Performance evaluation for DRED discrete-time queueing network analytical model

Hussein Abdel-Jaber\textsuperscript{a,},* Mike Woodward\textsuperscript{a},
Fadi Thabtah\textsuperscript{b}, Amer Abu-Ali\textsuperscript{b}

\textsuperscript{a}Department of Computing, University of Bradford, Bradford BD7 1DP, UK
\textsuperscript{b}MIS Department, Philadelphia University, Amman, Jordan

Received 10 April 2007; received in revised form 14 August 2007; accepted 5 September 2007

Abstract

Due to the rapid development in computer networks, congestion becomes a critical issue. Congestion usually occurs when the connection demands on network resources, i.e. buffer spaces, exceed the available ones. We propose in this paper a new discrete-time queueing network analytical model based on dynamic random early drop (DRED) algorithm to control the congestion in early stages. We apply our analytical model on two-queue nodes queueing network. Furthermore, we compare between the proposed analytical model and three known active queue management (AQM) algorithms, including DRED, random early detection (RED) and adaptive RED, in order to figure out which of them offers better quality of service (QoS). We also experimentally compare the queue nodes of the proposed analytical model and the three AQM methods in terms of different performance measures, including, average queue length, average queueing delay, throughput, packet loss probability, etc., aiming to determine the queue node that offers better performance.

\textsuperscript{c} 2007 Elsevier Ltd. All rights reserved.

Keywords: Adaptive RED; Congestion control; Analytical model; Discrete-time queueing network; DRED; RED
1. Introduction

With the rapid development in computer networks especially in the Internet, the competition of network sources on resources such as buffer space is increasing. This demand increase on the network resources may cause the router buffer to overflow and consequently results in congestion (Braden et al., 1998; Welzl, 2005). It has been shown experimentally that congestion deteriorates the network performance and may cause several drawbacks such as low throughput, high packets queueing delay, high packets loss rate and unfair share of network connections (Braden et al., 1998; Floyd and Jacobson, 1993; Welzl, 2005).

There are several methods developed in the network performance field for controlling congestion (e.g. Athuraliya et al., 2001; Braden et al., 1998; Brandauer et al., 2001; Feng et al., 2002; Floyd and Jacobson, 1993; Floyd et al., 2001; Ott et al., 1999). One of these methods is the drop-tail (DT) (Braden et al., 1998; Brandauer et al., 2001), which has been used for several years to control congestion in the Internet. The DT method depends on setting the routers buffers to maximum sizes and dropping packets when the routers buffers overflow. However, setting the routers buffers to maximum sizes may increase packets queueing delay. On the other hand, if the routers buffers are set to minimum sizes, the throughput will be decreased. DT method has several other drawbacks, including, lockout phenomenon, full queues, bias vs. bursty traffic and global synchronisation (Braden et al., 1998), where all of which contribute in degrading the Internet performance (Feng et al., 2002; Floyd and Jacobson, 1993; Floyd et al., 2001; Ott et al., 1999; Welzl, 2005). Many researchers in computer networks have proposed different active queue management (AQM) techniques (i.e. Athuraliya et al., 2000, 2001; Aweya et al., 2001; Brandauer et al., 2001; Feng et al., 1999, 2002; Floyd, 2000; Floyd and Jacobson, 1993; Floyd et al., 2001; Lapsley and Low, 1999a, b; Ott et al., 1999) in order to overcome some of the above limitations. Specifically, they aim to achieve the following:

1. Controlling the congestion at the routers buffers in the network.
2. Obtaining a satisfied quality of service (QoS) such as high throughput, low packets queueing delay and low packets loss rate.
3. Maintaining the queue length as small as possible.
4. Distributing fair share of the available resources among the network connections.

One of the known AQM algorithms is random early detection (RED) (Floyd and Jacobson, 1993) and its variants, including, adaptive RED (Floyd et al., 2001), gentle RED (Floyd, 2000), random exponential marking (REM) (Athuraliya et al., 2000, 2001; Lapsley and Low, 1999a, b), stabilised random early drop (SRED) (Ott et al., 1999), BLUE (Feng et al., 1999, 2002) and dynamic random early drop (DRED) (Aweya et al., 2001). In this paper, we introduce a new discrete-time queueing network analytical model based on DRED algorithm. The proposed analytical model can be utilised as a congestion control method in fixed and wireless networks. We apply our analytical model and simulation models of DRED, RED and adaptive RED methods on a network that has two-queue nodes (1, 2) in order to evaluate the efficiency of the proposed analytical model. In near future, we intend to apply our model on a network that consists of N queue nodes.

Furthermore, we compare between the proposed analytical model and (DRED, RED, adaptive RED) in order to determine the one which gives better QoS. We want also to
determine which queue node in each model produces better performance. The bases of our comparison between the models are: average queue length \((aqlj)\), throughput \((Tj)\), average queuing delay \((Dj)\), packet loss probability \((P_{lossj})\) and packet dropping probability \((Dpj)\), where \(j\) represents the queue nodes, i.e. \(j = 1,2\).

The paper is outlined as follows: Section 2 introduces DRED algorithm. The new discrete-time queueing network analytical model is presented in Section 3. Section 4 is devoted to a comparison between the queue nodes of the proposed model and (DRED, RED, adaptive RED) with reference to the performance measures mentioned above. Finally, we conclude this paper and suggest future works in Section 5.

2. Dynamic random early drop (DRED) algorithm

DRED (Aweya et al., 2001) was proposed to control the problem of congestion in networks. Unlike RED (Floyd and Jacobson, 1993), which its average queue length \((aql)\) relies on the number of network sources, i.e. TCP connections, DRED stabilises the queue length \((ql)\) at a predetermined level (this level is also called the target level of the queue length \((Tql)\)) independently from the number of TCP connections in the network (Floyd, 2000; Floyd et al., 2001). In other words, when the number of TCP connections in the network is large, the calculated \(aql\) becomes large as well and may exceed the maximum threshold. As a result of that, the RED router buffer drops every arrival packet, which consequently increases the packets loss rate.

DRED depends on fixed time units \((Ct)\) and for each \(Ct\), the current \(ql\) and the error signal \((Err(i))\) are computed in order to obtain \(Dp\). The calculated \(Err(i)\) depends on both the current \(ql\) and \(Tql\) as shown below:

\[
Err(i) = ql(i) - Tql. \tag{1}
\]

Then, based on the \(Err(i)\) value, one can find the filtered error signal \(Fil(i)\) as follows:

\[
Fil(i) = Fil(i-1)(1 - qw) + Err(i)qw. \tag{2}
\]

We can observe from Eq. (2) that DRED uses a low-pass filter in calculating the \(Fil(i)\) similar to RED, where \(qw\) is a control parameter called the queue weight. In Eq. (3), the \(Dp\) value is updated for every \(Ct\) using \(Fil(i)\), and the capacity of the DRED router buffer is denoted \(K\). The control parameter \((\varepsilon)\) is used to control the feedback gain of \(Dp\):

\[
Dp(i) = \text{minimum}\left\{\text{maximum}\left(Dp(i-1) + \varepsilon \frac{Fil(i)}{K}, 0\right), 1\right\}. \tag{3}
\]

As mentioned in Aweya et al. (2001), the \(Dp\) parameter is updated only when the current \(ql\) is equal or greater than the no-drop threshold \((th)\) for the sake of maintaining high-link utilisation. In other words, there will be no packet dropping when \(ql < th\). We can infer that DRED relies on the \(ql\) parameter in order to decide whether or not to drop packets. Hence, the congestion metric for the DRED is the current \(ql\). DRED is similar to RED in terms of packets dropping policy in which it uses the randomisation approach of Aweya et al. (2001). In addition, DRED algorithm marks the arrival packet either by dropping it or by adding an explicit congestion notification (ECN) bit in its header (Floyd, 1994; Ramakrishnan and Floyd, 2001).
3. The proposed discrete-time queueing network analytical model

The classic DRED algorithm relies on a certain level threshold \( (th) \) for determining the congestion in its router buffer. We applied DRED algorithm and our analytical model on a queueing network consisting of two queue nodes \( (j = 1, 2) \) as shown in Fig. 1. Our analytical model depends on a time unit called slot in which this slot could exist in a single or multiple events (Woodward, 1993). An example of a single event is the arrival or the departure of a packet, whereas, the arrival and the departure of packets in the same slot is an example of multiple events.

The queue nodes shown in Fig. 1 have finite packet capacities, where \( K_1 \) and \( K_2 \) correspond to the capacities for queue nodes 1 and 2, respectively, including packets that are currently in service. The arrival process in this model is based on the identical independent distribution (i.i.d.) Poisson \( (a_n \in \{0, 1\}, n = 0, 1, 2, \ldots) \) where \( a_n \) denotes the arrival of a packet in slot \( n \).

Thresholds \( th_1 \) and \( th_2 \) are used in order to determine the congestion in the queueing network and are expressed, respectively, in the following equations:

\[
\begin{align*}
th_1 &= \frac{0.9 K_1}{2}, \\
th_2 &= \frac{0.9 K_2}{2}.
\end{align*}
\]

In Fig. 1, the DRED router buffer of queue node 1 receives packets from sources at \( \lambda_1 \) rate \( (\lambda_1 \text{ is defined below}) \), only when the buffer occupancy is below \( th_1 \), and thus there will be no packet dropping \( (Dp1 = 0) \). Whereas, if the buffer occupancy reaches \( th_1 \) index, the sources of queue node 1 will reduce their sending rate from \( \lambda_1 \) to \( \lambda_3 \) \( (\lambda_3 \text{ is defined below}) \) in order to alleviate the congestion at the router buffer, and therefore the \( Dp1 \) increases from 0 to \( ((\lambda_1 - \lambda_3)/\lambda_1) \). For queue node 2, the DRED router buffer receives packets from its

![Fig. 1. Two queue nodes queueing network system.](image)
sources at $\lambda_2$ rate ($\lambda_2$ is defined below) in cases where the buffer occupancy is less than th2
index, and therefore the dropping probability of queue node 2 ($Dp_2$) becomes zero. Though, if the buffer occupancy is equal or larger than th2 index, $Dp_2$ increases from 0 to $((\lambda_2-\lambda_4)/\lambda_2)$, and the sources decrease their transmitting rates from $\lambda_2$ to $\lambda_4$ ($\lambda_4$ is defined below) in order to control the congestion.

To explain the queueing network system shown in Fig. 1, we analyse each queue node separately in order to come up with the traffic equations for each one of them. We assume that $\lambda$ is the arrival rate probability of the external packets (packets arriving from outside the network) in a slot. $\lambda_1$ and $\lambda_3$ are the probabilities of packets that arrive at node 1 before reaching the th1 index and after reaching the th1 index, respectively. $\lambda_2$ and $\lambda_4$ are the probabilities of packets that arrive to queue node 2 when the buffer occupancy is smaller than th2 index and when it is greater than or equal to th2 index, respectively. Further, $\beta_1$ and $\beta_2$ represent the probabilities of packets that arrive in a slot from nodes 1 and 2, respectively. We also assume that the queueing network is equilibrium, and the buffer occupancy process of each queue node is Markov chain with finite state spaces. The state spaces for queue node 1 are $\{0, 1, 2, 3, \ldots, th1 - 1, th1, th1 + 1, \ldots, K_1 - 1, K_1\}$ and for queue node 2 are $\{0, 1, 2, 3, \ldots, th2 - 1, th2, th2 + 1, \ldots, K_2 - 1, K_2\}$. Finally, we consider that $\lambda_1 > \lambda_3$, $\lambda_2 > \lambda_4$, $\beta_1 > \lambda_1$ and $\beta_2 > \lambda_2$ thus, $\beta_1 > \lambda_3$ and $\beta_2 > \lambda_4$.

Since we analyse each queue node in the queueing network system separately, therefore, the arrival rate for each queue node requires an independent evaluation. The below equations represent the arrival rate of packets for queue nodes 1 and 2, respectively,

$$\lambda_1 = \lambda r_{01} + \lambda_1 r_{11} + \lambda 2r_{21},$$

$$\lambda_2 = \lambda r_{02} + \lambda_1 r_{12} + \lambda 2r_{22}.$$  \hspace{1cm} (6)  \hspace{1cm} (7)

After solving the above equations recursively, we obtain the final form for $\lambda_1$ and $\lambda_2$ as follows:

$$\lambda_1 = \frac{\lambda r_{01}(1 - r_{22}) + \lambda r_{02} r_{21}}{(1 - r_{11})(1 - r_{22}) - r_{12} r_{21}},$$

$$\lambda_2 = \frac{\lambda r_{02}(1 - r_{22}) + \lambda r_{01} r_{12} + \lambda r_{02} r_{12} r_{21}}{(1 - r_{11})(1 - r_{22})^2 - r_{12} r_{21}(1 - r_{22})}.$$  \hspace{1cm} (8)  \hspace{1cm} (9)

In Eqs. (7) and (8), $r_{ij}, i,j = 1, 2$ represents the routing probabilities of packets between the two queue nodes in the queueing network, and $(r_{i0}, r_{j0})$ represent the probabilities of packets leaving the queueing network from queue nodes 1 and 2, respectively. Lastly, $(r_{0i}, r_{0j})$ represent the routing probabilities of packets that arrive from outside the network to queue nodes 1 and 2, respectively. Moreover, we assume that the probability of external packets arriving to node 1 is 60%, whereas 40% for node 2. And thus, queue node 1 has higher priority than queue node 2 in serving packets, which arrive from outside the network. The state transition diagrams of queue nodes 1 and 2 are shown in Figs. 2 and 3, respectively.

When using the state transition diagrams in Figs. 2 and 3, one can derive the balance equations for queue nodes 1 and 2 as shown in the following equations:

$$\Pi_0 = (1 - \lambda_1)\Pi_0 + \beta_j(1 - \lambda_j)\Pi_1,$$

$$\Pi_1 = \lambda_j\Pi_0 + \left[\lambda_j\beta_j + (1 - \lambda_j)(1 - \beta_j)\right]\Pi_1 + \beta_j(1 - \lambda_j)\Pi_2.$$  \hspace{1cm} (10)  \hspace{1cm} (11)
In general we get
\[ P_i = \frac{1}{2} \left( \frac{1 - \lambda_i}{1/C_0} \right) P_i / C_0 + \frac{1}{2} \left( 1 \right) P_i / C_0 + 1, \quad (12) \]
where \( i = 2, 3, 4, \ldots, thj - 1. \)

\[ P_{thj-1} = [\lambda_j (1 - \beta_j)] P_{thj-2} + [\lambda_j \beta_j + (1 - \lambda_j)(1 - \beta_j)] P_{thj-1} + [\beta_j (1 - \lambda_{j+2})] P_{thj}, \quad (13) \]

\[ P_{thj} = [\lambda_j (1 - \beta_j)] P_{thj-1} + [\lambda_{j+2} \beta_j + (1 - \lambda_{j+2})(1 - \beta_j)] P_{thj} + [\beta_j (1 - \lambda_{j+2})] P_{thj+1}. \quad (14) \]

In general we get
\[ P_i = [\lambda_{j+2}(1 - \beta_j)] P_{i-1} + [\lambda_{j+2} \beta_j + (1 - \lambda_{j+2})(1 - \beta_j)] P_i + [\beta_j (1 - \lambda_{j+2})] P_{i+1}, \quad (15) \]
where \( i = thj + 1, thj + 2, thj + 3, \ldots, K_j - 1. \)
Finally,
\[ \Pi_{Kj} = [\lambda_{j+2}(1-\beta_j)]\Pi_{Kj-1} + [\lambda_{j+2}\beta_j + (1-\beta_j)]\Pi_{Kj}, \]
where \( Kj = thj + I \).

Let
\[ \gamma_m = \frac{\lambda_m(1 - \beta_m)}{\beta_m(1 - \lambda_m)}, \quad \text{where} \quad m = 1, 2, 3, 4. \] (17)

After obtaining the balance equations of the queue nodes, we apply Eq. (17) in Eqs. (10)–(16) in order to get the network queueing probabilities shown below:

\[ \Pi_j = \frac{\lambda_j(1 - \beta_j)^{j-1}}{\beta_j(1 - \lambda_j)} \Pi_0 \]
where \( j = 1, 2 \) and \( i = 1, 2, 3, \ldots, thj - 1 \),

\[ \Pi_{thj+i} = \frac{\lambda_j \gamma_{j+1}^{thj+i-1}}{\beta_j(1 - \lambda_j)(1 - \lambda_j)} \Pi_0 \]
where \( j = 1, 2 \) and \( I = 0, 1, 2, 3, \ldots, I \).

We also evaluate the probabilities that queue nodes 1 and 2 are idle using Eq. (19) as follows:

\[ \sum_{i=0}^{Kj} \Pi_i = 1, \quad \text{where} \quad j = 1, 2. \] (20)

Then by applying Eqs. (18) and (19) in Eq. (20), we get

\[ \Pi_0 = \left[ \frac{1 - \gamma_{thj} - \beta_j(1 - \gamma_j)}{(1 - \beta_j)(1 - \gamma_j)} + \frac{\gamma_{thj}(1 - \lambda_j)(1 - \gamma_{j+2})}{(1 - \beta_j)(1 - \lambda_j)(1 - \gamma_{j+2})} \right]^{-1}. \] (21)

After computing \( \Pi_0 \), we estimate the performance measures (\( aqlj, T_j, D_j, P_{lossj} \)) for the proposed analytical model. We start by calculating the \( aql \) for the queue nodes using the generating function \( P(z) \), where \( P(z) \) is represented as

\[ P(z) = \sum_{i=0}^{Kj} z^i \Pi_i, \quad \text{where} \quad j = 1, 2. \] (22)

Then by obtaining the first derivative of \( P(z) \) at \( z = 1 \), we can compute the \( aqlj \) for the queue nodes as

\[ aqlj = P^{(1)}(1) = \frac{\Pi_0}{(1 - \beta_j)} \left[ \frac{\gamma_j - \gamma_{thj} \gamma_j + thj(1 - \gamma_j)}{(1 - \gamma_j)^2} + \frac{\gamma_{thj}(1 - \lambda_j)}{1 - \lambda_{j+2}} \right] \]
\[ \left( \frac{thj(1 - \gamma_{j+2})(1 - \gamma_{j+1}^2) + \gamma_{j+2} - \gamma_{j+2}^2[1 + I(1 - \gamma_{j+2})]}{(1 - \gamma_{j+2})^2} \right). \] (23)

It should be noted that \( aql \) result can also be computed using the formula \( \sum_{i=0}^{Kj} i \Pi_i \), where \( j = 1, 2 \).
After \( aqlj \) is calculated, the throughput fraction of the queue nodes that represents the number of packets, which have passed through the queuing network system successfully \( (T_j) \), is evaluated. \( T_j \) is also known as the fraction of time that the router is busy (Woodward, 1993). \( T_j \) can be calculated using the following equation:

\[
T_j = \beta_j \sum_{i=1}^{K_j} \Pi_i = \beta_j(1 - \Pi_0) \text{ packets/slot, where } j = 1, 2. \tag{24}
\]

Using \( aqlj \) and \( T_j \), which are given in Eqs. (23) and (24), respectively, we can obtain the average queuing delay \( (D_j) \) for queue nodes 1 and 2 by utilising the littles law as shown below:

\[
D_j = \frac{aqlj}{T_j} \text{ slots} = \frac{P^{(1)}(1)}{T_j} \text{ slots} = \frac{\sum_{j=0}^{K_j} i \Pi_i}{T_j} \text{ slots}. \tag{25}
\]

After \( D_j \) is computed, we can easily obtain the packet loss probability for queue nodes 1 and 2 \( (P_{lossj}, \text{ where } j = 1, 2) \), where \( P_{lossj} \) is the proportion of packets that have lost the service at the first and the second routers buffers from all the packets that have arrived at the routers buffers. As explained earlier, the DRED router buffer for each queue node starts dropping packets when the buffer occupancy reaches \( th1 \) and \( th2 \) indexes, respectively. \( P_{lossj} \) for queue nodes 1 and 2 is given as

\[
P_{lossj} = \sum_{i=th}^{K_j} \Pi_i, \text{ where } j = 1, 2. \tag{26}
\]

The joint equilibrium probability of the queuing network system is expressed as

\[
\Pi(K_1, K_2) = \Pi_1(K_1)\Pi_2(K_2), \tag{27}
\]

where \( K_1 \) packets at node 1 and \( K_2 \) packets at node 2.

4. Comparison results

In this section, we conduct an experimental comparison between the proposed discrete-time queuing network analytical model and different AQM methods, including RED, DRED and adaptive RED. Further, we evaluate the performance of each queue node in the proposed model and the AQM methods we consider, in order to obtain the traffic loads QoS for each queue node in all methods. The comparison results are obtained using different performance measures \( (aqlj, T_j, D_j, P_{lossj}) \). We also computed the \( D_{pj} \) in order to decide which queue node in each model drops less number of packets.

The parameters used in the experiments of our analytical model are given as follows: The probability of packets arriving from outside the network \( (\lambda) \) is set to values between \([0.66 \text{ and } 0.70]\), queue node 1 has higher priority than the queue node 2 with respect to serving packets that routed from outside the network. The probability of packets routed from outside the network to queue node 1 \( (r_{01}) \) is set to 0.60, and the probability of packets routed to queue node 2 \( (r_{02}) \) is set to 0.40. Queue node 1 parameters \( (\lambda_3, \beta_1, K_1) \) are set to 0.75, 0.90 and 20 packets, respectively. Whereas, queue node 2 parameters \( (\lambda_4, \beta_2, K_2) \) are set to 0.75, 0.90 and 20 packets, respectively. The routing probabilities of queue node 1 \( (r_{10}, r_{11}, r_{12}) \) are set to 0.40, 0.30 and 0.30, respectively, and the routing probabilities \( (r_{20}, r_{21}, r_{22}) \) for queue node 2 are set to 0.40, 0.30 and 0.30, respectively. The parameters
of DRED, RED and adaptive RED for queue node 1 are given in Tables 1–3, respectively. Whereas, queue node 2 parameters for the three AQM methods, are provided in Tables 4–6, respectively.

The parameters of RED and adaptive RED shown in their corresponding tables above are: \(D_{\text{max}}\) = the maximum packets dropping probability value, \(qw\) = queue weight,

Table 1
DRED parameters for queue node 1

<table>
<thead>
<tr>
<th>(\lambda)</th>
<th>(\lambda_3)</th>
<th>(b_1)</th>
<th>(K_1)</th>
<th>(C_t)</th>
<th>(qw)</th>
<th>(\varepsilon)</th>
<th>(D_{p1})</th>
<th>Number of slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.66</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.665</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.67</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.675</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.68</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.685</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.69</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.695</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>1</td>
<td>0.002</td>
<td>0.0000005</td>
<td>0.05</td>
<td>10 000 000</td>
</tr>
</tbody>
</table>

Table 2
RED parameters for queue node 1

<table>
<thead>
<tr>
<th>(\lambda)</th>
<th>(\lambda_3)</th>
<th>(b_1)</th>
<th>(K_1)</th>
<th>(D_{\text{max}})</th>
<th>(qw)</th>
<th>(\text{min threshold} 1)</th>
<th>(\text{max threshold} 1)</th>
<th>Number of slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.66</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.665</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.67</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.675</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.68</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.685</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.69</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.695</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>12</td>
<td>10 000 000</td>
</tr>
</tbody>
</table>

Table 3
Adaptive RED parameters for queue node 1

<table>
<thead>
<tr>
<th>(\lambda)</th>
<th>(\lambda_3)</th>
<th>(b_1)</th>
<th>(K_1)</th>
<th>(D_{\text{max}})</th>
<th>(qw)</th>
<th>(\text{min threshold} 1)</th>
<th>Interval of time</th>
<th>(d)</th>
<th>Number of slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.66</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.665</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.67</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.675</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.68</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.685</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.69</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.695</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.75</td>
<td>0.9</td>
<td>20</td>
<td>0.1</td>
<td>0.002</td>
<td>3</td>
<td>0.5</td>
<td>0.9</td>
<td>10 000 000</td>
</tr>
</tbody>
</table>
min threshold1 = the minimum position at queue node 1, max threshold1 = the maximum position at queue node 1, min threshold2 = the minimum position at queue node 2, max threshold2 = the maximum position at queue node 2, $d = a$ decreasing parameter used in the adaptive RED, interval time = the interval time when $D_{\text{max}}$ is updated (Floyd et al.,...
It should be noted that the proposed queueing network analytical model, DRED, RED and adaptive RED were built in Java and the experiments have been conducted on Pentium Centrino Duo 1.6 Processor with 1 GB RAM.

Tables 7 and 8 show our analytical model performance measures results for queue nodes 1 and 2, respectively. Whereas, Tables 9 and 10 display DRED performance measures.
results of queue nodes 1 and 2, respectively. Further, RED performance measures results for queue nodes 1 and 2 are given in Tables 11 and 12, respectively, and Tables 13 and 14 display the adaptive RED performance measures results for queue nodes 1 and 2, respectively.
The numbers shown in Tables 7–14 indicate that when $\lambda$ increases, the probability of packets arriving to queue nodes 1 and 2 increases as well ($aql_1$, $T_1$, $D_1$, $P_{loss_1}$) as shown in Figs. 4–11. Specifically, Figs. 4–7 show the results of queue node 1 according to $\lambda$ against $aql_1$, $\lambda$ against $T_1$, $\lambda$ against $D_1$ and $\lambda$ against $P_{loss_1}$,
respectively. In proportion to queue node 2 results, Figs. 8–11 demonstrate them. In particular, Figs. 8 and 9 depict the results for $\lambda$ against $aql_2$ and $T_2$, respectively, and the results of $\lambda$ against both $D_2$ and $P_{loss_2}$, are provided in Figs. 10 and 11, respectively.
After analysing Figs. 4 and 8, we found out that queue nodes 1 and 2 in the DRED, RED and adaptive RED algorithms stabilise their average queue sizes better than queue nodes 1 and 2 in the proposed analytical model since they drop more packets in heavy
traffic loads than our model. Furthermore, aqlj results for the two queue nodes in the DRED, RED and adaptive RED algorithms shown in Tables (1–6), are similar with a little difference in which DRED achieved slightly lower aqlj results than both RED and adaptive RED, respectively. Queue node 2 in the proposed analytical model produces better aqlj results than queue node 1 since our analytical model maintains traffic loads regardless the traffic load status, i.e. light or heavy. In other words, queue node 2 stabilises the average queue size better than queue node 1 in our analytical model. Whereas, DRED, RED and adaptive RED produce smaller aqlj results for queue node 1 than queue node 2 due to the fact that queue node 1 drops larger number of packets.

According to Tj results, Figs. 5 and 9 indicate that our analytical model produces higher Tj results if compared with that of DRED, RED and adaptive RED techniques. Moreover, each queue node in the proposed analytical model outperformed its corresponding queue node in DRED. In particular, we observe in Fig. 9 that the introduced model and both RED and adaptive RED give almost the same T2 results on certain traffic loads levels (<0.7935). In addition, Fig. 5 exhibits that DRED offers better T1 results than RED and adaptive RED since it serves more packets and that explains the higher T1 results for DRED comparing to those of RED and adaptive RED. Nevertheless, both RED and adaptive RED produce larger T2 results than DRED when the traffic load is normal, but when the traffic load increases, both RED and adaptive RED drop many packets, and therefore they deteriorate their T2 performance. Hence, DRED produces better T2 performance than RED and adaptive RED when the traffic load is heavy.

Furthermore, RED and adaptive RED have very similar results with respect to T1 results since they drop large number of packets when the traffic load becomes heavy as demonstrated in Fig. 5. On the other hand and according to T2 results, RED and adaptive RED have similar performance especially when the traffic load is normal. Though, when the traffic load becomes aggressive, the adaptive RED generates slightly smaller T2 results than RED as noted in Fig. 9. Moreover, queue node 1 in DRED has better performance than queue node 2 with regards to Tj results since it serves larger number of packets that come from outside the network. Whereas, the Tj results for both RED and adaptive RED in Figs. 5 and 9 illustrate that queue node 2 achieves higher Tj results than queue node 1 since queue node 1 drops larger number of packets.

Fig. 11. $\lambda$ vs. aqlj2.
We observe from Figs. 6 and 10 that DRED produces better $D_j$ results than those of our model, RED and adaptive RED since it has the smallest packets average waiting time in queue nodes 1 and 2, respectively. Fig. 6 depicts that RED, adaptive RED and our analytical model offer very similar $D_1$ results when the traffic load is normal ($\lambda$ in [0.66–0.68]). On the other hand, our model generates better $D_1$ results than both RED and adaptive RED when $\lambda$ increases (0.675 $\leq \lambda$ for RED and 0.685 $\leq \lambda$ for adaptive RED). Additionally, we note in Fig. 6 that the adaptive RED is better than RED with respect to $D_1$ performance metric especially when the traffic load is high. In accordance to $D_2$ results, both RED and adaptive RED outperform our model when the traffic load status is not heavy. Further, queue nodes 1 and 2 have similar $D_j$ results in the DRED algorithm with slightly smaller results for $D_1$ (see Tables 9 and 10 for more details). Whereas, queue node 2 gives smaller $D_j$ results than queue node 1 in our analytical model, RED and adaptive RED since queue node 1 drops more packets.

The reason that the simulation models of DRED, RED and adaptive RED outperform our analytical model with reference to $D_2$ performance measure can be explained as follows: The difference between the analytical model and the simulations of DRED, RED and adaptive RED for both $aql_2$ (Fig. 8) and $D_2$ (Fig. 10) happens because our model utilises the instantaneous queue length associated with $th_2$ in the computation, whereas the simulation models use the average queue length ($aql_l$) instead. Now, the instantaneous queue length has a much larger variance than $aql_l$ and it is well known that low variance reduces delays (Woodward, 1993). Furthermore, since $aql_1$ and $aql_2$ given in Figs. 4 and 8 are linked to $D_1$ and $D_2$ that appeared in Figs. 6 and 10, this is why our model achieved higher results with respect to $aql_1$ and $aql_2$ than DRED.

Figs. 7 and 11 show that the queue nodes of the proposed model drop smaller number of packets than that of DRED, RED and adaptive RED. To be more specific, Fig. 11 demonstrates that queue node 2 in the proposed analytical model drops fewer packets than queue node 2 of the DRED, RED and adaptive RED algorithms regardless whether the traffic load is light or heavy. Fig. 7 also indicates that queue node 1 in our analytical model outperforms queue node 1 in DRED when the traffic load is less than 0.695. Furthermore, DRED loses packets at its queue nodes less than both RED and adaptive RED. Lastly, Figs. 7 and 11 illustrate that RED and adaptive RED have similar number of lost packets in their two queue nodes. It should noted that queue node 2 in all compared methods loses fewer packets than that queue node 1 due to the fact that queue

![Image](image_url)
node 1 has a higher priority than queue node 2 in serving packets from outside the queueing network.

Finally, the $D_p,j$ results of our model, DRED, RED and adaptive RED are given in Figs. 12–19. Specifically, Figs. 12 and 13 display our model $D_p,j$ results for queue nodes 1 and 2, respectively, and, Figs. 14 and 15 illustrate DRED $D_p,j$ results for queue nodes 1 and 2.
1 and 2, respectively. Moreover, Figs. 16 and 17 show RED $D_{p1}$ results for queue nodes 1 and 2, respectively, and Figs. 18 and 19 demonstrate the adaptive RED $D_{p1}$ results for queue nodes 1 and 2, respectively.

In $D_{p1}$ experiments, the parameters used for the DRED, RED and adaptive RED algorithms and our analytical model are set to the same values discussed at the beginning.
of this section with a single exception that is $\lambda$ which has been set to 0.66. Figs. 13, 15, 17 and 19 clarify in terms of $D_p^2$ results for queue node 2 that both RED and adaptive RED drop almost the same number packets, and this number of dropped packets is the smallest if compared to DRED and the proposed analytical model especially when the traffic load status is not heavy. Furthermore, the proposed model is better than DRED with regards to $D_p^2$ results. Particularly, when the traffic load becomes heavy, our model drops the lowest number packets ($D_p^1$) since it is not affected with the traffic load status. It is obvious from Figs. 12, 14, 16 and 18 that RED, adaptive RED and DRED $D_p^1$ results have been affected by the increase of the traffic. In other words, RED, DRED and adaptive RED increase their $D_p^1$ due to the existence of congestion in their queue nodes 1. Additionally, these figures indicate that DRED drops fewer packets than both RED and adaptive RED if the traffic is heavy. Finally, in queue node 1, both RED and adaptive RED drop huge amount of packets when their traffic loads are high and both methods derive similar $D_p^1$ results.

5. Conclusions

In this paper, we propose a new discrete-time queueing network analytical model, which consists of two queue nodes, where queue node 1 has higher priority than queue node 2 regardless of the traffic load. We compare the proposed analytical model with DRED, RED and adaptive RED algorithms using different performance measures in order to decide the one that provides better QoS. Specifically, we compute the average queue length, throughput, average queueing delay, packet loss rate and packet dropping probability for the queue nodes in the compared models. The experimental results pointed out that DRED, RED and adaptive RED outperform our analytical model with respect to average queue length in the two queue nodes since their average queue length is reduced when the traffic load is increasing. According to the average queueing delay results in queue nodes 1 and 2, the results showed that DRED outperforms RED, adaptive RED and our model. For queue node 1, the proposed analytical model provides an average queueing delay results better than both RED and adaptive RED when the traffic load is heavy. However, if the traffic load is not heavy, RED, adaptive RED and the proposed analytical model produce very similar average queueing delay results in queue node 1. According the throughput and packet loss rate results, our analytical model outperforms
DRED, RED and adaptive RED in queue node 1. Furthermore, queue node 1 in the proposed analytical model outperforms the corresponding queue node 1 in DRED according to the packet loss rate at certain traffic load levels, i.e. [0.66–0.69], whereas queue 1 of DRED derives better packet loss rate when the traffic load is larger than 0.69. According to the performance of each queue node in compared models, queue node 2 has better performance than queue node 1 with respect to packet loss rate. In addition, all the compared methods except our proposed model offer better average queue length results in queue node 1 than queue node 2 since they decrease their average queue length results when the traffic load becomes heavy. On the other hand, due to the fact that our model is not influenced by the traffic load status, its average queue length results in queue node 2 are better than those in queue node 1. We intend in near future to extend our queueing network model to handle $N$ queue nodes based on the priority policy using discrete-time queues.

References


