This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2006 proceedings.

Token Bucket Based CAC and Packet Scheduling for IEEE 802.16 Broadband Wireless Access Networks

Chi-Hong Jiang, Tzu-Chieh Tsai
Computer Science Department, National Chengchi University, Taipei, Taiwan, ROC
{g9203, ttsai}@cs.nccu.edu.tw

Abstract

The IEEE 802.16 standard was designed for Wireless Metropolitan Area Network (WMAN). It supports QoS and has very high transmission rate. The key part of 802.16 – packet scheduling, was undefined and is an open issue. We first proposed a token-bucket based uplink packet scheduling combined with call admission control (CAC) of 802.16. Then a model of characterizing traffic flows by token bucket is presented. The simulation results show that our CAC and uplink packet scheduling can promise the delay requirement of rtPS flows and our model can predict the delay and loss of a traffic flow precisely.

1. Introduction

After the 802.11 standard, a brand new standard that aims at WMAN was proposed. The members of IEEE 802.16 had finished the original 802.16 standard in 2001[1], and 802.16a, 802.16c in 2002, 2003, respectively. In 2004, a revision of IEEE 802.16[2] standard was published, which is a revision of original 802.16, 802.16a, and 802.16c and also known as 802.16 REVd. The original IEEE 802.16 is restricted to Line Of Sight (LOS) but the following standards, such as 802.16a and 802.16c, can be Non Line Of Sight (NLOS)[3]. The bit rate of 802.16 is 32-134 Mbps at 28MHz channelization. The transmission range of 802.16 is typically 4-6 miles.

The MAC part of 802.16 standard is not Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) that was adopted by 802.11 standard. The 802.16 uses Time Division Duplex (TDD) or Frequency Division Duplex (FDD) to access medium resource. Unlike 802.11, the first 802.16 standard has QoS support, it divides all traffic flows into four classes according to their application type and each class has different priority.

802.16 uses packet scheduling to achieve QoS support, but there is no clear definition or implementation detail about the algorithm. The 802.16 standard defines only the QoS parameters and some simple principles, the other parts are open issues left to vendors.

The remainder of this paper is organized as follows. In Section 2, we make a detailed description about the IEEE 802.16 standard. Our call admission control and uplink packet scheduling models are proposed and explained in Section 3 and Section 4. In Section 5, we present the token rate estimation model. Simulation results are shown in Section 6. We conclude this paper in Section 7.

2. The IEEE 802.16 standard

In 802.16, the client side is called Subscriber Station (SS), and the server side is called Base Station (BS).There are four QoS classes defined in 802.16, which are Unsolicited Grant Service (UGS), real-time Polling Service (rtPS), non-real-time Polling Service (nrtPS), Best Effort (BE). Table 1 shows the QoS classes of 802.16, and the upper class has higher priority than lower one.

Table 1. 802.16 QoS classes

Class name	Traffic type	Application
UGS	real-time CBR	Voice over IP
rtPS	real-time VBR	real-time video
nrtPS	non-real-time	FTP
BE	non-real-time	HTTP

Traffic flows in 802.16 are treated as connections. A traffic flow must establish connection with its BS before transmitting. The operation process of 802.16 is shown in Figure 1[4][5]. The blocks drawn with dotted line in Figure 1 are the parts undefined in 802.16. The traffic policing model can be simply achieved by applying token bucket mechanism[4].

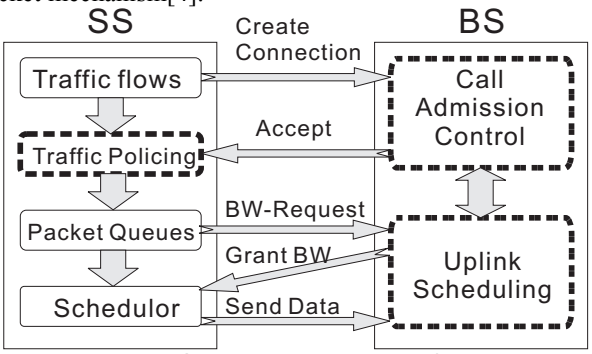

Figure 1. Operation process of 802.16

The 802.16 standard divides transmission time into super frames and each super frame can be divided into a downlink sub-frame and an uplink sub-frame. Downlink

1-4244-0086-4/06/$20.00 ©2006 IEEE.

means the direction of transmission is from BS to SS, and the uplink means inversely. The downlink scheduling is simple because the only sender is BS. Hence we focused on uplink scheduling.

After a BS accepts a new connection, it will poll this new connection and give it the opportunities of sending its BW requests. This connection should send its bandwidth request (BW request) to BS for receiving BW grants (e.g. time slots for transmitting data) from BS. In this paper, we use the architecture in [4]: a connection sends its queue length as BW request. These grants are the result of uplink packet scheduling at BS and will be included in uplink MAP (UL-MAP) field in the downlink sub-frame. The 802.16 frame structure is as Figure 2.

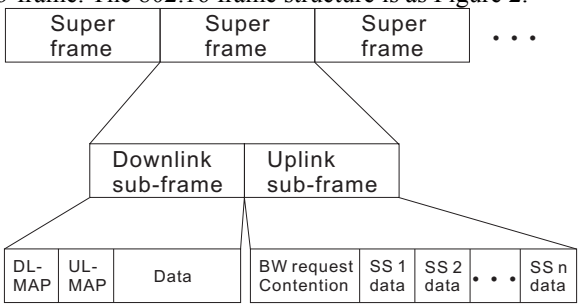

Figure 2. 802.16 frame structure

The BW request contention period is for the classes of lower priority, such as nrtPS and BE, to get opportunities of sending BW requests when system is too busy to poll all connections.

3. Call admission control (CAC)

In our scenario each connection is controlled by two token bucket parameters: token rate r_i (bps) and bucket size b_i (bits). When a traffic flow wants to establish a connection with BS, it sends these two parameters to BS and waits for response from BS. An extra parameter, delay requirement d_i, will be sent by rtPS flow.

Before presenting our CAC model, we should analyze the bandwidth requirement of an rtPS flow. Assume f is the duration of a super frame and there is an rtPS flow with parameter r_i, b_i, and d_i. The d_i must be two times of f at least. Here we assume $4f > d_i \geqq 3f$. So we set its delay requirement to 3f.

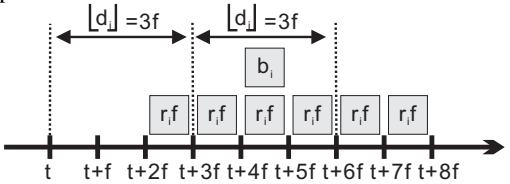

Figure 3. A burst of an rtPS connection from time t to t+6f

If this connection has a burst from time t to t+6f, the maximum size should be sent during each frame is shown in Figure 3. The gray blocks means the maximum size should be sent during that frame. Because the delay requirement of this connection is 3f, data arrived during frame[t, t+f] must be sent out during frame[t+2f, t+3f] at least. We know that a connection with token bucket parameters r_i and b_i sends data of r_it+b_i bits at most during a time period t. Hence during a frame f this connection will send data of r_if+b_i bits at most. If the rate of generating data is over r_i, b_i is consumed. In the extreme case this connection may run out of b_i during a certain frame. In Figure 3, b_i is totally consumed during frame[t+2f, t+3f], this situation makes the maximum size should be sent out during frame[t+4f, t+5f] to be r_if+b_i bits.

In Figure 3, the b_i comes from the frame[t+2f, t+3f]. So only the frame[t+3f, t+4f] can share the b_i bits of the frame[t+4f, t+5f]. Let $m_i=d_i \bmod f$ (m_i is 3 in Figure 3), we can find there are m-1 frames can share the b_i bits at most. Through analysis above we can easily find the maximum size during a frame is

$$r_if + \frac{b_i}{m_i - 1} \quad (1)$$

Let N_{rtPS} be the number of rtPS connections, C_{demand} be the maximum size (bits) required by all rtPS connections, we can know

$$C_{demand} = \sum_{i}^{NrtPS} \left(r_if + \frac{b_i}{m_i} \right) \quad (2)$$

In order to avoid starvation of some classes, we set a threshold for each class. They are four parameters: t_{UGS}, t_{rtPS}, t_{nrtPS}, t_{BE}, and $t_{UGS}+t_{rtPS}+t_{nrtPS}+t_{BE} \leqq C_{uplink}$, where C_{uplink} is the total capacity of uplink. When the total capacity occupied by a class is over its threshold, this class will have lower priority to use the capacity. When a new traffic flow with parameters r_i, bi, and d_i (only rtPS traffic flow has it) wants to establish a connection with BS, our call admission control algorithm is shown below.

Step 1. Calculate the remainder uplink capacity C_{remain}: $C_{remain}=C_{uplink}-C_{UGS}-C_{rtPS}-C_{nrtPS}-C_{BE}$, where C_{UGS}, C_{rtPS}, C_{nrtPS}, C_{BE} means the capacity occupied by each class.

Step 2. Compare C_{remain} to the bandwidth requirement of the new connection. For rtPS, we use (2) to calculate its bandwidth requirement, for other UGS and nrtPS, we take r_i as its bandwidth requirement. If there is enough capacity we accept it. If not, go to Step 3.

Step 3. First look at the connections that belong to lower classes than this new connection. If there is a class that uses more capacity than its threshold, we calculate C_L, which means how much capacity we can steal from it. If the sum of C_L and C_{remain} is greater than or equal to the bandwidth requirement of this new connection, we accept it. If not, we look at if the capacity occupied by the class of the new connection is less than its threshold. If not we deny it. Else we can try to steal some capacity from higher classes.

A connection can steal capacity from connections of a higher class only if the class it belongs to occupied less capacity than its threshold and the higher class use more capacity than its threshold. According the principle above, we can calculate C_U, which means how much capacity we can steal from connections of higher classes. If the sum of C_U, C_L, and C_{remain} is greater than or equal to the bandwidth requirement of this new connection, we accept it. Else we deny it.

When the new class of the new flow is BE, we just accept it and steal capacity from other classes as much as possible if the capacity occupied by BE is less than its threshold.

Stealing capacity from BE and nrtPS is relatively simple. We can easily decrease the capacity used by them because of they are not real-time. To steal capacity from the other two real-time classes we can make some connections of these two classes degrade their r_i, e.g. make r_i to be $c \bullet r_i$, where $0 < c < 1$.

4. Uplink packet scheduling

Our uplink packet scheduling algorithm is described below.

Step 1. Calculate arrival packets of rtPS connections and their deadline. We adopt Earliest Deadline First (EDF) mechanism proposed in [4]. There is a database that records the number of packets that need to be sent during each frame of every rtPS connection.

Step 2. Grant capacity to all UGS connections. According to the 802.16 standard[1][2], an UGS connection receives fixed bandwidth during each frame.

Step 3. Grant capacity to all rtPS connections according to the rtPS database. Due to possible degradation of r_i, we should restrict the maximum grant size of a connection to (1).

Step 4. Allocate the grants of nrtPS connections and BE connections. Let R_{nrtPS} and R_{BE} be the capacity needed by nrtPS and BE respectively. We first grant Min(R_{nrtPS}, B_{nrtPS}) to nrtPS connections and Min(R_{BE}, B_{BE}) to BE connections. After that, if there is remainder capacity, allocate them to rtPS and BE in order if necessary.

Step 5. If there is remainder capacity, allocate them to contention period of nrtPS and BE in order if necessary.

5. Token rate estimation model

Some research about characterizing a traffic flow by token bucket had been done such as [6]. They used measurement-based approaches to find appropriate token rate and bucket size. Main flow of our token rate estimation model is shown in Figure 4. We take a traffic flow that has Poisson arrival as example. First, the case of infinite queue is considered. Assume a traffic flow with parameters λ_i and d_q, which are mean arrival rate and queuing delay requirement of the flow. Now we introduce how to use λ_i, r_i, and b_i to predict the queuing delay.

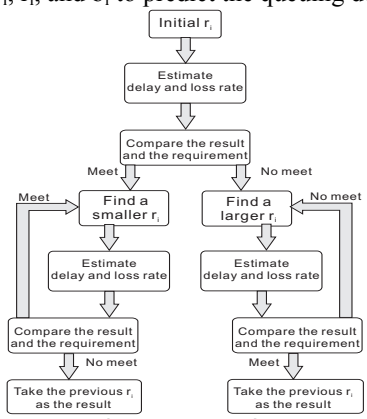

Figure 4. The flow chart of our token rate estimation model

In [7], traffic flows with different types of arrival and infinite queue were analyzed by simulation. We analyze the queuing delay by making use of Markov Chain. State(t, p) means there are t tokens in the bucket and p packets in the queue. Assume the time interval between two states is $1/r_i$, that is the time of generating a token. Hence we can find the probability P of n packets that arrive during the time interval $1/r_i$ is

$$P(n) = \frac{\alpha^n \cdot e^{-\alpha}}{n!}, \text{ where } \alpha = \frac{\lambda_i}{r_i}$$

This Markov Chain is shown in Figure 5 and 6.

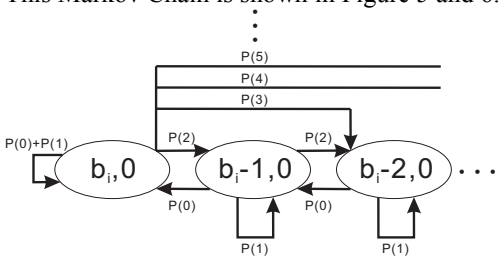

Figure 5. The beginning of the Markov Chain

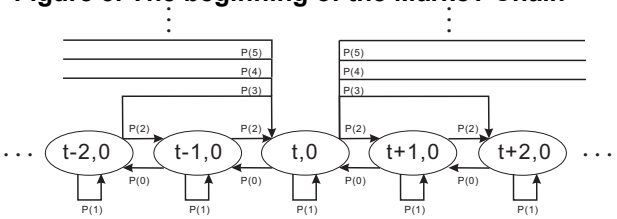

Figure 6. The general case of the Markov Chain

Assume state(t, p) is denoted by π (b_i-t+p) and let

$$M = \sum_{i=2}^{\infty} P(i)$$

We can list equations as follows.

$$\pi(0) \cdot M = P(0) \cdot \pi(1) \quad (3)$$

And for $n \geqq 1$, we have

$$\pi(n)\cdot(P(0)+M)= \sum_{k=0}^{n-1}\pi(k)\cdot P(n+1-k)+\pi(n+1)\cdot P(0) \quad (4)$$

Assume the Z-transform of Poisson and [$\pi(0)$, $\pi(1)$, $\pi(2)\ldots$] is Gp(z) and Gπ(z). Then apply Z-transform to (4) and simplify it, we can get

$$(M+P(0))\cdot G\pi(z)-\frac{(Gp(z)-P(0)-P(1)z)}{z}\cdot G\pi(z)-\frac{P(0)}{z}\cdot G\pi(z)=(A+P(0))\cdot\pi(0)-\frac{P(0)}{z}(\pi(0)+\pi(1)z) \quad (5)$$

From (3) we know

$$\pi(1)=\frac{M}{P(0)}\cdot\pi(0) \quad (6)$$

M+P(0)+P(1) is equal to 1, so substitute $\pi(1)$ in (5) for (6) and simplify it we get

$$G\pi(z)=\frac{P(0)\cdot\pi(0)\cdot(z-1)}{z-Gp(z)} \quad (7)$$

We know

$$\lim_{z\to 1} G\pi(z)=1 \quad (8)$$

Use (7) and (8) we can find

$$\pi(0)=\frac{r_i-\lambda_i}{r_i\cdot e^{-\alpha}}, \text{ where } \alpha=\frac{\lambda_i}{r_i} \quad (9)$$

To find [$\pi(0)$, $\pi(1)$, $\pi(2)\ldots$], we should make use of (6) to find $\pi(1)$ first. After we know $\pi(0)$ and $\pi(1)$, we can utilize (4), $\pi(0)$ and $\pi(1)$ to find $\pi(2)$. Go on this process we can find any $\pi(n)$. The average queuing delay d_{avg} can be expressed as

$$d_{avg}=P(\text{see token in the bucket})\cdot 0+P(\text{see no token in the bucket})\cdot d_{M/D/1/} \quad (10)$$

where $d_{M/D/1}$ is the mean delay of a M/D/1 queue and its value[8] is

$$\frac{2r_i-\lambda_i}{2r_i(r_i-\lambda_i)}, \text{ given } r_i > \lambda_i \quad (11)$$

And

$$P(\text{see no token in the bucket})=1-\sum_{k=0}^{b_i-1}\pi(k) \quad (12)$$

Substitute $d_{M/D/1}$ and P(see no token in the bucket) in (10) for (11) and (12) we can calculate the average delay of this flow if we give it parameters r_i and b_i.

Now we consider the case of finite queue with queue size q. This traffic flow has an extra parameter l_q, which means its loss rate requirement. We still use Markov Chain to solve this case, but the number of states become limited, e.g. [$\pi(0)$, $\pi(1)$, …, $\pi(b_i+q-1)$]. The equations are listed below.

$$\pi(0)\cdot(1-P(1)-P(0))=\pi(1)\cdot P(0) \quad (13)$$

For $b_i+q-2 \geqq n \geqq 1$,

$$\pi(n)\cdot(1-P(1))=\sum_{k=0}^{n-1}(\pi(k)\cdot P(n+1-k))+\pi(n+1)\cdot P(0) \quad (14)$$

There is no clear mathematic solution for the equations above found by us, so a recursion method was introduced. We can find all $\pi(n)$ very fast by means of computer.

The average queuing delay can be expressed as

$$d_{avg}=\frac{\sum_{k=0}^{b_i+q-1}\left(\pi(k)\cdot\left(\sum_{j=Max(b_i-k+1,\,0)}^{\infty}P(j)\cdot(Min(j+k-b_i,\,q)-N)\right)\right)}{r_i}$$

, in which N=0.5 when j=0. Otherwise N=0. The average loss rate can be expressed as

$$l_{avg}=\frac{\sum_{k=0}^{b_i+q-1}\left(\pi(k)\cdot\sum_{j=bi+q+1-k}^{\infty}P(j)\cdot(j+k-q-b_i)\right)}{\lambda_i/r_i}$$

After we solve [$\pi(0)$, $\pi(1)$, …, $\pi(b_i+q-1)$], we can estimate the average queuing delay and average loss rate by equations above. Given a reasonable b_i to the traffic flow, we can use a simple search algorithm to find appropriate r_i according to its d_q and l_q. We can put this model in the SS. When a new traffic flow comes, SS can use this model so as to find appropriate r_i, then the BS can schedule according this r_i.

6. Simulation results

We show the simulation results in this section. Section 7.1 is the results of section 3 and 4. Section 7.2 is the result of section 5.

6.1. CAC and uplink packet scheduling

Two methods of CAC and uplink packet scheduling are compared here. Method 2 was proposed in [4] and method 1 was proposed by us. The parameters of four classes are listed in Table 2.

The begin time of each flow is Poisson distribution. All flows send data during each frame in full speed. Frame duration f is 1ms and simulation time is 150ms.

Table 2. Simulation parameters

Class	Token rate(bps)	Bucket size(bits)	Delay req.(ms)	Packet size(bits)
UGS	192000	64	-	64
rtPS	640000	15000	20	256
nrtPS	2000000	15000	-	256
BE	512000	8000	-	128

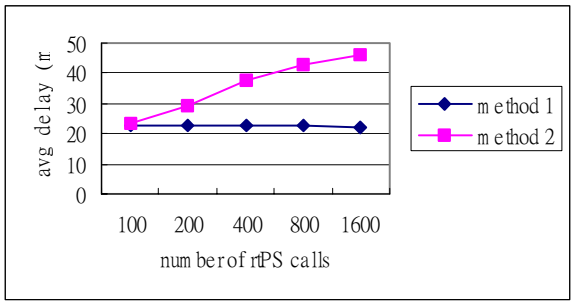

Figure 7. Average delay v.s. number of rtPS calls

From Figure 7 we can find that the average delay of rtPS used by our method is almost constant no matter how many rtPS calls exist.

6.2. Delay and loss rate estimation in token rate estimation model

Assume a traffic flow with Poisson arrival and its simulation parameters are listed in Table 3.

Table 3. Simulation parameters

Parameter	Value
mean arrival rate (bps)	640000
bucket size (bits)	5120
packet size (bits)	512
queue size (bits)	5120

The difference between estimated delay and simulated delay is 4.4% at most. Figure 8 shows this result. From Figure 9 we can see that our prediction of loss rate is very precise.

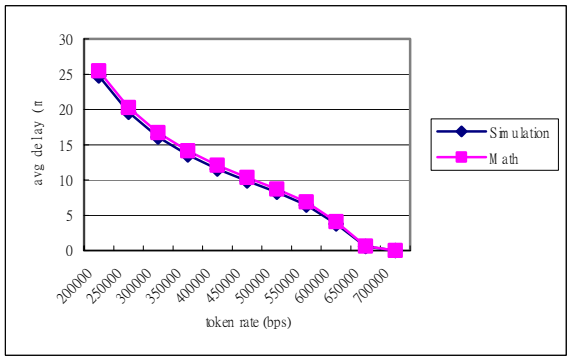

Figure 8. Average delay v.s. token rate

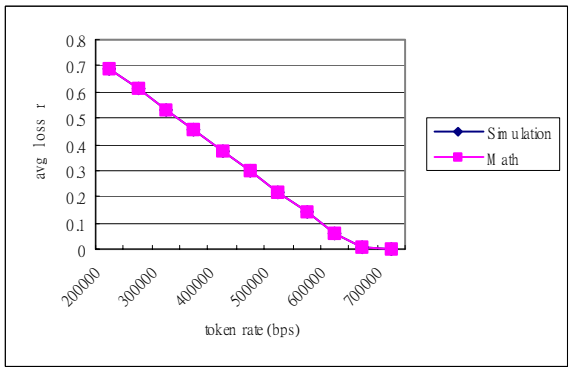

Figure 9. Average loss rate v.s. token rate

7. Conclusions and future work

In this paper we proposed a QoS-supported uplink packet scheduling and CAC mechanisms. Bandwidth needed by real-time flows is correctly reserved and their delay requirements can be promised. We also proposed a model to convert Poisson traffic flow into token bucket-based connection. In the future, how to integrate CAC and uplink scheduling with token rate estimation model may be another issue we will concern.

8. References

[1] IEEE, "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems", *IEEE standard*, December 2001.

[2] IEEE, "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems", *IEEE standard*, October 2004.

[3] Carl Eklund, Roger B. Marks, Kenneth L. Stanwood and Stanley Wang, "IEEE standard 802.16: A technical overview of the wirelessMAN air interface for broadband wireless access", *IEEE Communications Magazine*, vol. 40, no. 6, June 2002, pp. 98 – 107.

[4] Kitti Wongthavarawat, and Aura Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems", *International Journal of Communication Systems*, vol. 16, issue 1, February 2003, pp. 81-96.

[5] Dong-Hoon Cho, Jung-Hoon Song, Min-Su Kim, and Ki-Jun Han, "Performance Analysis of the IEEE 802.16 Wireless Metropolitan Area Network", *IEEE Computer Society*, DFMA'05, February 2005, pp. 130-137.

[6] Puqi Perry Tang and Tsung-Yuan Charles Tai, "Network traffic characterization using token bucket model", *IEEE INFOCOM 1999 - The Conference on Computer Communications*, no. 1, March 1999, pp. 51 – 62.

[7] Tarkan Taralp, Michael Devetsikiotis, and Ioannis Lambadaris, "Traffic Characterization for QoS Provisioning in High-Speed Networks", *IEEE Computer Society*, Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7, January 1998, pp. 485.

[8] Kleinrock L., "*Queueing Systems. Volume I: Theory*", John Wiley, New York, 1975.