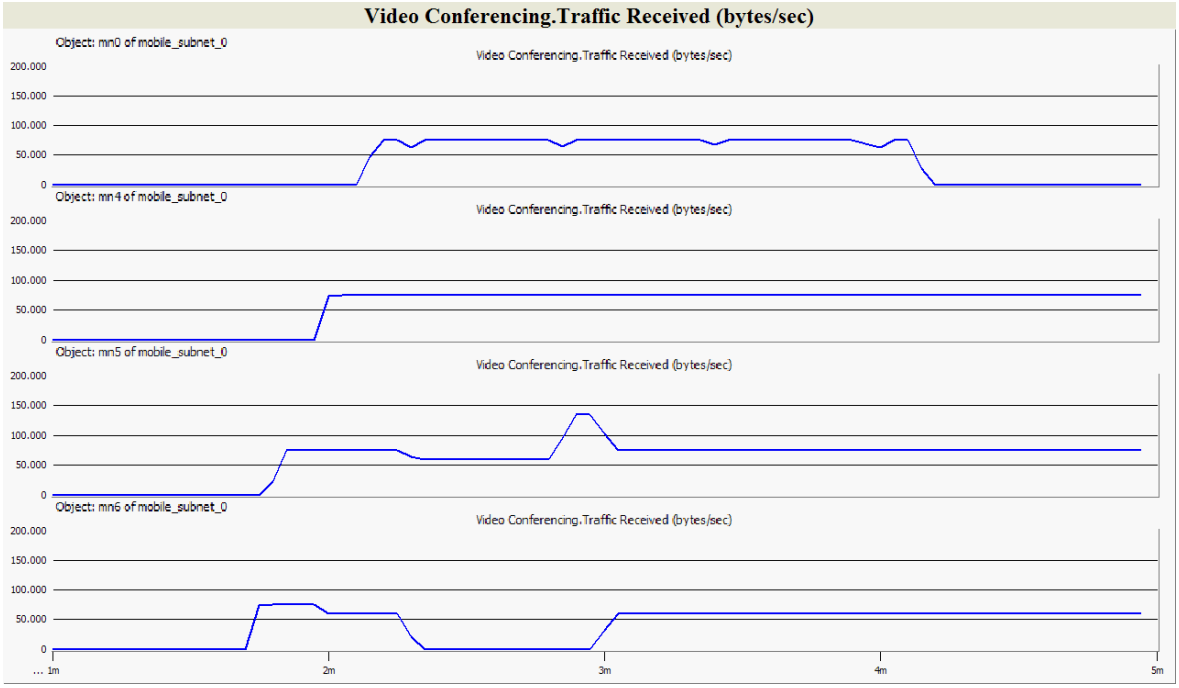


Figure 4.31. Aggregate Data Rate in Cell 2

In Figure 4.31, received flows for MN0, MN4, MN5, and MN6 are shown. At $120^{th}$ second, MN4 Gold/Standard flow starts which affects MN6 flow. When MN0 handoffs from cell1 to cell2 at around $140^{th}$ second, MN6 starts starving and can not receive any flow. MN5 is also affected. MN0 handoffs from cell2 to cell3 at around $170^{th}$ second. After that, with the available bandwidth, first MN5 receives all queued packets which results in the dramatic increase at the graph.

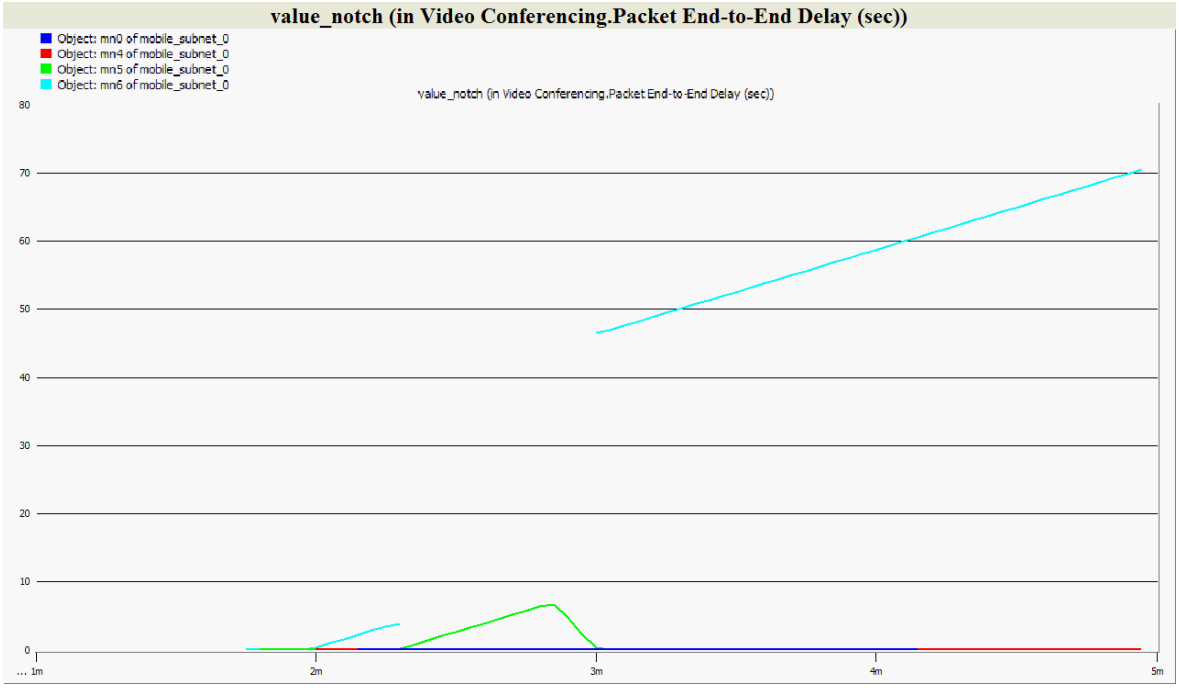


Figure 4.32. Delay in Cell 2

In Figure 4.32, delay values for MN0, MN4, MN5, and MN6 are shown. At $120^{th}$ second, MN4 Gold/Standard flow starts and delay for MN3 starts to increase. When MN0 handoffs from cell1 to cell2 at around $140^{th}$ second, MN6 can not receive any flow, hence no delay values are shown. For MN5, the delay values start to increase until MN0 handoffs to cell3 from cell2 at around $170^{th}$ second. When MN6 starts to receive flow after MN0 handoffs, its delay values also continue to increase since there is still not enough bandwidth.

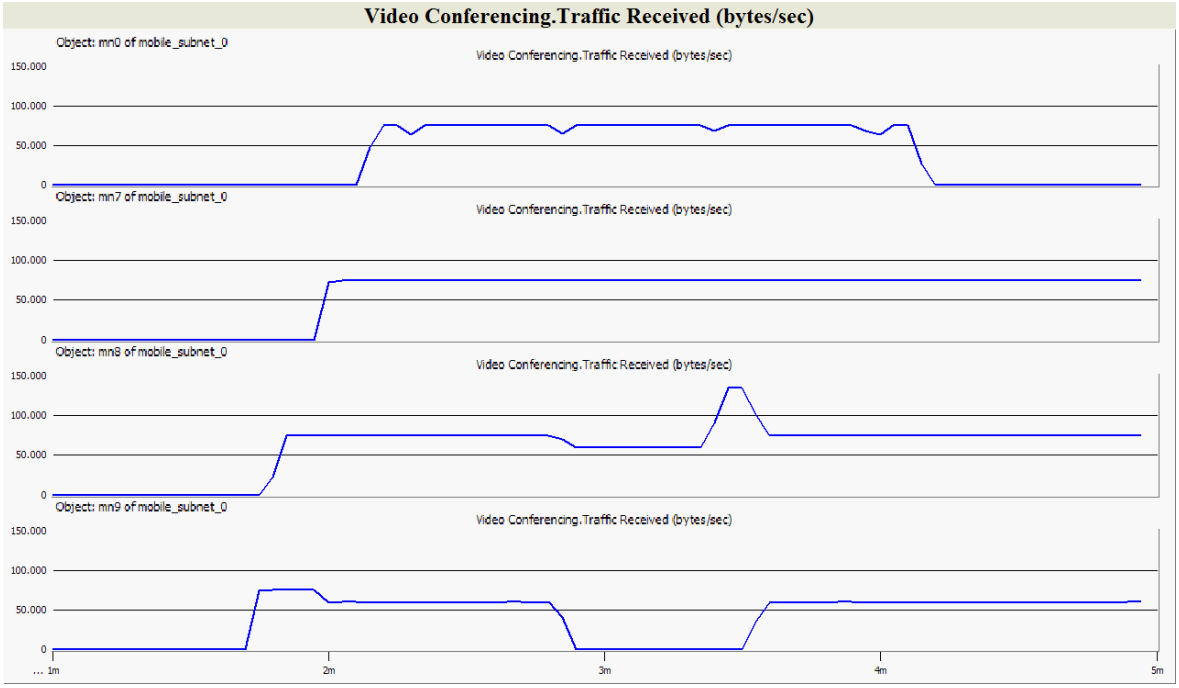


Figure 4.33. Aggregate Data Rate in Cell 3

In Figure 4.33, received flows for MN0, MN7, MN8, and MN9 are shown. At $120^{th}$ second, MN7 Gold/Standard flow starts which affects MN3 flow. When MN0 handoffs from cell2 to cell3 at around $170^{th}$ second, MN9 starts starving and can not receive any flow. MN8 is also affected. MN0 handoffs from cell3 to cell4 at around $200^{th}$ second. After that, with the available bandwidth, first MN8 receives all queued packets which results in the dramatic increase at the graph.

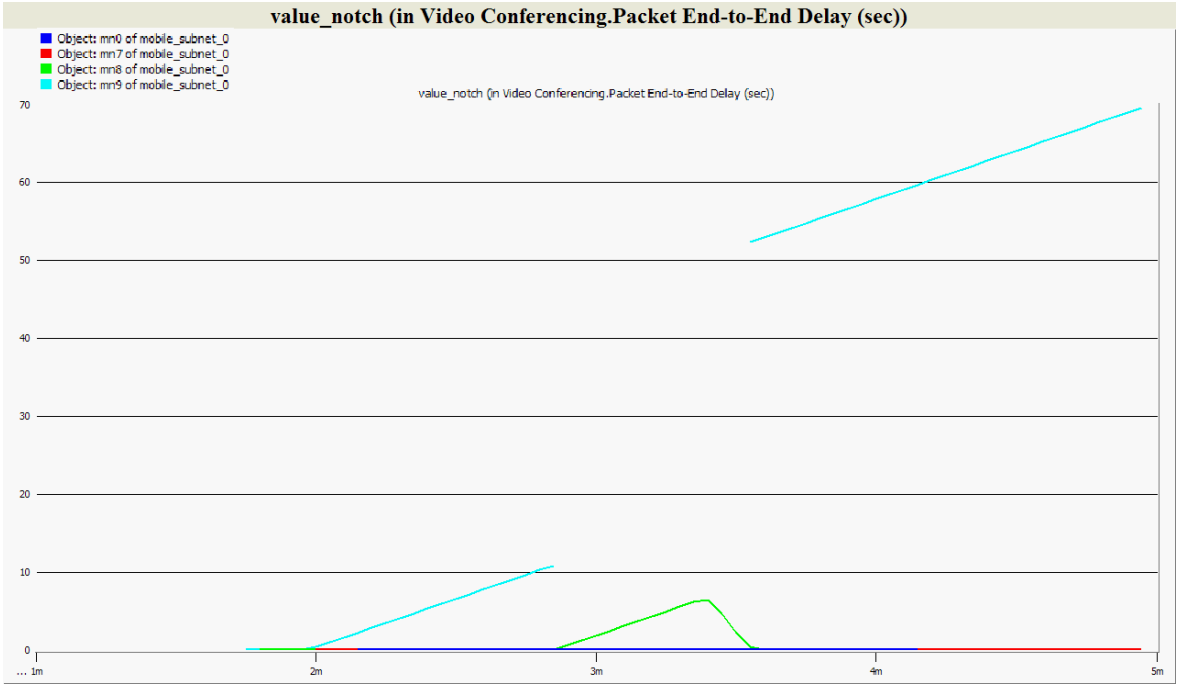


Figure 4.34. Delay in Cell 3

In Figure 4.34, delay values for MN0, MN7, MN8, and MN9 are shown. At $120^{th}$ second, MN7 Gold/Standard flow starts and delay for MN9 starts to increase. When MN0 handoffs from cell2 to cell3 at around $170^{th}$ second, MN9 can not receive any flow, hence no delay values are shown. For MN8, the delay values start to increase until MN0 handoffs to cell4 from cell3 at around $200^{th}$ second. When MN9 starts to receive flow after MN0 handoffs, its delay values also continue to increase since there is still not enough bandwidth.

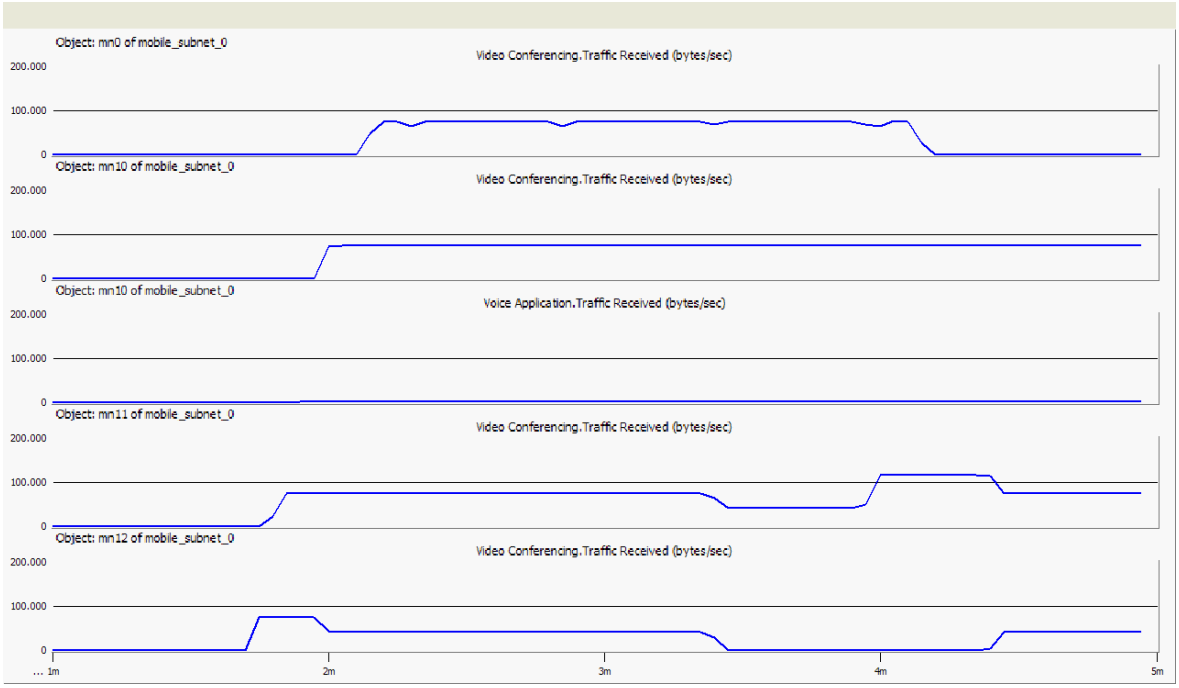


Figure 4.35. Aggregate Data Rate in Cell 4

In Figure 4.35, received flows for MN0, MN10, MN11, and MN12 are shown. At $120^{th}$ second, MN10 Gold/Standard flow starts which affects MN12 flow. When MN0 handoffs from cell3 to cell4 at around $200^{th}$ second, MN12 starts starving and can not receive any flow. MN11 is also affected. MN0 handoffs from cell4 to cell5 at around $230^{th}$ second. After that, with the available bandwidth, first MN12 receives all queued packets which results in the dramatic increase at the graph.

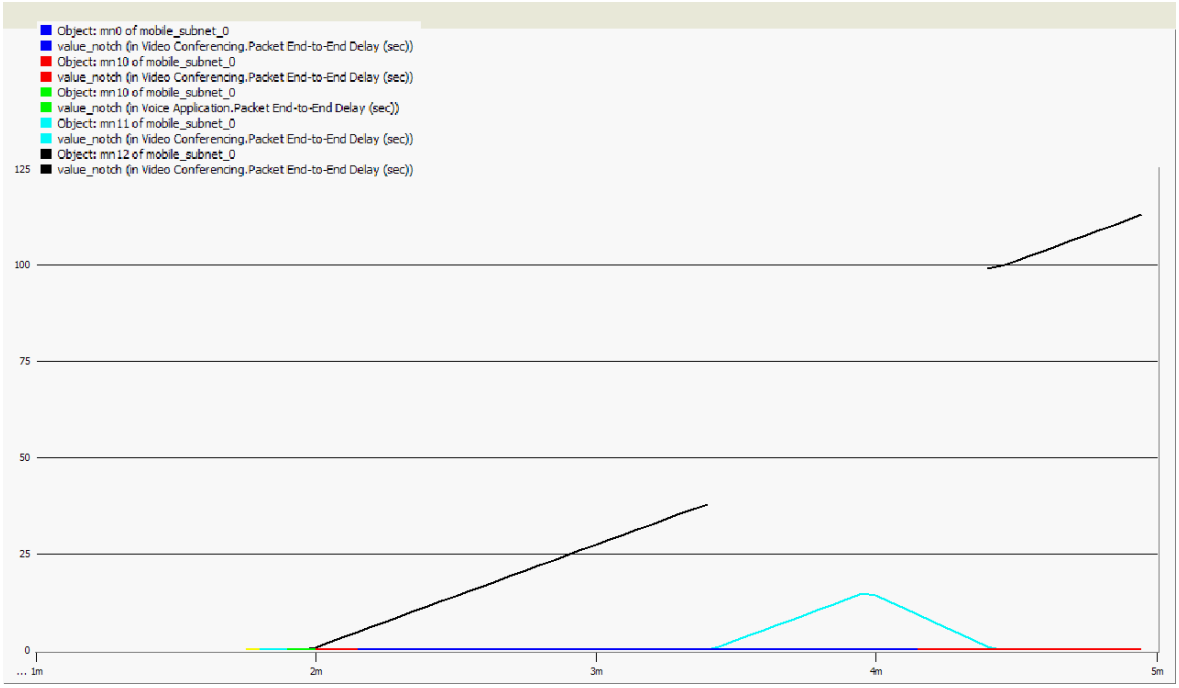


Figure 4.36. Delay in Cell 4

In Figure 4.36, delay values for MN0, MN10, MN11, and MN12 are shown. At $120^{th}$ second, MN10 Gold/Standard flow starts and delay for MN12 starts to increase. When MN0 handoffs from cell3 to cell4 at around $200^{th}$ second, MN12 can not receive any flow, hence no delay values are shown. For MN11, the delay values start to increase until MN0 handoffs to cell5 from cell4 at around $230^{th}$ second. When MN12 starts to receive flow after MN0 handoffs, its delay values also continue to increase since there is still not enough bandwidth.

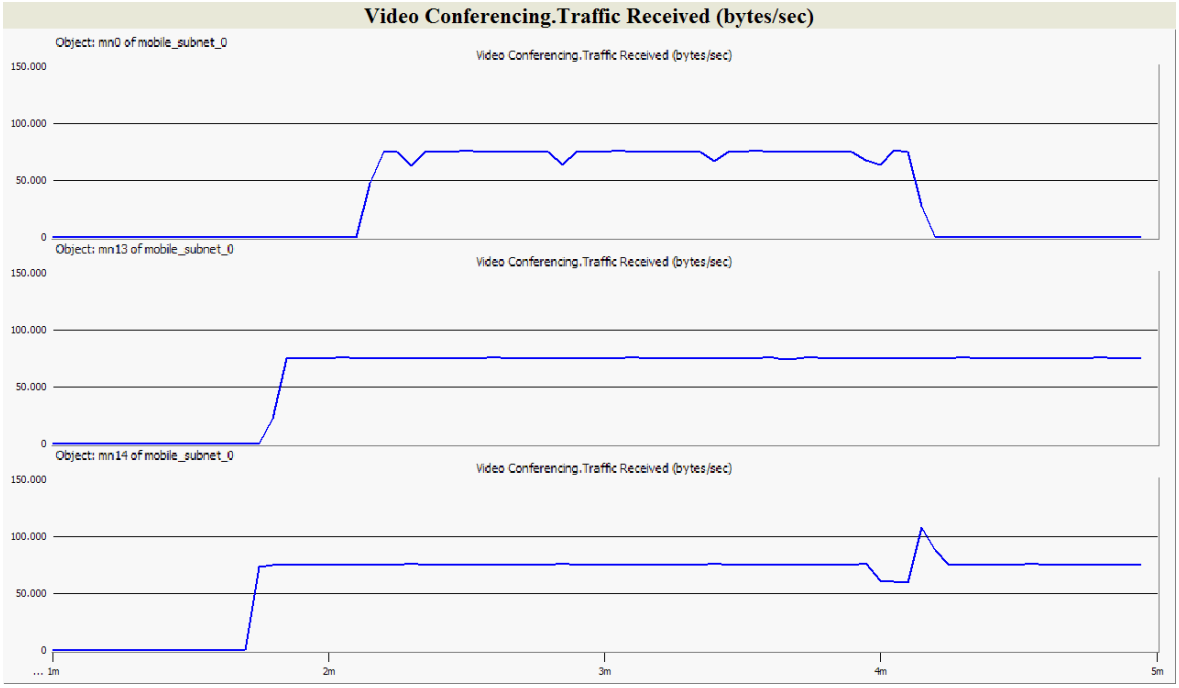


Figure 4.37. Aggregate Data Rate in Cell 5

In Figure 4.37, received flows for MN0, MN13, and MN14 are shown. When MN0 handoffs from cell4 to cell5 at around $230^{th}$ second, MN14 is affected. When the flow for MN0 ends at $250^{th}$ second, with the available bandwidth, MN14 receives all queued packets which results in the dramatic increase at the graph.

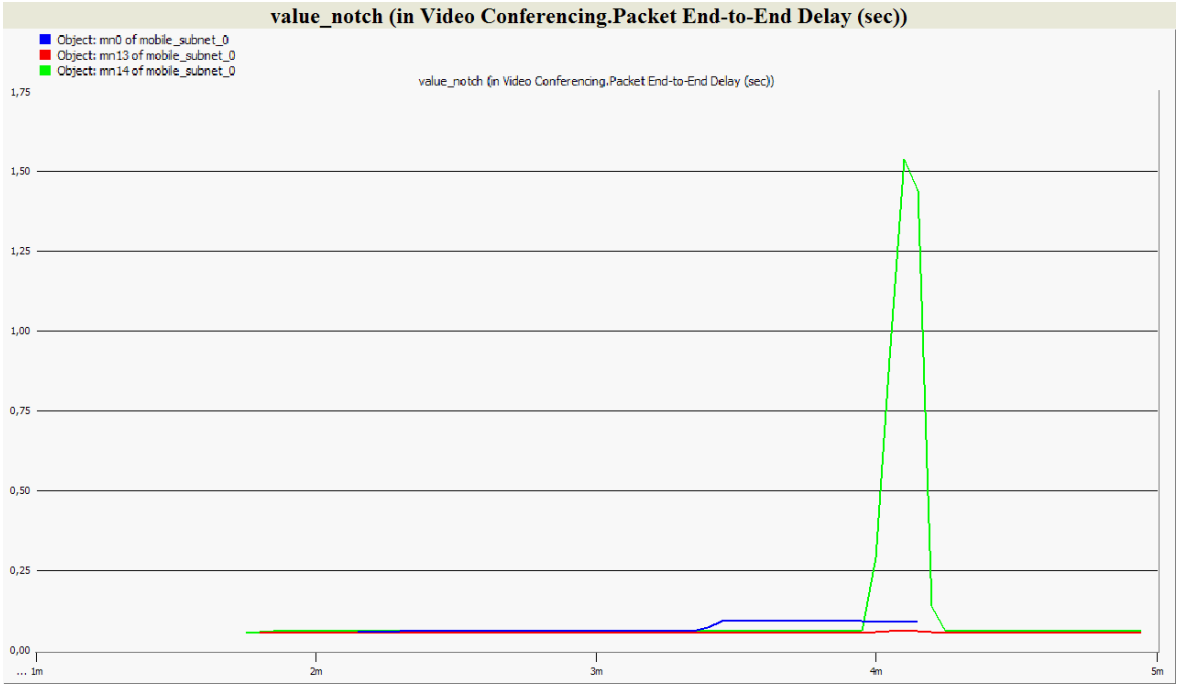


Figure 4.38. Delay in Cell 5

In Figure 4.38, delay values for MN0, MN13, and MN14 are shown. When MN0 handoffs from cell4 to cell5 at around $230^{th}$ second, the delay value for MN14 starts to increase until MN0 flow ends at $250^{th}$ second. Then with the available bandwidth delay values decrease to initial value.

### 4.2.9. Low Bandwidth at the Air Interface (Priority Scheduler with Reservation)

In this scenario, the bandwidth for the air interface is 2.4 MBps. There is no bandwidth problem for the core network. Priority Scheduler with Reservation algorithm is used for MAC scheduler. All mobile nodes generate flows according to the third pattern as shown in Table 4.5. For each cell received flow and delay values are shown at following figures. Following are the key events to mention in the figures:

- *At 130$^{th}$ sec,* mn0 flow starts, and since reservation is used nodes gets flow according to their QoS values.
- *At 145$^{th}$ sec,* mn0 handovers to cell2, mn2 and mn3 starts to receive flow since now there is available bandwidth, but mn5 and mn6 are affected. Since mn0 has the highest priority it still receives all flow.
- *At 170$^{th}$ sec,* mn0 handovers to cell3, mn5 and mn6 starts to receive flow since now there is available bandwidth, but mn8 and mn9 are affected.
- *At 195$^{th}$ sec,* mn0 handovers to cell4, mn8 and mn9 starts to receive flow since now there is available bandwidth, but mn11 and mn12 are affected.
- *At 220$^{th}$ sec,* mn0 handovers to cell5, since there is two nodes there only one node is affected.

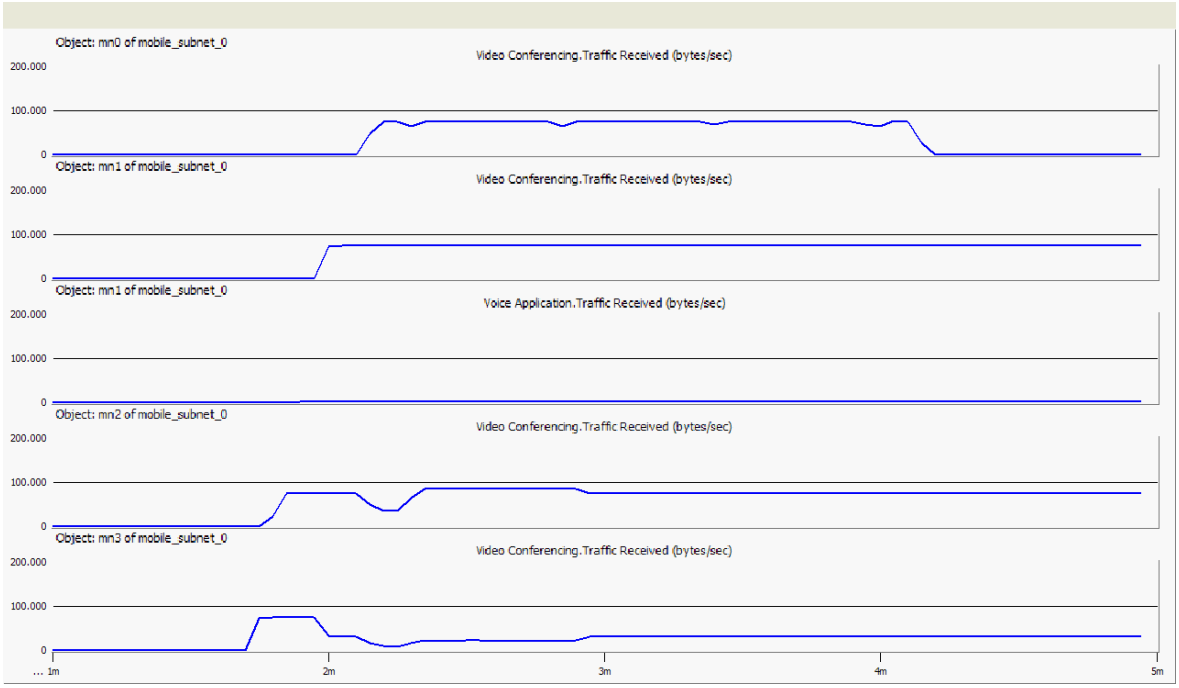


Figure 4.39. Aggregate Data Rate in Cell 1

In Figure 4.39, received flows for MN0, MN1, MN2, and MN3 are shown. When at $130^{th}$ second, MN0 flow starts, MN3 and MN2 are affected but get flow according to the reservation parameters. MN0 handoffs from cell1 to cell2 at around $140^{th}$ second. After that, with the available bandwidth, the received flow for MN2 and MN3 increases. After all the queued packets are delivered to MN2, the available bandwidth is also used to deliver packets to MN3.

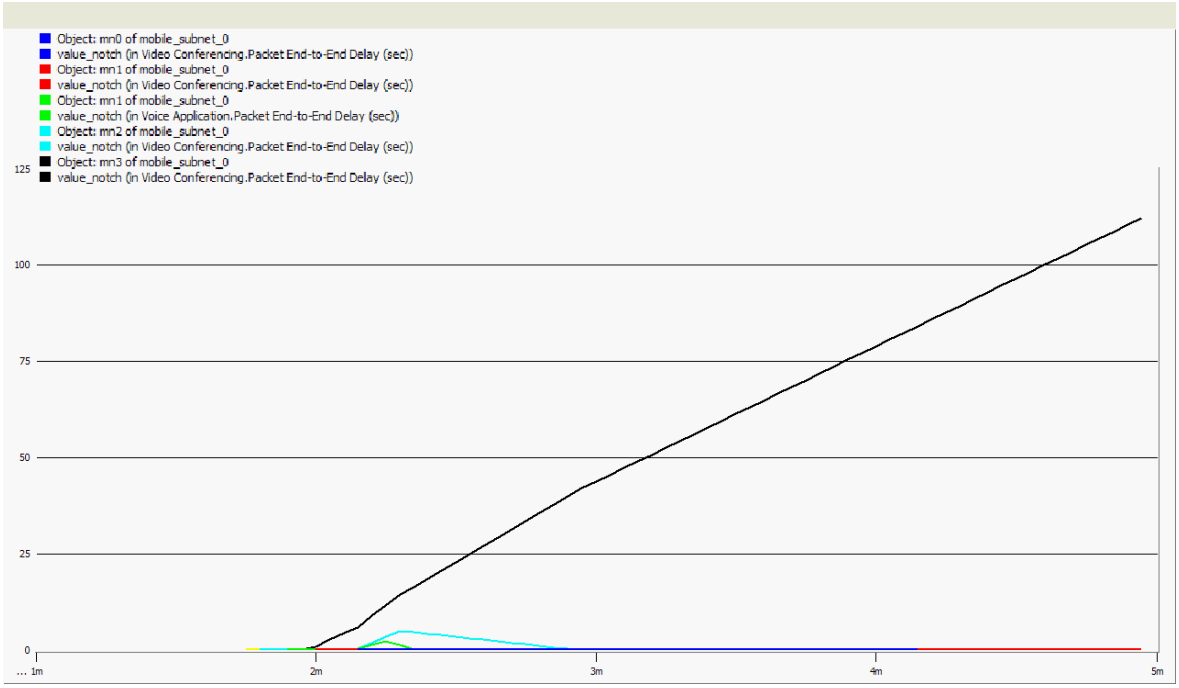


Figure 4.40. Delay in Cell 1

In Figure 4.40, delay values for MN0, MN1, MN2, and MN3 are shown. At $120^{th}$ second, MN1 Gold/Standard flow starts and delay for MN3 starts to increase. When at $130^{th}$ second, MN0 flow starts, delay value for MN2 also starts to increase until MN0 handoffs to cell2 from cell1 at around $140^{th}$ second.

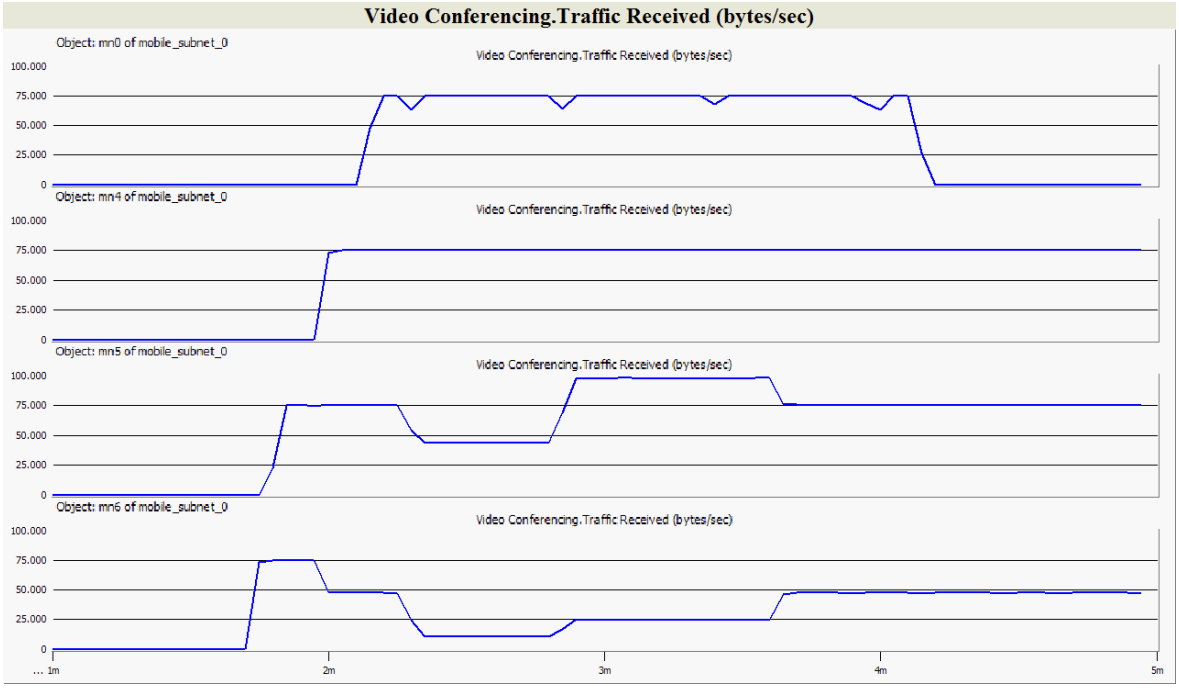


Figure 4.41. Aggregate Data Rate in Cell 2

In Figure 4.41, received flows for MN0, MN4, MN5, and MN6 are shown. When MN0 handoffs from cell1 to cell2 at around $140^{th}$ second, MN5 and MN6 are affected but get flow according to the reservation parameters. MN0 handoffs from cell2 to cell3 at around $170^{th}$ second. After all the queued packets are delivered to MN5, the available bandwidth is also used to deliver packets to MN6.

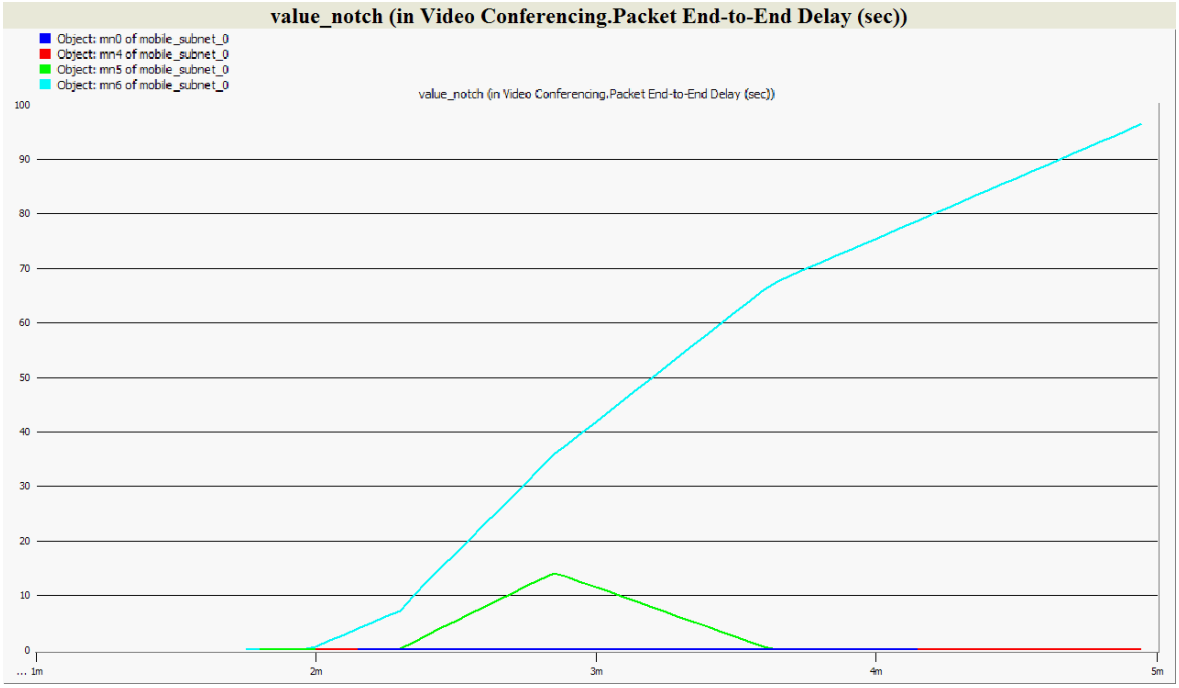


Figure 4.42. Delay in Cell 2

In Figure 4.42, delay values for MN0, MN4, MN5, and MN6 are shown. At $120^{th}$ second, MN4 Gold/Standard flow starts and delay for MN6 starts to increase. When at $130^{th}$ second, MN0 flow starts, delay value for MN5 also starts to increase until MN0 handoffs to cell2 from cell1 at around $140^{th}$ second. Then delay value for MN5 starts to decrease but since there is still inadequate bandwidth, delay value for MN6 continues to increase until the simulation ends.

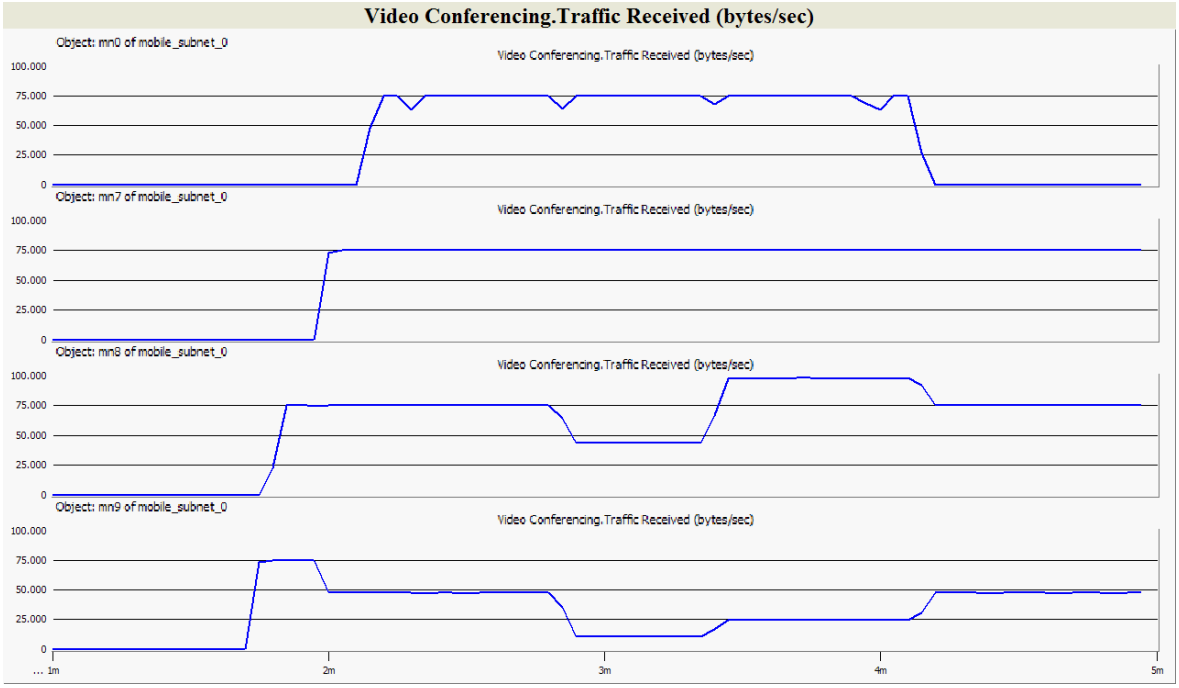


Figure 4.43. Aggregate Data Rate in Cell 3

In Figure 4.43, received flows for MN0, MN7, MN8, and MN9 are shown. When MN0 handoffs from cell2 to cell3 at around $170^{th}$ second, MN8 and MN9 are affected but get flow according to the reservation parameters. MN0 handoffs from cell3 to cell4 at around $200^{th}$ second. After all the queued packets are delivered to MN8, the available bandwidth is also used to deliver packets to MN9.

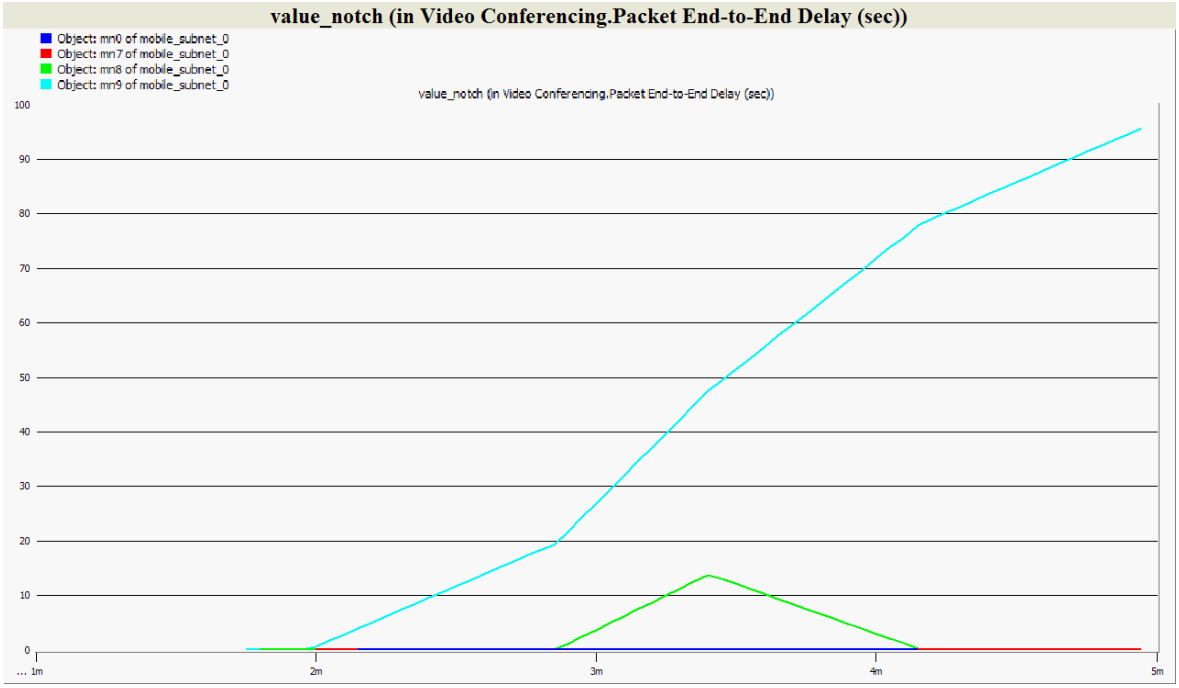


Figure 4.44. Delay in Cell 3

In Figure 4.44, delay values for MN0, MN7, MN8, and MN9 are shown. At $120^{th}$ second, MN7 Gold/Standard flow starts and delay for MN9 starts to increase. When at $130^{th}$ second, MN0 flow starts, delay value for MN8 also starts to increase until MN0 handoffs to cell3 from cell2 at around $170^{th}$ second. Then delay value for MN8 starts to decrease but since there is still inadequate bandwidth, delay value for MN9 continues to increase until the simulation ends.

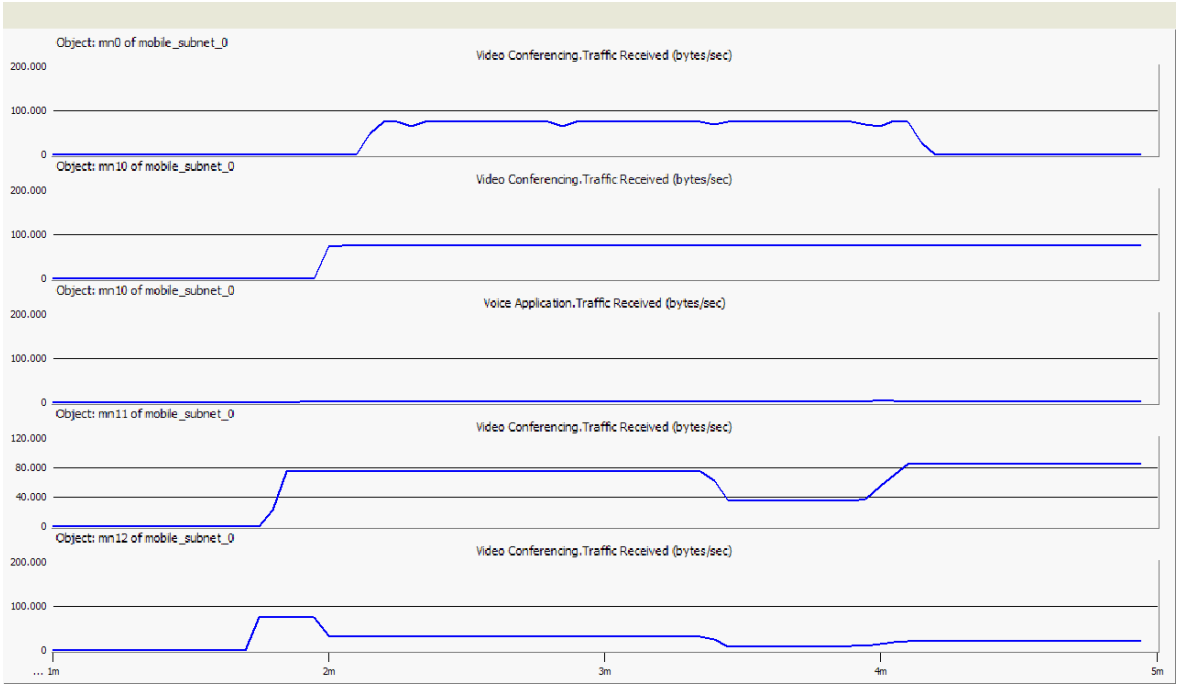


Figure 4.45. Aggregate Data Rate in Cell 4

In Figure 4.45, received flows for MN0, MN10, MN11, and MN12 are shown. When MN0 handoffs from cell3 to cell4 at around $200^{th}$ second, MN11 and MN12 are affected but get flow according to the reservation parameters. MN0 handoffs from cell4 to cell5 at around $230^{th}$ second. After all the queued packets are delivered to MN11, the available bandwidth is also used to deliver packets to MN12.

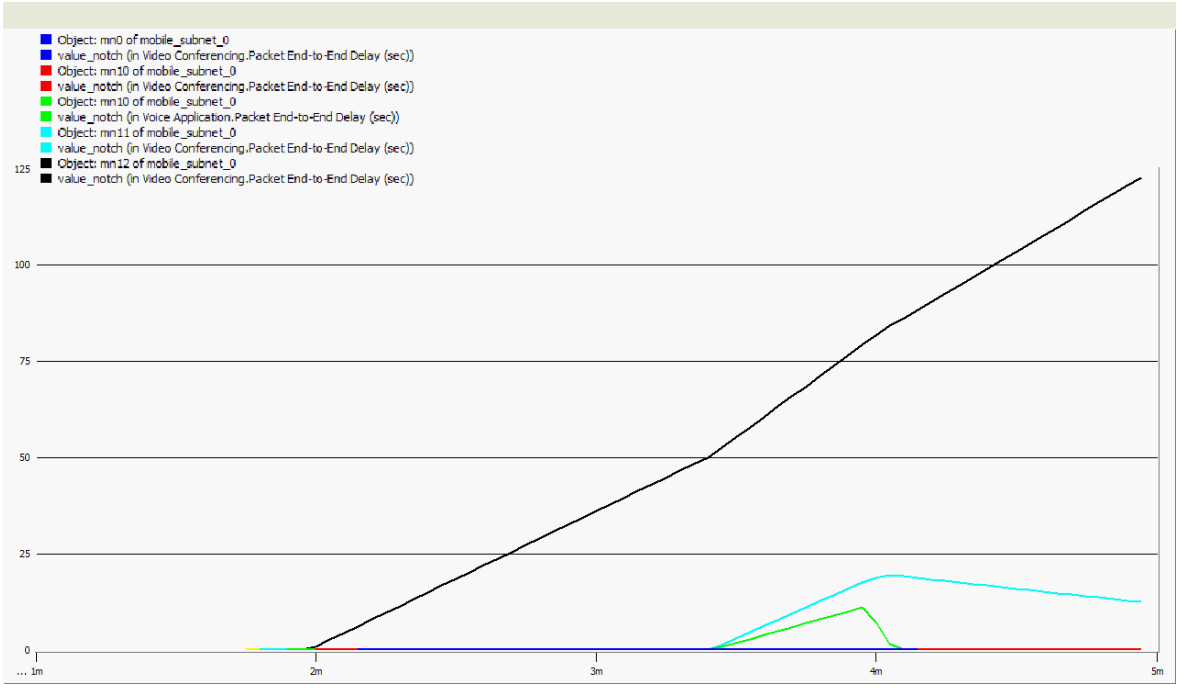


Figure 4.46. Delay in Cell 4

In Figure 4.46, delay values for MN0, MN10, MN11, and MN12 are shown. At $120^{th}$ second, MN10 Gold/Standard flow starts and delay for MN12 starts to increase. When at $130^{th}$ second, MN0 flow starts, delay value for MN11 also starts to increase until MN0 handoffs to cell4 from cell3 at around $200^{th}$ second. Then delay value for MN11 starts to decrease but since there is still inadequate bandwidth, delay value for MN12 continues to increase until the simulation ends.

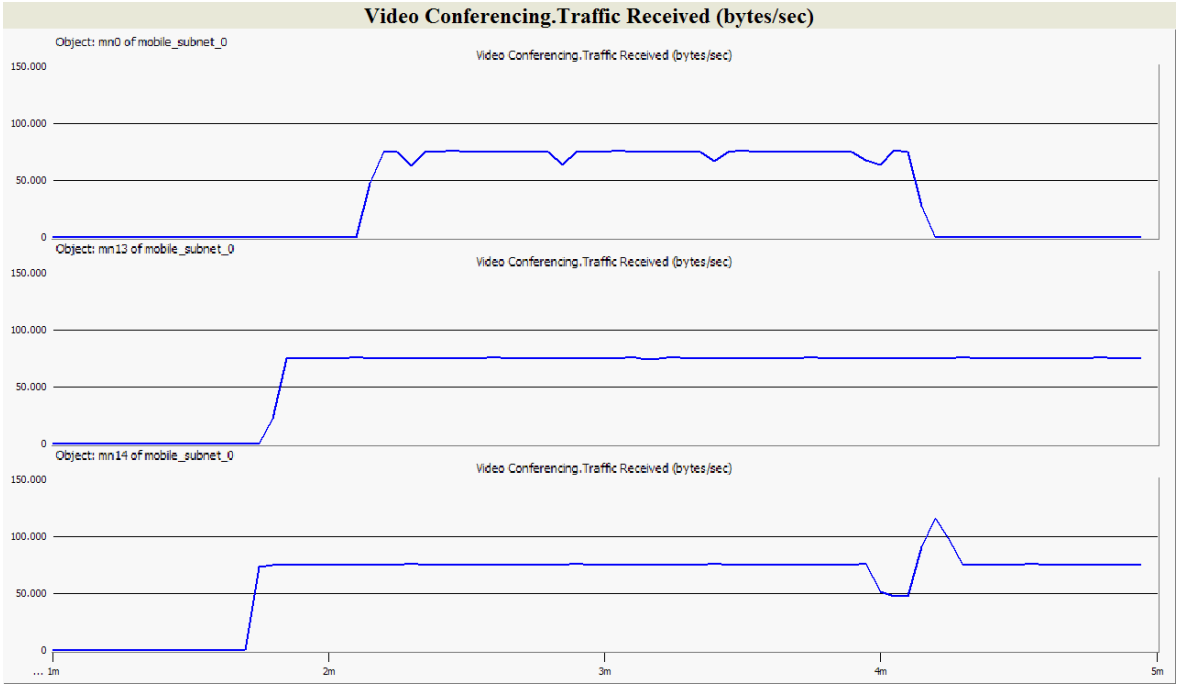


Figure 4.47. Aggregate Data Rate in Cell 5

In Figure 4.47, received flows for MN0, MN13, and MN14 are shown. When MN0 handoffs from cell4 to cell5 at around $230^{th}$ second, MN14 is affected. When the flow for MN0 ends at $250^{th}$ second, with the available bandwidth, MN14 receives all queued packets which results in the dramatic increase at the graph.

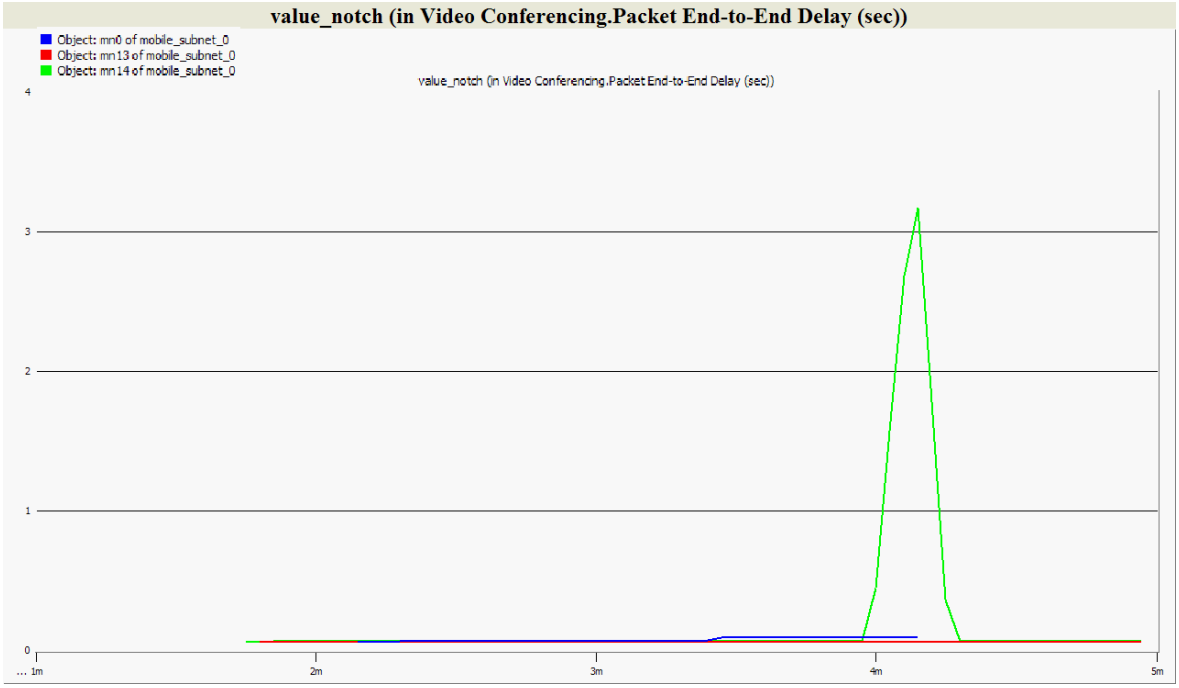


Figure 4.48. Delay in Cell 5

In Figure 4.48, delay values for MN0, MN13, and MN14 are shown. When MN0 handoffs from cell4 to cell5 at around $230^{th}$ second, the delay value for MN14 starts to increase until MN0 flow ends at $250^{th}$ second. Then with the available bandwidth delay values decrease to initial value.

### 4.2.10. No QoS at the Air Interface

This scenario depicts the default behaviour of the system without our proposed scheme. Therefore, we use it as a reference to show the improvement provided by our proposed QoS mechanism. All mobile nodes generate flows according to the fourth pattern as shown in Table 4.6. For each cell received flow and delay values are shown at following figures. Following are the key events to mention in the figures:

- *At 130$^{th}$ sec,* mn0 flow starts, and since all has the same QoS value, nodes receive same flow.
- *At 145$^{th}$ sec,* mn0 handovers to cell2, mn2 and mn3 starts to receive flow since now there is available bandwidth, but now all nodes at cell2 receive the same flow.
- *At 170$^{th}$ sec,* mn0 handovers to cell3, mn5 and mn6 starts to receive flow since now there is available bandwidth, but now all nodes at cell3 receive the same flow.
- *At 195$^{th}$ sec,* mn0 handovers to cell4, mn8 and mn9 starts to receive flow since now there is available bandwidth, but now all nodes at cell4 receive the same flow.

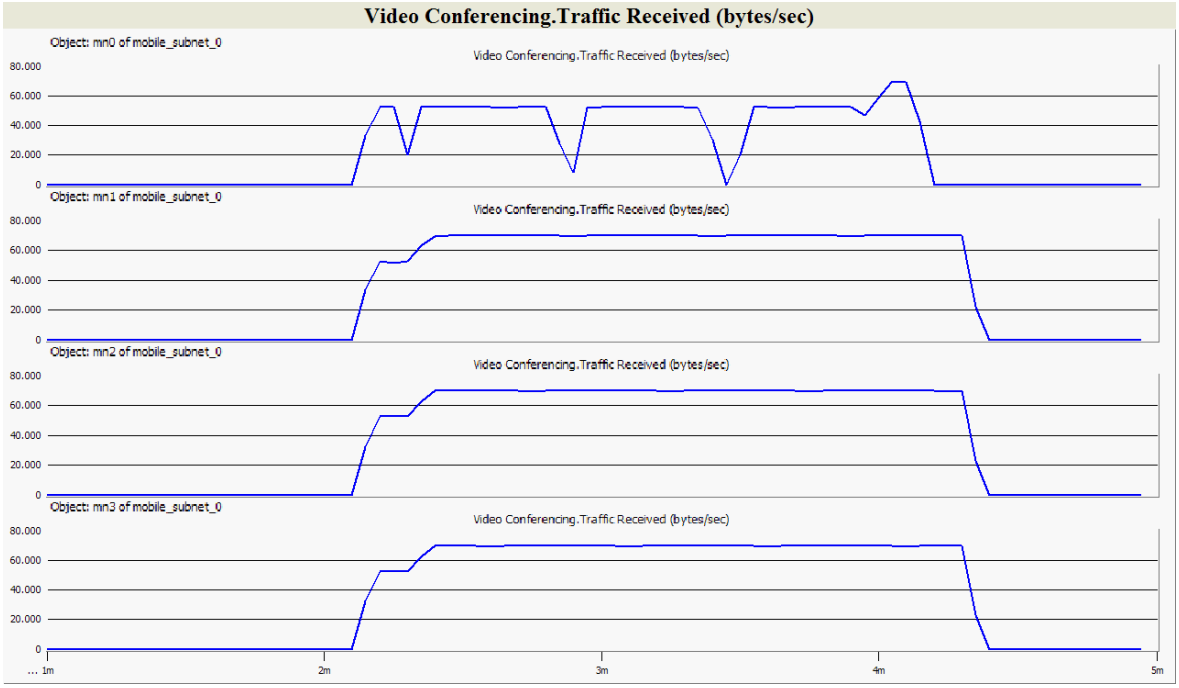


Figure 4.49. Aggregate Data Rate in Cell 1

In Figure 4.49, received flows for MN0, MN1, MN2, and MN3 are shown. When MN0 flow starts at $130^{th}$ second, all nodes at cell1 receive around 50,000 bps. After MN0 handoffs from cell1 to cell2 at around $140^{th}$ second, MN1, MN2, and MN3 start to receive 75,000 bps again.

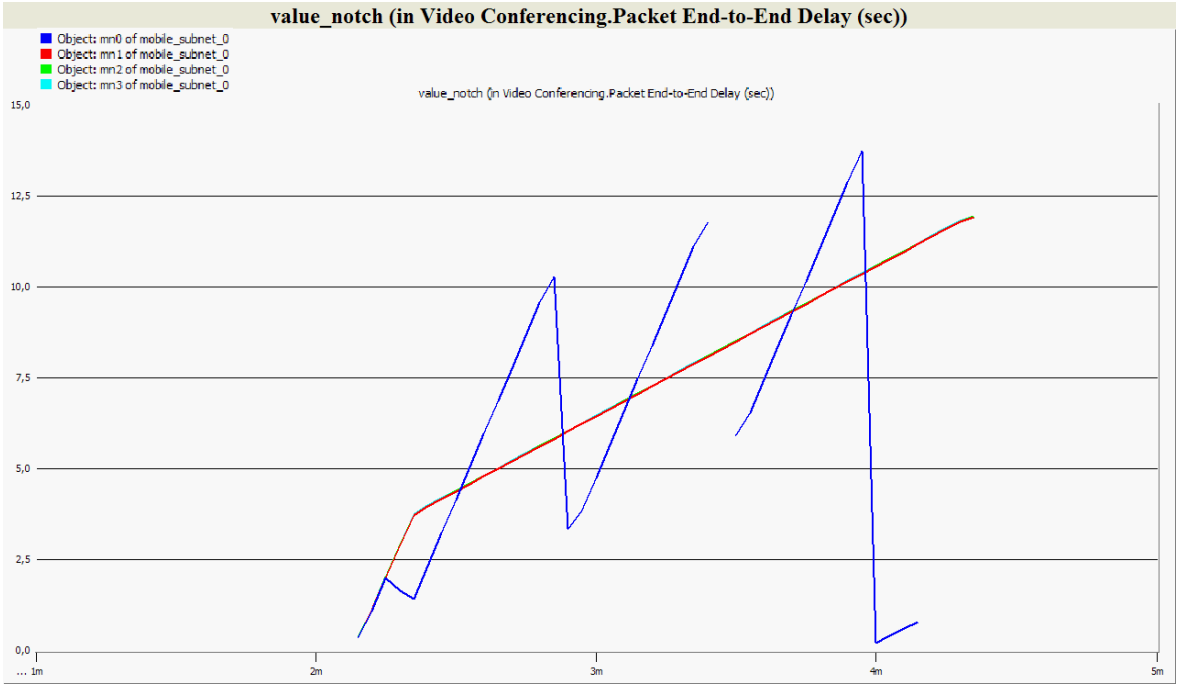


Figure 4.50. Delay in Cell 1

In Figure 4.50, delay values for MN0, MN1, MN2, and MN3 are shown. When at $130^{th}$ second, flow for all nodes start, delay value for all nodes start to increase. When MN0 handoffs to cell2 from cell1 at around $140^{th}$ second, the delay for MN1, MN2, and MN3 continue increasing with a slower rate. The delay for MN0 decreases while handoff since packets are dropped during handoff, then starts increasing at cell2.

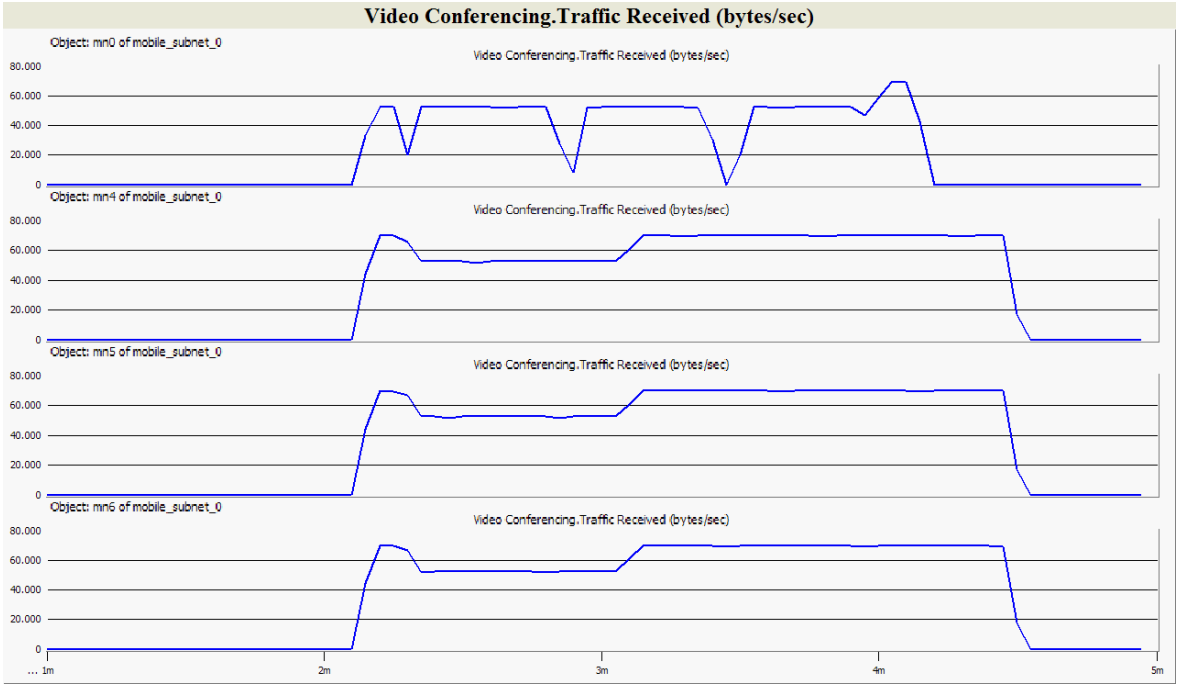


Figure 4.51. Aggregate Data Rate in Cell 2

In Figure 4.51, received flows for MN0, MN4, MN5, and MN6 are shown. When MN0 handoffs from cell2 to cell1 at around $140^{th}$ second, all nodes at cell2 receive around 50,000 bps. After MN0 handoffs from cell2 to cell3 at around $170^{th}$ second, MN4, MN5, and MN6 start to receive 75,000 bps again.

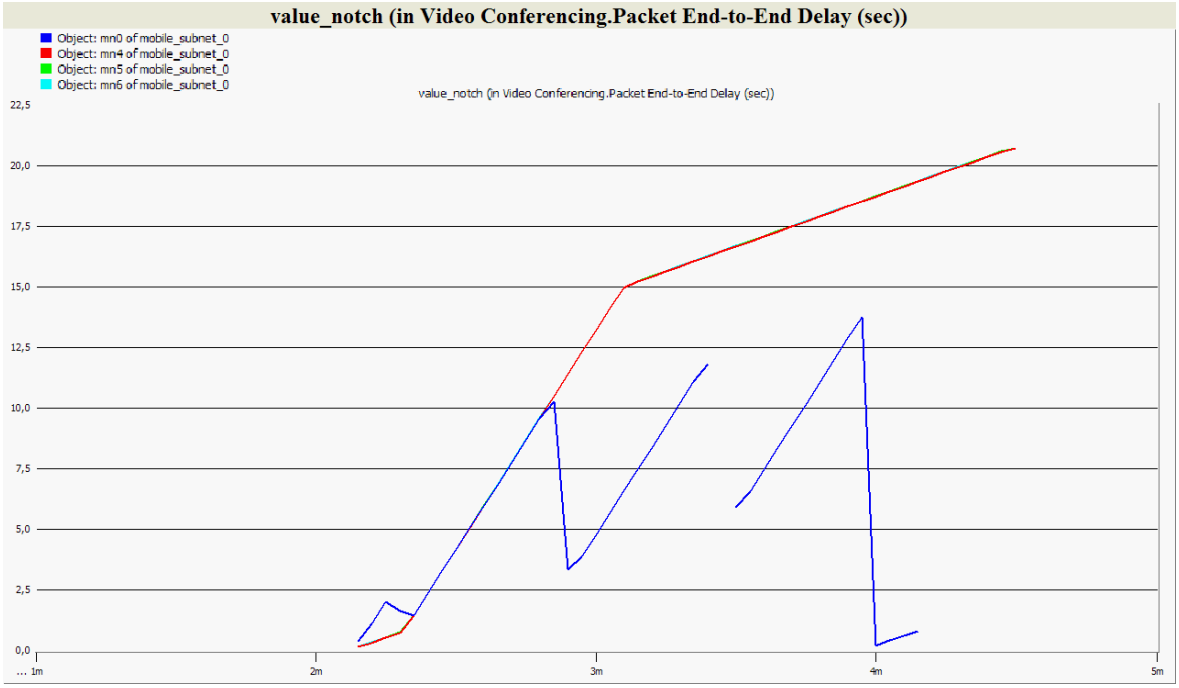


Figure 4.52. Delay in Cell 2

In Figure 4.52, delay values for MN0, MN4, MN5, and MN6 are shown. When MN0 handoffs from cell2 to cell1 at around $140^{th}$ second, delay value for all nodes start increasing with a higher rate. When MN0 handoffs to cell3 from cell2 at around $170^{th}$ second, the delay for MN4, MN5, and MN6 continue increasing with a slower rate. The delay for MN0 decreases while handoff since packets are dropped during handoff, then starts increasing at cell3.

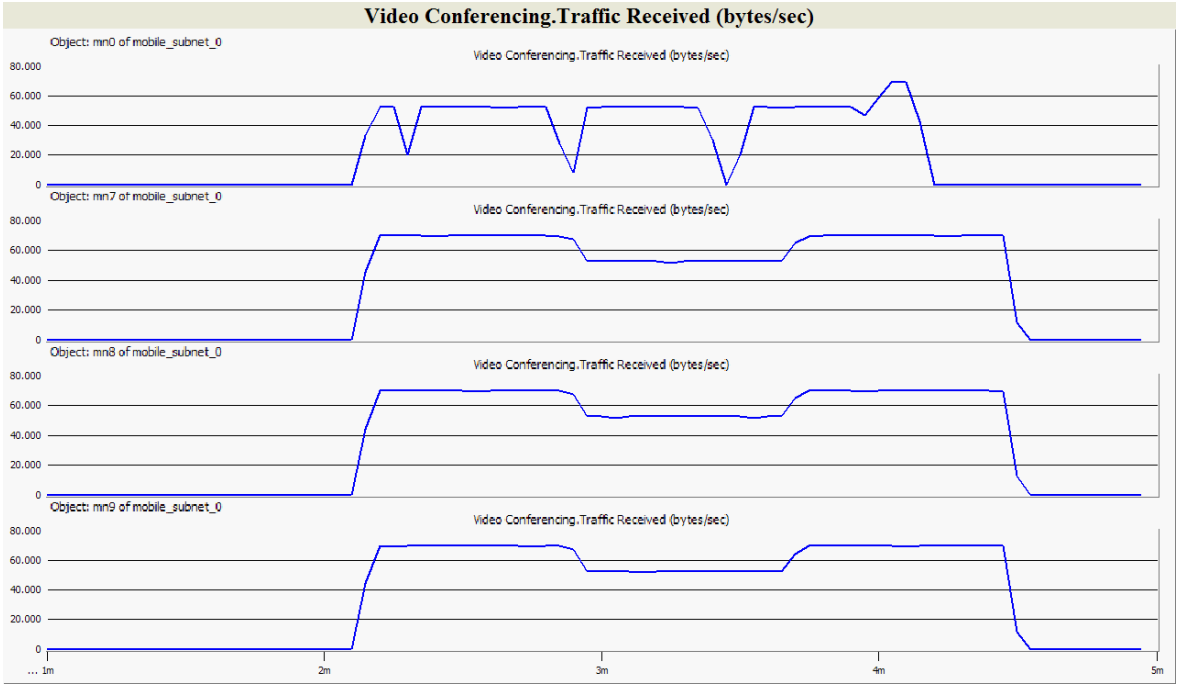


Figure 4.53. Aggregate Data Rate in Cell 3

In Figure 4.53, received flows for MN0, MN7, MN8, and MN9 are shown. When MN0 handoffs from cell3 to cell2 at around $170^{th}$ second, all nodes at cell3 receive around 50,000 bps. After MN0 handoffs from cell3 to cell4 at around $200^{th}$ second, MN7, MN8, and MN9 start to receive 75,000 bps again.

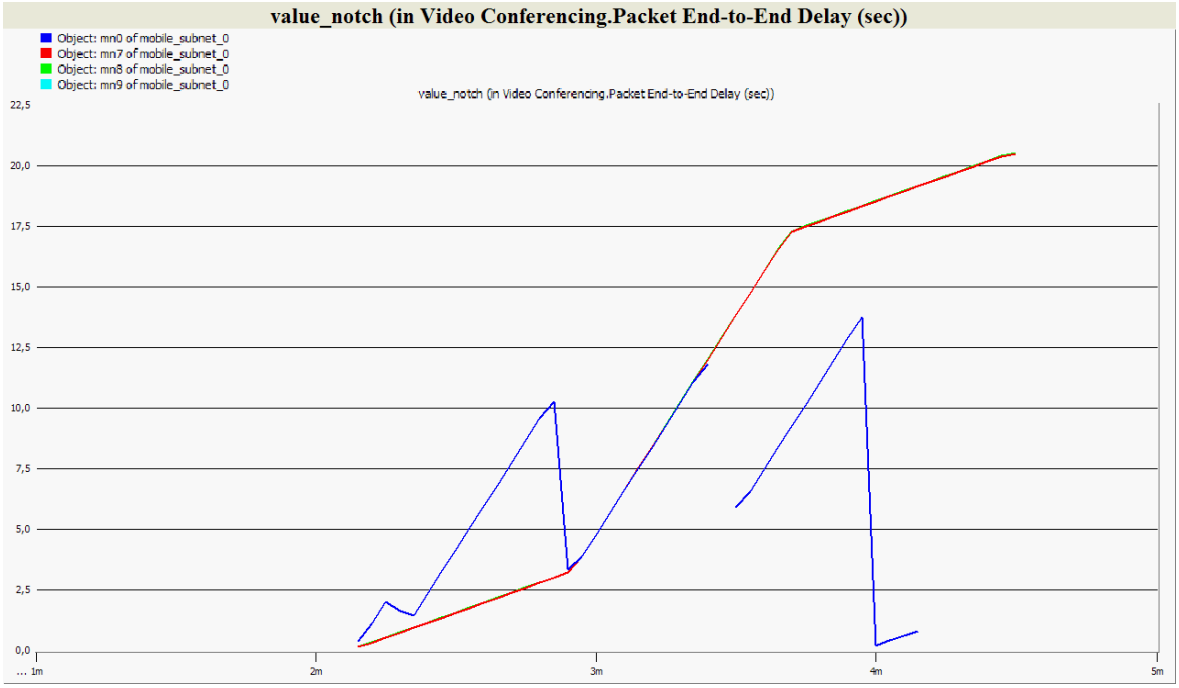


Figure 4.54. Delay in Cell 3

In Figure 4.54, delay values for MN0, MN7, MN8, and MN9 are shown. When MN0 handoffs from cell3 to cell2 at around $170^{th}$ second, delay value for all nodes start increasing with a higher rate. When MN0 handoffs to cell4 from cell3 at around $200^{th}$ second, the delay for MN7, MN8, and MN9 continue increasing with a slower rate. The delay for MN0 decreases while handoff since packets are dropped during handoff, then starts increasing at cell4.

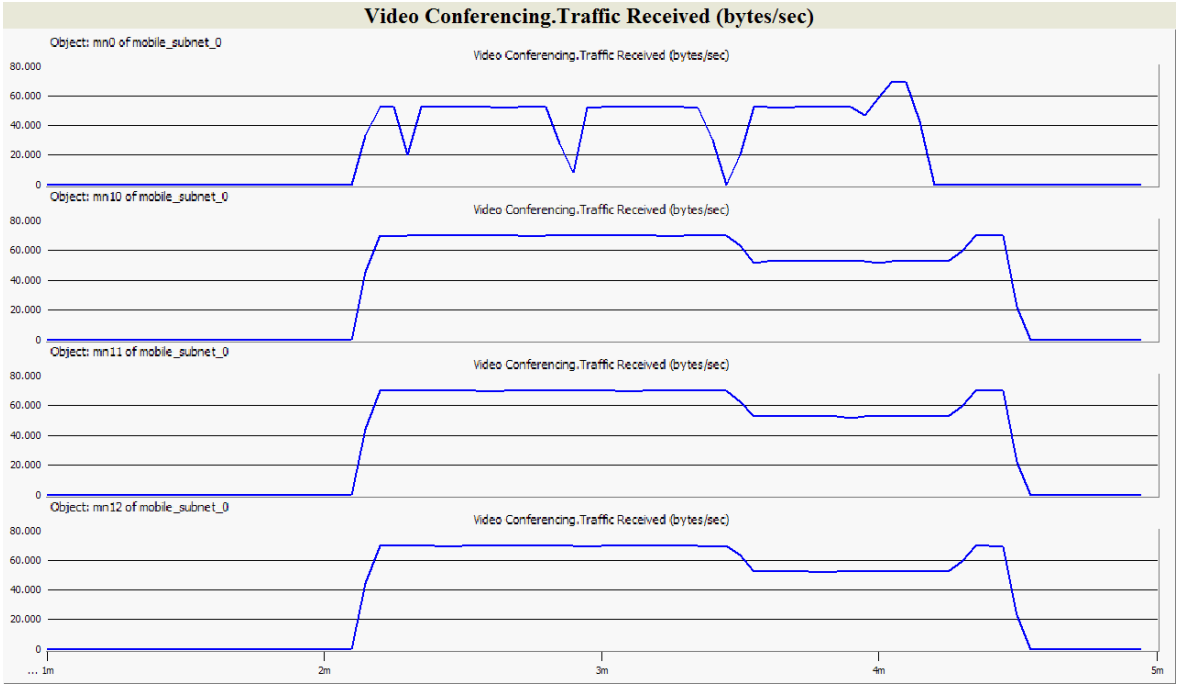


Figure 4.55. Aggregate Data Rate in Cell 4

In Figure 4.55, received flows for MN0, MN10, MN11, and MN12 are shown. When MN0 handoffs from cell4 to cell3 at around $200^{th}$ second, all nodes at cell4 receive around 50,000 bps. After MN0 handoffs from cell4 to cell5 at around $230^{th}$ second, MN10, MN11, and MN12 start to receive 75,000 bps again.

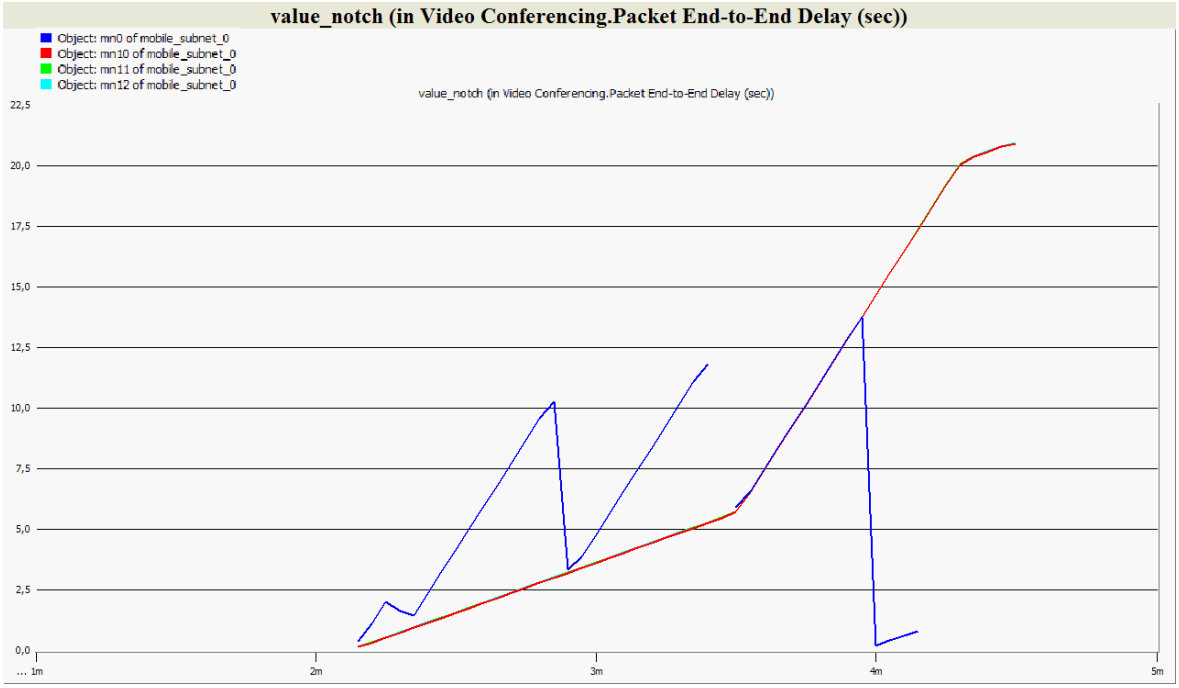


Figure 4.56. Delay in Cell 4

In Figure 4.56, delay values for MN0, MN10, MN11, and MN12 are shown. When MN0 handoffs from cell4 to cell3 at around $200^{th}$ second, delay value for all nodes start to increase. When MN0 handoffs to cell5 from cell4 at around $230^{th}$ second, the delay for MN10, MN11, and MN12 continue increasing with a slower rate. The delay for MN0 decreases while handoff since packets are dropped during handoff, then starts increasing at cell5.

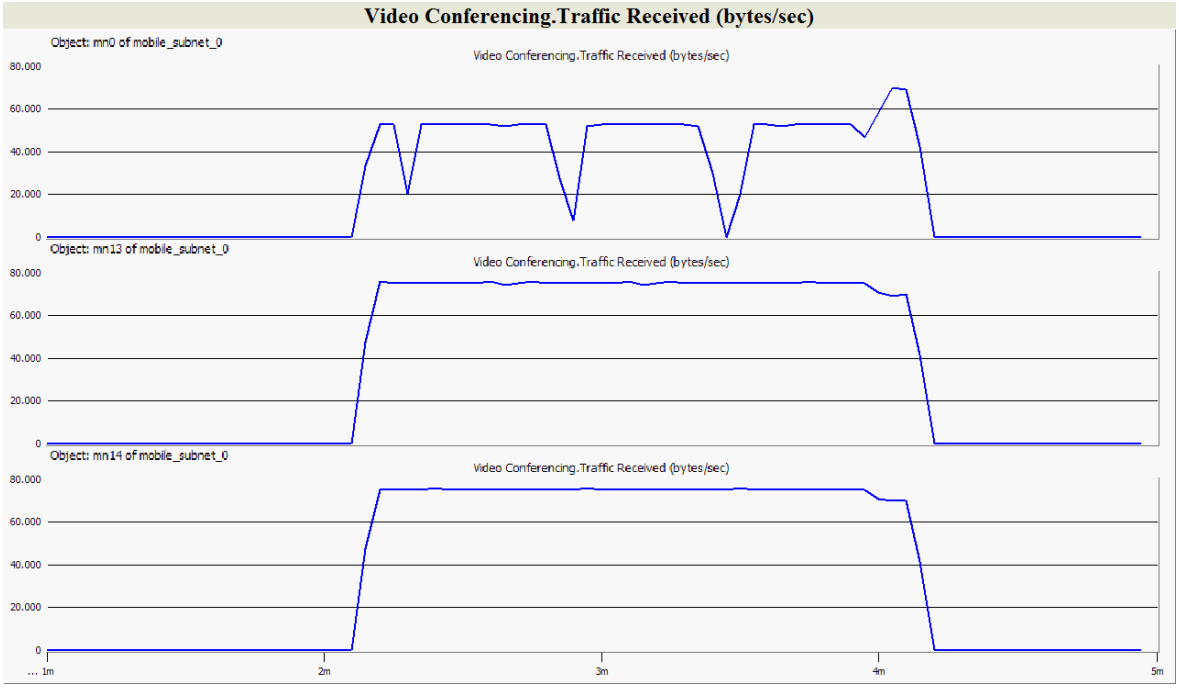


Figure 4.57. Aggregate Data Rate in Cell 5

In Figure 4.57, received flows for MN0, MN13, and MN14 are shown. When MN0 handoffs from cell5 to cell4 at around $230^{th}$ second, all nodes at cell5 receive around 70,000 bps. Then the traffic for all nodes end.

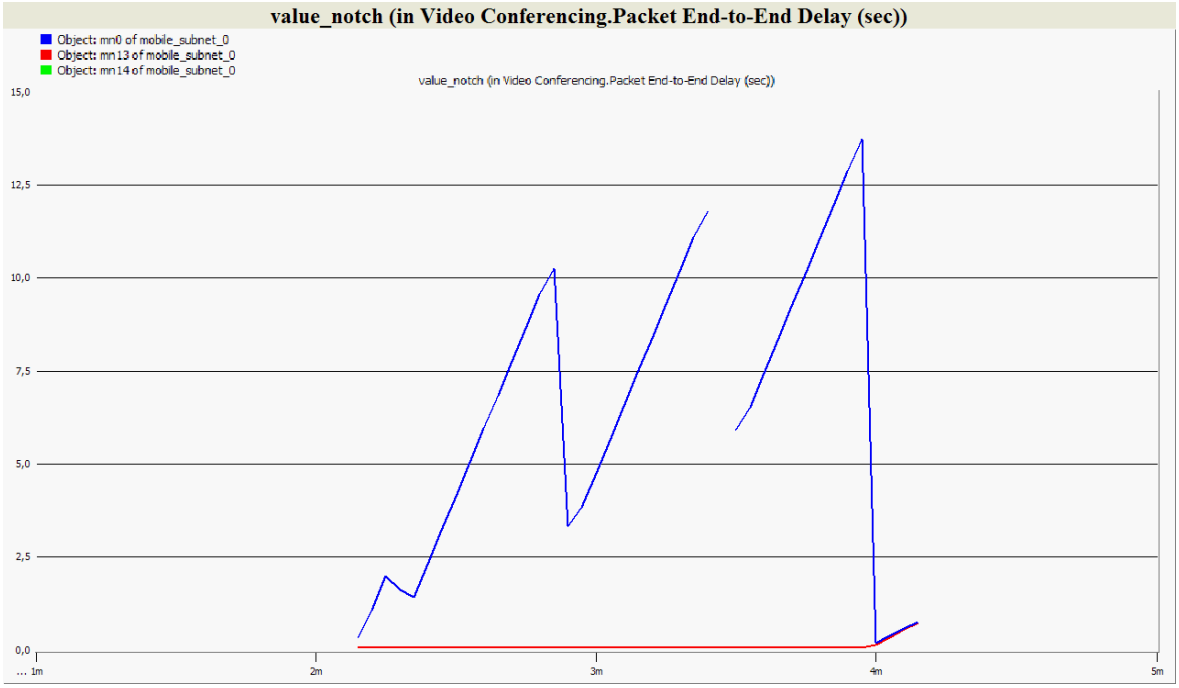


Figure 4.58. Delay in Cell 5

In Figure 4.58, delay values for MN0, MN13, and MN14 are shown. Until MN0 handoffs from cell5 to cell4 at around $230^{th}$ second, delay values for MN13 and MN14 are very low since there is available bandwidth. After MN0 handoffs to cell5, delay values for all nodes start to increase until the traffic for all nodes end. The peak values MN0 is due to its handoffs. Since all nodes have same QoS level, at each cell, delay value for MN0 increases with a high rate.

## 4.3. Overall Evaluation of Simulation Results

The architecture proposed in this thesis provides mechanisms to enable end-to-end QoS. Simulation results show that an IP packet traversing the whole network, including air interface and core network, experiences same QoS treatment that is based on Layer 3 parameters. The priority scheduling with reservation algorithm enables reserving air bandwidth for specific flow types. The amount of bandwith for each flow type can be defined in the architecture.

Mobile IP is utilized to provide mobility. Simulation results show that a mobile

node moving to another WAR stays connected over its HA and its packets are tunneled to new WAR by its HA. During handoff times, mobile node expriences small packet loss until it registers with HA. When a mobile node is connected to a new WAR, it acquires bandwith according to its SLA and QoS values. All air interface allocation in a cell changes when a new node enters the cell and leaves the cell.

# 5. CONCLUSIONS AND FUTURE WORK

As transition to all-IP next generation networks is ongoing, providing end-to-end QoS in these networks is drawing more attention, as users demand at least the same experience of legacy networks for voice and data services. In this thesis, this issue is addressed and an architecture that combines access network and the core network QoS mechanism to provide end-to-end QoS is proposed.

Firstly, a new MAC layer that runs over FH-OFDM physical layer is presented. The aim of the MAC layer is to extend the packet switching concept to the air interface. The QoS of the air interface is maintained by central control and scheduling of radio resources. For scheduling air resources, layer three QoS parameters are used, which results in same treatment of IP packets all over the network including the air interface. For the core network, MPLS is used as enabling technology for QoS. DiffServ support of MPLS and DiffServ aware traffic engineering are used to provide strict SLAs with strict QoS requirements. The mechanisms used for access and the core networks are integrated to meet end-to-end QoS requirements. The proposed architecture is implemented with OPNET Modeler simulator, and perceived QoS for different user and traffic types defined in the architecture are analyzed. It is shown that IP packets are treated similarly regarding QoS at access and core network.

There are some points open to further research. In our architecture, we utilized Mobile IP for user mobility. To provide better handoff experience, Layer 2 handoff and its effects should be investigated for micro mobility. Combined with Mobile IP for macro mobility, this architecture may result in reduced handoff overhead.

# APPENDIX A: OPNET SOURCE CODE

## A.1. MAC Implementation Function Block

```
static int wlan_hld_list_elem_add_comp(const void* list_elem_ptr1,
                                       const void* list_elem_ptr2) {
    WlanT_Hld_List_Elem*  hld_ptr1;
    WlanT_Hld_List_Elem*  hld_ptr2;

    /* This procedure is used in list processing to sort lists and find members of lists */
    /* containing higher layer data packets according to their MAC address destinations. */
    /* It returns 1 if hld_ptr1 is at a lower address than hld_ptr2 (closer to list head). */
    /* It returns -1 if hld_ptr1 is at a higher address than hld_ptr2 (closer to list tail). */
    /* It returns 0 if the addresses associated with the two elements are equal. */
    FIN(wlan_hld_list_elem_add_comp(list_elem_ptr1, list_elem_ptr2));

    hld_ptr1 = (WlanT_Hld_List_Elem *) list_elem_ptr1;
    hld_ptr2 = (WlanT_Hld_List_Elem *) list_elem_ptr2;

    if (hld_ptr1->destination_address > hld_ptr2->destination_address) {FRET(-1);}
    if (hld_ptr1->destination_address < hld_ptr2->destination_address) {FRET(1);} else {FRET(0);}
}

static int fhofdm_registered_ms_list_elem_add_comp(const void* list_elem_ptr1,
                                                   const void* list_elem_ptr2) {
    FHOFDM_Registered_MS_Elem*  hld_ptr1;
    FHOFDM_Registered_MS_Elem*  hld_ptr2;

    /* This procedure is used in list processing to sort lists and find members of lists */
    /* containing higher layer data packets according to their MAC address destinations. */
    /* It returns 1 if hld_ptr1 is at a lower address than hld_ptr2 (closer to list head). */
    /* It returns -1 if hld_ptr1 is at a higher address than hld_ptr2 (closer to list tail).*/
    /* It returns 0 if the addresses associated with the two elements are equal. */
    FIN(fhofdm_registered_ms_list_elem_add_comp(list_elem_ptr1, list_elem_ptr2));

    hld_ptr1 = (FHOFDM_Registered_MS_Elem *) list_elem_ptr1;
    hld_ptr2 = (FHOFDM_Registered_MS_Elem *) list_elem_ptr2;
    if (hld_ptr1->client_address > hld_ptr2->client_address) {FRET(-1);}
    if (hld_ptr1->client_address < hld_ptr2->client_address) {FRET(1);} else {FRET(0);}
}

static int fhofdm_registered_ms_list_elem_add_comp_ip(const void* list_elem_ptr1,
                                                      const void* list_elem_ptr2) {
```

```
    FHOFDM_Registered_MS_Elem*  hld_ptr1;
    FHOFDM_Registered_MS_Elem*  hld_ptr2;

    /* This procedure is used in list processing to sort lists and find members of lists */
    /* containing higher layer data packets according to their MAC address destinations. */
    /* It returns 1 if hld_ptr1 is at a lower address than hld_ptr2 (closer to list head). */
    /* It returns -1 if hld_ptr1 is at a higher address than hld_ptr2 (closer to list tail).*/
    /* It returns 0 if the addresses associated with the two elements are equal. */
    FIN(fhofdm_registered_ms_list_elem_add_compip(list_elem_ptr1, list_elem_ptr2));

    hld_ptr1 = (FHOFDM_Registered_MS_Elem *) list_elem_ptr1;
    hld_ptr2 = (FHOFDM_Registered_MS_Elem *) list_elem_ptr2;

    FRET(strcmp(hld_ptr1->ip_address, hld_ptr2->ip_address));
}

static void inc_slot() {
    current_slot = current_slot + 1;
    if (current_slot == NUM_OF_SLOTS )
        current_slot = 0;
}

static void send_dla_packet(Packet* pkt,
                            int ch,
                            int slot) {

    Packet* dla_ptr;
    int client_address;

    dla_ptr = op_pk_create_fmt("ofdm_dla");

    op_pk_fd_get_int32(pkt, 2,  &client_address);
    op_pk_fd_set_int32(dla_ptr, 0, DLAC_DLA , OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(dla_ptr, 1, client_address, OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(dla_ptr, 2, ch + TC_OFFSET + 1 , OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(dla_ptr, 3, slot, OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_dbl(dla_ptr, "Tx Data Rate", 900000, OPC_FIELD_SIZE_UNCHANGED);
    op_pk_send(dla_ptr, DLAC);
}

static void schedule_dl() {

    /*
    FHOFDM_In_Packet_Elem* in_packet_ptr;
    if (op_prg_list_size(in_packet_queue) > 0) {
            in_packet_ptr = (FHOFDM_In_Packet_Elem*)op_prg_list_remove(in_packet_queue, 0);
            if (in_packet_ptr != OPC_NIL) {
```

```
                if (in_packet_ptr->client_address == 1) {
                        op_prg_list_insert (tc_dl_queue[0][0],
                                            in_packet_ptr->pk,
                                            OPC_LISTPOS_TAIL);
                }
                else {
                        op_prg_list_insert (tc_dl_queue[1][0],
                                            in_packet_ptr->pk,
                                            OPC_LISTPOS_TAIL);
                }
                printf("scheduled packet of client %d \n",in_packet_ptr->client_address);
        }
}
*/
int i1, i2, found;
Packet* pk_ptr = OPC_NIL;
int i;
int tmp_index;

for (i = 0; i < NUM_PRIOS; i++) {
    if (op_prg_list_size(dl_packet_queue[i]) > 0) {
        //printf("dl_packet_queue %d is %d\n",i, op_prg_list_size (dl_packet_queue[i]));
    }
}

if (flow_control_enabled == 0) {
    for (i = 0; i < NUM_PRIOS; i++) {
        if (op_prg_list_size(dl_packet_queue[i]) > 0) {
            pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i], 0);
            break;
        }
    }
} else {

    found = 0;

    for (i2 = 2; i2 < 5; i2++) {
        if (found == 0 && priority_level_quota[i2] > 0 ) {
            if (op_prg_list_size(dl_packet_queue[i2]) > 0) {
                pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i2], 0);
                priority_level_quota[i2] = priority_level_quota[i2] - 1;
                found = 1;
                break;
            }
        }
    }

    if (found == 0) {
```

```
    for (i1 = 2; i1 < 5; i1++) {
        for (i2 = 2; i2 < 5; i2++) {
            if (found == 0 && priority_level_quota[i1] > 0 ) {
                if (op_prg_list_size(dl_packet_queue[i2]) > 0) {
                    pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i2], 0);
                    priority_level_quota[i1] = priority_level_quota[i1] - 1;
                    found = 1;
                    break;
                }
            }
        }
        if (found == 1) {
            break;
        }
    }
}

if (found == 0) {

    for (i2 = 5; i2 < 8; i2++) {
        if (found == 0 && priority_level_quota[i2] > 0 ) {
            if (op_prg_list_size(dl_packet_queue[i2]) > 0) {
                pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i2], 0);
                priority_level_quota[i2] = priority_level_quota[i2] - 1;
                found = 1;
                break;
            }
        }
    }
}

if (found == 0) {

    for (i1 = 5; i1 < 8; i1++) {
        for (i2 = 5; i2 < 8; i2++) {
            if (found == 0 && priority_level_quota[i1] > 0 ) {
                if (op_prg_list_size(dl_packet_queue[i2]) > 0) {
                    pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i2], 0);
                    priority_level_quota[i1] = priority_level_quota[i1] - 1;
                    found = 1;
                    break;
                }
            }
        }
        if (found == 1) {
            break;
        }
    }
```

```
    }

    if (found == 0) {

        for (i2 = 8; i2 < 11; i2++) {
            if (found == 0 && priority_level_quota[i2] > 0 ) {
                if (op_prg_list_size(dl_packet_queue[i2]) > 0) {
                    pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i2], 0);
                    priority_level_quota[i2] = priority_level_quota[i2] - 1;
                    found = 1;
                    break;
                }
            }
        }

    }
    if (found == 0) {
        for (i1 = 8; i1 < 11; i1++) {
            for (i2 = 8; i2 < 11; i2++) {
                if (found == 0 && priority_level_quota[i1] > 0 ) {
                    if (op_prg_list_size(dl_packet_queue[i2]) > 0) {
                        pk_ptr = (Packet*)op_prg_list_remove(dl_packet_queue[i2], 0);
                        priority_level_quota[i1] = priority_level_quota[i1] - 1;
                        found = 1;
                        break;
                    }
                }
            }
            if (found == 1) {
                break;
            }
        }
    }
}
if (pk_ptr != OPC_NIL) {
    /*
    op_pk_fd_get_int32 (pk_ptr, 2,   &client_address);
    if (client_address == 1) {
            op_prg_list_insert (tc_dl_queue[0][0], pk_ptr, OPC_LISTPOS_TAIL);
            dl_assignmens_count
    }
    else {
            op_prg_list_insert (tc_dl_queue[1][0], pk_ptr, OPC_LISTPOS_TAIL);
    }
    printf("scheduled packet of client %d \n",client_address);
     */
    found = 0;
    for (i1 = 0; i1 < NUM_OF_TC; i1++) {
```

```
            for (i2 = 0; i2 < NUM_OF_SLOTS; i2++) {
                if (dl_assignments_count[i1][i2] == 0) {
                    op_prg_list_insert(tc_dl_queue[i1][i2], pk_ptr, OPC_LISTPOS_TAIL);
                    dl_assignments_count[i1][i2] = dl_assignments_count[i1][i2]+1;
                    send_dla_packet(pk_ptr, i1, i2);
                    found = 1;
                    break;
                }
            }
            if (found == 1) {
                break;
            }
        }
        if (found == 0) {
            op_pk_destroy(pk_ptr);
        }

    } else {
        init_priority_level_quota();
    }
}

static void schedule_ul() {

    int client_address, i1, i2, found, more_packet;
    Packet* pk_ptr = OPC_NIL;
    Packet* ula_ptr = OPC_NIL;
    int i;
    int packet_count;
    int qos;
    for (i = 0; i < NUM_PRIOS; i++) {
        if (op_prg_list_size(ul_packet_queue[i]) > 0) {
            //printf("ul_packet_queue %d is %d\n",i, op_prg_list_size (ul_packet_queue[i]));
        }
    }
    found = 1;
    more_packet = 1;
    while (found == 1 && more_packet == 1) {

        pk_ptr = OPC_NIL;
        more_packet = 0;
        for (i = 0; i < NUM_PRIOS; i++) {
            // if (op_prg_list_size (ul_packet_queue[i]) > 0 && priority_level_quota[i] > 0 ) {
            if (op_prg_list_size(ul_packet_queue[i]) > 0 ) {
                pk_ptr = (Packet*)op_prg_list_remove(ul_packet_queue[i], 0);
                more_packet = 1;
                // priority_level_quota[i] = priority_level_quota[i] - 1;
                break;
```

```
            }
        }
        if (pk_ptr != OPC_NIL) {
            found = 0;
            for (i1 = 0; i1 < NUM_OF_TC; i1++) {
                for (i2 = 0; i2 < NUM_OF_SLOTS; i2++) {
                    if (ul_assignments_count[i1][i2] == 0) {
                        found = 1;



                        op_pk_fd_get_int32(pk_ptr, 1,  &client_address);
                        op_pk_fd_get_int32(pk_ptr, 2,  &packet_count);
                        op_pk_fd_get_int32(pk_ptr, 3,  &qos);

                        //ul_assignments_count[i1][i2] = ul_assignments_count[i1][i2]+packet_count;

                        ula_ptr = op_pk_create_fmt("ofdm_ula");

                        op_pk_fd_set_int32(ula_ptr, 0, ULAC_ULA, OPC_FIELD_SIZE_UNCHANGED);
                        op_pk_fd_set_int32(ula_ptr, 1, client_address, OPC_FIELD_SIZE_UNCHANGED);
                        op_pk_fd_set_int32(ula_ptr, 2, i1 + TC_OFFSET + 1, OPC_FIELD_SIZE_UNCHANGED);
                        op_pk_fd_set_int32(ula_ptr, 3, i2, OPC_FIELD_SIZE_UNCHANGED);
                        op_pk_fd_set_dbl(ula_ptr, "Tx Data Rate", 900000, OPC_FIELD_SIZE_UNCHANGED);
                        op_pk_send(ula_ptr, ULAC);

                        break;
                    }
                }
                if (found == 1) {
                    break;
                }
            }
            op_pk_destroy(pk_ptr);
        }
        //else {
        // init_priority_level_quota();
        //}
    }
}

static void fhofdm_send_ulr() {
    Packet *pkptr;

    if (current_slot == ms_ulrc_time_slot) {
        if (op_prg_list_size(ms_ulr_queue) > 0) {
            pkptr = (Packet*)op_prg_list_remove(ms_ulr_queue, 0);
            if (pkptr != OPC_NIL) {
                op_pk_send(pkptr, ms_ulrc_channel);
```

```
            }
        }
    }
}

static void fhofdm_send_ul_packets() {

    Packet *pkptr;
    int i1;
    for (i1 = 0; i1 < NUM_OF_TC; i1++) {

        if (op_prg_list_size(tc_ul_queue[i1][current_slot]) > 0) {
            pkptr = (Packet*)op_prg_list_remove(tc_ul_queue[i1][current_slot], 0);
            if (pkptr != OPC_NIL) {
                op_pk_send(pkptr, i1 + TC_OFFSET + 1);
                ul_assignments_count[i1][current_slot] = ul_assignments_count[i1][current_slot] - 1;
            }
        }
    }
}

static void fhofdm_send_dl_packets() {

    Packet *pkptr;
    int i1;
    for (i1 = 0; i1 < NUM_OF_TC; i1++) {

        if (op_prg_list_size(tc_dl_queue[i1][current_slot]) > 0) {
            pkptr = (Packet*)op_prg_list_remove(tc_dl_queue[i1][current_slot], 0);
            if (pkptr != OPC_NIL) {
                op_pk_send(pkptr, i1 + TC_OFFSET + 1);
                dl_assignments_count[i1][current_slot] = dl_assignments_count[i1][current_slot] - 1;
            }
        }
    }
}

static int get_priority(Packet* pkt) {
    /**************** TODO - Get priority *********************/
    Packet* pk_ptr;
    IpT_Dgram_Fields* pk_fd_ptr;
    int result;
    pk_ptr = op_pk_copy(pkt);
    op_pk_nfd_access(pk_ptr, "fields", &pk_fd_ptr);
    result = mapDSCPtoQoS(pk_fd_ptr->tos);
    op_pk_destroy(pk_ptr);
    return result;
}
```

```
static int mapDSCPtoQoS(int DSCP) {

    int aDSCP = DSCP / 4;
    switch (aDSCP) {
        case 0: return 0;
        case 10: return 13;
        case 12: return 12;
        case 14: return 11;
        case 18: return 10;
        case 20: return 9;
        case 22: return 8;
        case 26: return 7;
        case 28: return 6;
        case 30: return 5;
        case 34: return 4;
        case 36: return 3;
        case 38: return 2;
        case 46: return 1;
        default: return 0;
    }

}

static int mapQoStoDSCP(int QoS) {

    switch (QoS) {
        case 0: return 0;
        case 13: return 10 * 4;
        case 12: return 12 * 4;
        case 11: return 14 * 4;
        case 10: return 18 * 4;
        case 9: return 20 * 4;
        case 8: return 22 * 4;
        case 7: return 26 * 4;
        case 6: return 28 * 4;
        case 5: return 30 * 4;
        case 4: return 34 * 4;
        case 3: return 36 * 4;
        case 2: return 38 * 4;
        case 1: return 46 * 4;
        default: return 0;
    }
}

static int check_client(int ap_address,
                        int client_address) {
    int tmp = client_address / 10 * 10;
```

```
    if (tmp == ap_address) {
        return 1;
    } else {
        return 0;
    }
}

static void init_priority_level_quota() {
    /*int i;
    for (i=0; i < NUM_PRIOS; i++) {
            if(priority_level_quota[i] <= 0) {
                    if (i == 1 || i == 2 || i == 3 || i == 4) {
                            priority_level_quota[i] = platinum_priority;
                    }
                    else if (i == 5 || i == 6 || i == 7) {
                            priority_level_quota[i] = gold_priority;
                    }
                    else if (i == 8 || i == 9 || i == 10) {
                            priority_level_quota[i] = silver_priority;
                    }
                    else if (i == 11 || i == 12 || i == 13) {
                            priority_level_quota[i] = silver_priority;
                    }
             }
    }
     */

    /*
    platinum_priority_quota = platinum_priority;
    gold_priority_quota = gold_priority;
    silver_priority_quota = silver_priority;
    platinum_priority_pointer = 0;
    gold_priority_pointer = 0;
    silver_priority_pointer = 0;
     */

    if(priority_level_quota[2] <= 0) {
        priority_level_quota[2] = platinum_strm_priority;
    }
    if(priority_level_quota[3] <= 0) {
        priority_level_quota[3] = platinum_std_priority;
    }
    if(priority_level_quota[4] <= 0) {
        priority_level_quota[4] = platinum_bg_priority;
    }
    if(priority_level_quota[5] <= 0) {
        priority_level_quota[5] = gold_strm_priority;
    }
```

```
    if(priority_level_quota[6] <= 0) {
        priority_level_quota[6] = gold_std_priority;
    }
    if(priority_level_quota[7] <= 0) {
        priority_level_quota[7] = gold_bg_priority;
    }
    if(priority_level_quota[8] <= 0) {
        priority_level_quota[8] = silver_strm_priority;
    }
    if(priority_level_quota[9] <= 0) {
        priority_level_quota[9] = silver_std_priority;
    }
    if(priority_level_quota[10] <= 0) {
        priority_level_quota[10] = silver_bg_priority;
    }
}

static int inc_mod3(int a) {

    a = a + 1;
    if (a > 2)
        a = 0;
    return a;
}
```

## A.2. Mobile Node Idle

```
if (PACKET_RECEIVED) {

    Packet* in;
    Packet* src_pkptr;
    Packet* tmp;
    int comp_pk_count;
    int pk_size;
    int pk_index;
    int seg_size;
    int traffic_channel;
    int client_address;
    int header, address, tc, slot;

    strm = op_intrpt_strm();
    received_packet = op_pk_get(strm);

    if (strm == DLAC) {
```

```
    op_pk_fd_get_int32(received_packet, 0,  &header);
    op_pk_fd_get_int32(received_packet, 1,  &address);
    op_pk_fd_get_int32(received_packet, 2,   &tc);
    op_pk_fd_get_int32(received_packet, 3,   &slot);
    if (header == DLAC_DLA && address == my_address) {
        ms_dl_tc = tc;
        ms_dl_tc_slot = slot;
    }
    op_pk_destroy(received_packet);
} else if (strm == ULAC) {
    op_pk_fd_get_int32(received_packet, 0,  &header);
    op_pk_fd_get_int32(received_packet, 1,  &address);
    op_pk_fd_get_int32(received_packet, 2,   &tc);
    op_pk_fd_get_int32(received_packet, 3,   &slot);
    if (header == ULAC_ULA && address == my_address) {
        ms_ul_tc = tc - TC_OFFSET - 1;
        ms_ul_tc_slot = slot;
        while (op_prg_list_size(ms_packet_queue) > 0 ) {
            tmp = (Packet*)op_prg_list_remove(ms_packet_queue, 0);
            op_prg_list_insert(tc_ul_queue[ms_ul_tc][slot], tmp, OPC_LISTPOS_TAIL);
            ul_assignments_count[ms_ul_tc][slot] = ul_assignments_count[ms_ul_tc][slot]+1;
        }

    }
    op_pk_destroy(received_packet);

} else if (strm == ULRC) {
    op_pk_destroy(received_packet);
} else if (strm == RAC) {
    op_pk_destroy(received_packet);
} else {
    Packet* tmp;
    traffic_channel = strm - TC_OFFSET;
    //printf("MS: Packet received TC%d\n",traffic_channel);
    op_pk_fd_get_int32(received_packet, 2,   &client_address);
    op_pk_fd_get_pkt(received_packet, 3, &in);

    //printf("client_address %d my_address %d\n",client_address,my_address);
    if (client_address == my_address ) {
        tmp = op_pk_copy(in);
        seg_size = op_pk_total_size_get(tmp);
        /* Insert the segment into reassembly buffer. */
        op_sar_rsmbuf_seg_insert(ofdm_rsmbuf_ptr, tmp);
        /* Determine the number of complete source packets in reassembly buffer. */
        comp_pk_count = op_sar_rsmbuf_pk_count(ofdm_rsmbuf_ptr);
        /* Loop through all complete source packets in reassembly buffer. */
        for (pk_index = 0; pk_index < comp_pk_count; pk_index++) {
            /* Remove a complete source packet. */
```

```
                src_pkptr = op_sar_rsmbuf_pk_remove(ofdm_rsmbuf_ptr);
                /* Add source packet to internal sub-queue. */
                pk_size = op_pk_total_size_get(src_pkptr);
                wlan_rcvd_pkt_higher_layer_forward(src_pkptr,
                                                   OPC_FALSE,
                                                   mac_client_reassembly_buffer,
                                                   outstrm_to_mac_if);
            }
        }
        op_pk_destroy(in);
        op_pk_destroy(received_packet);
    }
} else if (DL_PACKET_SEND) {
    fhofdm_send_ulr();
    fhofdm_send_ul_packets();
    op_intrpt_schedule_self(op_sim_time()+TIME_SLOT , DL_PACKET_INTERRUPT);
}
```

## A.3. Mobile Node Request Uplink

```
Packet* transmit_frame_ptr;
Packet* ulr_ptr;
Packet* pk_ptr;
Packet* seg_pkptr;
IpT_Dgram_Fields* pk_fd_ptr;
char dest_addr_str[IPC_ADDR_STR_LEN];
char src_addr_str[IPC_ADDR_STR_LEN];
SimT_Pk_Id pkt_id, pkt_tree_id;
FHOFDM_Data d;
int i;
int rem_size;
int tos;
int packet_count;

++pk_count;
strm = op_intrpt_strm();
//printf("stream %d %d %d\n",strm,instrm_from_mac_if,op_intrpt_strm());
received_packet = op_pk_get(strm);

op_stat_write(pk_cnt_stathandle, pk_count);
//wlan_higher_layer_data_arrival();
//fhofdm_frame_transmit();
//op_pk_send( tmp_packet_ptr, LOW_LAYER_OUTPUT_STREAM);
//if (ap_flag != OPC_BOOLINT_ENABLED) {
```

```
//fhofdm_transmit_packet(received_packet);
pk_ptr = op_pk_copy(received_packet);
pkt_id = op_pk_id(pk_ptr);
pkt_tree_id = op_pk_tree_id(pk_ptr);
op_pk_nfd_access(pk_ptr, "fields", &pk_fd_ptr);
inet_address_print(dest_addr_str, pk_fd_ptr->dest_addr);
inet_address_print(src_addr_str, pk_fd_ptr->src_addr);

tos = get_priority(received_packet);
//printf("--------- tos %d \n",tos);
/* Determine the size of the source packet. */
rem_size = op_pk_total_size_get(received_packet);
/* Insert the source packet into segmentation buffer. */
op_sar_segbuf_pk_insert(ofdm_fragbuf_ptr, received_packet, 0);
/* Remove and send the first segment associated with arriving packet. */
/* If there is a remainder of source packet, schedule next segment transmission.*/
packet_count = 0;
while (rem_size > 0) {
    //op_intrpt_schedule_self (op_sim_time () + SEG_TX_TIME, SEG_TX_CODE);
    seg_pkptr = op_sar_srcbuf_seg_remove(ofdm_fragbuf_ptr, SEG_SIZE);
    transmit_frame_ptr = op_pk_create_fmt("ofdm_mac_data");
    op_pk_fd_set_int32(transmit_frame_ptr, 0,   1,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(transmit_frame_ptr, 1,   my_address,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(transmit_frame_ptr, 2,   0,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_pkt(transmit_frame_ptr, 3, seg_pkptr, OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_dbl(transmit_frame_ptr, "Tx Data Rate", 900000, OPC_FIELD_SIZE_UNCHANGED);

    packet_size_dcf = op_pk_total_size_get(transmit_frame_ptr);
    //op_pk_send (transmit_frame_ptr, 6);
    op_prg_list_insert(ms_packet_queue, transmit_frame_ptr, OPC_LISTPOS_TAIL);
    packet_count = packet_count + 1;
    rem_size -= SEG_SIZE;
}

ulr_ptr = op_pk_create_fmt("ofdm_ul_req");
op_pk_fd_set_int32(ulr_ptr, 0,   1,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 1,   my_address,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 2,   packet_count,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 3,   tos,           OPC_FIELD_SIZE_UNCHANGED);
op_prg_list_insert(ms_ulr_queue, ulr_ptr, OPC_LISTPOS_TAIL);
op_pk_destroy(pk_ptr);
```

## A.4. Wireless Access Router Idle

```
Packet* transmit_frame_ptr;
Packet* ulr_ptr;
Packet* pk_ptr;
Packet* seg_pkptr;
IpT_Dgram_Fields* pk_fd_ptr;
char dest_addr_str[IPC_ADDR_STR_LEN];
char src_addr_str[IPC_ADDR_STR_LEN];
SimT_Pk_Id pkt_id, pkt_tree_id;
FHOFDM_Data d;
int i;
int rem_size;
int tos;
int packet_count;

++pk_count;
strm = op_intrpt_strm();
//printf("stream %d %d %d\n",strm,instrm_from_mac_if,op_intrpt_strm());
received_packet = op_pk_get(strm);

op_stat_write(pk_cnt_stathandle, pk_count);
//wlan_higher_layer_data_arrival();
//fhofdm_frame_transmit();
//op_pk_send( tmp_packet_ptr, LOW_LAYER_OUTPUT_STREAM);
//if (ap_flag != OPC_BOOLINT_ENABLED) {
//fhofdm_transmit_packet(received_packet);

pk_ptr = op_pk_copy(received_packet);
pkt_id = op_pk_id(pk_ptr);
pkt_tree_id = op_pk_tree_id(pk_ptr);
op_pk_nfd_access(pk_ptr, "fields", &pk_fd_ptr);
inet_address_print(dest_addr_str, pk_fd_ptr->dest_addr);
inet_address_print(src_addr_str, pk_fd_ptr->src_addr);

tos = get_priority(received_packet);
//printf("--------- tos %d \n",tos);
/* Determine the size of the source packet. */
rem_size = op_pk_total_size_get(received_packet);
/* Insert the source packet into segmentation buffer. */
op_sar_segbuf_pk_insert(ofdm_fragbuf_ptr, received_packet, 0);
/* Remove and send the first segment associated with arriving packet. */
/* If there is a remainder of source packet, schedule next segment transmission.*/
packet_count = 0;
while (rem_size > 0) {
    //op_intrpt_schedule_self (op_sim_time () + SEG_TX_TIME, SEG_TX_CODE);
    seg_pkptr = op_sar_srcbuf_seg_remove(ofdm_fragbuf_ptr, SEG_SIZE);
    transmit_frame_ptr = op_pk_create_fmt("ofdm_mac_data");
    op_pk_fd_set_int32(transmit_frame_ptr, 0,   1,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(transmit_frame_ptr, 1,   my_address,           OPC_FIELD_SIZE_UNCHANGED);
```

```
    op_pk_fd_set_int32(transmit_frame_ptr, 2,   0,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_pkt(transmit_frame_ptr, 3, seg_pkptr, OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_dbl(transmit_frame_ptr, "Tx Data Rate", 900000, OPC_FIELD_SIZE_UNCHANGED);
    packet_size_dcf = op_pk_total_size_get(transmit_frame_ptr);
    //op_pk_send (transmit_frame_ptr, 6);
    op_prg_list_insert(ms_packet_queue, transmit_frame_ptr, OPC_LISTPOS_TAIL);
    packet_count = packet_count + 1;
    rem_size -= SEG_SIZE;
}
ulr_ptr = op_pk_create_fmt("ofdm_ul_req");
op_pk_fd_set_int32(ulr_ptr, 0,   1,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 1,   my_address,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 2,   packet_count,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 3,   tos,           OPC_FIELD_SIZE_UNCHANGED);
op_prg_list_insert(ms_ulr_queue, ulr_ptr, OPC_LISTPOS_TAIL);
op_pk_destroy(pk_ptr);
```

## A.5. Wireless Access Router Grant Access

```
Packet* transmit_frame_ptr;
Packet* ulr_ptr;
Packet* pk_ptr;
Packet* seg_pkptr;
IpT_Dgram_Fields* pk_fd_ptr;
char dest_addr_str[IPC_ADDR_STR_LEN];
char src_addr_str[IPC_ADDR_STR_LEN];
SimT_Pk_Id pkt_id, pkt_tree_id;
FHOFDM_Data d;
int i;
int rem_size;
int tos;
int packet_count;

++pk_count;
strm = op_intrpt_strm();
//printf("stream %d %d %d\n",strm,instrm_from_mac_if,op_intrpt_strm());
received_packet = op_pk_get(strm);
op_stat_write(pk_cnt_stathandle, pk_count);
//wlan_higher_layer_data_arrival();
//fhofdm_frame_transmit();
//op_pk_send( tmp_packet_ptr, LOW_LAYER_OUTPUT_STREAM);
//if (ap_flag != OPC_BOOLINT_ENABLED) {
//fhofdm_transmit_packet(received_packet);
pk_ptr = op_pk_copy(received_packet);
```

```
pkt_id = op_pk_id(pk_ptr);
pkt_tree_id = op_pk_tree_id(pk_ptr);
op_pk_nfd_access(pk_ptr, "fields", &pk_fd_ptr);
inet_address_print(dest_addr_str, pk_fd_ptr->dest_addr);
inet_address_print(src_addr_str, pk_fd_ptr->src_addr);

tos = get_priority(received_packet);
//printf("--------- tos %d \n",tos);
/* Determine the size of the source packet. */
rem_size = op_pk_total_size_get(received_packet);
/* Insert the source packet into segmentation buffer. */
op_sar_segbuf_pk_insert(ofdm_fragbuf_ptr, received_packet, 0);
/* Remove and send the first segment associated with arriving packet. */
/* If there is a remainder of source packet, schedule next segment transmission.*/
packet_count = 0;
while (rem_size > 0) {
    //op_intrpt_schedule_self (op_sim_time () + SEG_TX_TIME, SEG_TX_CODE);
    seg_pkptr = op_sar_srcbuf_seg_remove(ofdm_fragbuf_ptr, SEG_SIZE);
    transmit_frame_ptr = op_pk_create_fmt("ofdm_mac_data");
    op_pk_fd_set_int32(transmit_frame_ptr, 0,   1,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(transmit_frame_ptr, 1,   my_address,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_int32(transmit_frame_ptr, 2,   0,           OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_pkt(transmit_frame_ptr, 3, seg_pkptr, OPC_FIELD_SIZE_UNCHANGED);
    op_pk_fd_set_dbl(transmit_frame_ptr, "Tx Data Rate", 900000, OPC_FIELD_SIZE_UNCHANGED);
    packet_size_dcf = op_pk_total_size_get(transmit_frame_ptr);
    //op_pk_send (transmit_frame_ptr, 6);
    op_prg_list_insert(ms_packet_queue, transmit_frame_ptr, OPC_LISTPOS_TAIL);
    packet_count = packet_count + 1;
    rem_size -= SEG_SIZE;
}

ulr_ptr = op_pk_create_fmt("ofdm_ul_req");
op_pk_fd_set_int32(ulr_ptr, 0,   1,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 1,   my_address,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 2,   packet_count,           OPC_FIELD_SIZE_UNCHANGED);
op_pk_fd_set_int32(ulr_ptr, 3,   tos,           OPC_FIELD_SIZE_UNCHANGED);
op_prg_list_insert(ms_ulr_queue, ulr_ptr, OPC_LISTPOS_TAIL);

op_pk_destroy(pk_ptr);
```

# REFERENCES


1. Kim, Y., B.J. Jeong, J. Chung, C. Hwang, J.S. Ryu, K. Kim, and Y.K. Kim, "Beyond 3G: vision, requirements, and enabling technologies," *Communications Magazine, IEEE* , vol.41, no.3, pp. 120-124, Mar 2003.

2. "IEEE Std 802.20," in http://www.ieee802.org/20/, accessed 16 December 2007.

3. "OPNET Modeller," in *http://www.opnet.com/solutions/network_rd/modeler.html*, accessed 16 December 2007.

4. Hui, S.Y. and K.H. Yeung, "Challenges in the migration to 4G mobile systems," *Communications Magazine*, IEEE , vol.41, no.12, pp. 54-59, Dec. 2003.

5. Tugcu, T., H.B. Yilmaz, and F.S. Vainstein, "Analytical modeling of CAC in next generation wireless systems," *Computer Networks* vol 50, no 17, pp 3466-3484, Dec. 2006.

6. , Choi, Y., K.B. Lee, and S. Bahk, "All-IP 4G Network Architecture for Efficient Mobility and Resource Management," *IEEE Wireless Communications Magazine*, vol. 14, issue 2, pp. 42-46, Apr. 2007.

7. "Flash-OFDM," in *http://www.qualcomm.com/technology/flashofdm/index.html*, accessed 16 December 2007.

8. "Orthogonal Frequency Division Multiplexing," U.S. Patent No. 3, 488,4555, filed November 14, 1966, issued Jan 6, 1970.

9. Nee, R.V. and R. Prasad, "OFDM for Wireless Multimedia Communications," Norwood, MA: Artech House, Jan. 2000.

10. IEEE 802.11-1999, "IEEE Standard for Local and Metropolitan Area Networks Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC)

and Physical Layer (PHY) Specifications," Jun. 12, 1999.

11. ETSI HiperLAN/2, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; System Overview," Feb. 2002.

12. IEEE 802.16-2004, "IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems," Oct. 2004.

13. ETSI HiperMAN, "Broadband Radio Access Networks (BRAN); Functional Requirements for Fixed Wireless Access Systems Below 11 GHz: HIPERMAN," Mar. 2001.

14. Vaughan-Nichols, S.J., "OFDM: back to the wireless future," *Computer* , vol.35, no.12, pp. 19-21, Dec 2002.

15. Prasad, R., "OFDM for Wireless Communications Systems," Boston, Artech House, 2004.

16. Litwin, L. and M. Pugel, "The Principles of OFDM," *RF Signal Processing*, January 2001.

17. Laroia, R., S. Uppala, and L. Junyi, "Designing a mobile broadband wireless access network," *Signal Processing Magazine*, IEEE , vol.21, no.5, pp. 20-28, Sept. 2004.

18. Osborne, E. and A. Simha, "Traffic Engineering with MPLS," In Cisco Press, July 2002.

19. Rosen, E., A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *IETF RFC 3031*, January 2001.

20. Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *IETF RFC 2475*, December 1998.

21. Faucheur, F.L., L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval,

and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services," *IETF RFC 3270*, May 2002.

22. Fineberg, V., "QoS Support in MPLS Networks," *MPLS/Frame Relay Alliance White Paper*, May 2003.

23. C. Perkins, Ed., "IP Mobility Support for IPv4," *IETF RFC 3344*, August 2002.

24. Johnson, D., C. Perkins, and J. Arkko, "Mobility Support in IPv6," *IETF RFC 3775*, June 2004.

25. Zahariadis, T., "Trends in the path to 4G," *Communications Engineer* , vol.1, no.1, pp. 12-15, Feb. 2003.

26. Baybakov, K., T. Sviridova, M. Lobur, and R. Kohut, "Using OFDM for multiple access schemes in 4G communication system," *CAD Systems in Microelectronics, 2003. CADSM 2003. Proceedings of the 7th International Conference. The Experience of Designing and Application of*, vol.18, no.22, pp. 572-573, Feb. 2003.

27. Zhang, P. and L. Li, "Research on beyond 3G mobile communications," *Communication Technology Proceedings*, 2003. ICCT 2003. International Conference on , vol.1, no., pp. 28-31 vol.1, 9-11 April 2003.

28. Guo, Y., Z. Antoniou, and S. Dixit, "IP Transport in 3G Radio Access Networks: an MPLS-based Approach," in *Proc. IEEE Wireless Communications and Networking Conference*, vol. 1, pp. 1116, Mar. 2002.

29. Chueh, H. and K. Wang, "An all-MPLS approach for UMTS 3G core networks," *Vehicular Technology Conference*, 2003. VTC 2003-Fall. 2003 IEEE 58th , vol.4, no., pp. 2338-2342 Vol.4, 6-9 Oct. 2003.

30. Misra, I.S. and A. Banerjee, "MPLS based mobility framework in 4G architectures," *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region* , vol.2, no., pp. 670-674 Vol.2, 15-17 Oct. 2003.

31. Langar, R., S. Tohme, and G.L. Grand, "Micro Mobile MPLS: A New Scheme for Micro-mobility Management in 3G All-IP Networks," iscc, pp. 301-306, *10$^{th}$ IEEE Symposium on Computers and Communications (ISCC'05)*, 2005.

32. Zhou, H., C. Yeh, and H.T. Mouftah, "Dynamic hierarchical mobile MPLS for next generation all-IP wireless networks," *Vehicular Technology Conference*, 2005. VTC 2005-Spring. 2005 IEEE 61$^{st}$ , vol.4, no., pp. 2230-2234 Vol. 4, 30 May-1 June 2005.

33. Fineberg, V., C. Chen, and X. Xiao, "An end-to-end QoS architecture with the MPLS-based core," *IP Operations and Management, 2002 IEEE Workshop on* , vol., no., pp. 26-30, 2002

34. Malis, A. and D. Sinicrope, "MPLS PVC UNI Implementation Agreement: Baseline Text," *MPLS Forum draft*, 2002.75, May 2002.

35. Tanenbaum, A.S. and A.S. Woodhull, "Operating Systems Design and Implementation," New Jersey, Prentice Hall, 1997.

36. Zhang, D. and D. Ionescu, "QoS Performance Analysis in Deployment of DiffServ-aware MPLS Traffic Engineering," snpd, pp. 963-967, *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, 2007

37. Minei, I., "MPLS DiffServ-aware Traffic Engineering," in *http://www.juniper.net/solutions/literature/white_papers/200048.pdf*, accessed 21 December 2007.