CHAPTER 4

Routing and CAC Algorithms

In a QoS-provisioning mesh networks, cross layer design is a must. Since IEEE 802.16 is chosen as the MAC layer protocol, QoS in MAC layer can be achieved. However, in the network layer, a good routing algorithm is necessary and well-cooperates with the MAC layer. We proposed a new routing metric and a call admission control algorithm by modifying token bucket mechanism. In the following context, the routing metric and the algorithm are introduced in section 4.1 and the call admission control algorithm along with other details will be described in section 4.2.

4.1. Routing Algorithm

In this section, we obtain a routing metric(SWEB, Shortest-Widest Efficient Bandwidth) and algorithm which may achieve a good performance in IEEE 802.16 mesh mode. First, the criteria of a good routing algorithm should satisfy is discussed.

First, it would be desirable to have the same route for all traffic at a node. Although real-time traffic and best effort traffic tend to have different QoS requirements, such as delay, jitter or bandwidth, yet, it makes the routing considerably simple to have all traffics flowing over the same route. Second, the routing in IEEE 802.16 mesh mode should be fixed. Since the stations in the network do not move or have the minimum mobility, making topology and channel conditions do not change severely over a long period of time. Furthermore, this work is done upon the IEEE

802.16d standard, which does not support the mobility of stations. Third, also the most important one, to provide QoS of a traffic flow across the different routes can be difficult. For real-time traffics, the QoS requirements must be met by the reservation of network resources on the path to the MBS. If the traffic flows originating from a node have the different paths, and the routing is dynamic, the time of resource reservation is long, and serves as a major overhead. Based on the above arguments, we adopt a fixed routing algorithm and develop a routing metric for IEEE 802.16 mesh mode.

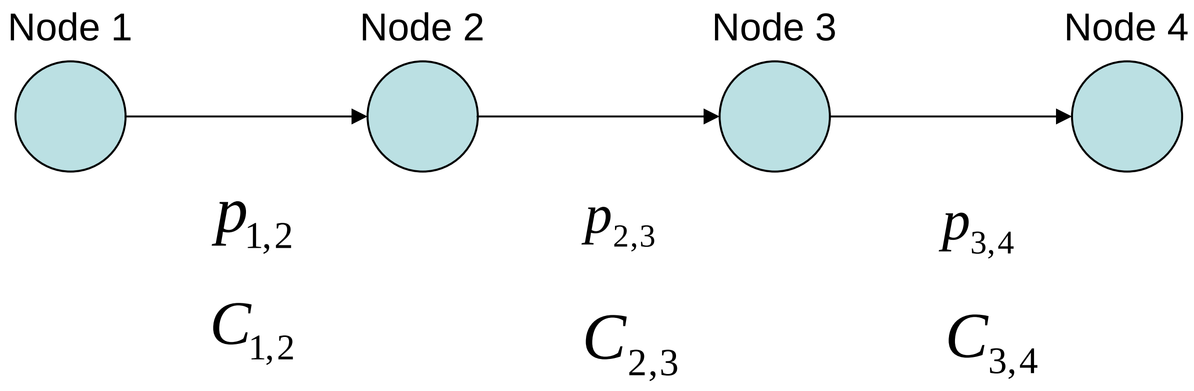

Figure 4.1: A Chain Topology

Assume that we have a chain-topology network like Figure 4.1, where $P_{i,j}$ and $C_{i,j}$ are the packet error rate and capacity of link (i,j), respectively. And we assume that the packet error rate of a link is symmetric, that is, the packet error rate in the forwarding direction is exactly same as in the reverse one. The capacity of a link can be retrieved by the modulation method used over the link. In IEEE 802.16, modulation method can be chosen from QPSK, 16 QAM or 64 QAM. Therefore, the capacity varies from link to link.

In other words, the capacity of a link can be determined by the burst profile given in MSH-NCFG messages. Since MSH-DSCH is exchange from time to time, the packet error rate can also be retrieved by the errors or drops of MSH-DSCH that is associated with a unique sequence number. Therefore, we argue that our SWEB is especially suitable for the IEEE 802.16 mesh networks. By these assumptions, the bandwidth of a link can be estimated as:

$$C_{i,j} \cdot (1 - p_{i,j})$$

By assigning this value on each link of the mesh network, the route from the source node to the destination node (MBS, usually) can be known by the traditional shortest path algorithm, like Bellman-Ford or Dijkstra’s algorithm. However, to minimize delay time of transmission in IEEE 802.16 mesh mode, the hop count of the source node to destination node should be considered as well. Therefore, among all potential routes, the efficient end-to-end bandwidth is estimated as:

$$\frac{\min(C_{1,2} \cdot (1 - p_{1,2}), C_{2,3} \cdot (1 - p_{2,3}), ..., C_{i,j} \cdot (1 - p_{i,j}))}{2}$$

And, the path metric is to be:

$$\frac{\min(C_{1,2} \cdot (1 - p_{1,2}), C_{2,3} \cdot (1 - p_{2,3}), ..., C_{i,j} \cdot (1 - p_{i,j}))}{2} \cdot \frac{1}{hopCount}$$

The path with the largest path metric will be chosen.

4.2. Call Admission Control Algorithm

There are several aspects that compose a better QoS provisioning network. We first modify the original 3-way handshake of the IEEE 802.16 standard. Second, we describe how to evaluate the bandwidth used by a flow. And third, the CAC algorithm is given.

4.2.1. Modified Three-way Handshake

The original 3-way handshake is in Figure 4.2:

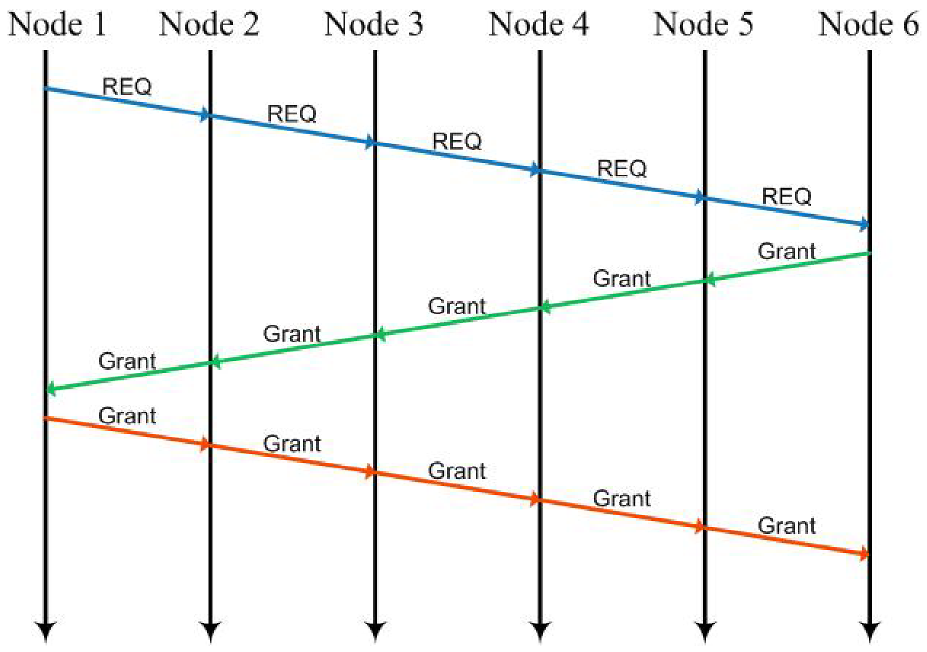

Figure 4.2: Original 3-way Handshake

After making a small modification to the original 3-way handshake, a new 3-way handshake is as in Figure 4.3. In our 3-way handshake, the call setup time is shorter than the original ones.

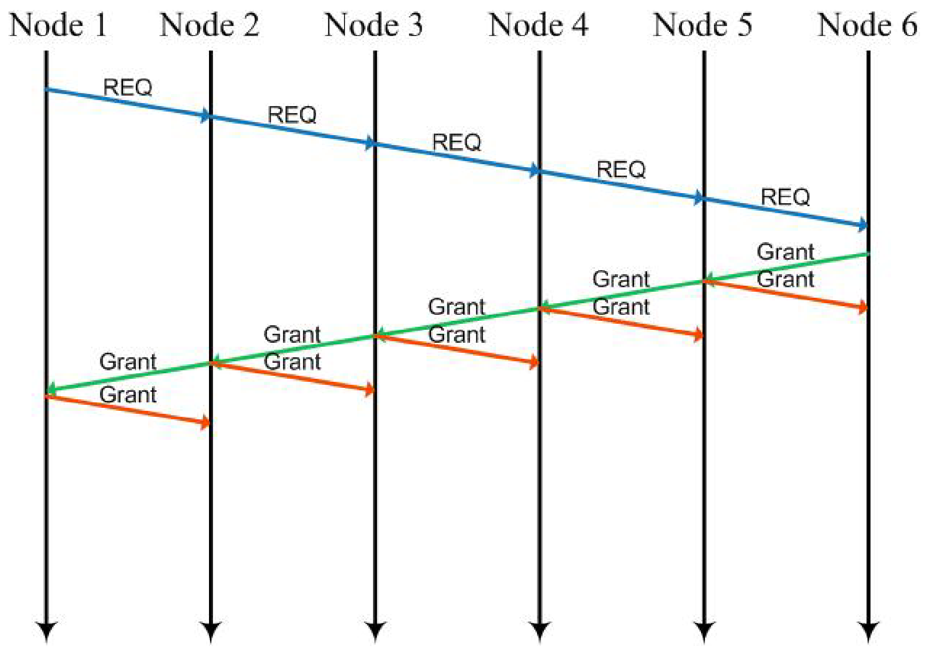

Figure 4.3: Modified 3-way Handshake.

In Figure 4.4, if the reservation is failed at node 4, the grant message sent by node 4 carries the information of rejecting the request. The modified 3-way handshake continues to convey the rejection to the source node. However, the reservations between node pairs (4,5) and (5,6) are successful, these reservation must be cancelled. Therefore, node 4 sends a MSH-DSCH:Request along with availabilities to release the reservations between node pairs (4,5) and (5,6).

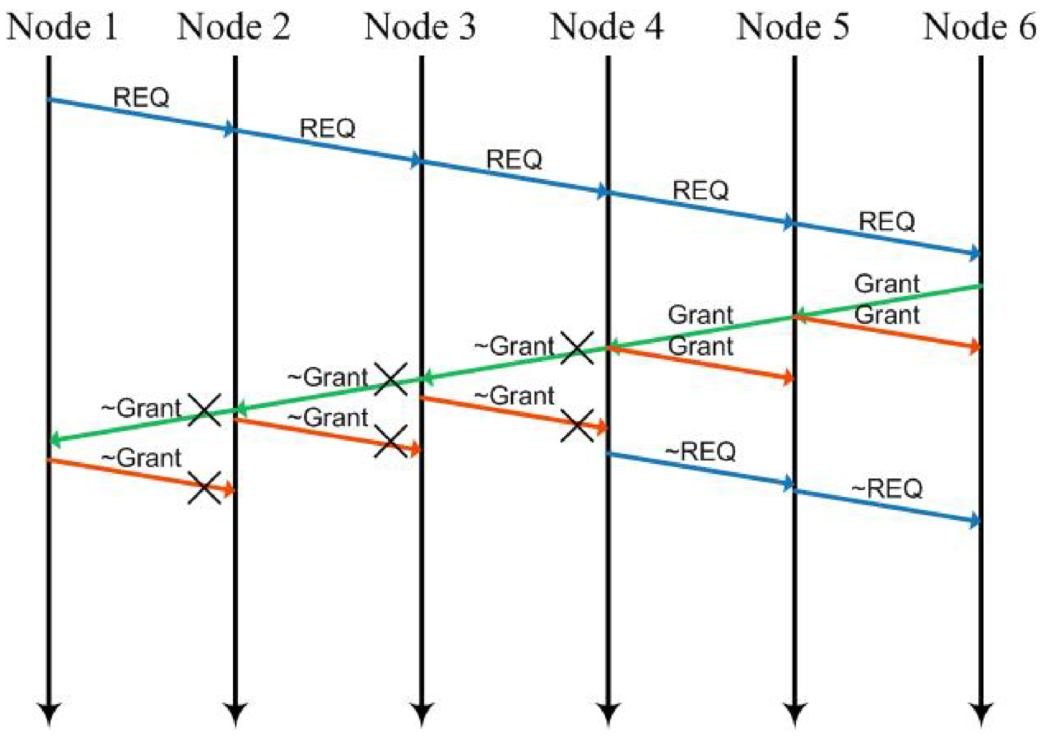

Figure 4.4: Handling the Reservation Failures.

4.2.2. Bandwidth Estimation

Before implementing any call admission control algorithm, the method to estimate the bandwidth used by a flow is necessary. Our bandwidth estimation is based on the token bucket mechanism. According to token bucket mechanism, the maximum volume of the packets transmitted within a frame is to be:

$$r \cdot f + b \tag{1}$$

Where definition of *r, f*, and *b* is same in Section 4.1., that is token rate, frame length, and token bucket size, respectively. Dividing (1) by *f*, the maximum bandwidth of a flow can be estimated as:

$$\frac{r \cdot f + b}{f} \tag{2}$$

Since the transmission does not always reach the maximum bandwidth allowed in any frame, the value of (2) would be too big for a flow, therefore, causing the bandwidth wastes. If the bandwidth is estimated as a smaller value, the delay requirements of real-time flows may not be met. Therefore, a better bandwidth estimation that considers the delay requirements of real-time flows is invented.

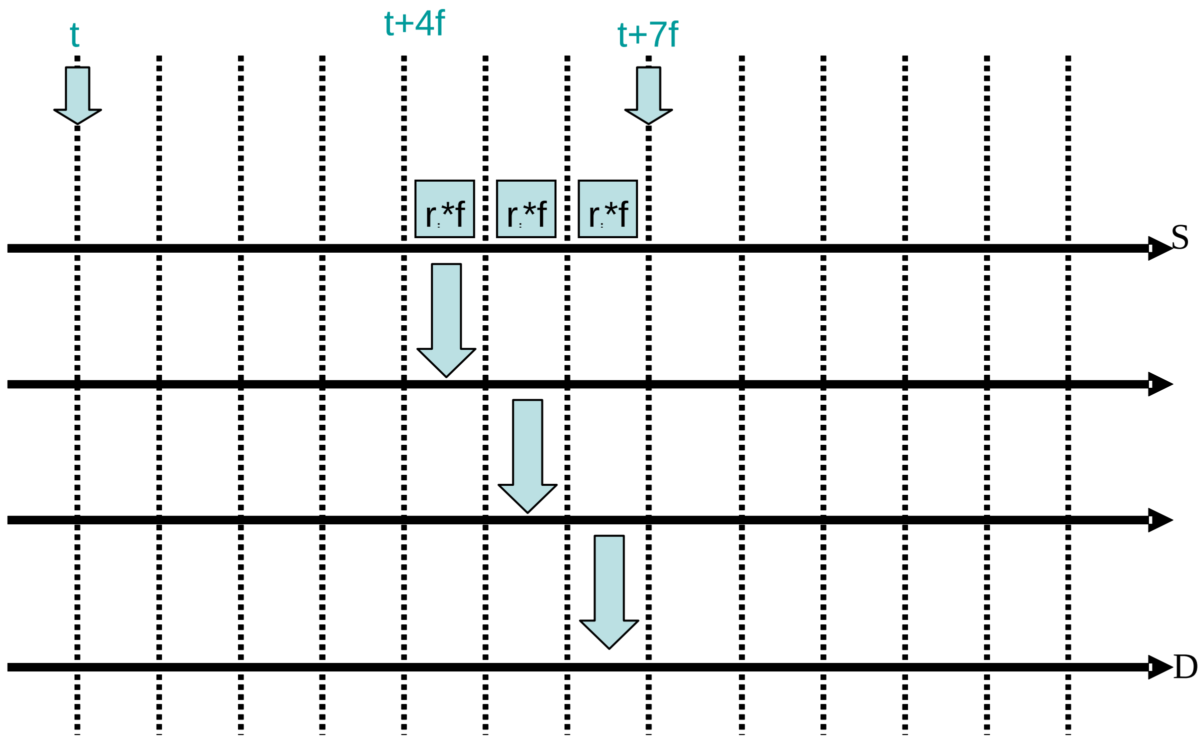

Figure 4.5: The Scenario of a Real-time Flow and Its Deadline.

Assume that a real-time flow generates the packets in time *t*, and its delay requirement is *7*f*. The hop count of this source station is 3. To satisfy its delay requirement, the packets generated in [*t*, *t+f*] must be started to send in [*t+3f*, *t+4f*]. The scenario is in Figure 4.5.

If the tokens stored in the token bucket are completely consumed in the time interval [*t+5f*, *t+6f*]. These *b* units of data must start to be sent in [*t+8f*, *t+9f*] at latest, because the deadline lays at *t+12f*. The scenario is in Figure 4.6.

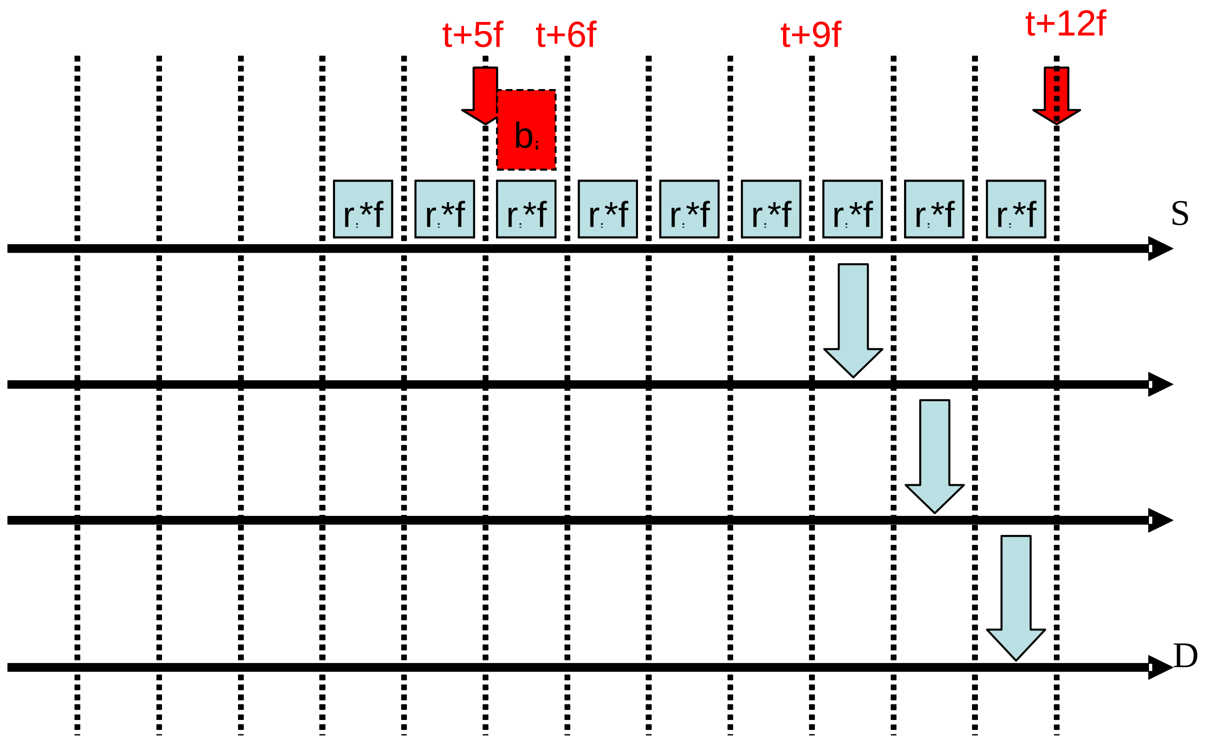

Figure 4.6: The Scenario of a Real-time Flow and Its Deadline for *b* Units of Packets

Under the network situation like this, there are two frames between the packets are generated and being sent. Therefore, we can share the b_i units of the packets in these frames. This is depicted in Figure 4.7

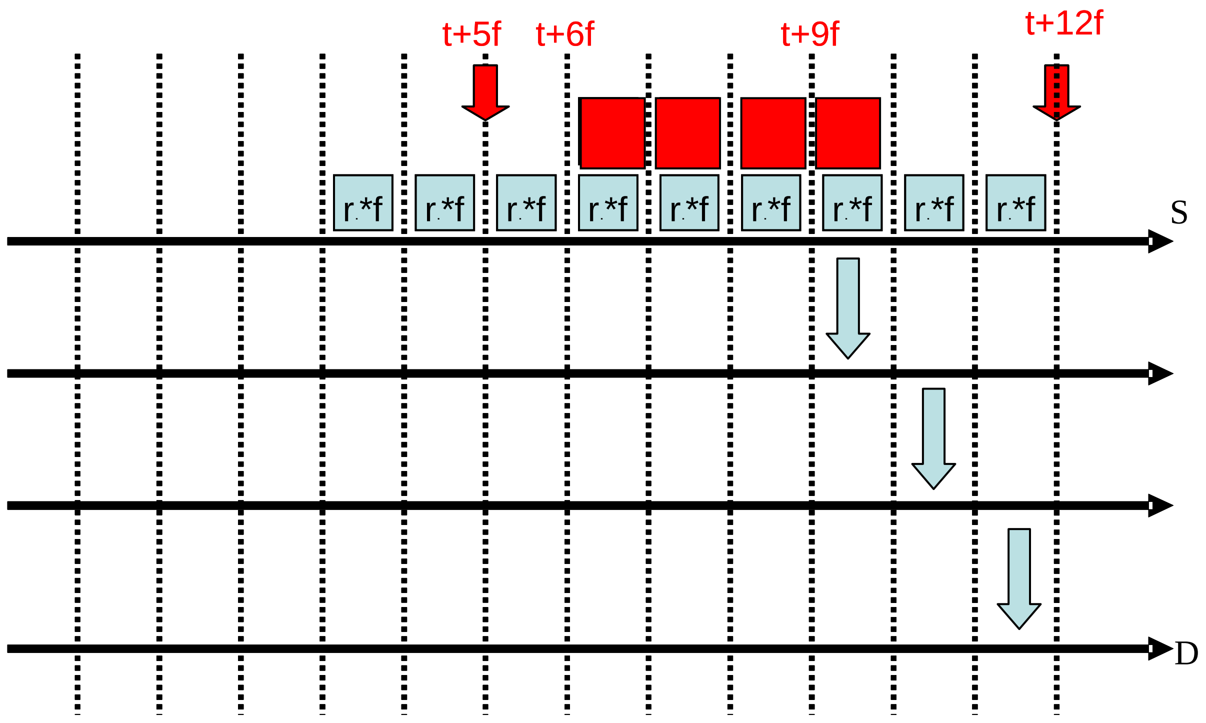

Figure 4.7: Sharing The *b* Units of Packets.

With the examples above, the better bandwidth estimation that is modified from token bucket mechanism is as follows. Assume that d_i is the delay requirement of a real-time flow *i*. r_i, b_i is its parameters of token bucket mechanism, and *f* still is the frame length. *h* is the transmission hop count of the source station to the MBS. Using theses parameters, the maximum volume transmitted by a traffic flow can be estimated as:

$$r_i \cdot f + \frac{b_i}{m_i} \tag{3}$$

, where $m_i = \left\lfloor \frac{d_i}{f} \right\rfloor - h_i$.

Dividing (3) by *f*, the bandwidth can be estimated as:

$$r_i + \frac{b_i}{m_i \cdot f} \tag{4}$$

We use (4) as the bandwidth estimation of each connection and adopt it in the CAC algorithm, which is introduced as follows.

4.2.3. Call Admission Control Algorithm

Our call admission control algorithm is called TAC, Token bucket-base Admission Control, which is designed to avoid starvation for low priority class traffic and guarantee the delay requirement of real-time traffics. The traffic flows are assumed to be divided into three classes: CBR (Constant Bit Rate), VBR (Variable Bit Rate), and BE (Best Effort), where CBR has the highest priority, and BE is lowest. The CBR traffic flows are assumed to be used for VoIP streams, having the stringent delay requirements; VBR traffic is used for real-time video streaming, such as MPEG4 streams, which has loose delay requirements. And BE is for text-based data streams, having minimum or none delay requirements.

By using (4) as bandwidth estimation of each flow, the delay requirement of real-time traffics can be met. To avoid starvation of low priority traffic flows, the minimum usage of each class is defined, they are: *CBR_min*, *VBR_min*, and *BE_min*. The minimum usage information is in terms of timeslots in data subframe. In the network, each traffic flow has its estimation of bandwidth. When the bandwidth used by a class exceeds its minimum usage, the new-coming flow will have lower priority to access the channel and tends to be preempted.

Overall concept of the CAC algorithm is as follows: When a station receives a MSH-DSCH:request, it examines whether the current usage of each class exceeds their minimum usage or not. If it does, the new-coming class will be marked as “downgraded” flows, If another MSH-DSCH:request from higher class traffic comes in, the “downgraded” flows will have bigger chance to be preempted. The concept is in figure 4.6.

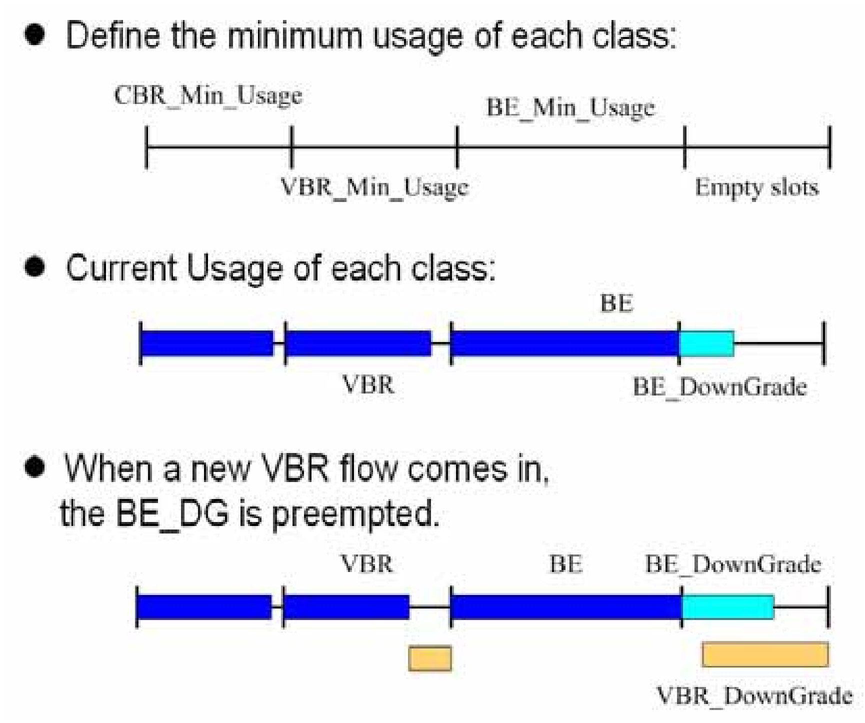
Define the minimum usage of each class:
CBR_Min_Usage
BE_Min_Usage
VBR_Min_Usage
Empty slots
Current Usage of each class:
BE
VBR
BE_DownGrade
When a new VBR flow comes in, the BE_DG is preempted.
VBR
BE
BE_DownGrade
VBR_DownGrade

Figure 4.8: Concept of CAC.

To make the downgraded and un-downgraded flows different, the fields in CID (Connection Identifier) can be used. In CID, three fields, reliability, priority and drop precedence, indicate the service level of a connection:

- Reliability: 1 bit. To indicate if the message needs re-transmission or not.
- Priority: 3 bits. Simply represents the class level of the message.
- Drop Precedence: 2 bits. Messages with larger Drop Precedence have higher drop likelihood during congestion.

By these three fields, the QoS mapping is in Table 4.1.

Table 4.1: QoS mappings.

	Priority	Reliability	Drop Precedence
CBR	7	0	0
CBR_Downgraded	4	0	1
VBR	6	0	0
VBR_ Downgraded	3	0	2
BE	5	1	0
BE_ Downgraded	2	1	3

In Table 4.1, as BE is for data or text-based connection, the retransmission is necessary. Therefore, the reliability field of both BE and BE_Downgrade is 1. Also, the order of these six traffic flow types is indicated in the priority field, and the values of their drop precedence field are given with the same philosophy.

We develop our CAC algorithm as follows:

CAC algorithm:

Step 1)

A new flow comes in with its *BW_req* (bandwidth request) in the unit of timeslots. Set *BW_avail* (Available Bandwidth) as the total empty timeslot number.

Step 2)

The ingress-station which handles the request checks if the BW_req < BW_avail or not, If yes, go to step 3. Or else go to step 4.

Step 3)

The station determines to downgrade the new-incoming flow or not, by comparing the current usage and the pre-defined minimum usage of the traffic class.

Step 4)

The station checks if the current usage exceeds the minimum usage of the traffic class. If yes, the flow is rejected. Or else, go to step 5.

Step 5)

Check the timeslots used by downgraded flows in the order of *BE_Downgrade, VBR_downgrade*, and *BE_downgrade*. If there is no such timeslots, the new-incoming flow is rejected. Or else, set these timeslots available, which means, preempt these timeslots. And update *BW_avail*. Go to step 2.

With the above algorithm and the bandwidth estimation in 4.2.2, our system works as follows: when a connection *i*, which is controlled by the token bucket mechanism, in initiated, the token rate, tr_i, bucket size, b_i, delay requirement for real-time traffic d_i, and session duration $n*f$, is given. And the bandwidth required is estimated by applying (4). The bandwidth information is further translated into the number of timeslots needed, and was put in MSH-DSCH messages. After applying our modified 3-way handshake for MSH-DSCH messages, while each station that handles MSH-DSCH message runs the CAC algorithm above, the new incoming flow can be determined to be accepted or not, to be downgraded or not. The flowcharts of CBR, VBR and BE traffics is in Figure 4.9, Figure 4.10 and Figure 4.11 respectively.

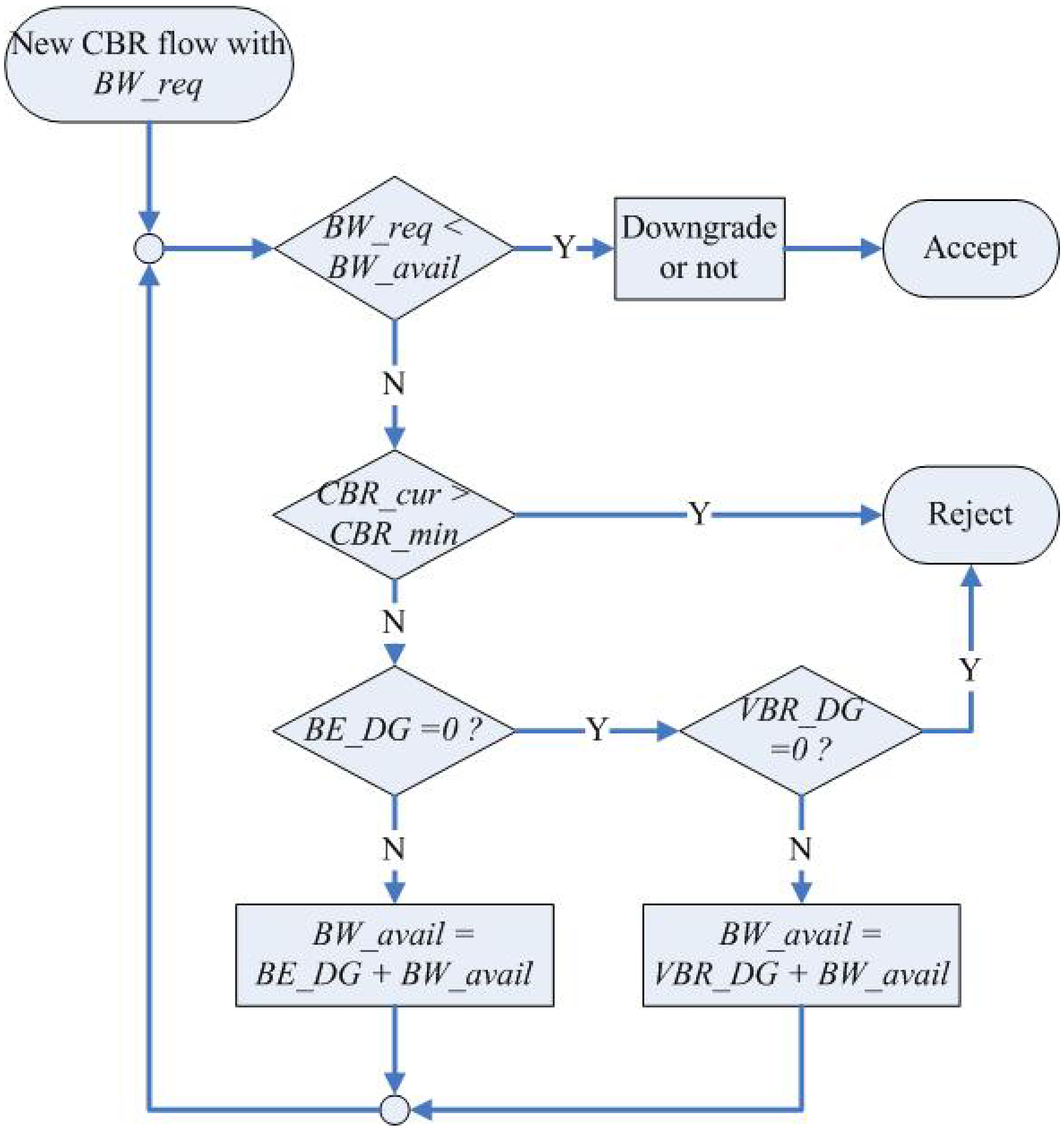

Figure 4.9: Flowchart of CAC algorithm for CBR flows.

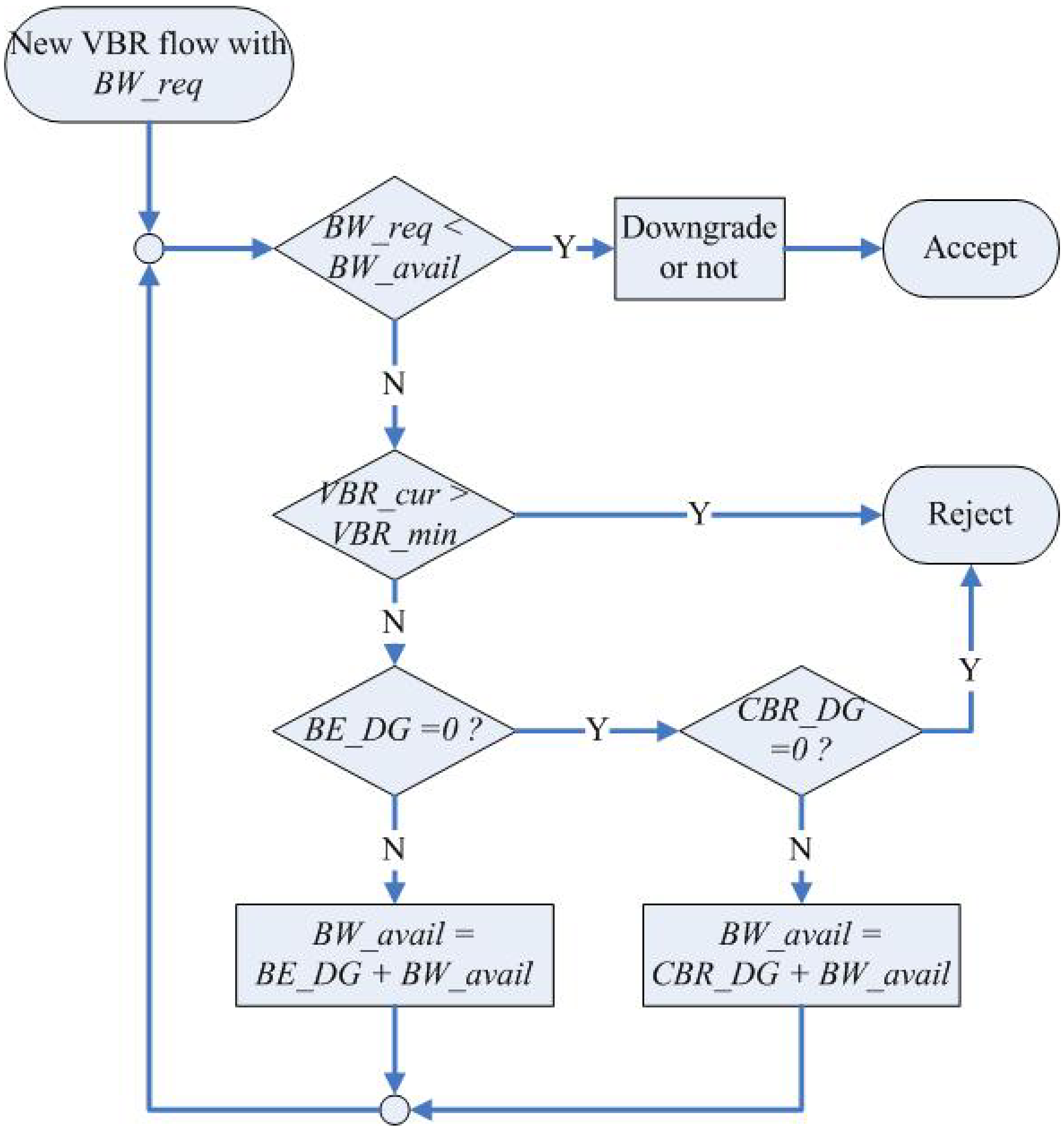

Figure 4.10: Flowchart of CAC algorithm for VBR flows.

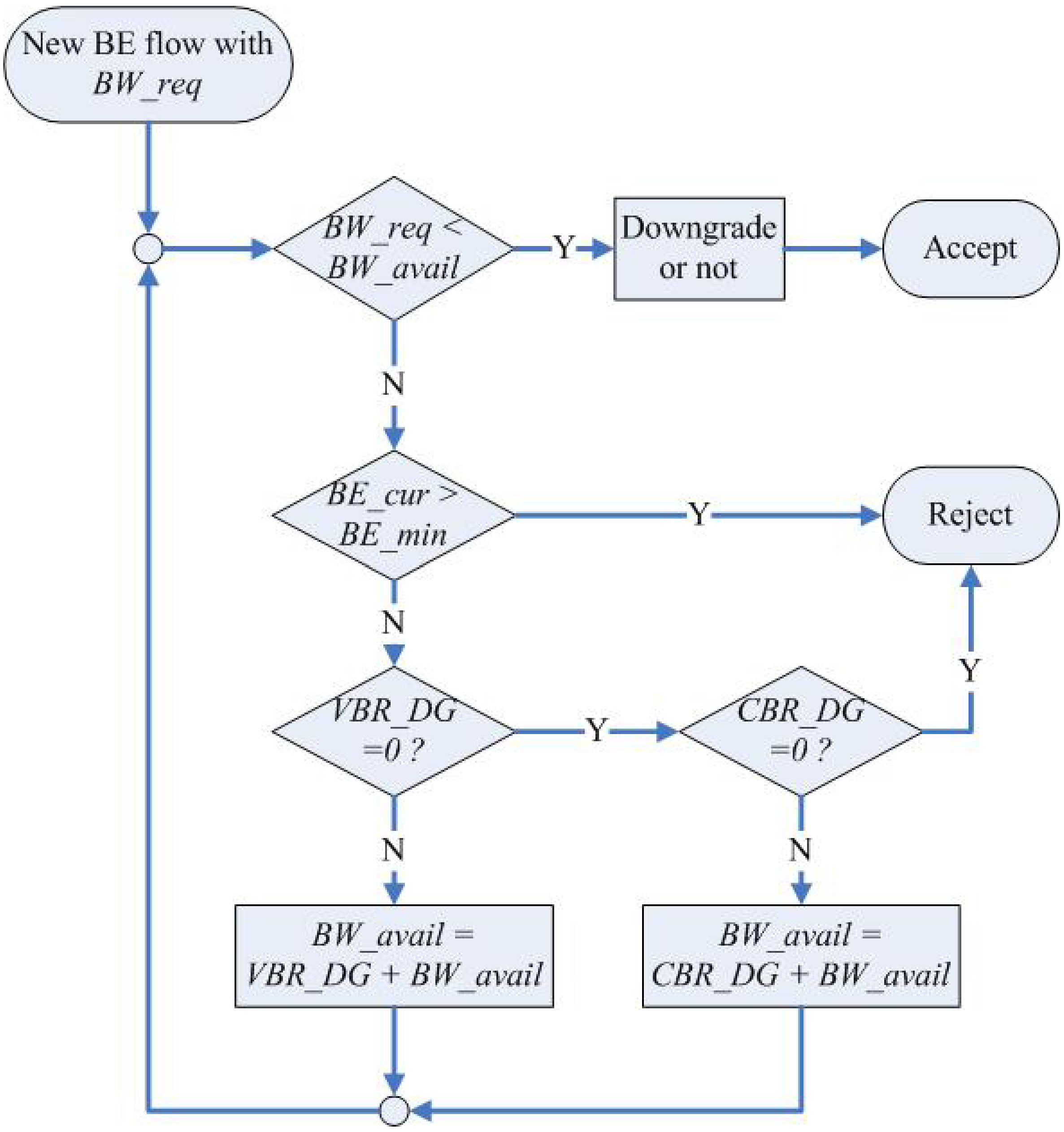

Figure 4.11: Flowchart of CAC algorithm for BE flows.