Cluster Comput
DOI 10.1007/s10586-015-0446-8

Congestion-aware TCP-friendly system for multimedia transmission based on UDP

Mohammed M. Kadhum[1,2] · **Hossam S. Hassanein**[1]

Received: 15 August 2013 / Revised: 31 January 2015 / Accepted: 11 March 2015

Abstract Although most of the data traffic in Internet is HTTP-based, multimedia applications will soon dominate a large percentage of the traffic. These applications require satisfactory level of bandwidth as they normally have large datasets. Under the limitation of network bandwidth, multimedia traffic may become a source for congestion; particularly when UDP is used due to the absence of flow control or congestion control mechanisms in this protocol. Avoiding congestion can be done if arrival rate to gateways is maintained close to outgoing link capacity while maintain small queue lengths at the routers. This guarantees the availability of buffer capacity for successful buffering and consequent forwarding in case of temporary traffic upsurges which could otherwise cause buffer overflows and packet loss. This research work introduces a congestion-aware and friendly UDP-based multimedia system which can help to lessening congestion occurrence and improve network performance, principally in real-time environments. The evaluation results show that the proposed system helps in controlling congestion, reducing packet loss, increasing throughput, and providing improved network utilization.

Keywords Congestion control · Multimedia · UDP · DCCP

✉ Mohammed M. Kadhum
kadhum@cs.queensu.ca; kadhum@nav6.usm.my

Hossam S. Hassanein
hossam@cs.queensu.ca

[1] Telecommunications Research Lab (TRL), School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada

[2] National Advanced IPv6 Center (NAv6), Universiti Sains Malaysia (USM), 11800 Gelugor, Pulau Pinang, Malaysia

1 Introduction

As the number of multimedia applications requiring high quality of service has rapidly increased, the need for optimizing the network has become critical. The diversity of multimedia applications on the Internet has led to different types of multimedia traffic [1,2]. Applications such as video conferencing, IP telephony, IPTV, and audio and video streaming are bursty in nature, making multimedia communication suffers from bandwidth requirement problems [3]. The changes in bandwidth requirements, for these applications, affect the network performance in terms of efficient utilization. Therefore, it is important to optimize the network bandwidth utilization; which would allow increasing the number of users using the multimedia applications with guaranteed quality of service.

In computer networking, the transport layer is responsible for providing end-to-end communication services for applications within a layered architecture of network components and protocols. The transport layer provides convenient services such as connection-oriented data stream support, reliability, flow control, and multiplexing. Transport layers are contained in both the transmission control protocol/Internet protocol (TCP/IP) model [4], which is the foundation of the Internet, and the open systems interconnection (OSI) [5] model of general networking.

1.1 Transport layer protocols for multimedia

The three well-known protocols in this group are the TCP, user datagram protocol (UDP), and datagram congestion control protocol (DCCP).

UDP is a popular transport protocol for the delivery of multimedia traffic over the Internet [6]. UDP is unfriendly and cannot share the bandwidth fairly when it coexists with

Published online: 24 March 2015

TCP under limited bandwidth because it does not have any congestion control mechanism. UDP usually utilizes the entire bandwidth in the same limited bandwidth link and TCP will be out of bandwidth. As most of Internet traffic is TCP flows, this may lead to a collapse of the entire network.

TCP can deliver best-effort service for error-intolerant and delay-tolerant data such as web, email, and file transport. These features of TCP make it suitable for the delivery of important, mission-critical, and error-free data which requires a reliable data connection [7]. TCP is not well suited for streaming media due to its reliable in-order delivery and congestion control that can cause random long delays.

DCCP is designed for the delivery of multimedia data over the network [8]. With the capability of having a congestion control mechanism, DCCP has the high potential to coexist fairly with TCP. The bandwidth can be shared fairly between other congestion-controlled transport protocols like TCP.

1.2 Multimedia transmission issues

Multimedia streaming is becoming a popular application in the Internet today. There are so many applications for multimedia streaming such as video conferencing, video-on-demand, voice over IP (VoIP) etc. Most of these applications are using UDP as their transport protocol due to the fact that UDP is a lighter protocol without any congestion control mechanism and is easier to implement. The increase of these applications resulted in the increase of multimedia data traffic which affected conventional data traffic that utilized TCP as their transport protocol, such as world wide web and email. The worst affected may be the mission-critical data such as online banking.

Transport layer protocols designed for delivering multimedia data over the Internet efficiently, such as DCCP, are working well for networks with a short delay link and minimum error rate. However, the performance of DCCP is degraded when delivering data over long delay links [9,10].

When it comes to a network link with long propagation delay, unlike UDP, the performance of DCCP delivering multimedia data is dropped significantly. This is due to the high round trip time (RTT) introduced in such link, for example, the links for satellite and wireless networks. DCCP over such links suffers from several problems such as higher packet drops, longer time to reach the optimum throughput, higher jitter and fluctuated throughput. This drawback of using DCCP as a transport protocol over long delay links drives researchers to improve the multimedia applications at the application layer over a simple transport protocol such as UDP. This in turn helps to improve the Internet's performance in terms of congestion avoidance and control.

Nowadays, many multimedia applications favor the UDP. The use of UDP as a transport protocol would endanger the network because there is no congestion control applied. To a certain extent, the entire network can collapse if too many applications deliberately use this protocol. Using conventional UDP as a transport protocol for carrying multimedia streaming data degrades the performance of other transport layer protocols in use.

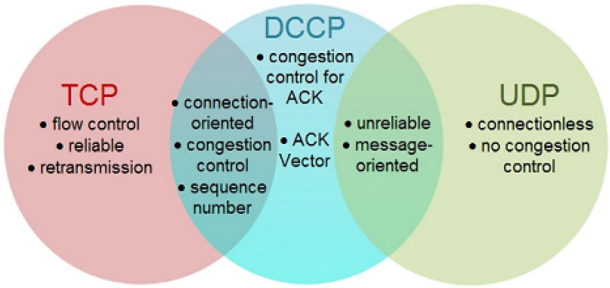


Fig. 1 DCCP protocol

To solve this problem, DCCP was invented with built-in congestion control schemes. For multimedia traffic, DCCP has several congestion control mechanisms, such as TCP-like, TFRC, and TFRC-SP. TCP-like is used for abrupt changes in data [7] while TFRC is used for more smooth (less abrupt) data [11] and TFRC-SP is used for small packets data [9]. DCCP can handle the congestion caused by the transmission of multimedia traffic even though it suffers from slow start. The problem is anticipated to occur with the delivery of multimedia data using DCCP over a long delay link network [11], since DCCP is designed to work best for normal or short delay links.

DCCP is well suited as a transport protocol for delivering multimedia traffic over wired or wireless networks [8]. It supports bidirectional unicast connections of congestion-controlled unreliable datagram. It uses acknowledgment mechanisms and explicit congestion notification (ECN) [12, 13] to figure out packet loss and congestion. It is also good for network health due to its built-in congestion control features.

With DCCP as a transport protocol, the multimedia traffic data carried by DCCP is supposed to share the bandwidth fairly with TCP data and co-existing in harmony. However, several studies [14–17] showed that DCCP has a weakness in competing the bandwidth with existing TCP connections. In addition, DCCP based VoIP presents poor performance in the presence of TCP traffics.

DCCP can be described as an intermediate transport protocol which is an unreliable transmission protocol, as is UDP, and has congestion control and connection-oriented features like TCP. Figure 1 loosely illustrates the DCCP.

DCCP provides a framework for congestion-controlled but unreliable data transmission. Within this framework, different congestion control identifiers (CCIDs) implement different TCP-friendly congestion control profiles. CCID-2, a TCP-like congestion control [16] specifies a profile similar to TCP's additive increase/multiplicative decrease (AIMD) congestion control mechanism. CCID-3, a TFRC mechanism

[17] provides rate-controlled congestion control, based on TCP-friendly rate control (TFRC) that is better suited for the transmission of data multimedia. Three types of media applications use DCCP: one-way pre-recorded media, one-way live media and two-way interactive media. The relevant difference between one-way and two-way media is delay sensitivity. Delays of tens of seconds or more from transmit time at the sender to play out time at the receiver are acceptable. In contrast, for two-way applications, transmission delays as little as 150–200 ms are often causing problems [5].

1.3 Motivation

One of the target applications of DCCP is Internet telephony (VoIP), which is an increasingly popular application. Interactive speech codec acts like constant-bit-rate sources, sending a fixed number of frames per second. Users are extremely sensitive to delay and quality fluctuation, so retransmissions are often useless. Retransmission mechanisms, as in other transport protocols like TCP, generally require more time and introduce higher delays. At this point, the receiver has to pass the playback point before the transmitted packet arrives. Quick adaptation to the available bandwidth is neither necessary nor desirable. The data rate is changed by adjusting the size of each compressed audio frame, either by adjusting codec parameters or by switching codec altogether. At the extreme, some speech codec can compress 20 ms of audio down to 64-bits of payload.

However, the packet rate is harder to adjust because buffering multiple frames per packet can cause an audible delay. Such small payloads pressurize the transport layer to reduce its header overhead, which becomes a significant contributor to connection bandwidth. A codec may also save bandwidth by not sending data during silent periods (when no one is talking), but immediately returns to its full rate as soon as speech resumes. Many of these issues are common to interactive video conferencing as well, although that involves much higher bandwidth.

Streaming media introduces a different set of trade-offs. Unlike interactive media, several seconds of buffer can be used to mask some rate variations, but since users prefer temporary video artifacts to frequent re-buffering, even streaming media generally prefers timeliness to absolute reliability. Video encoding standards often lead to application datagrams of widely varying sizes; for example, MPEG's key frames are many times larger than its incremental frames. An encoder may thus generate packets at a fixed rate, but with orders-of-magnitude size variation. Interactive games use unreliable transport to communicate position information. Since they can quickly make use of available bandwidth, games may prefer DCCP CCID-2 congestion response to the slower response desired by multimedia. One of the problems that may arise is that slow-start assumes that unacknowledged segments are due to network congestion. While this is an acceptable assumption, segments may be lost for other reasons, such as poor data link layer transmission quality. Thus, slow-start executes badly in situations with poor reception, such as wireless networks. Furthermore, according to Schier and Welzl [15], " the lack of transparency in the API with regard to packet loss, the coarse granularity of the lookup table used to calculate the TFRC equation, and the lack of history discounting in CCID-3—and demonstrates that they can significantly impair the performance of typical DCCP use cases such as live video streaming." It was found that CCID-3 and its variants result in lower voice qualities than those over UDP [18].

The goal of the research work presented in this paper is to develop congestion-aware and responsive UDP-based system for multimedia transmission that can reduce the occurrence of congestion situations and optimize the utilization of network resources that would meet the requirements of multimedia users and improve the network performance. The proposed system allows the end users to sense and respond to network congestion by adjusting transmission strategy and encoding pairs that are associated with different transmission rate values.

The remainder of the paper is organized as follows; Sect. 2 presents the proposed UDP-based system. The implementation of a UDP-based system is described in Sect. 3. The performance evaluation results are discussed in Sect. 4. Finally, conclusions and future work are presented in Sect. 5.

2 Congestion-aware UDP-based system

To take advantage of the features provided by UDP, applications that use UDP should have (i) congestion control at higher layers and (ii) a modified version of UDP that can keep the desirable features of UDP. Applications that use the enhanced version should understand its behavior. As the UDP agent is used to allocate and send network packets, information needed for communication at the application level should be passed to the UDP agent as a data stream. Nevertheless, packets that have a header stack only will be allocated by UDP implementation. Thus, the UDP implementation should be modified to add a new technique to send the incoming data from the application. As this application is going to be used with different gateway queue management mechanisms, this type of multimedia stream should be differentiated from other streams. Hence, the UDP agent should be modified to the right data type in a field of IP header.

2.1 UDP-based system requirements

The proposed congestion-aware UDP-based system would help in avoiding congestion, sharing the bandwidth with

other protocols and utilizing the network efficiently. Thus, the performance of the network will be improved overall. To develop such a system, it requires inclusion of UDP-based multimedia connections which exchange data traffic through the bottleneck link(s). These connections should be identified separately. Per-flow state should be maintained to guarantee good quality of service at the hosts ends that exchange the multimedia traffic. The visibility of an unencrypted IP and UDP packet header is required. Hosts should have knowledge about the IP or UDP options. They are capable of inspecting packets with tunnel encapsulations and suspend processing of packets with unknown formats. Nodes should have the ability to de-multiplex connections based on their IP address/protocol/port number, or an explicit identifier of a specific connection. Hosts should have the ability to differentiate between multimedia traffic and other traffic types.

This paper presents the development of a congestion-aware UDP-based multimedia application that runs over a modified version of UDP implementation in the network simulator 2 (ns-2). The multimedia application implements different media transmission rates to cope with different network conditions; and should respond to network congestion by changing its transmission rate.

2.2 UDP-based system design

In this proposed UDP-based system, the sender and the receiver should agree on different sets of transmission strategies. During the connection establishment phase, the sender and the receiver must also agree on sets of encoding strategies and associate them with different values of scales (transmission rates). Taking in mind that VoIP prefers constant packet rate to variable one, it requires application's cooperation to change packet size in order to change effective data rate [16]. Therefore, for each of the encoding and transmission strategy sets, constant transmission rate is determined, and every set uses same sized packets in spite of the encoding method.

The system is supposed to perform as follows: the multimedia source (sender) starts its transmission rate that is mapped to scale 0 (the first transmission rate). When data packets are first received at the receiver end, it should notify the sender of the transmission rate that the receiver is expecting to receive the next data packet. As a result, the sender has to change its transmission rates according to the rate value announced by the receiver. The receiver monitors the network and determines the transmission rate value based on the congestion condition indicated by the queue management mechanism used at the bottleneck router. In addition to this, packet loss and marked packet monitoring are used to detect congestion in the network. Every packet loss for each RTT is considered network congestion and action should be taken to lessen this congestion. When the loss or receiving of a marked packet happens, the receiver decreases the value of the transmission rate used recently to half. The receiver then informs the sender of the new value of the transmission rate it will receive or will expect to receive next. When no congestion is detected, the receiver increments the transmission rate value by one and informs the sender about the new value

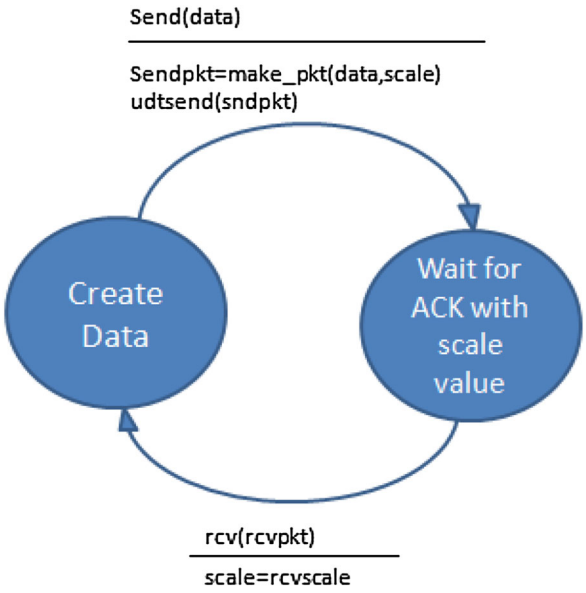


Fig. 2 FSM description of multimedia sender

Figure 2 shows the finite state machine (FSM) for the multimedia sender. It starts its transmission rate that is mapped to first transmission rate (scale 0) over unreliable data transfer channel. Then the sender has to wait for ACK (that includes the transmission rate that the receiver is expected to receive next) and change its transmission rates according to the rate value announced by the receiver.

Figure 3 shows the FSM for the multimedia receiver. When data packets are received at the receiver side, it should notify the sender about the transmission rate that the receiver is expected to receive next. The receiver monitors the network and determines the transmission rate value based on the congestion condition. When the receiver receives packet and no congestion is detected, increments the transmission rate value by one and informs the sender about the new value. When the loss is detected, the receiver decreases the value of the recent transmission rate to half. It then informs the sender of the new transmission rate value that is expected to receive next.

3 The implementation of UDP-based system

In this implementation, the UDP agent is used to allocate and send network packets. As mentioned earlier, information needed for communication at the application level should be passed to the UDP agent as a data stream and packets that

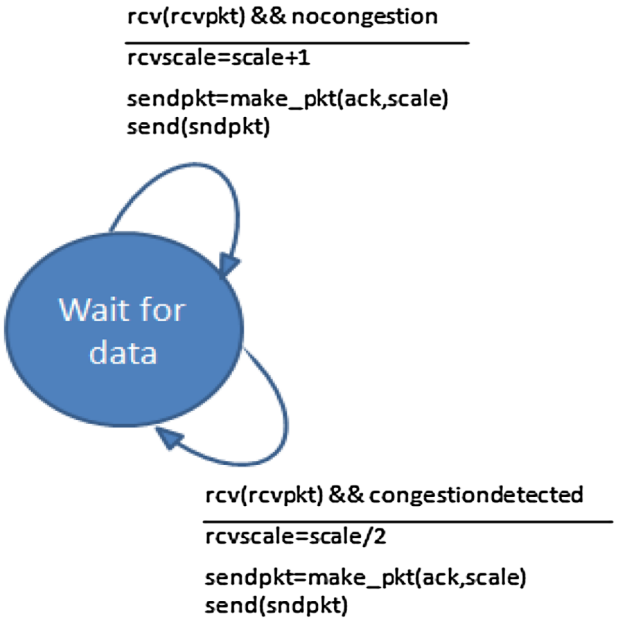


Fig. 3 FSM description of multimedia receiver

have a header stack only will be allocated by UDP implementations. Hence, the UDP implementation should be modified, to send the incoming data from the application. For the multimedia application, CBR implementation that is used in the ns-2 simulator is modified to have the ten level media scaling feature (ten transmission rates for testing experiment). The C++ class will be called "MApp" and implemented as a child class of "Application". The OTcl will be called "Application/MApp". The sender and receiver for this application are implemented in "MApp". The UDP agent used for MApp is called "UdpMAge". The OTcl for this agent is called "Agent/UDP/UDPm". For the application level communication, MApp header is defined in addition to MApp Sender and Receiver. Set of values for new parameter will be added to ns2 default file.

3.1 Class and header structure for the multimedia application

For the communication of multimedia applications, the header of the structure named in C++ is defined under "hdr_m". Once the multimedia application has information to send, it hands it to "UdpMAgent" in the structure format of "hdr_m". Then, "UdpMAgent" allocates packets (based on the data packet size) and writes the data to the multimedia header of each packet. By defining the "MultimediaHeaderClass" that should be derived from "PacketHeaderCalss" in the header, the header size and the OTcl name for the header (PacketHeader/Multimedia) are obtained. Note that the multimedia sender uses a timer for scheduling the next transmission of the data packet.

The "SendTimer" class that is derived from the "TimerHandler" class is defined in addition to its "expire()" member function. This function should call "send_m_pkt()" member function of the "MApp" object. At the beginning, "MApp" calculates the next data transmission time based on the transmission rate associated with the current scale value and the packet size that is given in the TCL simulation script. The "MApp" sender adjusts its scale parameter (next transmission rate) when an ACK application packet arrives from the receiver side.

3.2 The modified UDP for multimedia transmission

The modified UDP protocol for multimedia transmission, which is called UdpMAgent, is modified based on the original UDP protocol in ns-2 (known as "UdpAgent"). It is modified to allow for the following technical objectives:

- Writing the information received from a MApp object to the header of sending multimedia packet.
- Reading the information from the header of the received multimedia packet.
- Handing the header of the received multimedia packet to the "MApp" object.
- Segmentation and re-assembly.
- Prioritizing multimedia (MApp) packets (if need).

4 Evaluation of UDP-based system

To ensure that the congestion-aware UDP-based system is working properly and efficiently, a TCL model (presented in the Appendix) is applied for testing purposes and several scenarios are conducted to investigate the performance of a network utilizing the new system. The numerical results gained from simulations are analyzed statistically to evaluate the performance of the system.

4.1 Simulation scenarios

The experimental evaluation includes two scenarios. The first scenario is conducted for evaluating the congestion-aware UDP-based system, while the second scenario is performed using standard UDP for comparison purpose. The simulation scenario and architecture for studying the effectiveness of the proposed UDP-based system is illustrated in Fig. 4, which corresponds to the parameter settings included in the TCL model and presented in Table 1. This scenario includes two routers, two sources (one for the multimedia connection using the modified UDP and another for file transfer connection using TCP NewReno) and their corresponding receivers (Fig. 4). The traffic arriving on the incoming links at senders' gateway (R1) is aggregated and multiplexed on

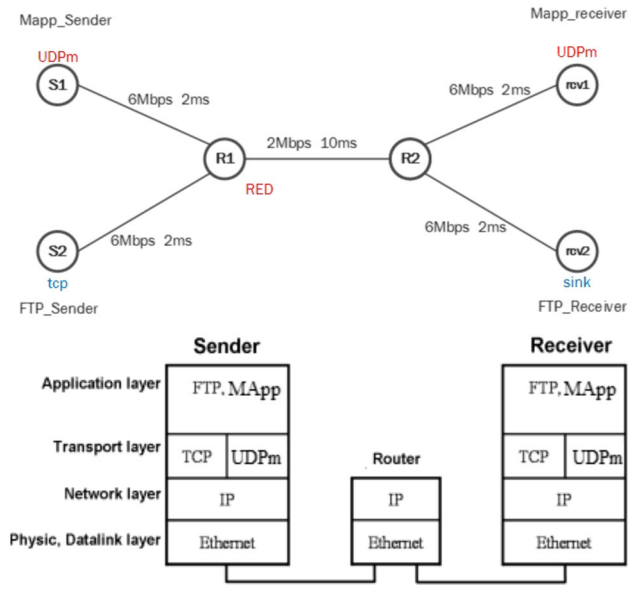


Fig. 4 Sample network scenario for evaluating the proposed system

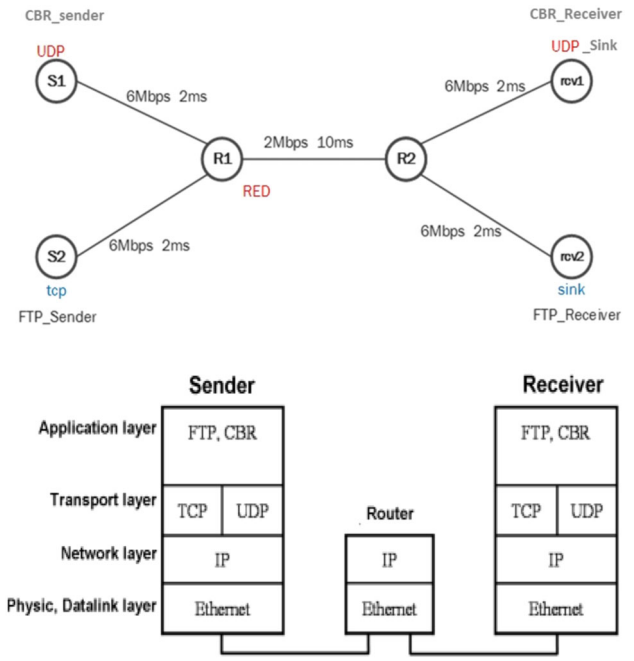


Fig. 5 Sample network scenario when the standard UDP is used

the outgoing link. The access links provide full-duplex connections at a data rate of 6 Mbps with a propagation delay of 2 ms between the sources and their gateway (R1). The bottleneck link between router 1 and router 2 has a data rate of 2 Mbps and a propagation delay of 10 ms. The capacity of access links has been chosen three times larger than the capacity of the bottleneck link to ensure that the link between router 1 and router 2 is the only bottle-neck in the network. The propagation delay of the downlink has been chosen five times larger than that of the access links to signify the large propagation delay across a network as opposed to the small delay experienced in accessing the network.

FTP application traffic generation module is connected to the TCP NewReno agent to generate data, while the developed multimedia application agent is connected to the modified UDP to generate the multimedia traffic. When the scenarios were run, the behavior of the proposed UDP-based system is recorded and examined by using performance metrics such as outgoing transmission link utilization, packet loss, queue size and throughput.

For comparison purpose, Fig. 5 demonstrates the second simulation scenario and architecture of utilizing standard UDP. Same parameter settings used to evaluate the proposed system are used in demonstrating the performance of the network when the standard UDP is used. In this scenario, CBR application traffic generation module is connected to standard UDP agent to generate data that represents multimedia traffic. UDP agent and attach it to the node S1, then attach a CBR traffic generator to the UDP agent. The packet size is being set to 500 bytes and a packet will be sent every 0.005 s (i.e. 200 packets per second).

The experimental parameters are summarized in Table 1.

Table 1 Experimental parameters

Parameters value	Parameters value
TCP version	NewReno
Application traffic type	FTP, CBR, MApp
FTP/MApp packet size	1000 bytes
CBR packet size	500 bytes
CBR sending interval	0.005 s
Router buffer size	20 packets
Bottleneck queue management mechanism	Random early detection (Min_thr= 5, Max_thr=10)
Access links bandwidth	6 Mbps
Bottleneck link bandwidth	2 Mbps
Simulation duration	50 s

4.2 Estimation of traffic generation

As the average transferred file over the Internet is about 10 Kbytes, the average file has no more than 10 TCP packets taking the typical packet length to be 1000 bytes. A typical distribution that describes the file length is the Pareto, with shape parameter of between 1 and 2 (and average of 10 Kbytes). Since the median of the file size is about 2.5 Kbytes [19], a Pareto distribution with mean 10 Kbytes and an average size of 2.5 Kbytes identifies a Pareto distribution with shape parameter b = 1.16 and with a minimum size of 1.37 Kbytes. The interarrival times of new connections are distributed exponentially. TCP connections are parameterized by the source host parameter and the session number

parameter from that host. A new FTP application is defined for each TCP source agent whenever TCP source has data to send. The arrival process of the new TCP flows is modeled according to a Poisson process. Therefore, the beginning of new TCP connection is generated using exponentially distributed random variables. In both scenarios, the average time between arrivals of new TCP sessions at the source is 45 ms. That is, on the average, 22.22 sessions arrive at the host. Therefore, the packet arrival rate of sessions is 22.22 sessions per second. Sessions are generated with random size with an average of 10 Kbytes, with Pareto distribution with shape 1.5. Therefore, the total bits generation rate is:

$$
\begin{aligned}
&22.2 \text{ sessions per second} \times 10 \text{ Kbytes per session} \\
&\quad \times 8 \text{ bitsper byte} \\
&= 22.2 \text{ sessions per second} \times 10 \times 1024 \text{ bytes per session} \\
&\quad \times 8 \text{ bitsper byte} \\
&= 1818624 \text{ bits per second} = 1.8 \text{ Mbps}
\end{aligned}
$$

Adding 1.8 Mbps generated by TCP source to 1.6 Mbps generated by UDP source equals 3.4 Mbps. Thus, it is clear that the rate of bits generation is higher than the bottleneck capacity (which is 2 Mbps). Hence, the congestion phenomenon is expected to appear.

4.3 Results and discussion

This subsection presents the performance evaluation of the proposed system based on the numerical results obtained from simulations. It discusses how multimedia traffic generated by the proposed UDP-based system is handled by the network and end hosts and also how more optimally the network resources are used. A comparison of the performance of the developed multimedia application that uses a modified UDP to that of the standard multimedia traffic that utilizes CBR protocol over original UDP protocols is presented here. This comparison is based on actual and average queue lengths, packet losses, and link utilization. Throughput and drop fairness property are also compared in terms of the number of dropped packets in a connection, considering the share in the bottleneck link bandwidth.

4.3.1 Actual and average queue size

This section describes the performance of the modified UDP-based system in terms of the actual and the average queue sizes. This comparison is done to show the queue occupancy level at the router's buffer and this is important as the impact of the multimedia traffic on the buffer utilization can be understood easily. As known, the key role of the queue management mechanism employed in the network routers is to keep the queue length as small as short as possible and to provide space for accommodating a sudden increase in multimedia traffic in order to avoid packet loss. Figure 6 illustrates the actual and average queue lengths of the bottleneck router. Here the multimedia traffic is generated using the multimedia application of the modified UDP-based system. Note that the traffic shown in the figure includes the accommodated FTP of the router's buffer.

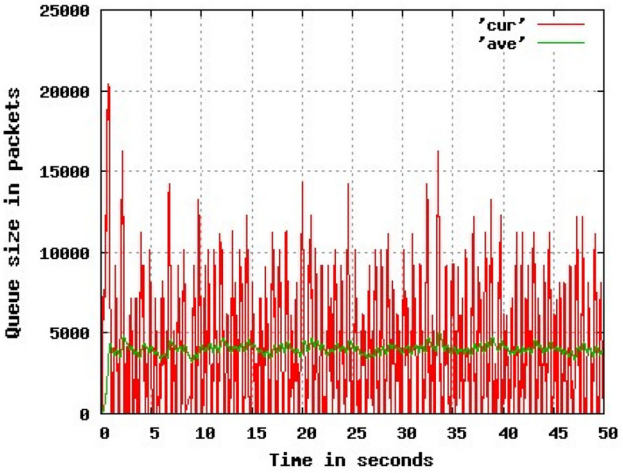


Fig. 6 Actual and average queue sizes in using modified UDP-based system

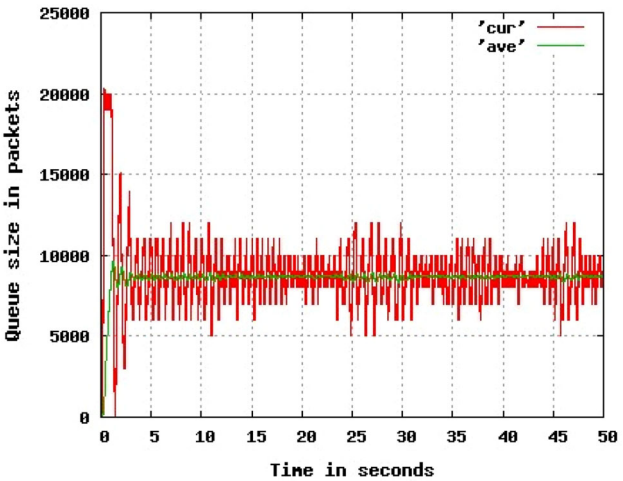


Fig. 7 Actual and average queue sizes using CBR and standard UDP

The actual and average queue sizes in the case of using the CBR application over the standard UDP to generate the multimedia traffic is shown in Fig. 7.

Figure 6 illustrates the queue management mechanism used in the router which allows the queue size to grow (indicated by RED) to a maximum queue size of 20 packets (20,000 bytes). A packet is then dropped in both TCP and modified UDP connections. This drives the sender to react to the loss by adjusting their transmission rates, and hence avoiding and controlling congestion at the gateway. In

the case of standard UDP (Fig. 7), it is clear that, initially, TCP increases its congestion window exponentially. This is because it is in the slow start phase, till a point when one of its packets gets dropped, causing the congestion window to become half of its original value. When transmission using the standard UDP begins, it occupies some bandwidth, preventing the TCP connection from increasing its congestion window any further. As a result of this, many more packets are dropped.

From Fig. 6, the average arrival rate at the gateway router is manageable due the fact that both resources (TCP and the modified UDP) are responsive. This can be realized from the variations observed in the window size. The average queue length is much lower compared to the value obtained in the previous case. It is approximately 4 packets, which is much lower as compared to 9 packets in the standard UDP case. Thus, the average delay of the connections is also smaller (shorter). It is,

$$D_q = (4 * 1000 * 8)/(2 * 10^3 * 10^3) = 16\,\text{ms}$$

In the case of standard UDP, shown in Fig. 7, there is a much faster but smaller variation in window size (due to the fact that only TCP is responsive). As seen from the figure, the high oscillations in the queue sizes correspond to those of the windows of TCP; and the average queue size is around 9 packets. This means that there is an additional average queuing delay which is calculated as,

$$D_q = (9 * 1000 * 8)/(2 * 10^3 * 10^3) = 36\,\text{ms}$$

4.3.2 Packet loss

Packet loss results in noticeable performance issues significantly affecting many network applications such as VoIP, online gaming, and video conferencing. The performance of these applications is degraded when packet loss is high. Figure 8 shows the total number of packets dropped at the gateway (350 packets), when the modified UDP-based system is used.

Figure 9 below shows the total number of packets dropped at the gateway when the standard UDP is used. The loss in this case is 6300 packets, a remarkable difference. The loss in the standard UDP connection is very high and shows the aggressiveness of UDP in utilizing the entire bandwidth leaving no space for TCP which is halted from transmitting data (see Fig. 10) due to numerous time outs. These time outs happen because all the links are occupied by UDP traffic.

Figure 11 shows the packet arrival rate at the router buffer when using the modified UDP-based system. The total number of packets that arrived successfully at the route is 12,000. Only 350 packets were dropped (see Fig. 8) (that is 2.916 % of total packets), because the system allows (as it is a respon-

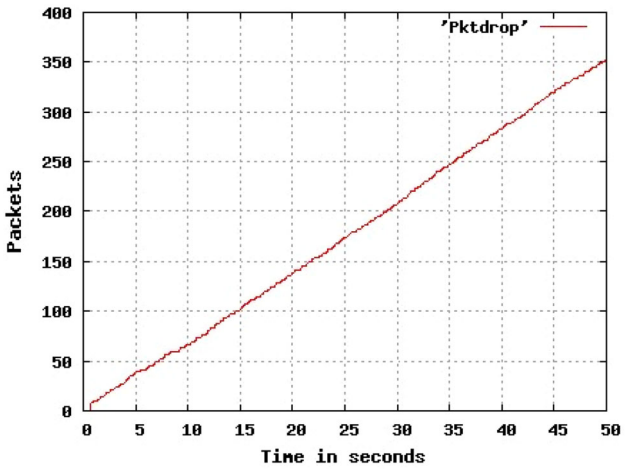


Fig. 8 Total packets drops using modified UDP-based system

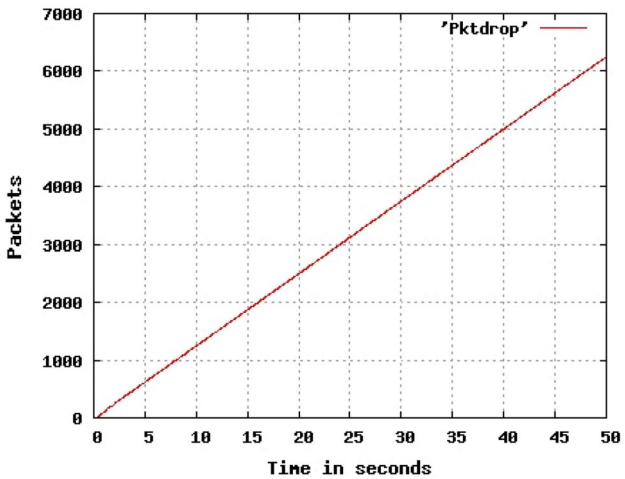


Fig. 9 Total packet drops using standard UDP

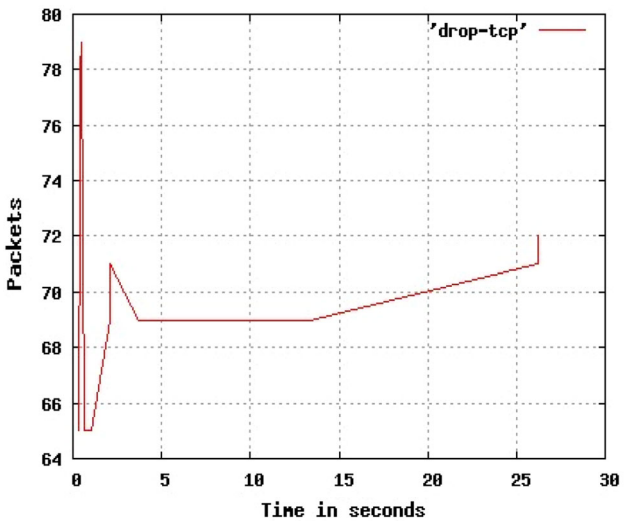


Fig. 10 Number packet drops when TCP connection is used

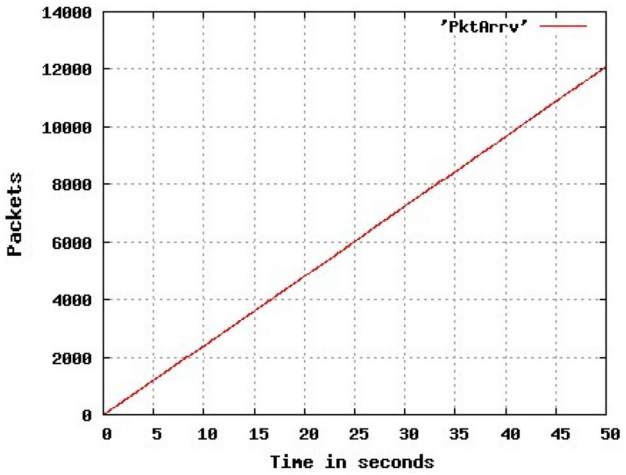


Fig. 11 Packet arrival rate when the modified UDP-based system is used

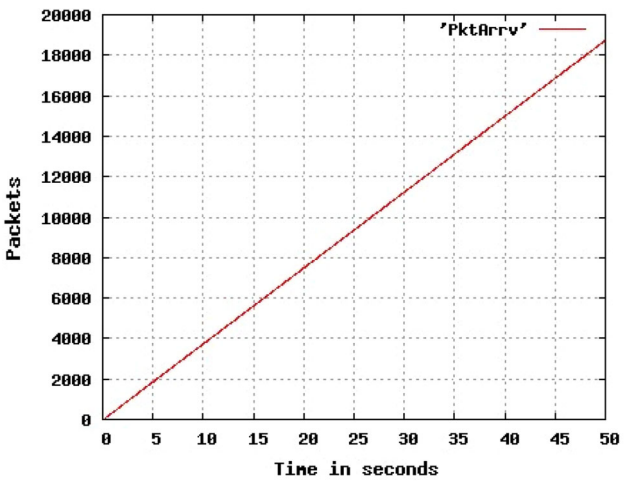


Fig. 12 Packets arrival rate when the standard UDP is used

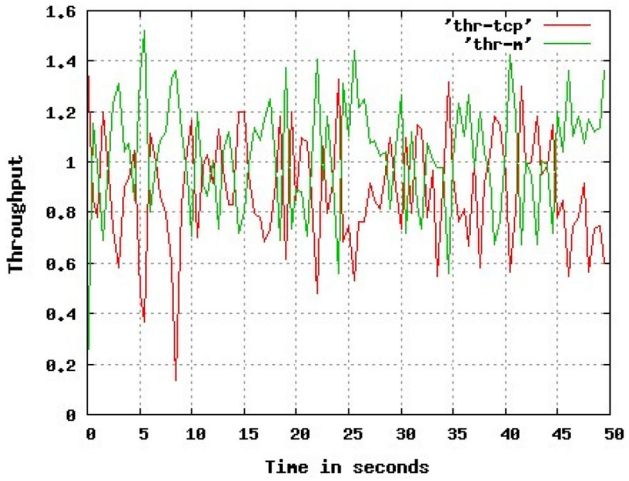


Fig. 13 Throughput gained for TCP and modified UDP

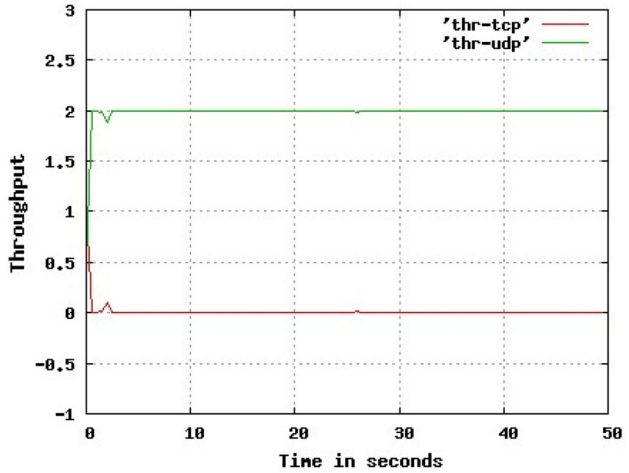


Fig. 14 Throughput gained for TCP and standard UDP

sive system) the queue management mechanism to control the packet arrival rate, thus avoid congestion. In contrast to the modified UDP, the standard UDP arrival rate cannot be controlled, as UDP is an unresponsive protocol.

Figure 12 shows the number of the UDP packets arrived at the router's buffer is 18400. 6300 packets of them were dropped (see Fig. 9) (that is 34.239 % of total packets) by the queue management mechanism at the router.

4.3.3 Throughput comparative drop fairness

Basically, the queue management mechanism employed in the bottleneck router tries to enforce fairness between the different connections crossing the router, in terms of the ratio of packets dropped from a connection with the connection's share of the bandwidth utilization on the bottleneck link. The fairness can be achieved by ensuring that the probability of a packet getting dropped from a specific connection is proportional to the connection's share of the throughput through the router. Figure 13 illustrates throughput gained for both TCP and the modified UDP. Figure 13 also shows that the modified UDP can share the bandwidth with TCP and the total throughput level is acceptable.

There is a big enhancement in TCP throughput compared to that of TCP that uses standard UDP depicted in Fig. 14 which confirms that TCP throughput is almost zero the majority of the time, except during the beginning of the TCP transmission. This low value of throughput is due to the fact that UDP consumes the entire bandwidth and its throughput is very high and almost consistent compared to that of the TCP connection. The Fig. 14 shows the unfairness presented by throughput values, when UDP co-exists with TCP.

4.3.4 Bandwidth link utilization

Network efficiency, or how network resources are utilized, can be measured based on link utilization as well. The per-

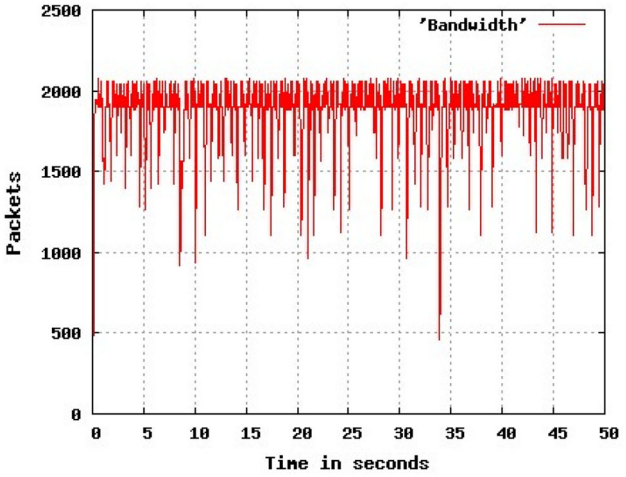


Fig. 15 Bandwidth usage at the bottleneck link using modified UDP-based system

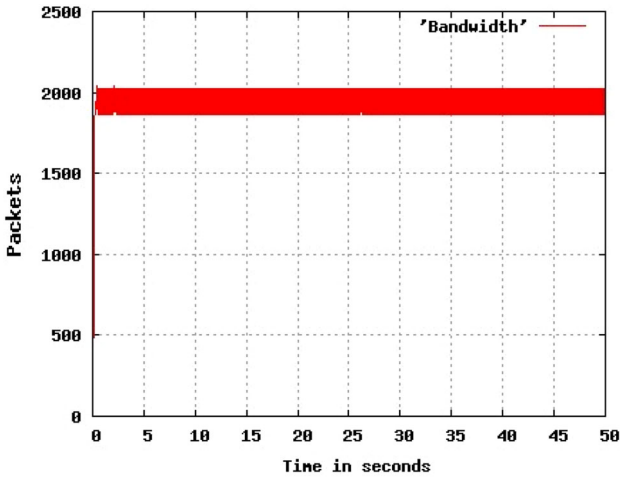


Fig. 16 Bandwidth usage at the bottleneck link using standard UDP

centage of time the queue waits before transmitting packets on the bottleneck link determines the level to which the link bandwidth is utilized. Link utilization also depends on the packet arrival rate and speed of the packet arrival at the router buffer. The link's utilization depends on how frequently the link is used. Thus, the utilization depends on whether the queue contains packets to be transmitted, and not on the exact number of packets present in the queue. A larger queue does not improve the link utilization. The utilization will be improved only when the possibility of the queue being left empty is reduced. Figures 15 and 16 show the bandwidth used at the bottleneck link in the case of the modified UDP and the standard UDP-based systems, respectively. The bandwidth link utilizations obtained on the bottleneck link can be understood from the packet arrival rate and queue size. Figure 15 confirms that the modified UDP-based system is a responsive, friendly one that shares the bandwidth fairly with other protocols like TCP. While Fig. 16 confirms that the standard UDP is an aggressive unfriendly protocol that tries to utilize the bandwidth fully leaving no chance for other protocols to transmit.

5 Conclusion and future work

With the rapid increase in multimedia applications and under the limitation of the network bandwidth, multimedia traffic can cause congestion which can degrade the performance experienced by the network users. Therefore, it is important to reduce the occurrence of congestion situations in a network to optimize the utilization of network resources to provide the network users with suitable performance. Popular transport protocols for transferring multimedia traffic over the Internet are TCP, UDP and DCCP all have several performance issues that affect the quality of service experienced by users.

This paper presents a congestion-aware UDP-based system developed for transmitting multimedia traffic that can help improve network performance. The system consists of a responsive multimedia application and a modified version of UDP. It is implemented in network simulator 2 (ns-2) to evaluate its performance. We found that the developed UDP-based system can improve the performance of the network and it is friendly in terms of sharing the bandwidth. The simulation results showed that the responsive multimedia application helps in alleviating the congestion, reducing the packet drops and increasing the throughput at the end hosts in addition to providing improved network utilization.

For further research, we are going to use the developed multimedia application with its modified UDP protocol on different IP router queue management mechanisms. This requires differentiating this type of multimedia traffic from other types of traffic. This can be achieved by adding some information in the IP header fields which, to the best of our knowledge, is not currently used. The newly arrived packet should be operated upon by the queue management mechanism used and it is either randomly dropped or inserted into the buffer. Multimedia packets can be identified (or labeled) by transport layer (as in ECN-capable packet) when sending. If the packet to be dropped is multimedia packet, it will be admitted and then might be marked at the head of the queue, when necessary, to provide fast congestion notification. Therefore, the queue management algorithm should be implemented in the arrival state. The length of the packet at the front of the queue is obtained, its service time is computed, and a self-interrupt is scheduled to signal the completion of the packet's service in the near future. If more packets are waiting in the queue to be served, the process moves from service-complete to service-start state to initiate the processing of the next packet at the head of the queue.

Acknowledgments This research work is funded by Universiti Sains Malaysia (USM), Short-term Grant No. 304/PNAV6312117.

Appendix

Tool command language (TCL) is utilized to test the performance the responsive UDP-based system in Network Simulator 2 (ns-2) compared to that of standard UDP. In the following, the details of TCL codes are provided.

- Start new simulation.

```
set ns [new Simulator]
```

- Define different colors for data flows

```
$ns color 1 Red
  $ns color 2 Blue
```

- Create the trace files to store information about every event happens.

```
set NamFile [open out.nam w]
  set TraceFile [open out.tr w]
  $ns namtrace-all $nf
  $ns trace-all $tf
```

- Define a 'finish' procedure to close the trace files once the simulation duration elapses and flash the information recorded and execute NAM for visualizing the simulation scenario.

```
proc finish {} {
  global ns nf tf
  $ns flush-trace
  #Close the trace file
  close $nf
  close $tf
  #Execute nam on the trace file
  exec nam out.nam &
  exit 0
  }
```

- Create the network nodes (sources, destinations, and routers nodes).

```
set node_(s1) [$ns node]
  set node_(s2) [$ns node]
  set node_(r1) [$ns node]
  set node_(r2) [$ns node]
  set node_(s3) [$ns node]
  set node_(s4) [$ns node]
```

- Create an access link between the sources and their gateway (R1); destinations and their gateway (R2). Create bidirectional link between bottleneck routers with specific bandwidth and delay.

```
$ns duplex-link $node_(s1) $node_(r1) 5Mb 3 ms Drop Tail
$ns duplex-link $node_(s2) $node_(r1) 5Mb 3 ms Drop Tail
$ns duplex-link $node_(r1) $node_(r2) 2Mb 10 ms RED
$ns duplex-link $node_(s3) $node_(r2) 5Mb 3 ms Drop Tail
$ns duplex-link $node_(s4) $node_(r2) 5Mb 3 ms Drop Tail
```

- Setup the parameters for the queue management mechanism used in this network scenario. Here Random Early Detection (RED) mechanism is used.

```
$ns queue-limit $node_(r1) $node_(r2) 20
  Queue/RED set thresh_ 5
  Queue/RED set maxthresh_ 10
  Queue/RED set q_weight_ 0.002
  Queue/RED set ave_ 0
```

- Monitor the queue link between the bottleneck routers.

```
$ns duplex-link-op $node_(r1) $node_(r2) queuePos 0.5
```

- Set the position for the network nodes for NAM visualization.

```
$ns duplex-link-op $node_(s1) $node_(r1) orient right-down
  $ns duplex-link-op $node_(s2) $node_(r1) orient right-up
  $ns duplex-link-op $node_(r1) $node_(r2) orient right
  $ns duplex-link-op $node_(s3) $node_(r2) orient left-down
  $ns duplex-link-op $node_(s4) $node_(r2) orient left-up
```

- Setup the modified UDP connection for Multimedia traffic between source 1 (s1) and its pair (s3). Define their packet sizes as well.

```
set udp_s [new Agent/UDP/UDPm]
  set udp_r [new Agent/UDP/UDPm]
  $ns attach-agent $node_(s1) $udp_s
  $ns attach-agent $node_(s3) $udp_r
  $ns connect $udp_s $udp_r
  $udp_s set packetSize_ 1000
  $udp_r set packetSize_ 1000
  $udp_s set fid_ 1
  $udp_r set fid_ 1
```

In the comparison scenario, this part is changed to the following:

```
set udp_s [new Agent/UDP]
set udp_r [new Agent/UDP/UDPm]
$ns attach-agent $node_(s1) $udp_s
$ns attach-agent $node_(s3) $udp_r
$ns connect $udp_s $udp_r
$udp_s set packetSize_ 1000
$udp_r set packetSize_ 1000
$udp_s set fid_ 1
$udp_r set fid_
```

- Define the multimedia application and attach it to s1 and s3.

```
set mapp_s [new Application/MApp]
set mapp_r [new Application/MApp]
$mapp_s attach-agent $udp_s
$mapp_r attach-agent $udp_r
$mapp_s set pktsize_ 1000
$mapp_s set random_ false
```

In the comparison scenario, this part is changed to the following:

```
set udp_s [new Application/Traffic/CBR]
set udp_r [new Application/Traffic/CBR]
$ udp_s attach-agent $udp_s
$ udp_r attach-agent $udp_r
$ udp_s set pktsize_ 500
$ udp_s set interval_ 0.005
$ udp_s set random_ false
```

- Setup the TCP connection, of type Newreno, for File Transfer traffic between source 2 (s2) and its destination (s4). Define the TCP source congestion window as well.

```
set tcp [$ns create-connection TCP/Newreno $node_(s2) TCPSink $node_(s4) 0]
$tcp set window_ 15
$tcp set fid_ 2
```

- Setup a FTP Application and attach it to TCP connection for source 2 (s2)

```
set ftp [$tcp attach-source FTP]
```

- Define the start time for FTP and multimedia applications. Define the simulation duration as well.

```
$ns at 0.0 "$ftp start"
$ns at 1.0 "$mapp_s start"
$ns at 50.0 "finish"
```

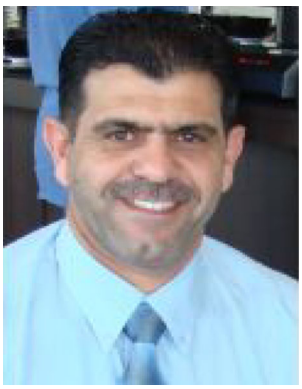
- Call ns to run this scenario.

```
$ns run
```

References

1. Thomas, S., Olivier, F., Alessio, B., Alberto, D., Antonio, P., Giorgio, V., et al.: Traffic analysis of peer-to-peer IPTV communities. Comput. Netw. **53**(4), 470–484 (2009)
2. Zoran, B., Bojan, B., Miodrag, B: Multimedia traffic in new generation networks: requirements, control and modeling. Paper presented at the Proceedings of the 13th WSEAS international conference on communications, 2009
3. Ramaswamy, R: Data confidentiality service on top of transmission control protocol/internet protocol. Paper presented at the CompEuro '90. Proceedings of the 1990 IEEE international conference on computer systems and software engineering, 8–10 May 1990
4. Postel, J.: User datagram protocol, RFC 0768, Internet Engineering Task Force (1980)
5. Floyd, S., Kohler, E.: Profile for datagram congestion control protocol (DCCP) congestion control ID 2: TCP-like congestion control, in RFC 4341 (2006)
6. Shahrudin Awang Nor, S.H., Ghazali, O., Kadhum, M.M.: Performance enhancement of DCCP TCP-like through reduction of congestion window. In: The 2010 International Conference on Modeling, Simulation and Control, ICMSC 2010, Egypt, pp. 247–251 (2010)
7. Yang, H., Zhang, F., Jiao, Q., Tang, X.: Dynamics of TCP-Like congestion control algorithm. In: The Sixth World Congress on Intelligent Control and Automation (WCICA), vol. 1, pp. 825–829 (2006)
8. Ishak, M.I., Ghani, M.A.H.A, Lynn, O.B.: Enhancement and analysis of TFRC performance for real-time data application: a survey. In: The 2nd International Conference on Computer and Automation Engineering (ICCAE), vol. 2, pp. 706–709, 26–28 Feb 2010
9. Sathiaseelan, A.; Fairhurst, G.: Use of quickstart for improving the performance of TFRC-SP over satellite networks. In: International Workshop on Satellite and Space Communications, vol., pp. 46–50, 14–15 Sept 2006
10. Nor, S.A., Hassan, S., Ghazali, O., Omar, M.H.: Enhancing DCCP congestion control mechanism for long delay link. In: International Symposium on Telecommunication Technologies (ISTT), pp. 313–318, 26–28 Nov 2012
11. Kohler, E., Floyd, S.: Profile for datagram congestion control protocol (DCCP) congestion control ID 2: TCP-like congestion control, RFC 4341, Internet Engineering Task Force (2006)
12. Ramakrishnan, K., Floyd, S., Black, D.: The addition of explicit congestion notification (ECN) to IP: RFC 3168 (2001)
13. Floyd, S., Kohler, E., Padhye, J.: Profile for datagram congestion control protocol (DCCP) congestion control ID 3: TCP-friendly rate control (TFRC), RFC 4342, Internet Engineering Task Force (2006)
14. Hoshikawa, T., Ishihara, S.: Estimation of sending rate of DCCP CCID3 flows based on jitter of probe packets on WLANs. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 701–706, 28–31 March 2011
15. Schier, M., Welzl, M.,: Using DCCP: Issues and improvements. In: 20th IEEE international conference on network protocols (ICNP), pp. 1–9 (2012)
16. Lien, Y.-N., Ding, Y.-C.: Can DCCP replace UDP in changing network conditions?. In: IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 716–723, 22–25 March 2011
17. Rahman, J., Saha, S., Hasan, S.F.: A new congestion control algorithm for datagram congestion control protocol (DCCP) based real-time multimedia applications. In: 7th International Conference on Electrical & Computer Engineering (ICECE), pp. 533–536, 20–22 Dec 2012

18. Yu, L., Li, X., Zhang, P., Li, G., Ma, M.,: An improved transport algorithm over DCCP in wireless ad hoc networks. In: IEEE International Conference on Communication Systems (ICCS), pp. 569–573, 19–21 Nov 2014
19. Biplab, S., Kalyanaraman, S., Kenneth, S.V.: An integrated model for the latency and steady-state throughput of TCP connections. Perform. Eval. **46**, 139–154 (2001)



Mohammed M. Kadhum is a Postdoctoral Fellow and staff member of the Telecommunication Research Lab (TRL), School of Computing, Queen's University, Canada. He is currently a senior lecturer at the National Advanced IPv6 Center (NAv6), Universiti Sains Malaysia (USM). He had completed his Ph.D. research in Computer Networking at Universiti Utara Malaysia (UUM). His research interest is on the areas of Computer Networking, Network Quality of Service (QoS), Network Performances, Network Traffic Engineering, Wireless and Mobile Networking, Local Area Networking, Telecommunications/Network Management, Internet Security. He has been awarded with several medals for his outstanding research projects. His professional activity includes being positioned as Technical Program Chair for NetApps2008, NetApps2010, and Co-General Chair for NETAPPS2012, a technical committee member for various well known journal and international conferences, a speaker for conferences, and a member of several science and technology societies.

Hossam S. Hassanein is a professor in the School of Computing at Queen's University. Before joining Queen's University in 1999, he worked at the department of Mathematics and Computer Science at Kuwait University (1993–1999) and the department of Electrical and Computer Engineering at the University of Waterloo (1991–1993). He obtained his Ph.D. in Computing Science from the University of Alberta in 1990, and M.A.Sc. in Computer Engineering from the University of Toronto in 1986. My B.Sc. is in Electrical Engineering from Kuwait University, 1984. He was born in Cairo, Egypt. His research is in the areas of broadband and wireless networks architectures, protocols, and control and performance evaluation. He is the founder and director of the Queen's Telecommunication Research Lab (Queen's TRL) in the School of Computing at Queen's.