國立交通大學

資訊科學與工程研究所

碩 士 論 文

IEEE 802.16e 行動 WiMAX 網路之省電類別管理

Management of Power Saving Classes in IEEE 802.16e Mobile WiMAX Networks

研 究 生：楊雁智

指導教授：曾煜棋　教授

中 華 民 國 九 十 七 年 八 月

IEEE 802.16e 行動 WiMAX 網路之省電類別管理
Management of Power Saving Classes in IEEE 802.16e Mobile WiMAX Networks

研 究 生：楊雁智　　Student：Yen-Chih Yang

指導教授：曾煜棋　　Advisor：Yu-Chee Tseng

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

August 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年八月

IEEE 802.16e 行動 WiMAX 網路之省電類別管理

學生：楊雁智　　　　　　　　　　　　指導教授：曾煜棋 博士

國立交通大學
資訊科學與工程研究所 碩士班

摘　　要

IEEE 802.16e 標準被制訂用來提供都會網路中的行動寬頻無線網路存取，行動性意謂了電源管理在行動用戶端 (Mobile Subscriber Station, MSS)的重要性。IEEE 802.16e 中定義了三種省電類別 (power saving class, PSC)，一個資料流 (traffic flow) 可以和一個省電類別作關聯。然而，從行動用戶端的角度來看，標準並未定義當有多個資料流時，哪些要被放在同一個省電類別中，以及在一個行動用戶端中，多個省電類別要如何相互配合運作，來達到更好的能源效率 (energy efficiency)。同時，也要解決如何決定省電類別的參數，像是起始訊框 (start frame)、聆聽視窗 (listening window) 大小、睡眠視窗 (sleep window) 大小，以及如何保證當多個省電類別存在時，服務品質 (Quality of Service, QoS) 能夠被維持。在本篇文章裡，我們提出了一個新穎的“交疊及解多工”方式，利用了 type I 和 type II 兩種省電類別，為 IEEE 802.16e 的網路提出了上述問題的解決方法。給定一組同一行動用戶端中的資料流，我們的方法首先試著為每個資料流制定出一個省電類別，以滿足各自的服務品質需求，然後我們會將它們交疊在一個很長的序列中，以計算總頻寬需求 (bandwidth requirement)。最後，我們會將這個序列解多工 (demultiplexing) 成多個省電類別，每一個用來支援一或多個資料流。此方法最後可達到高能源效率且能滿足每個資料流的服務品質需求。

Management of Power Saving Classes in IEEE 802.16e Mobile WiMAX Networks

Student：Yen-Chih Yang　　　　Advisor：Yu-Chee Tseng

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

The IEEE 802.16e standard has been defined to provide mobile broadband wireless access in metropolitan areas. Mobility implies the importance of power management at the Mobile Subscriber Station (MSS) side. In IEEE 802.16e, three types of power saving classes (PSCs) are defined. A traffic flow can be bound to a PSC. However, from an MSS's point of view, it does not define how multiple flows should be put into one PSC and how multiple PSCs of an MSS should cooperate with each other for better energy efficiency. At the same time, it needs to answer how to determine the parameters of each PSC, such as start frame, listening window size, and sleep window size, and how to guarantee QoS of traffic flows when multiple PSCs coexist. In this paper, we propose a novel ``fold-and-demultiplex'' method for an IEEE 802.16e network with PSCs of type I and type II. Given a set of traffic flows in an MSS, our method first tries to give each one a PSC satisfying its QoS requirement. Then we fold them together into one long series so as to calculate the total bandwidth requirement. Finally, we demultiplex the series into multiple PSCs, each supporting one or multiple flows. This ends up with high energy efficiency for MSSs while meeting flows' QoS requirements.

Index Terms —IEEE 802.16e, link protocol, MAC protocol, power management, WiMAX, wireless network.

ACKNOWLEDGE

本篇碩士學位論文的產生首先要感謝曾煜棋老師的指導，讓我有機會進到這個實驗室來學習，並且在一次次的討論中令我真正體驗到研究時如何細心觀察，大膽假設，小心求證的實際過程。您是極少數我見過工作認真，又能兼顧家庭，而且即使開始進入行政體系後，對於研究仍然相當有熱忱的教授之一。

接下來要感謝陳建智學長在論文上的諸多指導及付出的心力，讓我學到論文撰寫的概念及許多細節，以及在探討一個研究議題時從頭到尾的完整過程，讓原本粗糙的想法能夠變得完整而嚴謹。另外你也是我見過帶人最和氣、用心的學長。同時，也要感謝梁家銘學長在口試準備時的指導，讓我實際學到 presentation 的流程架構、言詞表達以及投影片的表現方式，讓原本沒人聽得懂的 presentation 變得深入簡出。

此外，感謝在這兩年中一起生活的研究所同學，志誠、書賢、亭瑩、婉婷、子文、文杰、小晉、怡廷，你們都是相當聰明而且認真的人才，而且大家相處起來也都很理性、和諧，完全沒有像一些其他實驗室會有相處不睦的現象發生。另外也感謝李淩、幼棣、倫武…等諸位實驗室的學長姐和學弟妹，共組了一個不但氣氛融洽，而且又很用功的好實驗室，可見當初我的選擇確實是對的。還有，特別感謝芳如、禹安和楷婷幾位朋友，豐富了我的碩士生活而不致於枯燥煩悶，也讓我看到其他碩士生辛苦的地方，雖然妳們的過程比較不順遂，但終究是熬過來了呢。

最後要感謝我的父母家人，讓我能夠放心自由地出門在外學習、工作及生活。感謝所有在這兩年中點綴過我生活的人，生活就是這所有的片段交織而成的，謝謝你們。

Contents

List of Tables

List of Figures

Chapter 1. Introduction

IEEE 802.16/WiMAX [1] has been considered as a promising approach for supporting broadband wireless access. The IEEE 802.16e standard [2] defines the mobility support for Mobile Subscriber Stations (MSSs). Similar to most wireless systems, conserving energy is a critical issue for MSSs. In IEEE 802.16e, three types of *Power Saving Classes (PSCs)* are defined to meet different traffic characteristics. Each PSC consists of a sequence of interleaved listening and sleep windows, and can support one or multiple traffic flows in an MSS with similar characteristics. Type I is designed for non-real-time traffic flows; it has an exponentially increasing size of sleep windows if no packet comes. Type II is designed for real-time traffic flows; it has a fixed size of sleep windows. Type III is designed for multicast connections or management operations. An MSS can turn off its radio interface when all its PSCs are during their sleep windows, but has to wake up when any of its PSCs is during a listening window. However, from an MSS's point of view, the specification does not define how multiple flows should be put into one PSC and how multiple PSCs of an MSS should cooperate with each other for better energy efficiency. At the same time, it needs to answer how to determine the parameters of each PSC, such as start frame, listening window size, and sleep window size, and how to guarantee QoS of traffic flows when multiple PSCs coexist.

Several works have addressed energy management in an 802.16e network. Some have focused on performance analysis [3, 4, 5]. Focusing on PSCs of type I, [6] proposes a *Longest Virtual Burst First (LVBF)* scheduling algorithm to improve the energy efficiency of MSSs, while [7] presents an adaptive scheme to dynamically adjust the initial and the maximum sleep window sizes. For a group of PSCs of Type II, [8] shows how to find the *Maximum Unavailability Interval (MUI)* by only adjusting their start frames. However, it does not consider how to satisfy the bandwidth requirement of each connection after the rescheduling. Considering real-time connections, [9] proposes a *Periodic On-Off* Scheme to form one PSC of Type II for all connections, such that the lengths of sleep and listening windows are constrained by delay and resource requirements. Since the scheme uses only one PSC, it may suffer from either longer wake-up time or mismatch of bandwidth requirements. As to be shown in our approach, using multiple PSCs will be more energy-efficient.

In this work, we are interested into PSCs of type I and II. Management of PSCs needs to answer the following questions: 1) How to form PSCs given multiple connections? 2) How to arrange each PSC's sleep and listening windows? 3) How to meet the QoS requirement of each connection while at the same time maximize the sleep time of each MSS? In this paper, we propose a novel "fold-and-demultiplex" method to answer these questions. Given a set of real-time traffic flows in an MSS and their QoS requirements, our method first tries to give each flow a PSC satisfying its QoS requirement. Then we fold them together into one long series so as to calculate the overall bandwidth requirement. Finally, we demultiplex the series into multiple PSCs each supporting one or multiple flows and include non-real-time flows. This ends up with longer sleeping time for the MSS while meeting all flows' QoS requirements. To the best of our knowledge, this "fold-and-then-demultiplex" method is also the first one of this kind for the power management purpose. This paper is organized as follow: Section 2 reviews the definitions of PSCs. Section 3 presents our system model and method. Sections 4 and 5 give some analytical and simulation results, respectively. Conclusions are drawn in Section 6.

Chapter 2. Backgrounds

IEEE 802.16e defines three types of PSCs for an MSS. Type I is to support Best Effort and NRT-VR (Non-Real-Time Variable Rate) connections. Type II is to support UGS (Unsolicited Grant Service), RT-VR (Real-Time Variable Rate), and ERT-VR (Extended-Real-Time Variable Rate) connections. When an MSS activates its sleep operation, each PSC will switch between listening and sleep modes. Each connection is bound to a PSC. During a sleep window, the corresponding connections cannot send or receive packets. So, when all PSCs of an MSS are in their sleep windows, the MSS can turn off its air interface to save energy. This period is called an *unavailability interval* of the MSS.

A PSC of type I is denoted by P^I. Its sleep windows are interleaved by fixed-length listening windows of size $P^I.T_L$. Its initial sleep window size is $P^I.T_{S\_init}$, and is doubled each time, until reaching the maximum size, $P^I.T_{S\_max}$, after which it remains the same. During a listening window, the MSS will check if there are incoming packets for P^I. If not, P^I will enter another sleep window; otherwise, it will deactivate P^I and return to normal state. A PSC of type II is denoted by P^{II}. Both its sleep windows and listening windows are of fixed lengths $P^{II}.T_S$ and $P^{II}.T_L$, respectively. However, if there is any transmission/reception during a listening window, it will not return to normal state unless being instructed. A PSC of type III has only one sleep, after which it will return to normal state immediately. Fig. 2.1 illustrates these definitions. The unit of sleep/listening windows is Orthogonal Frequency Division Multiplexing (OFDM) frame length, normally 5 ms. A frame can be divided into a downlink subframe and an uplink subframe.

In order to support different data delivery services, the BS needs to allocate band-

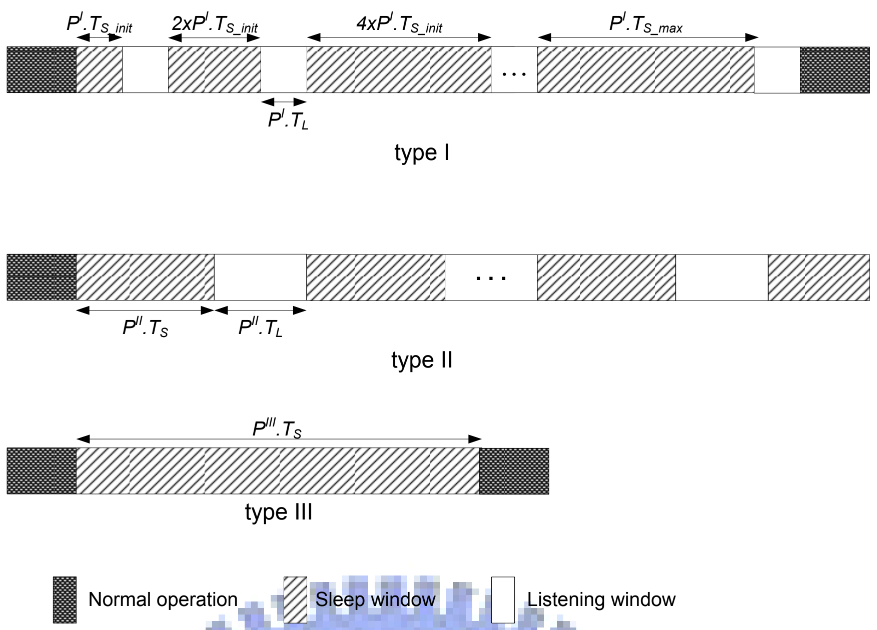


Figure 2.1: Three types of PSCs.

widths to flows according to their traffic parameters. For downlink flows, this is an easy job because the BS has all flows' information. However, for uplink flows, this is not the case. Below, we review three scheduling services for uplink transmissions. UGS is designed for real-time services with periodical fixed-size packets, such as VoIP connections. The BS will periodically assign fixed-size grants to an UGS connection. For real-time services with variable-size packets, such as MPEG videos, rtPS (real-time Polling Service) can be used. The BS can assign periodical uplink resource to an rtPS flow to send its bandwidth requests. In return, the BS can grant variable uplink bandwidths to it. For real-time services with periodical fixed-size packets interleaved by intermittent silence, such as VoIP connections with silence suppression, ertPS (extended real-time Polling Service) can be used. Initially, the BS will assign periodical fixed-size grants to the connection. Once the connection has nothing to send because of silence, it will inform the BS by using the granted bandwidth. Then, in the following frames, the granted bandwidth will be decreased to the size of a bandwidth request. When the connection becomes active again,

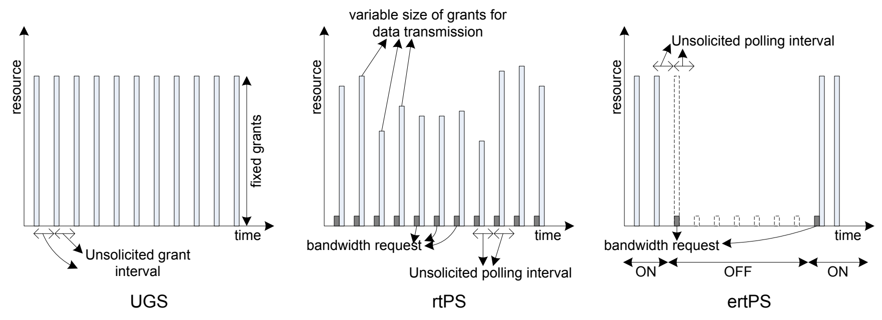

resource
fixed grants
time
Unsolicited grant interval
UGS
variable size of grants for data transmission
resource
time
bandwidth request
Unsolicited polling interval
rtPS
Unsolicited polling interval
resource
time
bandwidth request
ON
OFF
ON
ertPS

Figure 2.2: Bandwidth allocation of service classes UGS, rtPS, and ertPS.

it will inform the BS and the BS will restore its periodical fixed bandwidth. Fig. 2.2 illustrates some examples. Note that the actual allocation is controlled by QoS parameters such as Minimum Reserved Traffic Rate (MRTR), Maximum Sustained Traffic Rate (MSTR), Maximum Latency, Unsolicited Grant Interval (UGI), and Unsolicited Polling Interval (UPI).

Chapter 3. Fold-and-Demultiplex Method

Given an uplink to downlink subframe ratio, we consider an MSS with U non-real-time connections and $M+N$ real-time connections, among which M are for uplink and N are for downlink. The QoS parameters of each connection are already known to the MSS. We assume that the maximum acceptable delay of each connection is greater than or equal to three times of its packet inter-arrival time; this assumption will allow us to prove the delay bound in Section 4.1. Let F be the length of an OFDM frame. Our goal is to compute a set of PSCs to maximize the unavailability intervals of the MSS while meet all connections' QoS requirements. For each real-time connection C_i, $i = 1 \cdots M+N$, the following QoS parameters are known:

- $C_i.D_{max}$: The maximum delay that a packet can tolerate.
- $C_i.MRTR$: The minimum reserved traffic rate (bits/sec).
- $C_i.I_d$: The expected packet inter-arrival time.
- $C_i.I_{pol\_grt}$: For an uplink rtPS or ertPS connection, it is the UPI; for an uplink UGS connection, it is the UGI. (This parameter is not used for a downlink connection.)
- $C_i.S_u$: The Service Data Unit (SDU) size of C_i (this is only needed for a constant-rate VoIP connection.)

For non-real-time connections, they don't have maximum delay and jitter constraints. And each time the data transmission, the MSS should be keep awake until all the buffered packets are send/received, so there is no need to consider the bandwidth requirement in the power management scheme. In our method, all non-real-time connections are associated with the same one PSC of type I, and we let this PSC's listening windows overlap with those of type II PSCs' to receive traffic indication. The QoS parameters of non-real-time connections are not considered.

The outputs of our scheme are one PSC of type I and some PSCs of type II. The PSC of type I, denoted by P^I, is for non-real-time connections and has the following parameters:

- $P^I.T_L$: Size of a listening window.
- $P^I.T_{S\_init}$: Size of the initial sleep window.
- $P^I.T_{S\_max}$: Size of the maximum sleep window.
- $P^I.N_{start}$: Start frame number.

The ith PSC of type II, denoted by P_i^{II}, is for real-time connections and has the following parameters:

- $P_i^{II}.T_L$: Size of a listening window.
- $P_i^{II}.T_S$: Size of a sleep window.
- $P_i^{II}.N_{start}$: Start frame number.

Our scheme consists of four steps. The first one will consider only real-time connections and create a tentative PSC of type II for each connection, such that each one has a wake-up period that is an integer multiple of that with the shortest period. The second step will fold all these PSCs into one so as to calculate the overall bandwidth requirement. The third step will demultiplex the above result into multiple real PSCs. The last step will include non-real-time connections.

3.1 Creating Tentative PSCs

In this step, the MSS will consider only real-time connections. It will compute one tentative PSC for each connection and send the result to the BS via a MOB_SLP-REQ message. It includes three steps: a) For each real-time connection, create a tentative PSC of type II according to its QoS parameters. b) To increase the overlapping of listening windows, adjust these PSCs such that their periods are integer multiples of that with the smallest period. c) Adjust these connections' QoS parameters to adapt to the sleep behavior of PSCs.

Step a: For each C_i, $i = 1 \cdots M{+}N$, we define a temporary PSC P_i^{II} as follows:

$$P_i^{II}.T_L = \begin{cases} 2, & \text{if } C_i \text{ is of type rtPS} \\ 1, & \text{otherwise.} \end{cases} \tag{3.1}$$

$$P_i^{II}.T_S = \begin{cases} \left\lceil \dfrac{C_i.D_{max}}{2 * F} \right\rceil - P_i^{II}.T_L, & \text{if } C_i \text{ is of type UGS or ERT-VR for VoIP services} \\ \left\lceil \dfrac{C_i.D_{max}}{3 * F} \right\rceil - P_i^{II}.T_L, & \text{otherwise.} \end{cases} \tag{3.2}$$

For an rtPS connection, since it requires an additional frame to send its bandwidth request to the BS, its listening window should be two frames. For other connections, we assume that one frame is sufficient (later on, we will relax this.) $T_L{+}T_S$ can be considered as the wake-up period of a PSC. For an UGS/ERT-VR connection for VoIP services, Eq. (3.2) ensures that its wake-up period is no more than $\frac{1}{2}$ of its maximum delay. For other cases, it is no more than $\frac{1}{3}$ of its maximum delay. This guarentees that each packet will not exceed its maximum delay (refer to section 4.1). Note that these parameters are only tentative.

Step b: Next, the MSS will adjust these PSCs to increase their overlapping. Let P_{min}^{II} be the PSC with the smallest wake-up period. We will enforce each PSC's wake-up

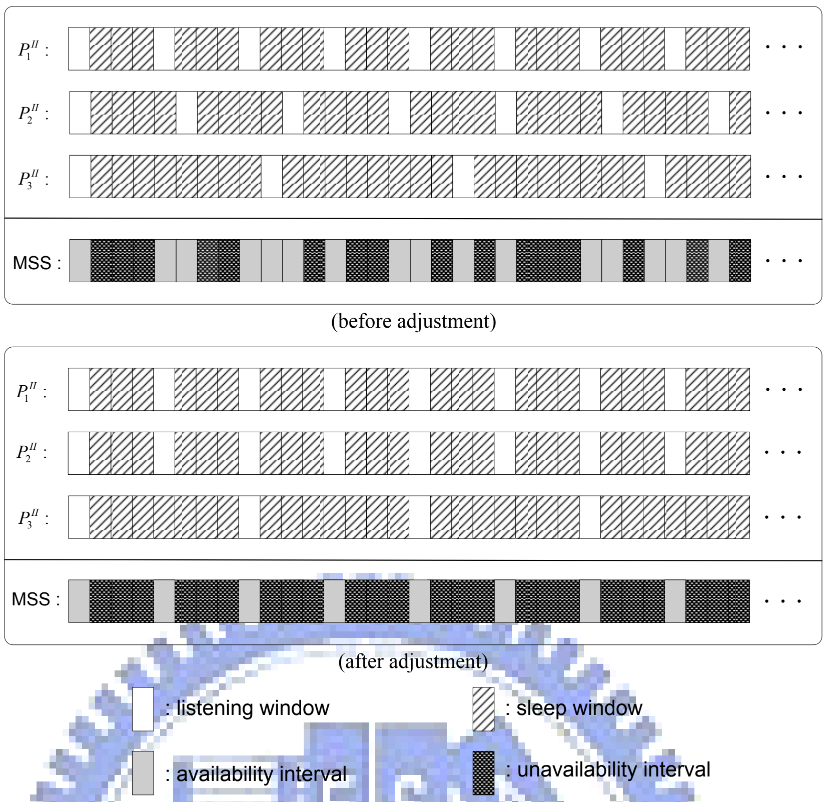


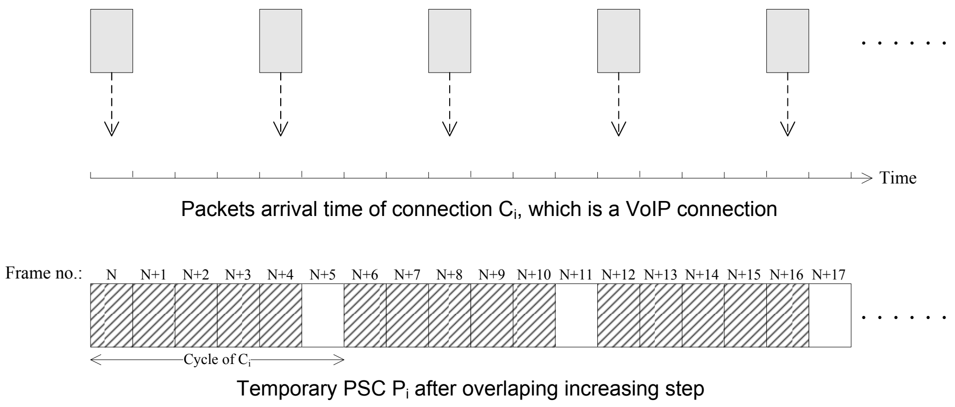
Figure 3.1: Example of the step b.

period to be an integer multiple of $(P_{min}^{II}.T_L + P_{min}^{II}.T_S)$. That is, for each P_i^{II}, we adjust its T_S as follows:

$$P_i^{II}.T_S = \left\lfloor \frac{(P_i^{II}.T_L + P_i^{II}.T_S)}{(P_{min}^{II}.T_L + P_{min}^{II}.T_S)} \right\rfloor \times (P_{min}^{II}.T_L + P_{min}^{II}.T_S) - P_i^{II}.T_L. \qquad (3.3)$$

For example, if there are three PSCs with parameters $P_1^{II}.T_L = 1$, $P_1^{II}.T_S = 3$, $P_2^{II}.T_L = 1$, $P_2^{II}.T_S = 4$, $P_3^{II}.T_L = 1$, and $P_3^{II}.T_S = 8$, then the smallest wake-up period is 4. So the adjustment result will be $P_2^{II}.T_S = 3$ and $P_3^{II}.T_S = 7$. Before the adjustment, the MSS has to wake-up 84 frames per 180 frames. After the adjustment, there will be only 1 wake-up frame per 4 frames (see Fig. 3.1).

Step c: Because of the MSS's sleeping behavior, a flow may need to deliver more traffic during a listening window. So changing its QoS parameters maybe needed. Consider the example in Fig. 3.2, where a connection has a packet inter-arrival time of 4



Time
Packets arrival time of connection C_i, which is a VoIP connection
Frame no.: N N+1 N+2 N+3 N+4 N+5 N+6 N+7 N+8 N+9 N+10 N+11 N+12 N+13 N+14 N+15 N+16 N+17
Cycle of C_i
Temporary PSC P_i after overlaping increasing step

Figure 3.2: Example of the mismatch between the packet arrival time of a flow and its wake-up period.

frames but its wake-up period is 6 frames. It may need to deliver two packets in a listening window if each packet is indivisible (such as VoIP packets). So we suggest to change the *MRTR* of each C_i of type UGS/ERT-VR as follows:

$$C_i.MRTR = \frac{1000}{(P_i.T_L + P_i.T_S)\cdot F}\cdot\left(\left\lceil\frac{(P_i.T_L + P_i.T_S)\cdot F}{C_i.I_d}\right\rceil\cdot C_i.S_u\right). \tag{3.4}$$

The term in parentheses is the bandwidth required per listening window. Spreading this on a wake-up period is C_i's *MRTR*. Besides, the grant interval/polling interval of C_i should be changed to its wake-up period:

$$C_i.I_{pol\_grt} = (P_i.T_L + P_i.T_S)\cdot F. \tag{3.5}$$

These changes can be updated by the management message DSC-REQ. The effect is a slight increase of bandwidth demand, but it can reduce the buffering delay of such as voice data. In Fig. 3.2, we will allocate space to deliver $\left\lceil\frac{6}{4}\right\rceil = 2$ packets per listening window, so that the *MRTR* will be changed to $\frac{1000}{6F}\times 2\times C_i.S_u$. The new grant interval will be $6F$.

3.2 Folding PSCs into a State Series

Next, we will fold the above PSCs into an infinite periodical series of real numbers standing for bandwidth requirements. Then depending on the available bandwidth per frame, the real series is converted into a binary series, standing for the active or sleeping state of each frame. Note that the series does not actually follow the definition of PSC. Later on, we will demultiplex it into real PSCs.

Consider PSC P_i^{II} of any type except rtPS. Due to its sleeping behavior, the amount of data b_i that it has to transmit during a listening window is its traffic rate times its wake-up period times, i.e.,

$$b_i = C_i.MRTR \times (P_i.T_L + P_i.T_S) \times F. \tag{3.6}$$

For PSC P_i^{II} of type rtPS, in each listening window, it only requires a fixed bandwidth Δb to submit its request in the first frame. In the 2nd frame, if needed, the same bandwidth of b_i in Eq. (3.6) is required.

For a non-rtPS connection C_i, its bandwidth requirement in each frame can be represented by an infinite real series $S_i(t), t = 0 \cdots \infty$, where $k \in N$ and $T_i = P_i.T_L + P_i.T_S$ is its wake-up period:

$$S_i\left(k \cdot T_i + t\right) = \begin{cases} b_i, & t = 0 \\ 0, & \text{otherwise.} \end{cases}$$

For an rtPS connection C_i, the series can be defined as

$$S_i\left(k \cdot T_i + t\right) = \begin{cases} \Delta b, & t = 0 \\ b_i, & t = 1 \\ 0, & \text{otherwise.} \end{cases}$$

To represent the bandwidth requirement in each frame, we fold these series into one by summing corresponding values together:

$$\hat{S}\left(t\right)=\sum_{t\geq 0} S_i\left(t\right).$$

Note that the above discussion does not distinguish uplink from downlink. In fact, the above series $\hat{S}(t)$ should be separated into two series, one for uplink and one for downlink. With this understanding, we will still use $\hat{S}(t)$ for simplicity.

Theorem 1. *$\hat{S}\left(t\right)$ is a periodical series of period $= lcm\{T_i \mid i = 1\cdots M+N\}$.*

Proof. Let $n = lcm\{T_i \mid i = 1\cdots M+N\}, k \in N, t \geq 0,$

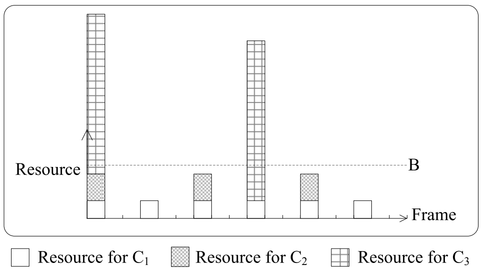
$S_i\left(k\cdot n+t\right)=S_i\left(k\cdot \frac{n}{T_i}\cdot T_i+t\right)=S_i\left(t\right)$, where $i = 1\cdots M{+}N$

That means $S_i\left(t\right)$ is a periodical series of period n. Here, $\hat{S}\left(t\right)=\sum_{t\geq 0} S_i\left(t\right)$, so $\hat{S}\left(t\right)$ is also a periodical sequence of period n. □

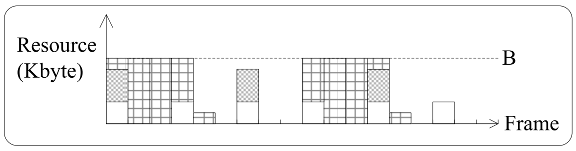
Next, we convert the bandwidth requirement series $\hat{S}(t)$ to a binary *state series* $\tilde{S}(t)$, where 1 and 0 mean avtive and idle states respectively:

$$\tilde{S}\left(t\right)=\begin{cases}1, & \text{if } a\left(t\right)>0\\ 0, & \text{otherwise}\end{cases}, \tag{3.7}$$

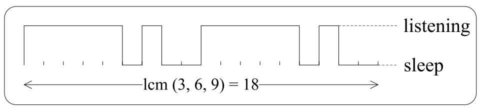
where $a(t)$ is the actual bandwidth to be consumed by the MSS in the t-th frame and B is the maximum bandwidth that can be allocated to the MSS in a frame decided by the BS. Intuitively, $\sum_{k=0}^{t}\hat{S}\left(k\right)$ is the total bandwidth required up to the t-th frame and $\sum_{k=0}^{t-1}a\left(k\right)$ is the actual bandwidth consumed by the MSS up to the $(t-1)$-th frame. So the first term in min() is the actual bandwidth required in the t-th frame; however, the actual bandwidth consumed should not exceed B. If $a(t)>0$, the MSS should by active



(a) Resource requirement of the MSS.



(b) Resource allocation for the MSS.



(c) The sleep-listening state waveform of the MSS.

Figure 3.3: Example of the state series construction.

in the t-th frame, enforcing $\tilde{S}(t) = 1$; otherwise it can go to sleep.

$$a(t) = \min\left(\sum_{k=0}^{t} \hat{S}(k) - \sum_{k=0}^{t-1} a(k), B\right). \tag{3.8}$$

Figure 3.3 is an example of above procedure. Fig. 3.3(a) shows the total bandwidth requirement of the MSS of three connections (i.e., $\hat{S}(t)$). Fig. 3.3(b) is the actual consumed resources by the MSS (i.e., $a(t)$). Fig. 3.3(c) is the state series $\tilde{S}(t)$ of the MSS.

Below, we will show that $\tilde{S}(t)$ is a periodical binary series. Define $n = lcm\{T_i \mid i = 1 \cdots M + N\}$ and $m = \min\{T_i \mid i = 1 \cdots M + N\}$

Theorem 2. *$\tilde{S}(t)$ is a periodical binary series of period* $= lcm\{T_i \mid i = 1 \cdots M + N\}$.

Proof. In the beginning, we first shows that $\tilde{S}(t)$ can be divided into multiple periodical series of period p, where $m|p$ and $p|n$. $\tilde{S}(t)$ is a binary series which contains lots of continuous 0s and 1s. For each group of continuous 1s, which starts from t_0 and ends in t_s, it contains resource requirements from a set of connections $S_c = \left\{C_i \,\middle|\, \sum_{k=t_0}^{t_s} S_i(k) > 0\right\}$, such that the MSS has to be awake for $(t_s - t_0 + 1)$ frames long to satisfy these bandwidth requirements. Since all $S_i(k)$ of $C_i \in S_c$ are periodical, the same bandwidth requirements occur during time duration $[t_0, t_s]$ must occur again in duration $[t_0 + i \cdot p, t_s + i \cdot p]$, where

$i \in Z$ and $p = lcm\{T_i \mid C_i \in S_c\}$. Therefore, series $\tilde{S}(t)$ must be all the during time $[t_0 + i \cdot p, t_s + i \cdot p]$. So, we can use a periodical series with cycle p and length of 1s as $(t_s - t_0 + 1)$ to cover the continuous 1s during $[t_0, t_s]$ and others. Note that p is a multiple of m and a factor of n because $S_c \subset S = \{C_i \mid i = 1 \cdots M + N\}$. Now, we have known that, for each group of continuous 1s in $\tilde{S}(t)$, say group G_j, we can generate a periodical series $\tilde{S}_j(t)$ for it. So, $\tilde{S}(t)$ is a periodical series. Also, we know that each $\tilde{S}_j(t)$ is with period P_j, where P_j is the lcm of a subset of MSS's real-time connections' T_i. Since a connection's resource requirement will always occur in some G_js, the period of $\tilde{S}(t) = lcm\{P_j \mid \forall j\} = lcm\{T_i \mid i = 1 \cdots M + N\}$. □

3.3 Demultiplexing the State Series into PSCs

The above state series $\tilde{S}(t)$ does not follow the definition of PSC. Recall that a PSC contains three components: listening window size, sleep window size, and start frame. So, we have to divide $\tilde{S}(t)$ into multiple state series each meeting the definition of PSC. Actually, the proof of Thm. 2 implies a way to divide $\tilde{S}(t)$ into multiple PSCs. However, for each continuous 1s period, this way will generate a PSC for it. To decrease the number of PSCs, we propose another method in the following.

1. [Initialization] Let p_{lcm} be the period of the state series $\tilde{S}(t)$ and p_{min} be the smallest wake-up period of the tentative PSCs in Sec. 3.1. Define set $C = \emptyset$, which for storing binary strings, each to interpested as a PSC. We define a temporary binary array $W[0 : p_{lcm} - 1]$ which contains all elements in C. Our goal is to construct a set of PSCs eventually reach the condition $W[0 : p_{lcm} - 1] = \tilde{S}[0 : p_{lcm} - 1]$.

2. Recall that p_{lcm} is an integer multiple of p_{min}. For each $i = 1$ to $\frac{p_{lcm}}{p_{min}}$, such that

$i \cdot p_{min}$ is a factor of p_{lcm}, we will try to compute at most one PSC by executing the following steps:

(a) Let's denote by $\tilde{S}[j : k]$ the substring of $\tilde{S}(t)$ from the j-th bit to the k-th bit. Since $(i \times p_{min})$ is a factor of p_{lcm}, we cut $\tilde{S}\,[0 : p_{lcm} - 1]$ into $\frac{p_{lcm}}{i \times p_{min}}$ segments, each of length $i \times p_{min}$. Then we let binary string $T\,[0 : i \cdot p_{min} - 1]$ be the bit-wise-AND of these $\frac{p_{lcm}}{i \cdot p_{min}}$ segments, i.e.,

$$T[j] = \tilde{S}(j) \wedge \tilde{S}\,(j + i \cdot p_{min}) \wedge \tilde{S}\,(j + 2i \cdot p_{min}) \wedge \cdots$$

. Next, except the 1st group of continuous 1 in binary string T, we set 0 to all other bits.

(b) For each binary string $T'\,[0 : k \cdot p_{min} - 1]$ in set C, if its period is a factor of binary string T (means $k|i$), we compare T' against T bit by bit. When the same bit in both T' and T are 1 (means an active frame is covered by both T' and T), we set the bit in T as 0, i.e., when $T[j] = T'[j] = 1$, then $T[j] = 0$.

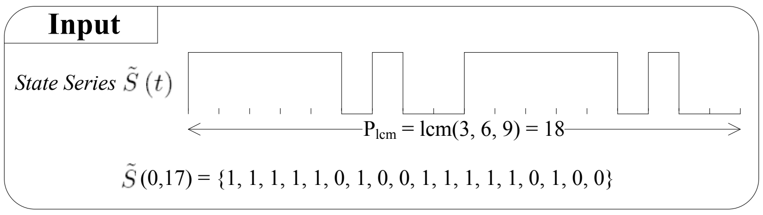
(c) If all bits in T are 0 then do nothing, else insert T into set C and we get a PSC with $T_L =$ number of 1s in binary string T and $T_S = i \cdot p_{min} - T_L$. Then we do bit-wise-OR to binary array $W[0 : p_{lcm} - 1]$ and T to update $W[0 : p_{lcm} - 1]$, i.e., $W[j] = W[j] \vee T[j\%\,(i \cdot p_{min})]$. If $W[0 : p_{lcm} - 1] = \tilde{S}[0 : p_{lcm} - 1]$, stop computing PSCs and go to step 4.

3. If $W[0 \cdots p_{lcm} - 1] \neq \tilde{S}[0 \cdots p_{lcm} - 1]$, mean there are some missing continuous 1s period in $W[0 \cdots p_{lcm} - 1]$, where $W[0 \cdots p_{lcm} - 1]$ contains all 0s in these periods. For each such period, generate a binary string T with length of p_{lcm}, T_L= number of 1s in the continuous 1s period, and $T_S = p_{lcm} - T_L$ and then insert T into set C.

4. Now, all the PSCs are in set C. Assume we active the sleep mode and the first listening frame will be at frame N, then for each PSC (each binary string T' in set C), we define its $N_{start} = N + N_0 - (T_S \text{ of } T')$, where N_0 is the number of continuous 0 from the start of T'.

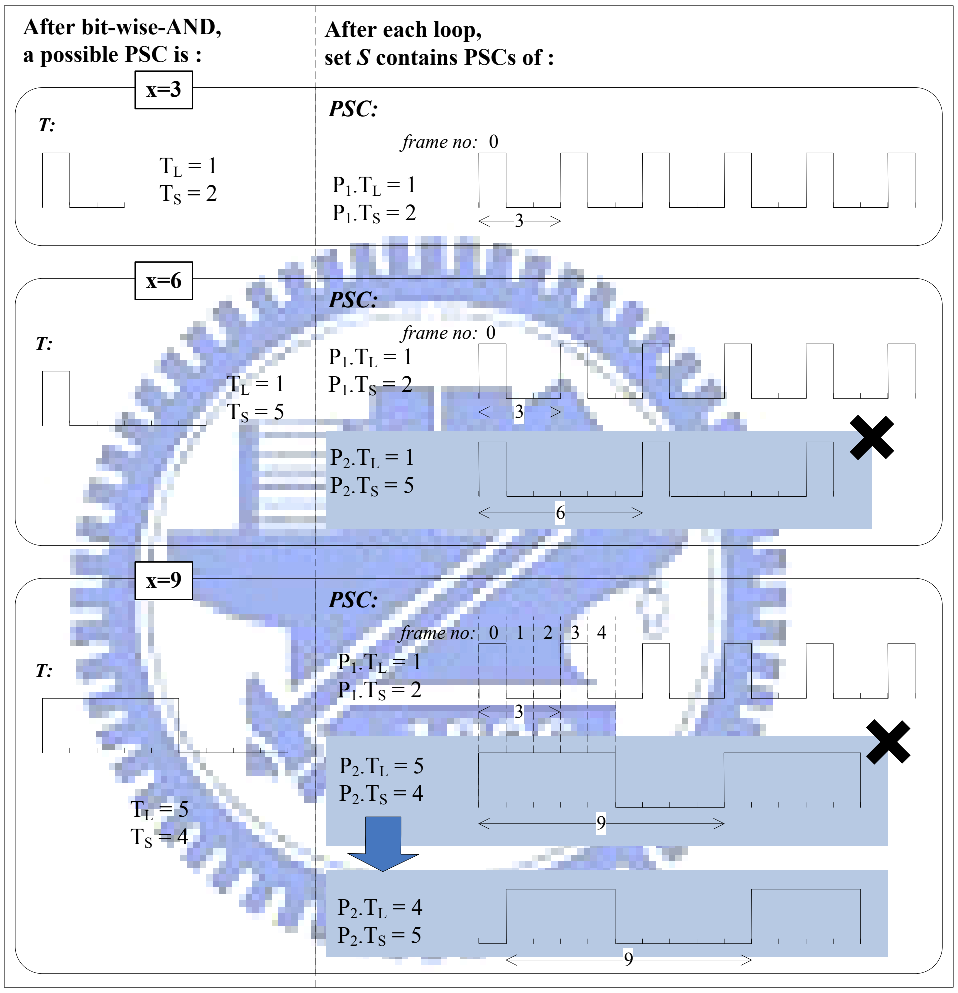
After above PSC demultiplexing procedure, the derived final PSCs are replied to the MSS by using MOB_SLP-RES message. For each final PSC, it is with all the real-time connections of the MSS as its member connections.

Below, let's see an example which is shown in Fig. 3.4. Suppose the state series $\tilde{S}(t)$ of the MSS has been known as Fig. 3.4(a) with $p_{lcm} = 18$ and $p_{min} = 3$. Then, we can start to divide $\tilde{S}(t)$ into multiple PSCs. In Fig. 3.4(b), first, $\tilde{S}(t)$ is cut into segments and each is with length of 3. By bit-wise-AND all the segments, we will get a binary string $T = \{1, 0, 0\}$. So, we get a PSC with $T_L = 1$ and $T_S = 2$. In the next loop, again, $\tilde{S}(t)$ is divided into segments with length of 6. A binary string $T = \{1, 0, 0, 0, 0, 0\}$ is obtained. But its active frames can be covered by the PSC in set C. So, we don't need to add it to set C. Next, $\tilde{S}(t)$ is divided into segments with length 9, and a binary string $T = \{1, 1, 1, 1, 1, 0, 0, 0, 0\}$ is conducted. After comparing the PSCs in set C and T, we will get a nonzero $T = \{0, 1, 1, 1, 1, 0, 0, 0, 0\}$. This is a PSC with $T_L = 4$ and $T_S = 5$. (shown in the 3rd subfigure of Fig. 3.4(b)). As yet the PSCs we find in set C are enough to compose the state series $\tilde{S}(t)$. Finally, two PSCs are conducted as shown in Fig. 3.4(c): P_1^{II} is with $T_L = 1$, $T_S = 2$, and $N_{start} = N - 2$, P_2^{II} is with $T_L = 4$, $T_S = 5$, and $N_{start} = N - 4$.

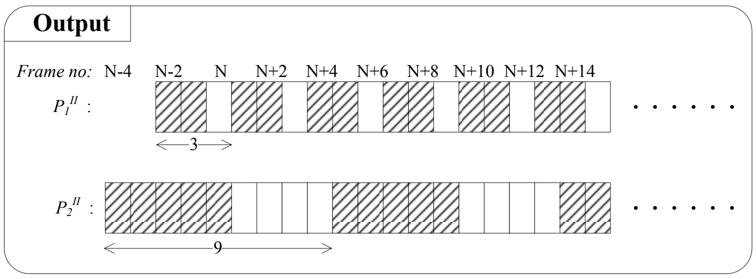


Input
State Series $\tilde{S}(t)$
P_{lcm} = lcm(3, 6, 9) = 18
$\tilde{S}$(0,17) = {1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0}

(a) State series $\tilde{S}(t)$ with $p_{lcm} = 18$ and $p_{min} = 3$.



(b) For each length x, find the longest listening period in T, and construct a PSC if needed.



(c) Final PSCs of type II.

Figure 3.4: Example of the PSC demultiplexing procedure.

3.4 Considering Non-realtime Connections: Allowing for Type I PSC

For non-real-time connections, we will define a PSC of type I, P^I, to be in charge of them. When P^I truns into listening mode, according to the standard, the MSS will receive a periodical broadcasting message called MOB_TRF-IND message to indicate whether there are any packets for the PSC buffered in the BS. The MOB_TRF-IND message is a broadcast message with a bitmap, where a certain bit will be set while there are packets for the corresponding PSC arrive the BS. Whenever there are packets arrive, P^I will be deactivated until the packets are transmitted, and then starts from the first sleep window again. To save energy, we can choose to let the listening windows of P^I to overlap with the listening windows of an existing PSC of type II.

After executing the steps in Sec. 3.1~3.3, we have derived multiple PSCs of type II. Then pick up the one with the smallest wake-up period p_{min} from them. Because the MSS has to wake up at least per p_{min}, we can let the listening windows of P^I overlap with these awake frames. It means that the maximum cycle of P^I will be multiple of p_{min}. The following shows how the T_L, T_{S_init}, T_{S_final}, and N_{start} of P^I be set.

Suppose the next listening window of the PSC with smallest cycle is at frame t. The listening window of P^I is assigned to 1 frame. For the initial sleep window, if there is a suggestion value from upper layer, we will adopt it; otherwise $P^I.T_{S\_init}$ will be assigned to 1 frame for less packet delay. To save evergy, it can be longer, but the value doesn't affect the average sleep ratio severely under the operation of existing PSCs of type II. The final sleep window is set to the multiple of p_{lcm}, because it should be larger than the initial sleep window, likewise we choose the smallest one which greater than $P^I.P_{S\_init}$ for less packet delay.

$$
\begin{aligned}
P^I.T_L &= 1 && (3.9)\\
P^I.T_{S\_init} &= \begin{cases} \text{Default value from upper layer if exist.} \\ 1, \ \text{otherwise.} \end{cases} && (3.10)\\
P^I.T_{S\_final} &= \left\lceil \frac{P^I.T_{S\_init}}{p_{lcm}} \right\rceil \cdot p_{lcm} && (3.11)\\
\text{Let } j &= P^I.T_{S\_init} \cdot \left(2^{k+1} - 1\right) + (k-1)\,, \\
& \quad \text{where } k = \max\left\{x \,\middle|\, P^I.T_{S\_init} \cdot 2^x < P^I.T_{S\_final}, x \in N\right\}. \\
P^I.N_{start} &= t + \left\lceil \frac{j}{p_{lcm}} \right\rceil \cdot p_{lcm} - j. && (3.12)
\end{aligned}
$$

3.5 Extended Method

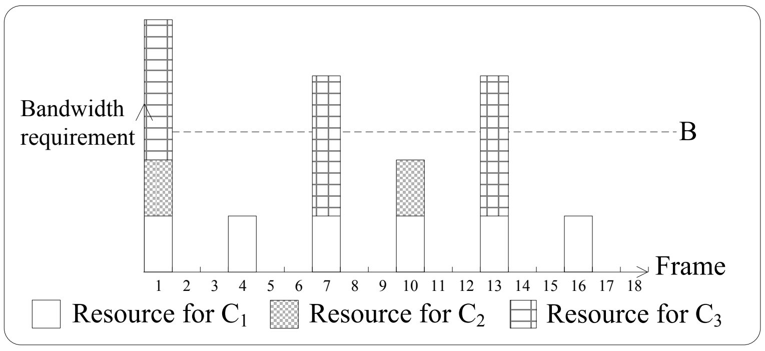
The method described above is complete to decide the listening and sleep periods of an MSS. However, we can further optimize the power consumption by adjusting the resource allocation for the MSS. Recall the resource reservation procedure in section 3.2, when the bandwidth requirement is larger than the maximum reserved size per frame (B_U/B_D), BS will always allocate resource in next frame for the exceeded bandwidth requirement until the requirement can be served. In this case, in an availability interval, the reserved resource for the MSS in the last awaked frame is very likely to be less than B_U/B_D. That is, for the MSS, this frame is not fully utilized. An extended method to improve the energy consumption is to delay the resource allocation until the next bandwidth requirement. By this way, availability intervals can be shortened and then, the power saving can be further promoted. However, the final sleep-listening state waveform may become aperiodic. In consequence, the PSC demultiplexing procedure will be inexecutable. To achieve fully utilization of an awaked frame and sustain the periodicity of sleep-listening state waveform, we propose Algorithm 3: *Extended Method for Resource Reservation* which replaces Equation (**??**)∼(3.8) in resource reservation procedure.

Algorithm 1: Extended Method for Resource Reservation

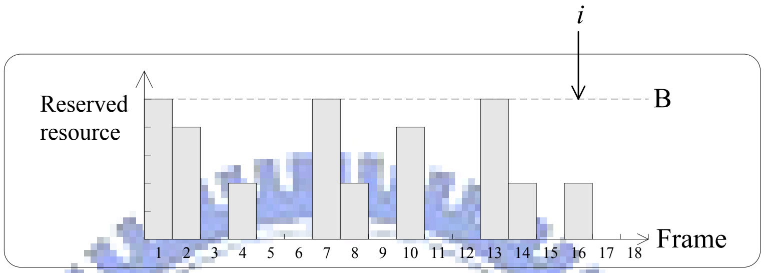
```
input  : $Br(x)$, $n$, where $n = lcm\{(P_i^{II}.T_L + P_i^{II}.T_S)\,|\forall P_i^{II}\}$
output: Revised $Br(x)$

1  $r = 0$;
2  for $i = 1$ to $n$ do
3      $r = r + Br(i)$;
4      $Br(i) = min(B, r)$;
5      $r = r - Br(i)$;
6  for $i = n$ to $1$ do
7      if $0 < Br(i) < B$ then
8          $r = Br(i)$;
9          for $j = i - 1$ to $1$ do
10             if $0 < Br(j) < B$ then
11                 $r = r + Br(j)$;
12                 if $r \leq B$ then $Br(j) = 0$;
13                 else
14                     $Br(j) = r - B$;
15                     $Br(i) = B$;
16                     goto label;
17                 if $j$=1 then
18                     $Br(i) = r$;
19                     return;
20     label:
```

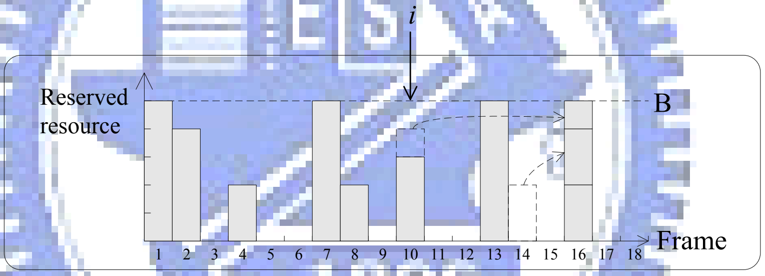
We know the bandwidth requirement is a periodic waveform with cycle n, where n is the *lcm* of periods of all uplink or downlink PSCs (input). Therefore, we pick up an interval n to do resource reservation. Lines 2-5 are the same as Equations (**??**)∼(3.8).



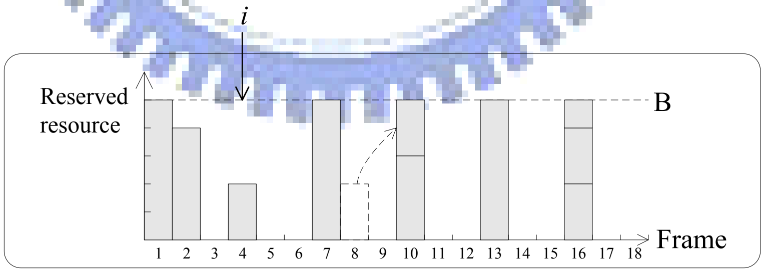
(a) Resource requirement of an MSS.



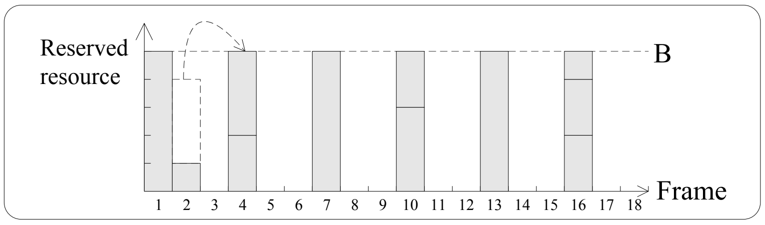
(b) Searching for the non-fully utilized frames backward direction. The first non-fully utilized frame we will find is 16th frame.



(c) Delay the resource allocation of frame 10 and 14 to frame 16. Then, the next non-fully utilized frame is frame 10.



(d) Delay the resource allocation of frame 8 to frame 10. Then, the next non-fully utilized frame is frame 4.



(e) Delay the resource allocation of frame 2 to frame 4. The final resource allocation is shown as the figure.

Figure 3.5: Example of the extended method.

So, after lines 2-5, we derive an sleep-listening state waveform as section 3.2. Then, from the end of the n frames,we try to find a un-fully utilized awaked frame in an descending order (lines 7-8). If finding a frame i which is not fully used, we will keep searching next unused awaked frame j and then delay the resource allocation in jth frame until frame i (lines 10-16). When frame i is full, we will repeat seeking next un-fully utilized frame and filling it. In this way, some awaked frames can be saved. Also, compared the original resource reservation procedure, this extended one is still with complexity $O(n)$.

This adjustment will increase some delay, but promote the energy saving of an MSS. It can be applied when delay is not a strict requirement.

Figure 3.5 is an example which demonstrates the extended method. Suppose the bandwidth requirement of an MSS is shown as Fig. 3.5(a). Then BS executes the extended method to decide the resource allocation for the MSS and derives the sleep-listening state waveform of the MSS as follows. Fig. 3.5(b) is the result of resource reservation after executing lines 2-5. Then, the last non-fully utilized frame i, where $i = 16$, is found. We reallocate the resource by delaying the former non-fully utilized allocation, frames 14 and 10, to this frame, shown as Fig. 3.5(c). Next, $i = 10$, and likewise, the resource allocation on frames 8 and 4 are delayed to frame 10, shown as Fig. 3.5(d). Finally, $i = 5$ and the resource allocation on frame 2 are reallocated to here. The eventual resource allocation is derived as Fig. 3.5(e).

Chapter 4. Proof and Analysis

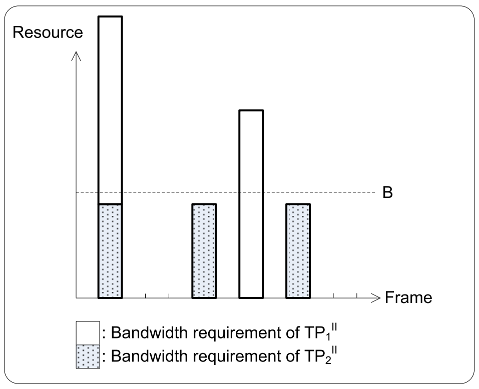
4.1 Proof of Delay Bound

Our sleep scheduling scheme tries to make data transmitting and receiving of different connections being concentrated in the same frames; so, it can decrease the total awake time. However, the packet delay for a connection may be increased because the real transmission/receiving time is delayed due to our resource allocation scheme. In this subsection, we will show the maximum packet delay of each connection will less than or equal to its maximum delay constraint when applying our scheme.

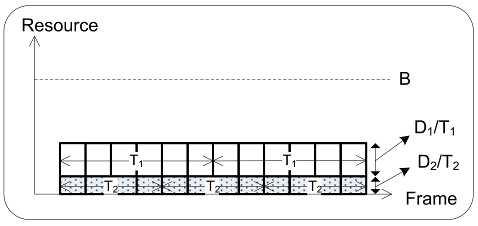
Assume there are n real-time connections and each connection C_i has a temporary PSC P_i^{II} with $T_L = 1$, $T_S = T_i - 1$, and the data transmission/receiving requirement equals to D_i in each interval T_i. Besides, the maximum resource in a frame that the BS can supply to an MSS is B. Then, the first property of resource allocation is introduced below.

Lemma 1. *When each time the resource requirement occurs, say the resource requirement of P_i^{II}, if BS allocates at least $\frac{D_i}{T_i}$ resource to P_i^{II} in a per frame basis until D_i is served, the resource requirement of P_i^{II} can be guaranteed in T_i.*

It is obvious that if each connection transmits or receives data size $\frac{D_i}{T_i}$ per frame, the resource requirement of each P_i^{II} can be guaranteed in T_i. Fig. 4.1 shows the case. In Fig. 4.1(a), the resource requirement of each P_i^{II} over frames of the time is presented. Fig. 4.1(b) shows, if each P_i^{II} can be allocated $\frac{D_i}{T_i}$ resource each frame, then the resource requirement can be served before next resource requirement comes.



(a) An example, there are 2 connections with temporary PSC P_1^{II}, $T_L = 1$, $T_S = 5$ and P_2^{II}, $T_L = 1$, $T_S = 3$, respectively.



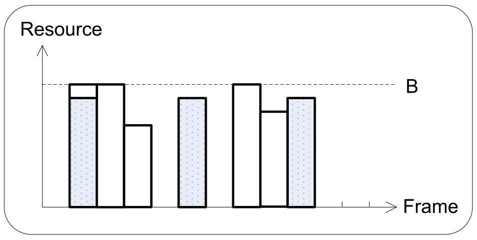
(b) BS allocates P_1^{II} and P_2^{II} resources every frame with $\frac{D_1}{T_1}$ and $\frac{D_2}{T_2}$, respectively.

Figure 4.1: Example of allocating bandwidth averagely in every frame for each connection.

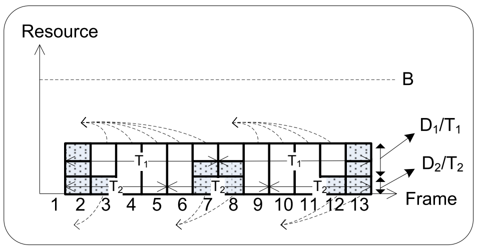
Allocation scheme in Lemma 1 can guarantee the resource requirement being satisfied in T_i. Following, we will show that our resource allocation scheme can also guarantee the property.

Lemma 2. *The allocation method in Lemma 1 can be revised to generate the same resource allocation as our allocation method without changing its property. Therefore, adopting our resource allocation method, the bandwidth requirement occurs in a listening window of P_i^{II} also can be guaranteed within T_i.*

Fig. 4.2(a) shows the resource allocation result of our method with the example of Fig. 4.1(a). Each time when the resource is not enough to serve the remaining requests, we first serve the request of the temporary PSC with closest next listening window. This is known as *earliest deadline first* scheduling. For the method in Lemma 1, first, we exchange the resource allocations for C_1 and C_2 such that resources with closest next listening window will be moved forward. If we assume $\frac{D_2}{T_2} = \frac{1}{2}\left(\frac{D_1}{T_1}\right)$, resource allocation in Fig. 4.1(b) will be as Fig. 4.2(b). Apparently, the property in Lemma 1 still hold. Now, we move the resource allocation for each C_i forward to fill the unused resource until the



(a) The example in Fig. 4.1(a) would turn out like this by using our resource allocation method.



(b) Revise the allocation result in Fig. 4.1(b) and then move the resource allocation forward (C_i with earliest deadline gets higher priority). In the end, the same result as (a) would be conducted.

Figure 4.2: The bandwidth allocation of our scheme can be transformed to the above one in Fig. 4.1

threshold B reached (C_i with earliest deadline gets higher priority). Take Fig. 4.2(b) for example again, moving resource allocation for C_1 in frame 3~7 forward so that allocation can be as close to frame 2 as possible. Apparently, the same result as Fig. 4.2(a) will be conducted in the end. Still, the property in Lemma 1 holds. So, by *earliest deadline first* scheuduling scheme, Lemma 2 is proved.

Theorem 3. *The packet delay of each connection will be less or equal to its maximum delay constraint.*

Recall that, for UGS/ERT-VR connections for VoIP service, we define the cycle of the P_i^{II} is maximum delay divided by 2, and for other real-time connections, the cycle of the P_i^{II} is maximum delay divided by 3. So, the property of this theorem can be discussed from these two aspects separately.

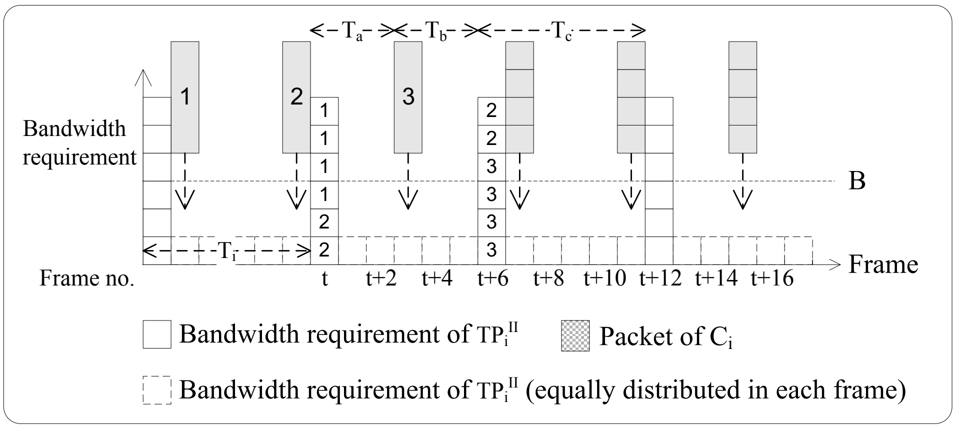
UGS/ERT-VR connections for VoIP service :

For these types of connections, the bandwidth requirement in a listening window of P_i^{II} is the quantity to serve the maximum possible size of arriving data within a cycle T_i. In other words, MSS always request enough bandwidth for an arrival packet of C_i on the upcoming listening window. We specify the length of the time between the upcoming listening window and the packet arrival as T_A. Also, by Lemma 2, the bandwidth requirement of C_i on frame t can be satisfied before $t+T_i$. So,

$$\text{packet delay} \le T_A + T_i \le T_i + T_i \le \left(\frac{C_i.D_{max}}{2}\right) + \left(\frac{C_i.D_{max}}{2}\right) = C_i.D_{max}$$

Other connections :

For these types of connections, an arrival packet is not assured being included in the bandwidth request of upcoming listening window of its P_i^{II}. This is because the total resource requirement of C_i is averagely distributed in each cycle. Fig. 4.3 displays an example of packet delay. The part of dashed line shows the virtual bandwidth requirement if the bandwidth requirement is equally distributed in each frame. Then, clearly, the bandwidth request of a packet is done prior to the arrival of next packet. But, actually, the bandwidth requirement of our scheme is gathered up periodically in the solid line plcae. For instance, the bandwidth requirement on frame t only prepares for part of packet 2, because the requirement at frame t is only for the virtual requirement of frame $t-5$ to t. The remaining bandwidth request of packet 2 is on the next listening window, say frame $t+T_i$ ($T_i = 6$). That is, the bandwidth request for a packet is done on the listening window after the next packet arrival time. So, the time from the arrival of packet 2 to the listening window right after packet 3 can be represented as $T_a + T_b$ (as shown in Fig. 4.3), where T_a is the packet inter-arrival time and T_b is the time between the arrival of packet 3 and its upcoming listening window of P_i^{II}. And likewise, this request at



<-Ta-><-Tb-><----Tc---->
Bandwidth requirement
Frame no.
<----Ti---->
t t+2 t+4 t+6 t+8 t+10 t+12 t+14 t+16
B
Frame
Bandwidth requirement of TP_i^II
Packet of C_i
Bandwidth requirement of TP_i^II (equally distributed in each frame)

Figure 4.3: Packet delay of arrival packet 2 of C_i can be guaranteed shorter than T_a+T_b+T_c while the bandwidth requirement and allocation are done by our scheme.

frame $t+6$ can be satisfied within T_i by Lemma. 2. So, we derives

$$\begin{aligned}\text{packet delay} &= T_a + T_b + T_c \leq C_i.I_d + T_i + T_i \\ &\leq \left(\frac{C_i.D_{max}}{3}\right) + \left(\frac{C_i.D_{max}}{3}\right) + \left(\frac{C_i.D_{max}}{3}\right) = C_i.D_{max}\end{aligned}$$

4.2 Power Consumption Analysis

We assume an MSS consume energy of P_A and P_S in an awake and sleep frame, respectively. Then the average power consumption of an MSS per frame can be described as following:

$$\begin{aligned}P_{MSS} &= (Awake\ ratio) \times P_A + (Sleep\ ratio) \times P_S \\ &= \frac{\sum_{t=1}^{n} \tilde{S}(t)}{n} \times P_A + \left(1 - \frac{\sum_{t=1}^{n} \tilde{S}(t)}{n}\right) \times P_S, \qquad (4.1)\end{aligned}$$

where $n = lcm\{T_i \mid i = 1 \cdots M + N\}$.

4.3 Average Delay Analysis

We assume there are n connections $C_1, ...C_n$, each with first packet burst arrival time S_i, data inter-arrival time $C_i.I_d$. Suppose a packet size is D, for each connection C_i, there are N_i packets in a data burst. The average packet delay can be represented as

$$\begin{aligned}
&\frac{\text{(Cumulative time of arrival packets)} - \text{(Cumulative time of departure packets)}}{\text{Served packets}} \\
&= \lim_{\tau\to\infty} \frac{\int_0^\tau F_a(t)\mathrm{d}t - \int_0^\tau F_s(t)\mathrm{d}t}{F_s(\tau)}, \qquad (4.2)
\end{aligned}$$

where $F_a(t)$ is the cumulative number of arrival packets over time, and $F_s(t)$ is the cumulative number of departure packets over time.

The cumulative time of arrival packets for C_i : $\int_0^\tau F_{a,C_i}(t)\mathrm{d}t$ is equal to the sum of each packet's ($\tau - \mathit{arrival\ time}$). Consider that all the packets in the kth burst arrival of C_i have the same ($\tau - \mathit{arrival\ time}$), which is $(\tau - (S_i + (k-1) \times C_i.I_d))$, and there are totally $\left\lceil \frac{\tau - S_i}{C_i.I_d} \right\rceil$ bursts arriving from S_i to time τ, we can conduct that

$$\int_0^\tau F_{a,C_i}(t)\mathrm{d}t = N_i \left\lceil \frac{\tau - S_i}{C_i.I_d} \right\rceil \left[\tau - S_i - \left(\left\lceil \frac{\tau - S_i}{C_i.I_d} \right\rceil - 1 \right) \frac{C_i.I_d}{2} \right].$$

So, the cumulative time of arrival packets is

$$\int_0^\tau F_a(t)\mathrm{d}t = \sum_{i=1}^{n} \left\{ N_i \left\lceil \frac{\tau - S_i}{C_i.I_d} \right\rceil \left[\tau - S_i - \left(\left\lceil \frac{\tau - S_i}{C_i.I_d} \right\rceil - 1 \right) \frac{C_i.I_d}{2} \right] \right\}. \qquad (4.3)$$

To get $\int_0^\tau F_s(t)\mathrm{d}t$, we first recall that $\tilde{S}(k)$ represents the state series of the MSS, where we use k to express frame no. (not t in Sec. 3 and Sec. 4.2). To conduct $\tilde{S}(k)$, we can use N_i and $C_i.I_d$ information of each connection C_i. Here, the detail deduction is omitted, which is described in Sec. 3.2. Then, we use $f_s(k)$ to represent the number of served packets at frame k, which is shown as

$$f_s(k) = \min\left\{b(k)\cdot\tilde{S}\left(k-k_{MSS}\right), \frac{B}{D}\right\}, \tag{4.4}$$

where $b(k)$ presents the number of buffered packets in queue at frame k and k_{MSS} is the first listening frame of MSS. $b(k)$ can be further described as

$$b(k) = b(k-1) - f_s(k-1) + \tilde{a}(k-1),$$

where $\tilde{a}(k)$ is the number of arrived packets at frame k. $\tilde{a}(k)$ can be expressed as

$$\begin{aligned}\tilde{a}(k) &= \sum_{i=1}^{n} I_i(k)\cdot N_i, \\ &\text{where } I_i(k) = \begin{cases}1, & \text{if } 0 < ((k+1)\cdot F - S_i) \mod C_i.I_d \le F \\ 0, & \text{otherwise}\end{cases}\end{aligned}$$

Therefore, $\int_0^{\tau} F_s(t)\mathrm{d}t$ can be conducted as

$$\int_0^{\tau} F_s(t)\mathrm{d}t = \sum_{k=k_{MSS}}^{\lceil\frac{\tau}{F}\rceil-1}\left(\left\lceil\frac{\tau}{F}\right\rceil - 1 - k\right)\cdot f_s(k)\cdot F. \tag{4.5}$$

So, from Eqs. (4.3), (4.4), and (4.5), the average packet delay is

$$\lim_{\tau\to\infty}\frac{\sum_{i=1}^{n}\left\{N_i\left\lceil\frac{\tau-S_i}{C_i.I_d}\right\rceil\left[\tau - S_i + \left(\left\lceil\frac{\tau-S_i}{C_i.I_d}\right\rceil - 1\right)\frac{C_i.I_d}{2}\right]\right\} - \sum_{k=k_{MSS}}^{\lceil\frac{\tau}{F}\rceil-1}\left(\left\lceil\frac{\tau}{F}\right\rceil - 1 - k\right)\cdot f_s(k)\cdot F}{\sum_{k=k_{MSS}}^{\lceil\frac{\tau}{F}\rceil-1} f_s(k)} \tag{4.6}$$

Chapter 5. Simulation Results

We developed a simulation program in C to simulate and evaluate the performance of proposed sleep scheme. In this simulation program, a frame is set to 5ms, and the size of resource limit per frame of an MS is fixed, we can set this value as an input parameter beforehand.

There are lots of connections between a BS and an MS, each connection has its own QoS parameters. The program takes the delay constraints, packet inter-arrival time, and packet size of these connections, then derives the availability/unavilability interval of MS and PSCs according to the proposed algorithm. For each connection, its packet will arrive per inter-arrival time, but MS can only transmit/receive packets on the PSCs' listening window. So most of the packets cannot be transmitted immediately.

The sleep rate (sleep time of the MS / total time) and the statistic of packet delay will be reported. Changing input connections and resource limits can cause different simulation results. In the simulations, we define a connection set as the minimum unit of input connections. There are 8 real-time connections with different QoS parameters in this set as shown in Table.5.1, each connection has random direction (UPLINK/DOWNLINK) and random start frame. Every time we add the number of connections between BS and MS, these eight connections will be added together.

The following are some simulation cases, each has different meaning. Most of simulations are compare to PS scheme in reference [9], that's because the assumption, input and environment of this work is more similar to ours.

Fig.5.1 fixes the resource limit of an MS to 5000 bytes/frame, and increases the

Table 5.1: A connection set.

Type	Delay Constraint (ms)	Data arrival interval (ms)	Data Size (bytes)
UGS	135	45	160
UGS	105	35	30
UGS	180	60	160
ERT-VR for VoIP services	60	20	60
ERT-VR for VoIP services	90	30	24
ERT-VR for VoIP services	75	25	20
RT-VR	200	30	375
RT-VR	200	50	925

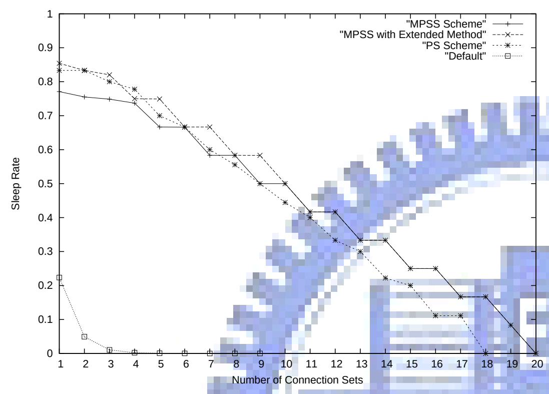


Figure 5.1: (Sleep Time/Total Time) of an MSS while the resource limit is fixed

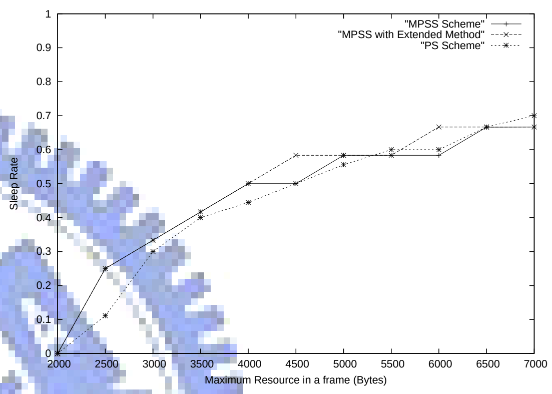


Figure 5.2: (Sleep Time/Total Time) of an MSS while the number of connections is fixed

number of connection sets between BS and MS step by step, to observe the percentage of sleep time of an MS in total time. According to the simulation results, although at first our scheme has slightly less sleep time, but with more and more connections added, our scheme has better sleep rate apparently. That's because the advantages of using multiple PSCs was revealed, the situation that MS is awake but no packet arrives is rarely appeared.

Differ to Fig.5.1, Fig.5.2 fixes the number of connection sets to eight sets, and changes the resource limit of an MS. When MS has less resource limit, our scheme get better performance. In other words, resources are allocated well when resource is not plenty.

The maximum packet delay is shown on Fig.5.3. The circumstance of this simulation

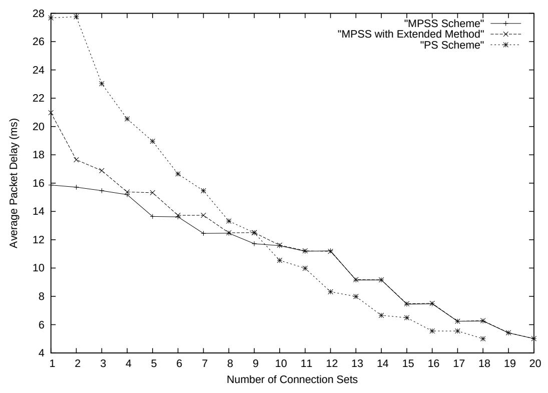


Figure 5.3: Average packet delay (connection 1)

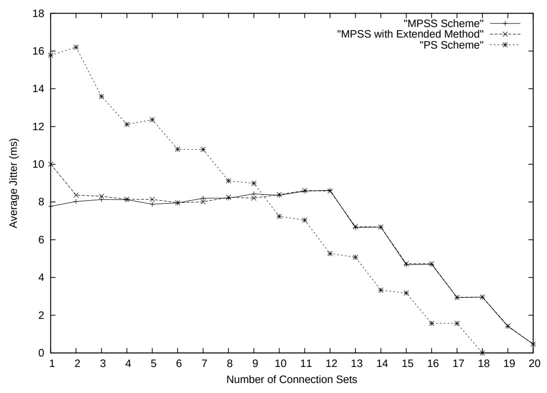


Figure 5.4: Jitter of packets (connection 1)

is the same with Fig.5.1. Actually, different connection's packet has different delay bound, we take one of the connection (connection 1) in the connection set to describe the value will be less or equal to its delay constraint. All the packet delay of this type of connections will be recorded and the maximum one in a long time will be reported. This result shows even if the traffic load is heavy, no packets delay broke the delay constraint. Noted that the connection sets cannot be added without limit, in case of traffic load more than all the resource of an MS, some connections will be denied during admission control.

Fig.5.4 shows the jitter of the connections. Like above, these values are belong to connection 1. The jitter is defined as standard deviation of all packet delays. We can see the curve raised gently with increasing number of connections, but the value of jitter is in the reasonable range all the time. In this figure, the compared method has less jitter because it allocates more resources for packets and MS has longer awake time, arrived packets can be transmitted quickly.

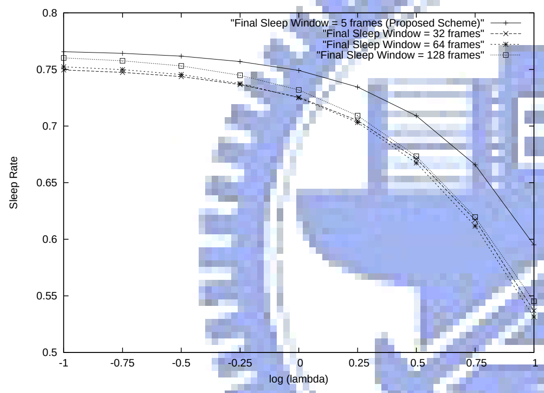


Figure 5.5: (Sleep Time/Total Time) of an MSS while the data rate of type I connections is λ

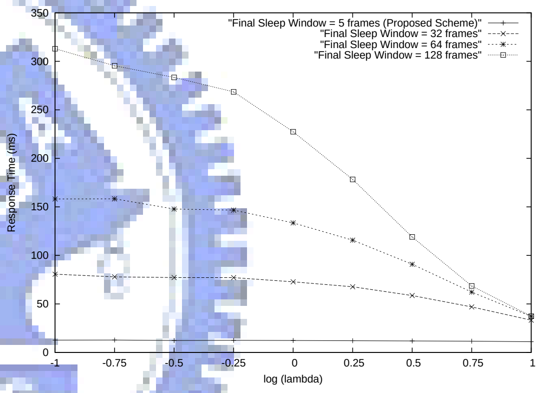


Figure 5.6: Response time of the packets of type I connections

Chapter 6. Conclusions

We proposed a sleep scheduling scheme MPSS which conform to the sleep mode mechanism of IEEE 802.16e standard. In this scheme, the connections' maximum delay constraints, packet inter-arrival time, and packet sizes are considered to determine the parameters of PSCs. The data of a connection can be transmitted during the listening window of corresponding PSCs. Multiple type II PSCs will be conducted for real-time connections, and one type I PSC will be in charge of non-real-time connections. The problem of mismatch between uplink scheduling and listening frames of PSCs is handled. Besides, an extended method to save more energy but doesn't guarantee the packet delay constraint is also introduced.

When MS enters sleep mode, we proved the delay of each packet will not larger than its maximum delay constraint, and present the performance and average analysis. The simulation result also shown that our scheme can save more energy of an MS than previous work, and the delay of each packet was not larger than its delay constraint.

This scheme considered the operations of sleep mode in the standard, and doesn't take heavy computation effort, that is to say it is a practical strategy to save energy of an MS. In the future, we will consider multiple MSs to dynamically adjust the resource limitation in resource reservation procedure instead of fixing the resource of an MS.

Bibliography

[1] IEEE Std. 802.16-2004. IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems. Oct. 2004.

[2] IEEE Std. 802.16e 2005. Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems - Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands. Feb. 2006.

[3] Y. Xiao. Energy Saving Mechanism in the IEEE 802.16e Wireless MAN. *IEEE Communications Letters*, 9(7):595–597, July 2005.

[4] Y. Zhang and M. Fujise. Energy Management in the IEEE 802.16e MAC. *IEEE Communications Letters*, 10(4):311–313, April 2006.

[5] L. Kong and H.-K. Tsang. Performance Study of Power Saving Classes of Type I and II in IEEE 802.16e. In *Proc. of the 31st IEEE Conference on Local Computer Networks*, pages 20–27, Nov. 2006.

[6] J. Shi, G. Fang, Y. Sun, J. Zhou, Z. Li, and E. Dutkiewicz. Improving Mobile Station Energy Efficiency in IEEE 802.16e WMAN by Burst Scheduling. In *IEEE GLOBECOM*, Nov. 2006.

[7] F. Xu, W. Zhong, and Z. Zhou. A Novel Adaptive Energy Saving Mode in IEEE 802.16e System. In *Proc. of Military Communications Conference (MILCOM2006)*, Oct. 2006.

[8] T.-C. Chen, Y.-Y. Chen, and J.-C. Chen. An Efficient Energy Saving Mechanism for IEEE 802.16e Wireless MANs. *IEEE Transanctions on Wireless Communications*, to appear.

[9] Y.-L. Chen and S.-L. Tsao. Energy-efficient Sleep-mode Operations for Broadband Wireless Access Systems. In *IEEE VTC 2006-Fall*, 2006.