CHAPTER TWO

LITERATURE REVIEW

This chapter presents the background of the congestion control process and provides a survey of existing approaches used to address the congestion control problem. The classification criteria are set, and existing congestion control approaches are categorized in Section 2.1. Each category is discussed in the subsequent subsection according to the aforementioned classification. TCP congestion control approaches are discussed in Section 2.2. The router buffer approach is discussed in Section 2.3. Scheduling is described in Section 2.4. The buffer management approach is explored in Section 2.5. Controlling congestion at a late stage is discussed in Section 2.5.1. Parameter-based early congestion control approaches are examined in Section 2.5.2. Fuzzy-based congestion control approaches are examined in Section 2.5.3. The discrete-time queue approach is analyzed in Section 2.6. Traffic modeling is studied in Sections 2.7. Simulation environment is discussed in Section 2.8. A chapter summary is provided in Section 2.9.

2.1 Congestion Control

Congestion is one of the main issues prevalent in network router buffers (Ryu et al., 2003; Welzl, 2005; Baklizi et al., 2012; Attiya & El-Khobby, 2012) because it affects network efficiency by causing low throughput functionality, high queuing delay, PL, aql mismanagement, and potential buffer overflow (Braden et al., 1998; Welzl, 2005). Many approaches have been developed to immediately control

congested networks. The instantaneous and effective control of congestion leads to high QoS for traffic load and ensures the fair sharing of resources. Existing congestion control approaches can be divided broadly into two categories (see Figure 2.1), namely, TCP congestion control approaches and router-based approaches. Router-based approaches can be furthered classified into buffer-based approaches and scheduling approaches. Buffer-based approaches can also be classified into approaches that control congestion at a late stage and approaches that control congestion at an early stage. Methods for congestion control at an early stage are classified into parameter-based early congestion methods and fuzzy-based early congestion methods. The goal in developing these various approaches is to detect network congestion before the buffer overflows, mitigate congestion, and ensure QoS.

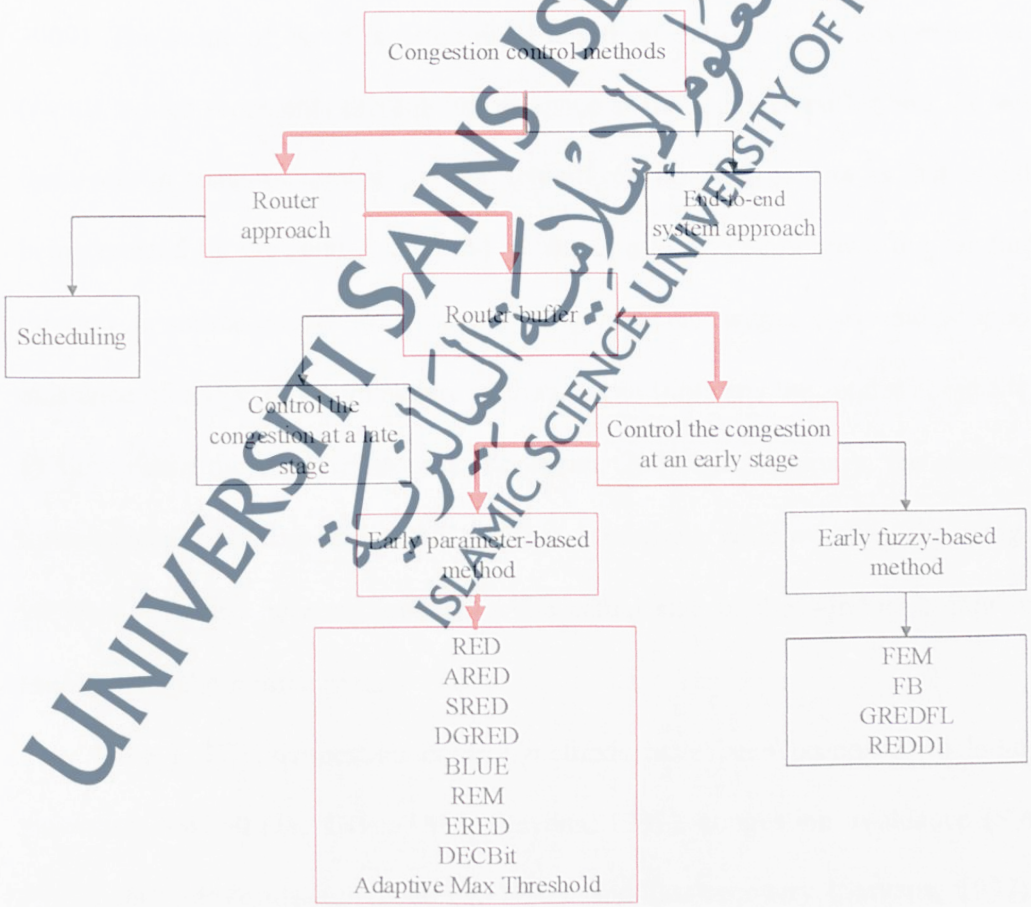


Figure 2.1: Classification of Congestion Control Approaches

2.2 TCP Congestion Control

TCP is mainly utilized to provide a reliable connection between the source and the destination in the Internet (Postel, 1981; Forouzan, 2009; Fall & Stevens, 2011). TCP offers various advantages. First, it establishes a connection between two ends using a three-way handshake. Second, TCP ensures the delivery of packets by using acknowledgment (ACK) sent by the receiver and then received by the sender as proof of data delivery. Third, TCP uses a timer to determine when the data packets are sent or retransmitted (Postel, 1981; Forouzan, 2009).

In addition, TCP was invented with a windowing mechanism, in which each source sends packets based on a congestion window (cwnd), which represents the number of packets that the sender can transmit before receiving ACK (Allman et al., 2009). The value of cwnd is determined based on the receiver's advertised window (rwnd), which represents the amount of space at the receiver buffer and the network feedback. In terms of network, if the network cannot deliver data as fast as data are being created by the sender, then the sender is asked to slow down the sending rate through a source-quenched message. This message warns the sender about the existence of congestion somewhere in the path and informs the sender about the need to slow down the sending process (Forouzan, 2009). In summary, the sender holds three pieces of information: the receiver-advertised window size, the congestion window size, and network feedback. The actual size of the window comprises the minimum of these three parts.

Several TCP congestion control methods have been proposed, including the slow start method (Jacobson, 1988; Stevens, 1997), congestion avoidance (Stevens, 1997), fast retransmission (Stevens, 1997), and fast recovery (Stevens, 1997). The slow start method works when a new TCP connection is established or when TCP

connections remain active after either a long idle period or expiration timeout. The source initializes its cwnd to the size of a single TCP segment, which is the segment size announced by the other end (the default size, typically 536 or 512) (Stevens, 1997). The cwnd corresponds to the amount of data that can be sent from the source to its destination across the network before any congestion is noticed; its issue prevents rapid build-up in router buffers (Stevens, 1997).

The slow start method (Jacobson, 1988; Stevens, 1997; Forouzan, 2009) is used by a TCP sender to control the amount of outstanding data being injected into the network. Initially, the size of the congestion window (cwnd) is set to one segment. Once ACK is received by destination, the cwnd is increased by one to increase the number of transmission packets. The slow start method increases the number of segments exponentially as long as the sender continues to receive ACK from the other side. The sender stops sending the segment using the slow start method in two situations: if the value of cwnd exceeds the destination receiving window and if congestion occurs at the network and is announced to the source through duplicated ACKs as shown in Figure 2.2. In both situations, the sender stops applying the slow start method and starts congestion avoidance.

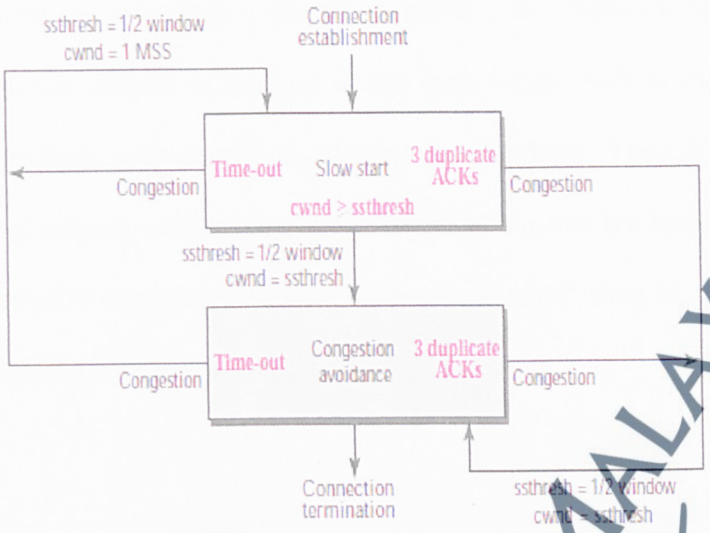
In the congestion avoidance method (Jacobson, 1988; Forouzan, 2009), two parameters, namely, cwnd and the slow start threshold (ssthreshold) that decides whether the slow start method or the congestion avoidance method should be used (Allman et al., 2009), are determined (Stevens, 1997). As the network becomes congested, the source reduces its cwnd size by setting the ssthreshold parameter to half of the cwnd size and entering the congestion avoidance mode. The source linearly increases the number of segments by one when an ACK is successfully received.

The TCP source detects congestion based on the following scenarios: 1) if the source receives three duplicate ACKs and 2) if retransmission timeout occurs (end of retransmission timeout period). As shown in Figure 2.2, most TCP implementations involve two reactions. A timeout is indicative of congestion and the dropping of transmitted packets. In such a case, the TCP should first ensure that the threshold value is half of the current window size. It should then ensure that the cwnd is returned to one segment and that the TCP starts with the slow start phase. In case three duplicated ACKs are received, the notification for congestion is weak. TCP retransmits the lost segment in this scenario.

The fast retransmission method (Stevens, 1997) reduces transmission by entering the slow start method and reducing the cwnd size to one segment, setting up the ssthreshold to half of the cwnd, and finally retransmitting the missing segments to the destination.

The fast recovery method was proposed by Van Jacobson in (1990) to avoid throughput deterioration caused by rapid retransmission and to manage congestion (Stevens, 1997). The ssthreshold value is set to half of the cwnd, and the cwnd is set to the ssthreshold value.

In summary, the main goal of any TCP congestion control method is to control network congestion by decreasing the transmission rate of the source. TCP is a typical example of an end-to-end system transmission protocol (Chen et al., 2007; Chen & Yang, 2009; Wang et al., 2010). The TCP source discovers congestion by identifying the packet dropping rate at router buffers. The source responds to this type of congestion by decreasing the transmission rates.



Source: TCP/IP protocol suite, 2009

Figure 2.2: Summary of TCP Congestion Policy

2.3 Router Congestion Control

The Internet is a group of connected networks in which packets are transferred to their potential destination. Subsequently, transmitted packets are passed through intermediate routers until they reach their destination. Figure 2.3 shows a typical network topology.

A packet received by a router is queued at the router buffer before being forwarded to their correct path based on the router table. The buffer is important because it alleviates the loss of incoming packets when other packets are being processed by the router. The router buffer has a finite size for holding queuing packets. When the number of arriving packets exceeds the amount of available space, the router will discard some incoming packets. Consequently, queuing or dropping decision must be made wisely to alleviate packet loss (PL) and buffering delay (D). Packet delay occurs when the number of packets that are currently situated in the

buffer increases relatively (Senthilkumaran & Sankaranarayanan, 2013). Subsequently, the packets at the end of the queue must wait at the buffer until the packets before them are transmitted, thereby causing delay. Thus, if a router buffer is too large, then packets can be accommodated and their number increases, which leads to packet delay. By contrast, if a router buffer is too small, then PL will occur because incoming packets cannot be accommodated.

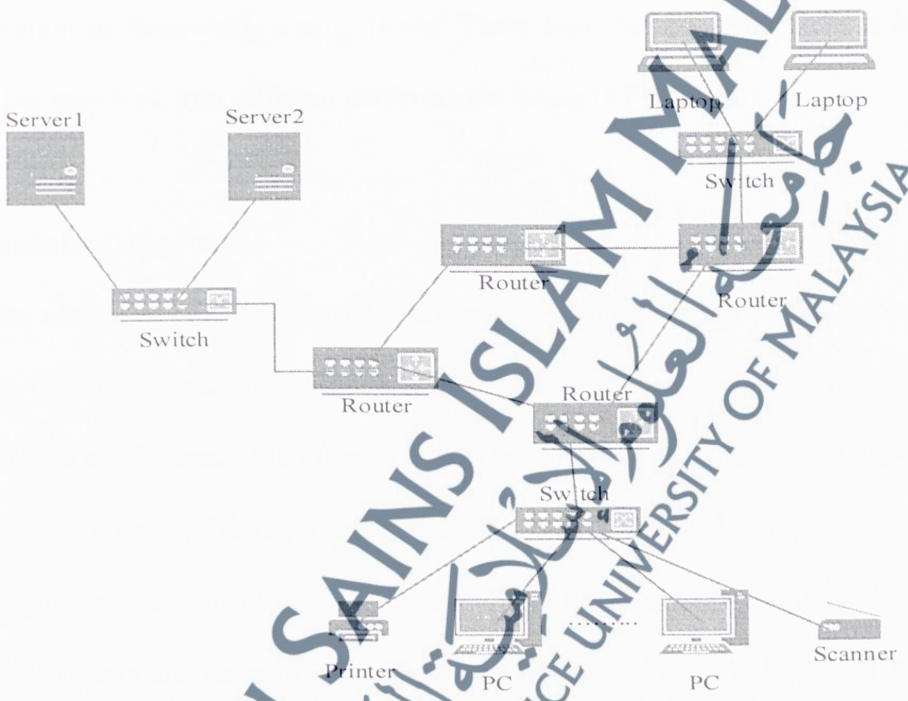


Figure 2.3: Typical Network Topology

Routers perform an important function in the congestion phenomenon by switching packets among a number of networks. In a typical manner, router-based congestion control mechanisms are categorized into two types: queue management (Ryu et al., 2003; Zhang et al., 2011) and scheduling (Malhotra & Sharma, 2012). The former includes methods for administering ql in a buffer by dropping packets (Joshi et al., 2005). The latter includes methods that determine which order of packets should be sent next and are used to allocate available bandwidth among flows. However, scheduling alone will be unable to help avoid packet loss. Buffer management

methods are an important aspect of congestion control because network load can change rapidly and probably cause packet dropping, except in the case in which sufficient buffer space is available to hold packets temporarily (Lefelhocz et al., 1996). "Queue management" and "scheduling" differ in terms of how ql at the router buffer is managed; that is, queue management drops packets when necessary, whereas scheduling selects which packet to be served next and is used primarily to administer the allocation of bandwidth among flows. These two router approaches are closely related, but they look into different performance issues (RFC2309).

2.4 Scheduling Approach

The network resources can be managed by using either a packet scheduling approach or a buffer management approach. When a router receives a few arriving packets (light traffic load), the effects of both scheduling and buffer management are unnoticeable. When traffic load increases, both scheduling and buffer management play a significant role in controlling flows. Scheduling controls the order in which individual packets are served (or whether they get served at all). The most popular scheduling methods include first in first out (FIFO), round robin (RR), priority queue (PQ), fairness queue (FQ), and weighted fair queuing (WFQ).

FIFO is the most popular scheduling method because it simply serves packets in the order of their arrival by placing all packets in one queue. Packets are treated equally and served in the same order by placing packets arriving from different flows into a single queue. Although the amount of buffer space (queue) at each router is finite, the router becomes inclined to drop incoming packets if the queue of the router buffer becomes full. Packet dropping can be done regardless of which flow the packet

belongs to or how significant the packet is (Silberschatz et al., 1998; Rashed & Kabir, 2010; Rastogi & Srivastava, 2013; Musa et al., 2014).

RR (Silberschatz et al., 1998) is one of the oldest and simplest scheduling methods. It is similar to FIFO because of its preemption functionality that allows switching between processes (Matarneh, 2009; Rastogi & Srivastava, 2013; Matthew et al., 2014). RR allocates a fixed time (10–100 ms), called quantum, for each process to serve in circular order without any priority. Thus, it is used widely in time-sharing systems.

PQ is a simple method for supporting differentiated service classes. The packets are placed into different queues according to the priority assigned to each of them by a classifier. The packet with a high priority is served before the other packets. Thus, PQ provides the fastest handling for important traffic during transmission by providing the high-priority queues with absolute preferential treatment over their low-priority counterparts. PQ is useful with mission-critical traffic, such as real-time VoIP traffic. FIFO is used in PQ within each priority queue (Demers et al., 1989; Silberschatz et al., 1998; Rastogi & Srivastava, 2013; Musa et al., 2014).

FQ, a concept recommended by John Nagle (1987), is the foundation for a class of queue scheduling disciplines designed to ensure fairness among all network traffic and to prevent bursty traffic from consuming more than its fair share of the output port bandwidth. In FQ, packets are classified by the system into their associated flow and then assigned to a queue that is especially dedicated to that flow. FQ serves one packet at a time in RR order. The concept behind FQ is to maintain a separate queue for each flow that is being handled by the router (Demers et al., 1989; Silberschatz et al., 1998; Rastogi & Srivastava, 2013; Musa et al., 2014).

WFQ was proposed by Lixia Zhang, Alan Demers, Sriinivasan Keshav, and Scott Shenke in (1989). WFQ was designed to overcome the limitations of FQ. WFQ supports flows with a variety of bandwidth requirements by providing a weight to each queue, which represents a percentage usability of the output port bandwidth. WFQ ensures fairness by preventing the allocation of more bandwidth for flows that contain a large number of packets. This situation is achieved by assigning a queue for each flow and applying weight to provide bandwidth for each flow relative to other flows (Demers et al., 1989; Silberschatz et al., 1998; Rastogi & Srivastava, 2013; Musa et al., 2014).

Overall, scheduling alone does not prevent substantial PLs. Given that network load can change rapidly, packets are likely to be dropped whenever a surge of packets occurs, unless an adequate buffer space is available to harbor the packets temporarily. Thus, buffer management methods comprise an important aspect of congestion control.

2.5 Buffer Management Approach

Router buffers detect congestion by evaluating network behavior. In particular, routers detect congestion based on available spaces at router buffers and the numbers of arriving packets. If the number of packets exceeds a specific level, then the router generates a congestion notification and sends it back to the source, which is then prompted to reduce its transmission rate.

2.5.1 Congestion Control at a Late Stage (Passive Queue Management)

In the late congestion control category, the router buffer can detect congestion after the buffer overflows. A late congestion notification is sent back to the source, and a late congestion response takes place from the router buffer. One of the mechanisms of this category is to set up the buffer size in the router to

a fixed value and drop all packets that exceed this value. Drop tail (DT) serves as an example of this category.

2.5.1.1 Drop Tail (DT)

The DT method is one of the traditional congestion control methods that detects congestion based on the fixed size of the router buffer (Braden et al., 1998). The size of the router buffer can vary (i.e., intermediate, maximum, or minimum). If the size of the router buffer is set to the maximum value, then the number of queuing packets at the router buffer increases; consequently, queuing delays increase. If the size of the router buffer is set to the minimum value, then throughput rationally decreases. In all cases, the router buffer drops all packets if size reaches the threshold. DT parameters are not adaptive, and thus, parameter values cannot be modified after initialization (Figure 2.4) (Brandauer et al., 2001; Stanojevic et al., 2006; Chandra & Subramani, 2010).

Figure 2.4: Router Buffer Control Using DT

The disadvantages of DT lie in the lockout phenomenon (Braden et al., 1998; Sharma & Behra, 2016), which is explained as follows. One or more flows can monopolize router buffer spaces and prevent other flows from sharing the space of the router buffer. Another drawback is the full

queue (Braden et al., 1998), which is explained as the detection of congestion after the buffer overflows. The third drawback is global synchronization (Ryu et al., 2003; Sharma & Behra, 2016), in which a congestion notification is sent back to all sources, such that they reduce their transmission rates and thereby reduce throughput.

2.5.2 Parameter Setting- Based Early Congestion Control Methods

In these methods, congestion is detected and controlled at an early stage prior to router overflow and to achieve high QoS for traffic loads by efficiently managing the queue buffer; consequently, network resources are utilized efficiently (Tassiulas et al., 1994; Salim & Ahmed, 2000; Bitorika et al., 2004; Muezerie et al., 2005; Guffens & Bastin, 2005; Patel & Bhatnagar, 2017).These methods are called Active Queue Management (AQM) methods (Das et al., 2013; Singh & Balveer, 2013; Kiruthiga & Raj, 2014). Unlike the DT method, which provides a late response to the congestion problem, AQM methods provide an early prediction of congestion and do not wait until the router buffer overflows. Sources can be informed about congestion either explicitly or implicitly. In the former, the method sends back an explicit notification (explicit feedback) to the source using an explicit congestion notification (ECN) bit in the packet header (Floyd, 1994; Ramakrishnan et al., 2001). The destination that receives this packet is informed about the congestion and sends back an acknowledgment to the source. In the latter, an implicit feedback is sent by dropping packets at the router buffer. AQM methods include DECbit, Random Early Detection (RED), Adaptive Random Early Detection (ARED), Stabilized Random Early Drop (SRED), Gentle Random Early Detection (GRED), Adaptive GRED (AGRED),

Random Exponential Marking (REM), Effective RED (ERED), Dynamic Gentle Random Early Detection (DGRED), and BLUE. These methods are adapted to the congestion status at the router buffer by altering the values of the utilized parameters based on the network and the congestion indicator in an attempt to ease the difficulties of parameter setting.

2.5.2.1 DECbit Method

The DECbit method is built based on aql as a congestion indicator (see Figure 2.5). DECbit computes the size of the buffer for every arriving packet under the condition that the size is larger than 1. DECbit marks and transfers the packet to its potential destination. Once the packet arrives at the destination, the destination notifies the sender about the occurrence of congestion by resending the marked packet. The source reduces the transmission rate by decreasing their transmission window by one packet for two round trip times (Sharifahmadian & Latifi, 2012). The transmission window size is decreased exponentially if more than half of the packets in the last window are returned with marked ACK (Lapsley & Low, 1999; Athuraliya et al., 2001). Otherwise, the transmission rate is increased linearly (Floyd & Jacobson, 1993; Floyd, 2000; Floyd et al., 2001).

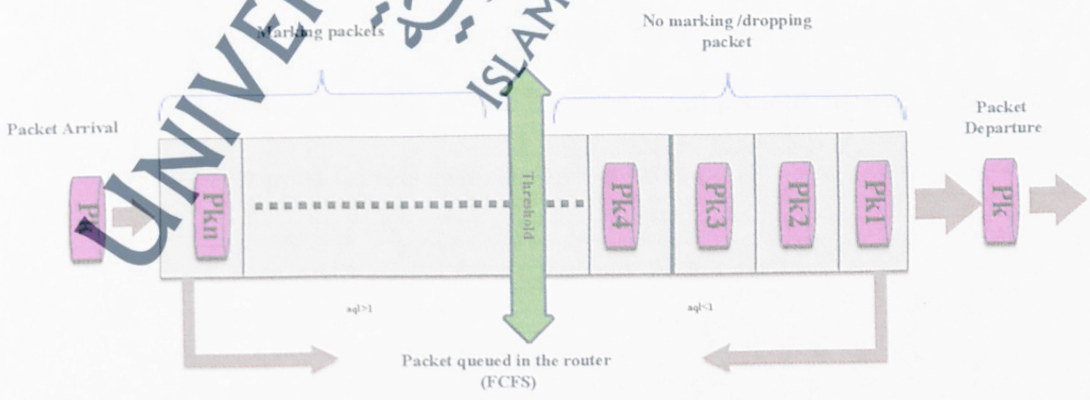


Figure 2.5: Router Buffer Control Using DECbit.

In general, the DECbit method offers the following advantageous properties: simple, distributed, optimized, and low cost in terms of the overhead for notifying the sender by marking packets dynamically. By contrast, the DECbit method is unable to serve bursty flow because it takes time to send a notification for the source to reduce the transmission rate.

2.5.2.2 RED Method

The RED was proposed by Floyd and Jacobson (1993). It is a non-adaptive method that is successfully adopted by the Internet Engineering Task Force in RFC2309 (Braden et al., 1998). RED does not notify the sender in an explicit manner like DECbit does, but it does notify the source in an implicit manner through dropped packets resulting from the congestion control implemented by RED. DP is calculated based on aql and two thresholds (minthreshold and maxthreshold) (Floyd & Jacobson, 1993), as shown in Figure 2.6. These thresholds represent two positions in the router buffer and serve as good indicators of congestion status. RED detects congestion by comparing the computed aql with the thresholds. When the aql is less than the minthreshold, no congestion occurs and no packet is dropped. When the aql falls between the two thresholds, then the arriving packets are dropped randomly to reduce the probability of congestion. Moreover, when the aql is above the maxthreshold, then all arriving packets are dropped. In this case, the DP value is 1.

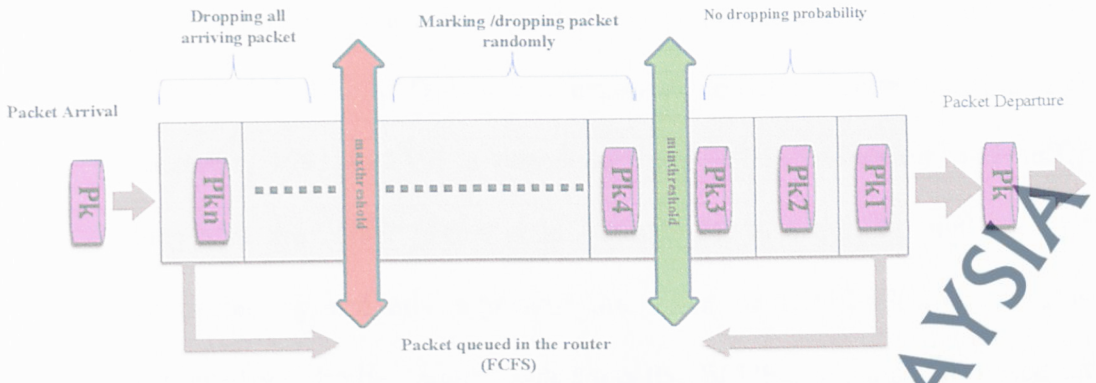


Figure 2.6: Single Router Buffer for RED.

RED makes the leap between traditional methods and AQM methods that detect congestion at an early stage and addresses the global synchronization problem (Fan et al., 2012). However, RED exhibits several drawbacks. For example, RED calculates DP by computing aql based on traffic load (number of connections) and not according to the actual status of the packet load. This method further leads to reduced network performance in the form of delays and PL. In effect, the goal of using DP is to keep aql between the minthreshold and maxthreshold. However, RED does not achieve stability when traffic load changes promptly (i.e., bursty traffic) (Floyd et al., 2001; Feng et al., 2002; Abdel-Jaber et al., 2014).

2.5.2.3 BLUE Method

The BLUE method was proposed by Feng, Shin, Kandlur, and Saha in (2002) to improve the performance of the RED method. The BLUE method is mainly dependent on the calculated DP that is based on a single threshold (see Figure 2.7). BLUE computes ql and compares it with the threshold. When ql at the router buffer surpasses the threshold, BLUE increases DP with a fixed value Pin (DP increment) to manage the congestion and attempts to maintain ql at a reasonable value while avoiding

buffer overflow. When buffer length is empty or the link is idle, DP is decreased. Unlike RED, which relies on aql to detect congestion (Floyd & Jacobson, 1993), BLUE is dependent on PL, link utilization, and buffer length at the router (Feng et al., 2002). PL and buffer length can be controlled by properly adjusting the value of DP. Link utilization is achieved via the link status. Subsequently, BLUE calculates DP based on several parameters, such as freeze time (f_time). Freeze time represents the shortest time required between two successive DPs; it can be set as a fixed value or a random value to avoid global synchronization (Braden et al., 1998; Feng et al., 2002). Pin and Pde identify the value by which DP can be increased or decreased. Furthermore, Pin must be higher than Pde to prevent under-utilization (Feng et al., 2002). BLUE addresses the global synchronization problem by either selecting an f_time parameter randomly or dropping packets randomly similar to RED. The limitation of the BLUE That is, BLUE needs to set up its parameters to particular values to ensure good performance (parameterization). The DP method is calculated as shown in Figure 2.8.

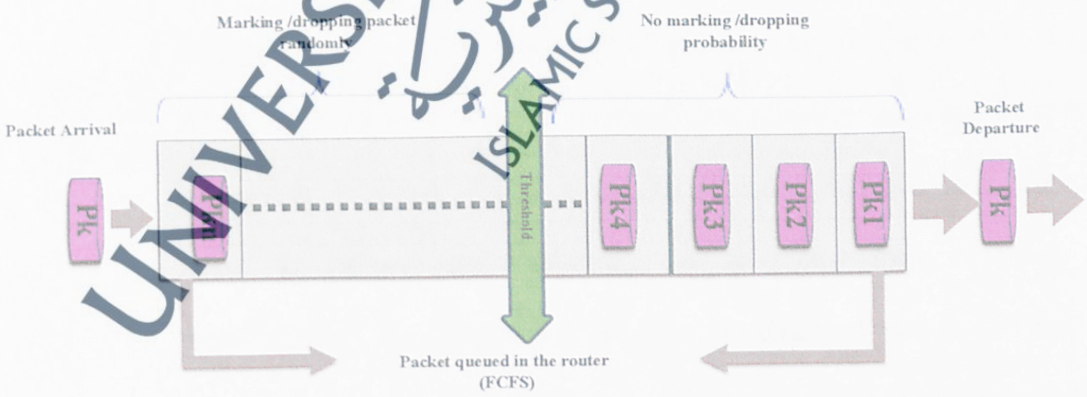


Figure 2.7: Single Router Buffer for BLUE.

```
1.  Begin
2.  if (buff1e>th)or Plos
3.  if (time _of the1_adj>f_time)
4.  DP= DP+ Pin
5.  L_adj = current
6.  if (que_buff == 0) // (buffer is empty)
7.  DP= DP-Pde
8.  L_adj = current
9.  End
    Where the que_buff=queue buffer
    buff_1e=buffer length
    Th=threshold
    plos=packet losing
    ladj=last adjustment
    time of the ladj=time of the last adjustment
    pin=pincrementpde=pdecrement
```

Figure 2.8: BLUE Method.

2.5.2.4 ARED Method

The ARED method is an adaptive method that aims to solve the aql stabilization problem of RED (Floyd et al., 2001). Despite the fact that ARED and RED have nearly the same congestion indicator techniques, ARED attempts to maintain aql at a certain level called target aql (Taql), which is calculated between the minthreshold and maxthreshold (Floyd et al., 2001) and is equal to (minthreshold+maxthreshold)/2. In addition, the D_{max} parameter, which is used in RED, is always kept between [0.01, 0.5] in ARED. ARED uses additive increase multiplicative decrease to control congestion. In effect, ARED controls congestion and does not allow the queue to increase by preventing the aql value from reaching the maxthreshold. The computed value of aql is compared with that of Taql. If the value of aql is less than that of Taql and the value of D_{max} is greater than or equal to 0.01, then D_{max} is decreased along with DP. If aql is above Taql

and D_{max} is approximately 0.5, then D_{max} is increased along with DP, as shown in Figure 2.9.

Every period of time in seconds

$$if(aql < T_{aql} \;\&\&\; D_{max} \geq 0.01)$$

$$D_{max} = D_{max} \times d1; \quad // \text{Decrease } D_{max} \text{ value}$$

$$esleif(aql > T_{aql} \;\&\&\; D_{max} \leq 0.50)$$

$$D_{max} = D_{max} + d2; \quad // \text{Increase } D_{max} \text{ value}$$

Figure 2.9: DP of ARED Method.

The main advantages of ARED include its capability to ease the parameterization problem of RED and partially stabilize aql. However, ARED has several restrictions. First, in case of heavy congestion, ARED cannot stabilize aql between the minthreshold and maxthreshold because DP calculation is based on two scenarios: no congestion emerges and congestion reaches a high level. Second, similar to that under RED, if heavy congestion occurs in ARED while aql is less than the minthreshold, then the router buffer is likely to overflow and lose all packets. Finally, the parameters of ARED must be set to certain values to obtain satisfactory performance (parameterization).

2.5.2.5 SRED Method

The SRED method (Ott et al., 1999) was proposed to overcome some of the limitations of RED, particularly in cases involving a large number of encountered connections; RED calculates aql with regard to the number of connections and is unable to ensure a fair share of connections (Ott et al., 1999). SRED uses a zombie list to store information of recently active flows by counting their variables, source addresses, destination addresses, and time stamps as shown in Figure 2.10.

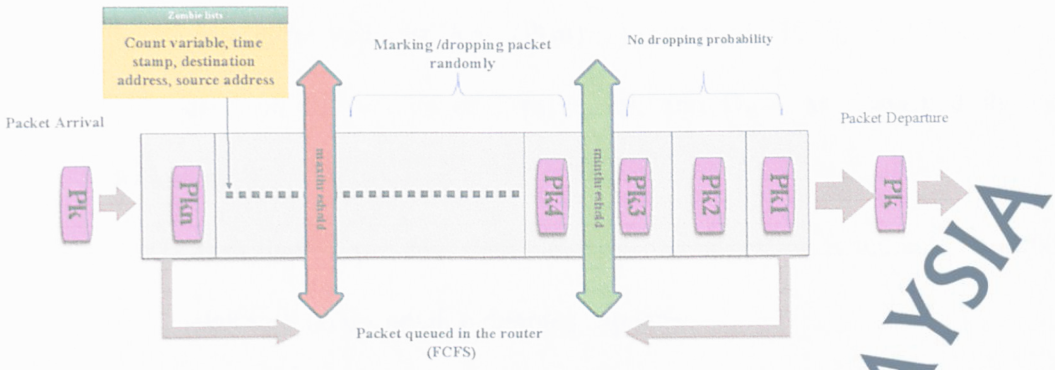


Figure 2.10: Single Router Buffer for SRED

The zombie list is initially empty. The list starts to accumulate records when traffic arrives at the router. As the zombie list becomes full, every arriving packet is compared with a randomly selected packet from the zombie list. If the record and the arriving packet correspond to the same flow, then the count increases by one, a hit process is conducted, and the timestamp value is changed to become the value of the arrival time. Otherwise, the miss value is conducted (arrived packets is not in the zombie list), the count value is set to zero, and the timestamp is set to the time that the packet arrives at the router buffer. The hit value can distinguish between misbehaving flow and behaving flow because misbehaving flows have more hit values than behaving flows. In general, if arrival packets match several zombie records, then many hits are recorded and counts are accordingly increased numerous times. From the definition of misbehaving flows, that is, "flows that have gained more than their fair share in the bandwidth," flows with high count values and several hits have the tendency to misbehave. SRED gauges the frequency of packet hits P(m) = z/p, where z represents the zombie list size and p represents the probability of mismatch between two packets. The number of active flows is calculated

by inversing the value of P(m) (P(m)−1) (Ott et al., 1999). SRED controls DP based on the values of P(m)−1, ql, and D_{max}, as described in the following:

1) If the value of ql is less than c/6, then no congestion is indicated and no packet will be dropped. c denotes capacity.
2) If the value of ql is between c/6 and c/3, then the D_{max} value is set to $D_{max}/4$. Thus, DP is increased in this case.
3) If the value of ql is equal to c/3, then the D_{max} value is set to $D_{max}/4$. Thus, DP is increased.
4) If $1 \geq P(m) \geq 1/2566$, then DP is increased and the selection of 2566 is made stochastically, thereby indicating the need for further investigation (Ott et al., 1999).

The advantage of SRED lies in its reduced dependency on aql by using the zombie list. However, SRED entails a high number of PLs because it is based on keeping the queue small by losing packets, particularly when the number of connections increases (Morris, 2000).

2.5.2.6 GRED Method

GRED is an adaptive method proposed by Floyd (2000) to solve the limitations of RED. GRED relies on a single aql and three threshold values to control congestion. Compared with RED, GRED uses the same congestion metric, i.e., aql; however, GRED adopts an additional threshold value called doublemaxthreshold (Figure 2.11). The dropping rate is based on the location of the aql relative to the three thresholds. The additional threshold allows GRED to stabilize aql at a specific level, thereby making it

better than the RED method. Subsequently, GRED deals with arriving packets based on the following:

1) No packets will be dropped if the aql value at the router buffer is less than the minthreshold.

2) The random DP mechanism of RED is used if the aql value is between the minthreshold and maxthreshold.

3) If the aql value is between the maxthreshold and doublemaxthreshold, then GRED increases the random DP compared with that in the previous case to avoid congestion.

4) If the aql value is above or equal to the doublemaxthreshold, then GRED drops every arriving packet; that is, the value of DP is set to 1.

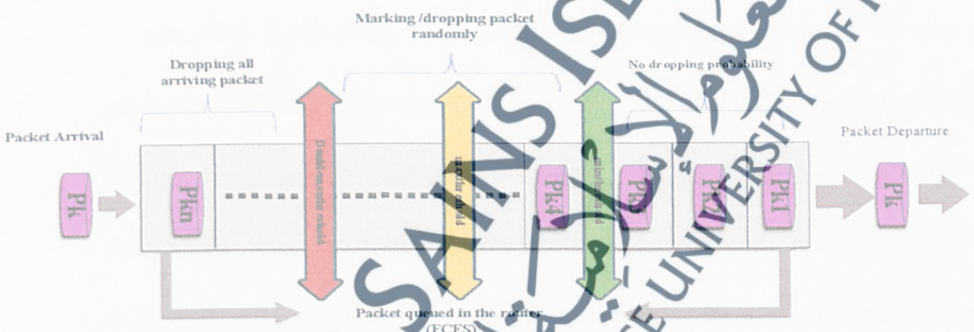


Figure 2.11: Single Router Buffer for GRED.

However, GRED exhibits the following limitations, First, GRED suffers from the parameter setting problem. The value for its parameters must be set to specific value to achieve satisfactory performance. Second, GRED slowly controls and detects congestion when heavy traffic suddenly arrives at a non-congested router buffer; thus, no packet will be dropped but the router buffer will likely overflow (Floyd, 2000; Floyd et al., 2001; Aweya et al., 2001).

2.5.2.7 REM Method

REM is an adaptive method that was proposed to improve the performance aspects, including PL and queuing delay (Lapsley & Low, 1999; Athuraliya et al., 2001). REM has two main properties: transmission rate at the source (R) and ql. These properties must be stabilized to control congestion. REM adopts the price property as the congestion metric. Price is computed based on rate mismatch and queue mismatch, where the former represents the difference between the source transmission rate and the link capacity, whereas the latter represents the difference between ql and the Taql (Lapsley & Low, 1999; Athuraliya et al., 2001).

A transmitted packet will pass through many routers until it reaches its potential destination. The router buffer computes the price value as the packet arrives at any intermediate router, and the total prices for the router buffers are used to calculate DP along the path from the source to the destination. If a congested router exists along the path, then the price values will increase, and consequently, DP will also increase. If congestion occurs, the routers will notify the sources to assuage congestion and the sources will reduce their transmission rates. The REM method stabilizes the rate around the link capacity and the queue around a small target (Zhou et al., 2013). However, REM has limitations, such as parameterization and low throughput when traffic is high.

2.5.2.8 Adaptive Maximum Threshold

The adaptive maximum threshold (Geat et al., 2007) is an adaptive congestion control method that controls congestion by regarding aql as the congestion indicator. This method works in basically the same way as RED

with only one difference, i.e., it uses an adaptive maxthreshold position to keep aql at a specific value between the minimum and maximum thresholds. Such difference implies using additional maxthresholds, called maxthreshold2, and two DPs, similar to that in GRED, to stabilize aql at Taql, as shown in Figure 2.12 (Geat et al., 2007).

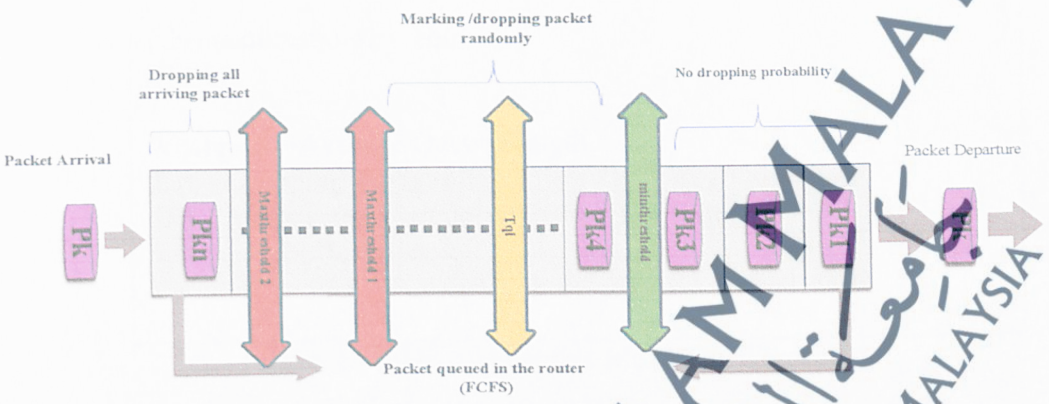


Figure 2.12: Single Router Buffer for Adaptive Maximum Threshold.

The adaptive maximum threshold keeps the aql at a specific value (Taql), prevents the router buffer from building up, and consequently, avoids PL. This threshold uses aql to determine the number of dropping packets (Geat et al., 2007). The DP method is calculated as shown in Figure 2.13.

Calculate aql
IF (0≤aql<minthreshold)
Dp = 0
Else
{ IF (minthreshold≤aql≤maxthreshold1)
DP= (DP_{max}/(maxthreshold - minthreshold))*(aql- minthreshold)}
Else
{IF (maxthreshold1 ≤aql≤ maxthreshold2)
DP=DP_{max}}
Else
{IF(maxthreshold1 < aql≤ k)
DP=1}

Where aql =Average Queue Length
DP = Dropping Probability
DP_{max} = Maximum Dropping Probability Value
k = router buffer capacity

Figure 2.13: DP of Adaptive Maximum Threshold

Congestion is controlled using one of the following conditions:

1) No packet is dropped if aql is less than the minthreshold.
2) If the aql value is between the minthreshold and maxthreshold1, then packets are randomly dropped.
3) If the aql value is increased to a value between maxthreshold1 and maxthreshold2, then DP will increase accordingly.
4) All arriving packets will be dropped if the value of aql exceeds maxthreshold2. Meanwhile, the DP value is set as one.

The adaptive maximum threshold partially stabilizes aql between the minthreshold and maxthreshold. A drawback of this method is the occurrence of parameterization given the existence of multiple thresholds. Moreover, this method will take long a time to adjust in case of high traffic.

2.5.2.9 Effective RED (ERED) Method

The ERED (Abbasov & Korukoglu, 2009) is an adaptive method that was proposed to overcome the limitations of RED. ERED uses instantaneous ql as the congestion indicator in addition to aql. It also adopts several thresholds to control congestion. In effect, the value of the maxthreshold is set to 2 maxthreshold and minthreshold equal to $(\frac{\text{minthreshold} + \text{maxthreshold}}{2} + \text{minthreshold})$. When heavy traffic occurs and the number of queued packets increases. ERED controls congestion as follows:

1) When aql is between the minthreshold and maxthreshold and ql is higher than the minthreshold, ERED drops every arriving packet with DP being equal to one.
2) If aql is less than the minthreshold and ql is less than 1.75* the maxthreshold, then DP is increased.

ERED controls congestion by controlling the packet dropping function using both aql and ql. It partially stabilizes aql between the minthreshold and maxthreshold. The drawback of ERED lies in its parameterization.

2.5.2.10 DGRED Method

DGRED (Baklizi et al., 2013) was proposed to overcome the limitations that prevail in the GRED method. DGRED uses a dynamic updating technique for the maxthreshold and doublemaxthreshold positions to control congestion at an early stage. The difference between DGRED and GRED is as follows. The former uses a target aql (Taql), as shown in Figure 2.14, whereas the latter uses a fixed threshold value. The Taql

should be stabilized between the minthreshold and maxthreshold to achieve better performance and to prevent the router buffer from expanding and overflowing. The DP method is calculated as shown in Figure 2.15.

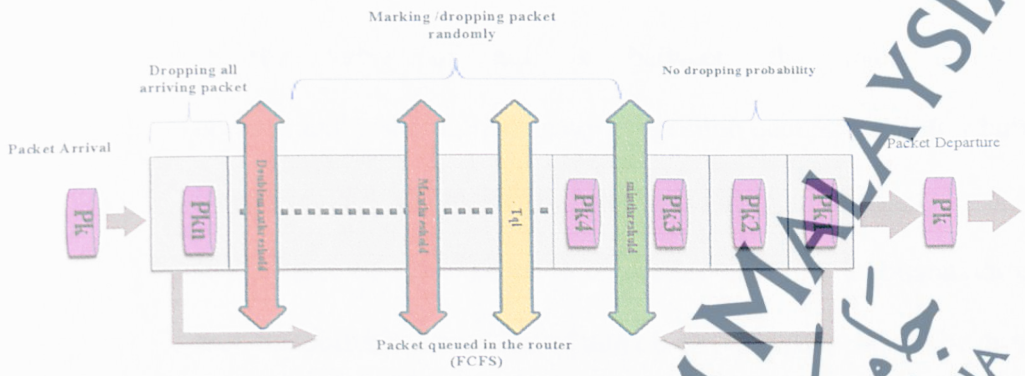


Figure 2.14: Single Router Buffer for the Proposed DGRED

Calculate aql
If (aql<minthreshold)
DP =0
Else
{ if (minthreshold ≤ aql <maxthreshold)

$$D_P = \frac{\frac{D_{max} \times (aql - minthreshold)}{maxthreshold - minthreshold}}{(1 - C \times D_{init})}\}$$

Else
{if (maxthreshold ≤aql<doublemaxthreshold)

$$Dp = D_P = \frac{D_{max} + \frac{(1 - D_{max}) \times (aql - maxthreshold)}{maxthreshold}}{(1 - C \times D_{init})}\}$$

Else
{If(aql≥ doublemaxthreshold)
Dp=1}

Where aql= average queue length
D_{init} =intial dropping probability
C= is the Counter

Figure 2.15: DP of DGRED Method

DGRED controls congestion as follows:

1. The value of aql is computed and compared with the threshold.

2. If the value of aql is less than the minthreshold, then no congestion occurs and no packet will be dropped.
3. If the value of aql is between the minthreshold and maxthreshold, then light congestion is anticipated and packets will be dropped randomly.
4. If the value of aql is between the maxthreshold and doublemaxthreshold, then heavy congestion occurs, the buffer builds up its size rapidly, and indiscriminate packet dropping occurs.
5. If the value of aql is equal to or greater than the doublemaxthreshold, then the buffer has overflowed and all arriving packets will be dropped.

Consequently, the DGRED method controls congestion in the router buffer. The drawbacks of DGRED appear to be its parameterization and its incapability to stabilize congestion around the Taql, particularly when heavy congestion occurs.

2.5.3 Fuzzy-based Early Congestion Control Methods

Many researchers have proposed AQM methods based on FL to reduce performance dependency on parameter values and to enhance network performance (Abualhaj et al., 2016; Khatari & Samara, 2017). FL is based on human decision-making and experiences (Negnevitsky, 2005). FL in AQM is used to decide whether congestion occurs. FL is designed with a good (intuitive) understanding of a system, and the limitations resulting from the complexity of the system parameters can probably be avoided (Murshid et al., 2012). Several FL-based methods have been proposed, such as FL-based BLUE (Feng et al., 1999; Feng et al., 2002) and FL-based RED (Chrysostomou et al. (a), 2003; Chrysostomou et al. (b), 2003) methods.

2.5.3.1 Fuzzy logic (FL)

FL, which is a part of computational intelligence, is a popular research area that deals with processing numerical data (Pedrycz & Vasilakos, 2000; Negnevitsky, 2005; Ingoley & Nashipudi, 2012). FL is defined as the encoding of facts using mathematical expressions with relevant membership values (Loukas et al., 2000; Abdel-Jaber et al. (a), 2008). The striking difference between FL and binary logic (BL) (Zadeh, 1975) is that the truth value for BL is only true or false (0, 1), whereas that for FL is extended to deal with the values for variables between [0, 1], thereby indicating that the truth is partial.

2.5.3.2 Fuzzy Sets

A fuzzy set (Novák, 1995) is a set with fuzzy boundaries (Negnevitsky, 2005). It can also be defined as a collection of elements with associated membership values. In BL, the membership value for each element (truth value) can either be 0 or 1. For example, an element x has 1 membership value if and only if x is an element of set X, which indicates that $x \in X$. Otherwise, $x \notin X$ has a membership value of 0, which indicates that element x does not belong to set X. Nevertheless, the membership value in FL can be partially true, which implies that any element can obtain a value between 0 and 1 for its membership association.

For example, we can produce five sets that indicate temperature, namely, "very low," "low," "medium," "high," and "very high." An item in BL with a value of −5 belongs to the "very low" binary set. The membership value of this set is equal to 1, and its membership values for the other sets are equal to 0. By contrast, an item in FL can simultaneously

belong to multiple fuzzy sets with varying membership values. Figure 2.16 illustrates the five sets in BL, whereas Figure 2.17 displays the five sets in FL in a particular period.

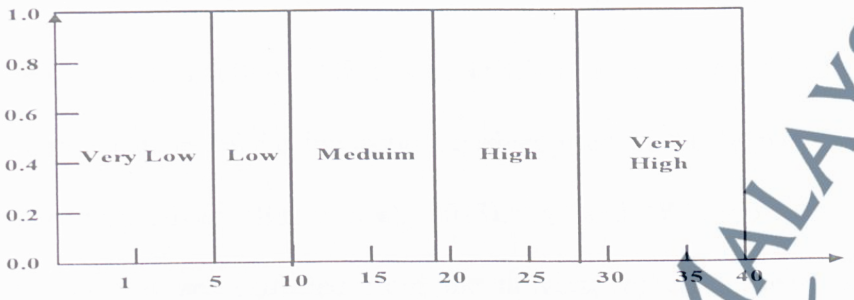


Figure 2.16: Binary Sets.

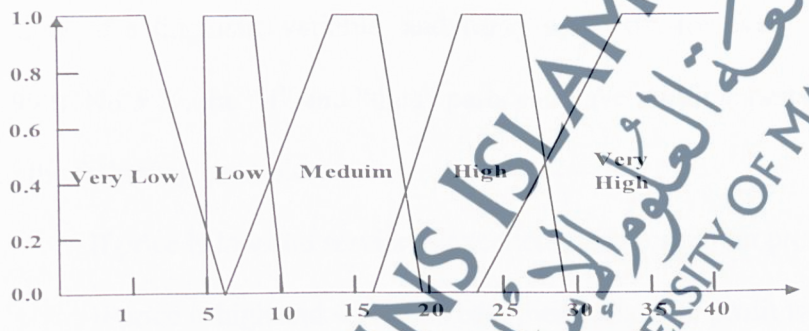


Figure 2.17: Fuzzy Sets.

The function that computes the binary membership values in BL is called the characteristic function (Negnevitsky, 2005) and is defined in Equation 2.1.

$F_B(A) \rightarrow 0,1$ *where*

$$F_B(A) = \begin{cases} 0, if\ a \notin B \\ 1, if\ a \in B \end{cases} \quad \text{..... (2.1)}$$

Meanwhile, the function that calculates the fuzzy membership values is given in Equation 2.2

$F_B(a): A \rightarrow [0,1]$, ... (2.2)

where, $F_B(a)$: 0 if a does not belong to B

$F_B(a)$: 1 if a completely belong to B

$1 > F_B(a) > 0$ if a partly belongs to B

2.5.3.3 Fuzzy Rules

A fuzzy rule is defined as a conditional statement that can be represented as follows: "IF f is F, THEN h is H," where "f" and "h" are representatives of the linguistic variables used in the IF part and THEN part, respectively (Bazaz et al., 2013). "X" and "Y" denote the linguistic values that are extracted from the universe of discourse (Uod). Each linguistic variable has its own Uod, which represents the range of possible values of a linguistic variable, and fuzzy sets exist for every Uod (Lee, 1990). Notably, the "if" and "then" parts can have multiple parts, as shown in the following examples:

If price is low and service is excellent, then restaurant profit is high.

If price is high and service is bad, then restaurant profit is low.

The input and output linguistic variables and the sets represented in Uod are provided by experts with knowledge and experience in the domain, and fuzzy rules are listed based on a theory or through trial and error.

The trial and error method works based on knowledge and experience. After fuzzy linguistic rules are created, an expert will ensure that the rules fit the system by implementing several experiments. Otherwise, the process will be repeated and other rules will be created until suitable rules are obtained. Meanwhile, the theory method works on the basis of tuning the extreme parameters of a system by a domain expert. The combination of the trial and error and theory designing methods may yield better fuzzy linguistic rules than using each method separately.

2.5.3.4 Fuzzy Inference Process (FIP)

FIP can be defined as "the process of mapping from a given input to an output using the theory of fuzzy sets" (Mamdani, 1975; Gottwald, 1993). The Mamdani-style FIP is one of the most commonly used fuzzy inference (FI) techniques. This FIP has four steps (Mamdani, 1975): fuzzification of inputs, evaluation of rules, aggregation of output rules, and defuzzification.

In congestion control methods, FL is used to calculate packet DP based on the Mamdani process using the following steps: 1) the fuzzification of the input crisp values for the input linguistic variables, 2) the evaluation of the rules, 3) the aggregation of the output rules into a single fuzzy set, and 4) defuzzification (Negnevitsky, 2005).

2.5.3.5 Fuzzy Explicit Marking (FEM) Method

The FEM method was proposed to address the drawbacks of the RED method, particularly in cases with heavy congestion and reduced network performance (Hiok & Qiu, 2000; Chrysostomou et al. (a), 2003; Negnevitsky, 2005). FEM locates congestion in basically the same way as RED. However, FEM uses FIP instead of thresholds (i.e., maxthreshold and minthreshold) and sends an explicit congestion notification to the source by adding an ECN bit to its header.

FEM uses two input linguistic variables, namely, the changing rate in the buffer (CRB) and current buffer length (CBL), and one single output linguistic variable DP to control congestion by calculating the amount of DP (Chrysostomou et al. (b), 2003; Negnevitsky, 2005). FEM uses trapezoidal (Klir, 1995) or triangular (Negnevitsky, 2005) for inputs and output linguistic variables to denote the fuzzy set characterized by the

accurate representation feature of expert knowledge and the simple computation of output linguistic variables.

The input and output linguistic variables and fuzzy sets are as follows:

CBL = {empty, low, moderate, full}

CRB = {zero, decreasing, increasing}

DP = {zero, low, medium, high}

FEM adopts two mechanisms to calculate DP: best efforts and assured efforts. These types of efforts are changeable as determined by the types of services. Tables 2.1 and 2.2 represent the rules for best efforts and assured efforts, respectively.

In best efforts, DP is calculated as follows:

1. If the CBL value is moderate and the CRB value is zero, then the DP value is low.
2. If the CBL value is full and the CRB value is decreasing, then the DP value is medium.
3. If the CBL value is full and the CRB value is zero or increasing, then the DP value is high.

In an assured class of service, FEM drops packets as follows:

1. If the CBL value is moderate and the CRB value is increasing, then the DP value is low.
2. If the CBL value is full and the CRB value is zero or decreasing, or if the CBL value is full and the CRB value is increasing, then the DP value is medium.

The drawback of FEM is as follows: membership requires further investigation and membership will not be optimal if investigation is performed based on trial and error.

Table 2.1: Set of FL Rules for the Best Effort Class of Service

Condition	Result
If CBL is empty	DP value is zero
If CBL is low and CRB is decreasing	DP value is zero
If CBL is low and CRB is zero	DP value is zero
If CBL is low and CRB is increasing	DP value is zero
If CBL is moderate and CRB is decreasing	DP value is zero
If CBL is moderate and CRB is zero	DP value is low
If CBL is moderate and CRB is increasing	DP value is medium
If CBL is full and CRB is decreasing	DP value is medium
If CBL is full and CRB is zero	DP value is high
If CBL is full and CRB is increasing	DP value is high

Table 2.2: Set of FL Rules for the Assured Class of Service.

Condition	Result
If CBL is empty	DP value is zero
If CBL is low and CRB is decreasing	DP value is zero
If CBL is low and CRB is zero	DP value is zero
If CBL is low and CRB is increasing	DP value is zero
If CBL is moderate and CRB is decreasing	DP value is zero
If CBL is moderate and CRB is zero	DP value is zero
If CBL is moderate and CRB is increasing	DP value is low
If CBL is full and CRB is decreasing	DP value is low
If CBL is full and CRB is zero	DP value is low
If CBL is full and CRB is increasing	DP value is medium

2.5.3.6 The Fuzzy BLUE (FB)

FB is an adaptive method that was proposed to improve the BLUE method and to provide fair share among all connections. FB that uses FL to

control congestion and compute DP is known as the FB controller (FBC). FB relies on two input linguistic variables, namely, buffer occupancy (BO) and PL, and a single output linguistic variable DP, which uses trapezoidal (Klir, 1995) or triangular (Negnevitsky, 2005) for the inputs and output linguistic variables. The input and output linguistic variables and the fuzzy sets work as follows:

Buffer occupancy (BO) = {low, middle, high}.

Packet Loss (PL) = {small, medium, big}.

Packets dropping/marking probability (DP) = {zero, low, moderate, high}.

The Fuzzy variables, sets, and rules have been created by domain experts. Table 2.3 and Figures 2.18 to 2.20 denote the membership functions and their associated BO, PL, and DP values, respectively. FB is based on the trial and error and theory methods in terms of establishing the fuzzy rules (Yaghmaee & Amintoosi, 2003). After constructing the fuzzy rules, FI is applied to obtain a crisp value (numerical value) for the output linguistic variable DP (Mamdani, 1975; Negnevitsky, 2005).

The drawback of FB lies in the fact that membership requires further investigation and will not be optimal if based only on trial and error.

Table 2.3: The FB Rules

Condition	Result
If BO is low and PL is small	DP value is zero
If BO is low and PL is medium	DP value is zero
If BO is low and PL is big	DP value is zero
If BO is middle and PL is small	DP value is zero
If BO is middle and PL is medium	DP value is zero
If BO is middle and PL is big	DP value is low
If BO is high and PL is small	DP value is zero
If BO is high and PL is medium	DP value is moderate
If BO is high and PL is big	DP value is high

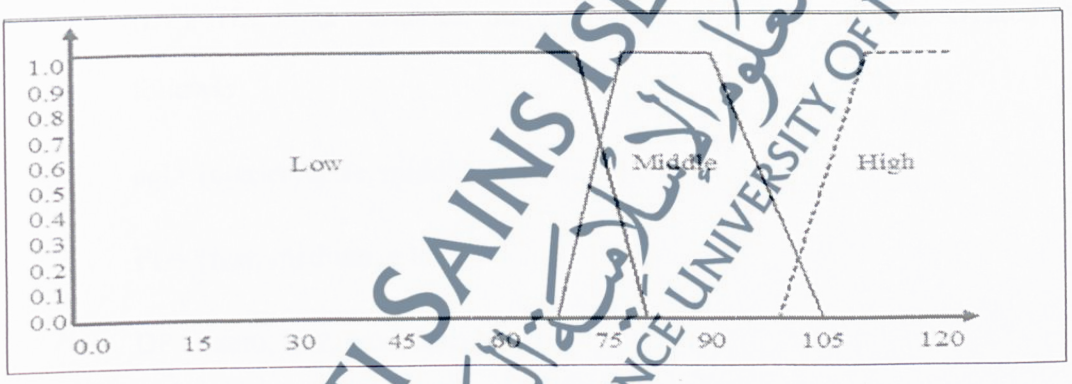


Figure 2.18: Membership Function for BO.

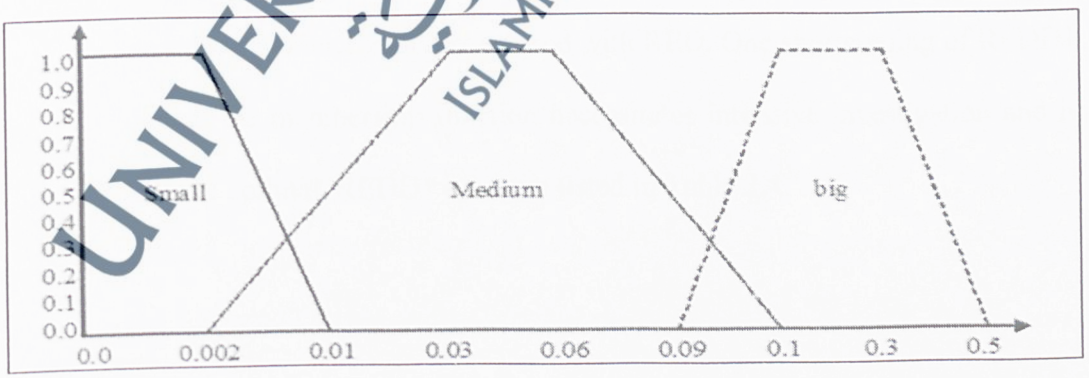


Figure 2.19: Membership Function for PL Linguistic Variable.

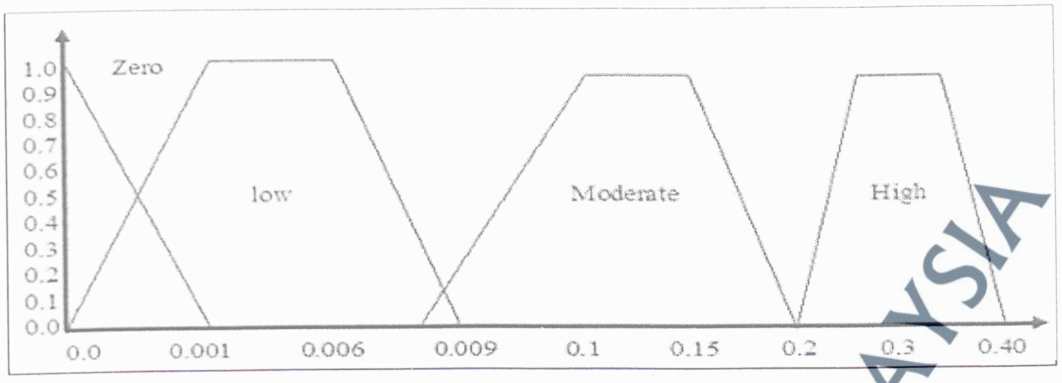


Figure 2.20: Membership Function for DP Linguistic Variable.

2.5.3.7 REDD1 Method

REDD1 was proposed to overcome the limitations of the RED method (Abdel-Jaber et al. (a), 2008). REDD1 works based on two input linguistic variables, namely, aql and PL, and one output linguistic variable (DP). The input variables, output variable and fuzzy sets are created as follows:

aql ={conservative, middle, aggressive}

PL= {few, medium, a lot}

DP= {zero, low, moderate, high}.

REDD1 exhibits a more satisfactory performance measure than the original RED method because it provides fewer PL results. In addition, it reduces parameterization compared with RED. One shortcoming of REDD1 is that the membership function necessitates intensive investigation and is not yet optimal. REDD1 rules are listed in Table 2.4.

Table 2.4: The REDD1 Rules

Condition	Result
If aql is conservative and PL is few	DP value is zero
If aql is conservative and PL is medium	DP value is zero
If aql is conservative and PL is a lot	DP value is zero
If aql is middle and PL is few	DP value is zero
If aql is middle and PL is medium	DP value is zero
If aql is middle and PL is a lot	DP value is low
If aql is aggressive and PL is few	DP value is zero
If aql is aggressive and PL is medium	DP value is moderate
If aql is aggressive and PL is a lot	DP value is high

2.5.3.8 Gentle Random Early Detection Fuzzy Logic Method (GREDFL)

GREDFL (Baklizi et al., 2014) was proposed to address certain limitations of the GRED method. GREDFL is based on two input linguistic values, namely, aql and delay (D), and a single output (DP). GREDFL rules are presented in Table 2.5. The inputs, output, and fuzzy sets are created as follows:

aql ={conservative, middle, aggressive}.

D = {little, average, long}.

DP = {zero, low, moderate, high}.

Compared with GRED, GREDFL reduces the problem of parameter setting dependency. The drawback of GREDFL is that the membership function requires intensive investigation and is not yet optimal.

Table 2.5: GREDFL Rules

Condition	Result
IF aql is conservative and D is little	DP is zero
IF aql is conservative and D is average	DP is zero
IF aql is conservative and D is a long	DP is zero
IF aql is middle and D is little	DP is zero
IF aql is middle and D is average	DP is zero
IF aql is middle and D is a long	DP is low
IF aql is aggressive and D is little	DP is zero
IF aql is aggressive and D is average	DP is moderate
IF aql is aggressive and D is a long	DP is high

2.6 Discrete-time Queues Approach

To compute and study the performance of a network, the proceedings in a system, which materialize either as packet arrival or packet departure must be recorded and analyzed at various time intervals. The events that happen in each interval, which are either packet arrival or departure, are cached, and the associated performance of the systems is analyzed accordingly.

In general, two main approaches can be used to model a queuing network: continuous and discrete models (Woodward, 1993; Alfa, 2010). Network time is split into similar intervals. The main flaw of a continuous model is its failure to monitor system performance that is linked to a particular event. By contrast, discrete-time queue is divided into finite intervals of a number of lengths. Thus, system performance that is linked to a particular event can be monitored (Woodward, 1993; Alfa, 2010; ´Chydziński & CHRÓST, 2011). Moreover, a continuous model can be characterized as a time-driven model, whereas a discrete model can be characterized

as an event-driven discretization. A queue can be modelled based on Kendall's notation of six items, A, B, C, K, P, Q which are listed as follows:

1. Process of arrival (A): A random process that explains how many packets arrive at the queuing system and is denoted by A.
2. Process of service (B): A random process that explains the time distribution required by the packet in the server and is denoted by B.
3. Number of servers (Cs): The number of servers involved in the queuing system and is denoted by Cs.
4. System capacity (K): The maximum total number of packets that can be accommodated in the system, including packets that are currently in service, and is denoted by K.
5. Customer population (P): The limit for the total number of packets that participate in the arrival process and is denoted by P.
6. Queuing service discipline: Rules that determine which packet at the queuing system should be served.

The Cs, K, and P components are positive integers. Subsequently, the fourth and the fifth components should be disregarded if the capacity and the source are infinite. Many descriptors can determine the A and B components, such as deterministic (D), where inter-arrival and service time distributions are constant. If Markovian (M) is used, then inter-arrival and service time distributions are exponential in the continuous-time queues. By contrast, inter-arrival and service times are "geometrically" distributed in discrete-time queues, as denoted by Geo; exponential distribution can only be used if multiple arrivals and departures occurred (Alfa, 2010). Lastly, if the descriptor is "generally" distributed (G), then no restriction is applied to the type of distribution. In discrete-time queues (Alfa, 2010), a basic time unit, called

a slot, is used. Single or multiple events can occur in each slot. Packets may arrive and/or depart from the same slot.

The state of a discrete-time queue with arrivals and departures in each slot is depicted in Figure 2.21 (Alfa, 2010).

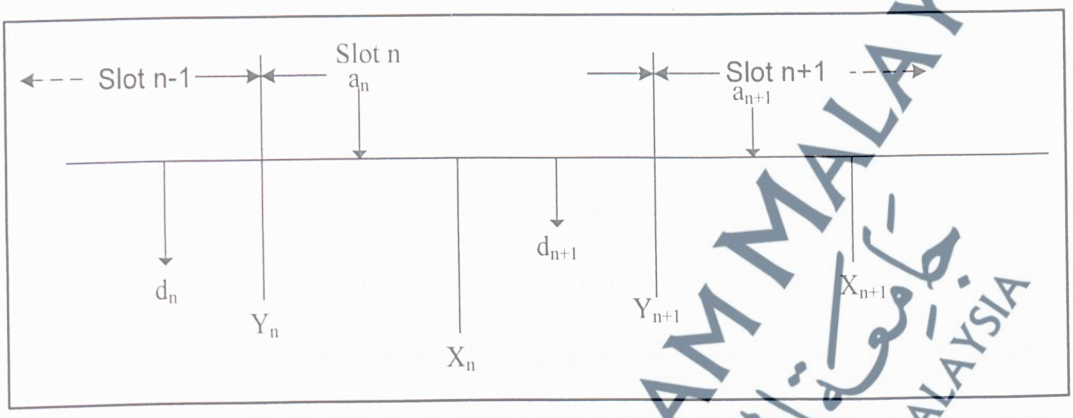


Figure 2.21: State of a Discrete-time Queue.

An event-driven discretization in a system is described as follows:

- Packet arrival occurs after the start of a slot.
- Packet departure occurs before the end of the same slot. A time slot is denoted by n and n = {0, 1, 2, 3…}. A packet that is arriving in and departing from a slot n is defined by a_n and d_{n+1}, respectively.
- The d_0 value is equal to zero because a packet must arrive first before it can depart.
- Y_n represents the number of packets in time slot n.
- a_n denotes the number of arriving packets in time slot n.
- d_{n+1} denotes the number of departing packets in time slot n+1.
- Y_{n+1} represents queue size after slot n has ended, which is calculated using the following equation: $Y_{n+1} = Y_n + a_n - d_{n+1}$. The number of packets in the slot

before packet departure is denoted by X_n, where X_{n+1} is calculated using the following equation: $X_{n+1} = X_n - d_{n+1} + a_{n+1}$.

2.7 Traffic Modeling

The key issues in evaluating AQM algorithm performance are queue modeling and parameter tuning. Queue modeling is used to evaluate or validate performance. Traffic modeling plays an important role in the evaluation process. Renewal traffic processes, such as BP and Poisson processes (PPs), are traditional modeling approaches based on simple mathematical calculation (Lee et al., 1997). These approaches cannot represent aggregated traffic or are unable to capture the properties of aggregation, such as correlation and burstiness (Lee et al., 1997; Ng et al., 1999). In addition, renewal traffic models use discrete-time queuing processes that deal with network traffic as a single class. However, network traffic exists in different classes. Renewal traffic limitation is overcome by using Markov-modulated Arrival (MMA). Three MMA types are used to model packet arrival, namely, Markov-modulated Fluid Flow (MMFF), Markov-modulated Poisson Processes (MMPPs), and Markov-modulated BP (MMBP) (Ng et al., 1999). MMA models represent discrete- and continuous-time scales and packet arrival. Only a single event is allowed for a packet at any time (the packet may arrive or depart) in a continuous approach. By contrast, multiple events can occur for a packet at a single time slot (the packet may arrive and/or depart) in a discrete-time queue (Abdel-Jaber et al. (b), 2007; Al-Diabat et al., 2012). The MMA types based on time scale and packet arrival are summarized in Table 2.6.

MMFF is used to represent the characteristics of a continuous approach in packet arrival and time scale (Ng et al., 1999). By contrast, MMPP (Fischer &

Hellstern, 1993) is widely used in modeling applications that have discrete packet arrival with continuous time (Lim et al., 2011). Computer and communication are discrete in nature, which indicates that MMPP and MMFF do not capture the characteristics that represent digital communications (Lim et al., 2011). Thus, MMBP is a suitable candidate for modeling nature traffic due to its capability to model traffic using a discrete-time approach with bursty and correlated traffic properties (Ng et al., 1999; Guan et al. (a), 2004; Guan et al. (b), 2004; Guan et al., 2006; Lim et al., 2009; Kim et al., 2010; Lim et al., 2011).

MMBP is equivalent to MMPP under discrete time (Ng et al., 1999). Numerous studies have modeled arrival traffic using MMBP. In particular, Guan et al. (2006) and Lim et al. (2009) proposed an approach for controlling delay on a router at a specific level under aggregated Internet traffic flows using MMBP. Similarly, Guan et al. (2004b) evaluated RED under bursty and correlated traffic using MMBP. Lim et al. (2010) evaluated RED and WRED using MMBP-2. Kim et al. (2010) proposed the Active-WRED method and used MMBP to model its arrival process.

In summary, Internet traffic is naturally aggregated, and thus, modeling this type of traffic requires an approach that captures the discretized, bursty, and correlation characteristics of traffic. Accordingly, MMBP is a suitable candidate that can represent all the aforementioned characteristics. By contrast, MMPP and MMFF are unable to capture these characteristics, particularly correlated traffic and bursty traffic.

Table 2.6: MMA Categories

MMA	MMFF	MMPP	MMBP
Time scale	Continuous	Continuous	Discrete
Packet Arrival	Continuous	Discrete	Discrete

2.7.1 Bernoulli Process (BP)

BP is a discrete-time stochastic process because it comprises a finite or infinite sequence of binary random variables. In particular, it is a sequence of independent random variables Xi (Boucherie and Dijk 2010; Neely et al. 2008). BP arrivals can only occur at time slot k. The probability of arrival p in time slot k is independent from other arrivals, such that BP is memoryless. The arrival in slot k is distributed binomially (Woodward, 1993; Abdel-jaber et al. (b), 2007), and the two possible values of each Xi are success with probability p (which implies the probability of arrival in a slot) and failure with probability (1−p) (which indicates the probability of no arrival in a slot). The number of time slots between two arrivals is geometrically distributed (the probability of the next slot at which the arrival occurs) with parameter p (Abdel-jaber et al. (b), 2007). BP stands out as a good model for an initial study of discrete time systems. Its simplicity aids tractability (Robertazzi, 2012).

2.7.2 Two State Markov Modulated Bernoulli Process (MMBP-2)

MMBP-2 is used to model the packet arrival process. It can assess this process in both bursty traffic and correlated traffic. Two states are selected to help smooth the numerical calculation, analysis, presentation, and discussion (Ng et al., 1999). The M-state Markov chain can be broadened using the same techniques (Ng et al., 1999). The proposed source model is illustrated in Figure 2.22.

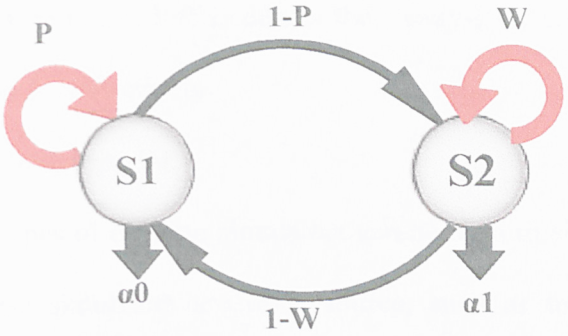


Figure 2.22: MMBP-2

When a process is in state 1 in time slot N, packets arrive at the system with probability α0. The process can then remain in the same state in the next time slot (N+1) with probability P or transit to state 2 with probability 1−P. Similarly, when the process is in state 2 in time slot N, a packet with probability α1 arrives. The arrival process can then remain in state 2 in the next time slot (N+1) with probability W or transit to state 1 with probability 1−W. Thus, the probability that a time slot contains a packet arrival is a BP with parameters (α0 or α1), which vary with regard to a two-state Markov process that is free from the arrival distribution (Guan et al., 2007). Thus, MMBP-2 is characterized by a transition probability matrix (TP) for its process and a diagonal matrix (DM) for its arrival probabilities as follows:

$$TP = \begin{bmatrix} P & 1-P \\ 1-W & q \end{bmatrix} \quad \text{and} \quad DM = \begin{bmatrix} \alpha 0 & 0 \\ 0 & \alpha 1 \end{bmatrix}.$$

Markov theory states that the steady-state probabilities of MMBP-2 in states 1 and 2 [P (S1) and P (S2)] can be obtained from the balance equations of these states. Equations 2.3 and 2.4 are the probabilities of MMBP-2 states 1 and 2 as follows:

$$P(S1) = PP(S1) + (1 - W)P(S2), \quad \text{(2.3)}$$

$$P(S2) = (1 - P)P(S1) + qP(S2), \quad \text{(2.4)}$$

Where P (S1) and P (S2) denote the steady-state probabilities of MMBP-2 in states 1 and 2, respectively.

2.8 Simulation Environment

Many types of existing simulators can be used to simulate AQM methods; some of these simulators are open source, such as the NS2 and OMNet++ (Pongor, 1993) simulators (Issariyakul & Hossain, 2011), or commercial simulators, such as OPNET (Guo et al., 2007) These simulators face many drawbacks, such as large integrated components, licensing cost, complexity, and being generalized, such that if someone intends to use the simulator, several modifications are necessary. Mishra and Jangale (2014) pointed out that researchers should specify the advantages and disadvantages of a computer programming language that uses simulation. Thus, the proposed method GB is simulated in NetBeans Integrated Development Environment (IDE) and Java Development Kit (JDK 1.6) (ORACLE, 2013). We use a Java simulator to avoid some of the drawbacks of existing simulators and for the following reasons. First, simulating a method using Java will produce the same results as those of an NS2 simulator (Seifaddini et al., 2013). Second, Java is faster than C++ and exhibits better performance according to Reinholtz (2000). Third, a Java simulator is easier to use in terms of error handling and debugging. Fourth, a Java simulator is compatible with any operating system and improves simulation quality according to Kilgore et al. (1998). The proposed method is simulated using a single router buffer, which indicates a single node using a Java simulator because of the aforementioned reasons.

2.9 Summary

This chapter presents a survey of congestion control that includes various types of congestion control methods. A large number of AQM methods, such as drop tail, DECbit, RED, REM, SRED, GRED, ARED, BLUE, FB, FEM, REDD1, DGRED, and ERED, have been discussed, thereby shedding light on the congestion measures (i.e., aql, instantaneous ql, and price), the manner in which the DP function is calculated, and the packet dropping policies (e.g., random or periodic) of each method. Table 2.7 presents a comparison among these AQMs in terms of the following: (1) objectives, (2) open or closed-loop (in open loop congestion control methods, control decisions do not rely on any sort of feedback information from congested spots in the network; by contrast, closed-loop control methods make control decisions based on certain feedback information to the sources), (3) control factors that detect congestion, (4) congestion metric design, (5) adaptive or non-adaptive (parameter values are changed dynamically from one situation to another), (6) implicit or explicit feedback, (7) the dropping policy, and (8) advantages and disadvantages. In addition, several queuing models have been studied. In particular, BP and MMBP are used to validate or evaluate the performance of AQM methods.

The main objective of all the congestion control methods is to detect congestion at an early stage before the buffer overflows. To date, the dependence of the BLUE method on its parameters has resulted in a problem called BLUE parameterization. BLUE parameter setting remains an important unsolved issue. The setting values of the δd, δi, and threshold parameters contribute to ascertain performance evaluation. Thus, this thesis suggests several AQM techniques to stabilize the value of ql at a specific level and to reduce the dependency of BLUE on its parameters to a large extent. These techniques are called GB and GBFL.

Table 2.7: Comparison among Various Types of Congestion Control Methods.

Methods	Objectives of the method	Open or closed-loop control	Factors that detect congestion	Congestion metric design	Adaptive or non-adaptive	Implicit or explicit feedback	Dropping policy	Advantages	Disadvantages
Drop Tail	Detect congestion and enhance PL and D by fixing the size of the router buffer to optimize D and PL	Closed-loop	Congestion metric without flow information	ql	Non-adaptive	Implicit feedback	If the queue of the router buffer overflows, then all arriving packets will be dropped.	Easy to implement with no computation overhead	PL, saturation
DECbit	Control congestion by sending back marked packets	Open loop	Congestion metric with flow information	Binary indication bit and aql	Adaptive	Explicit feedback	Marks and forwards every incoming packet when aql is more than one and sends them to their potential destination.	Simple, distributed, and optimized	Unsuitable for bursty traffic
RED	Overcome the limitations of drop tail and detect congestion at an early stage	Closed-loop	Congestion metric without flow information	aql	Non-adaptive	Implicit feedback	Randomly drops incoming packets when aql is between the minthreshold and maxthreshold; also drops all incoming packets if aql is greater than the maxthrshold.	Eliminates global synchronization problems	Unable to keep aql between the minthreshold and maxthreshold, particularly when heavy congestion occurs
ARED	Overcome the limitation in RED by stabilizing the aql at specific level	Closed-loop	Congestion metric without flow information	Modify D_{max} based on aql	Adaptive	Implicit feedback	Increases random dropping of incoming packets by a fixed value when aql is greater the Taql, Dmax is around	Eliminates parameter sensitivity	Unable to keep aql between the minthreshold and maxthreshold, particularly when heavy congestion,

	to control the congestion						approximately 0.5, then Dmax is increased and in effect, the along with DP. Moreover, random dropping of packets decreases when aql is less than the Taql.		parameterization, and PL occur
SRED	Overcome certain limitations of RED, particularly when a large number of connections and misbehaving traffic occur	Closed-loop	Flow information only	Estimate the number of active flows and ql	Adaptive	Implicit feedback	Drops arriving packets based on the number of active queues and current ql	Relies on aql	High PL
GRED	Overcome the limitations of RED by stabilizing aql when aql exceeds the maxthreshold	Closed-loop	Congestion metric without flow information	aql	Adaptive	Implicit feedback	Randomly drops packets (same as RED) if the value of aql is between the minthreshold and maxthreshold. If the value of aql is between the maxthreshold and doublemaxthreshold, then GRED increases random DP. If the value of aql is doublemaxthreshold or higher, then GRED drops every arriving packet.	Relies on aql	GRED deals with several threshold values, parameterization, and overflow.

BLUE	Low loss rates and low queue length oscillation	Closed-loop	Congestion metric without flow information	ql, link idle. and PL	Adaptive	Implicit feedback	Randomly increase the number of dropped packets when ql exceeds the threshold and D_P decreases when current ql is less than the threshold.	Avoids the problems associated with bursty traffic flows by allowing space to accommodate bursty packets in the router buffer and global synchronization	Parameterization and queuing delay increase with increasing congestion.
REM	Enhance certain performance measures, such as PL and D	Open loop	Congestion metric without flow information	Price	Adaptive	Explicit feedback	Increase DP if the weighted total (W_{Total}) is positive, then DP increases. Otherwise, DP decreases.	Stabilize the rate around the link capacity and stabilize the queue around a small target	Parameter sensitivity and low throughput
Adaptive Max Threshold	Stabilized aql between between the minimum and maximum thresholds	Closed-loop	Congestion metric with flow information	aql	Adaptive	Implicit feedback	Randomization using adaptive maxthreshold to stabilize aql at $T_{aq,}$ و if the aql value is between the minthreshold and maxthreshold1. Randomly drop the packet if the value of aql increases to between maxthreshold1 and maxthreshold2, packet dropping is also increased accordingly. All arriving packets will be dropped if the value of aql exceeds	Stabilize aql partially	Parameterization, too many thresholds, aql takes time to arrive at the minthreshold when traffic is high

							maxthreshold2.		
ERED	To overcome the limitations of the RED method	Closed-loop	Congestion metric	aql and ql	Adaptive	Implicit feedback	Controlling DP using aql and ql	Stabilize aql partially	Parameterization, too many thresholds, aql takes time to arrive at the minthreshold when traffic is high
DGRED	To overcome the limitations of GRED and stabilize ql at a specific level	Closed-loop	Congestion metric	aql	Adaptive	Implicit feedback	Randomization when aql is between the minthreshold and maxthreshold and ql is greater than the minthreshold. When aql is less than the minthreshold, ql is less than 1.75* the maxthreshold, and DP is increased.	Stabilize aql partially	Parameterization, too many thresholds, aql takes time to arrive at the minthreshold when traffic is high
FEM	To overcome the drawbacks of the RED method, particularly during heavy congestion	Closed-loop	Congestion metric	FIP	Adaptive	Implicit feedback	Two input linguistic variables, namely, CBL and CRB, are used to calculate a single output linguistic variable (DP).	Does not depend on threshold	Input membership and output membership are inaccurate because they are built based on trial and error.
FB	To enhance the BLUE method and provide fair share among all connections	Closed-loop	Congestion metric	FBC	Adaptive	Implicit feedback	DP is calculated based on BO and PL.	Does not depend on threshold	Input membership and output membership are inaccurate because they are built based on trial and error.
REDD1	To overcome the drawbacks of the RED method,	Closed-loop	Congestion metric	FIP	Adaptive	Implicit feedback	DP is calculated based on aql and PL.	Does not depend on threshold	Input membership and output membership are inaccurate because they are built based on trial and error.

	particularly during heavy congestion								
GREDFL	To overcome the drawbacks of the GRED method, particularly during heavy congestion	Closed-loop	Congestion metric	FIP	Adaptive	Implicit feedback	DP is calculated based on aql and D.	Does not depend on threshold	Input membership and output membership are inaccurate because they are built based on trial and error.