QoE-aware Congestion Control Algorithm for Conversational Services in Wireless Environments

Contents

7.1 Introduction

When the bottleneck link is overloaded or channel conditions are bad, the TCP throughput decreases and cannot satisfy the source rate of the multimedia application. This increases, generally, the jitter and the packet loss rate that could impact the user-perceived quality, which is also known as the QoE. Although the QoE is affected by some factors, such as the audio quality, devices, echo, etc., we focus, in this chapter, on improving the QoE through a novel congestion control algorithm. The impact of non-networking factors could be cataloged into a protocol stack to form a conceptual

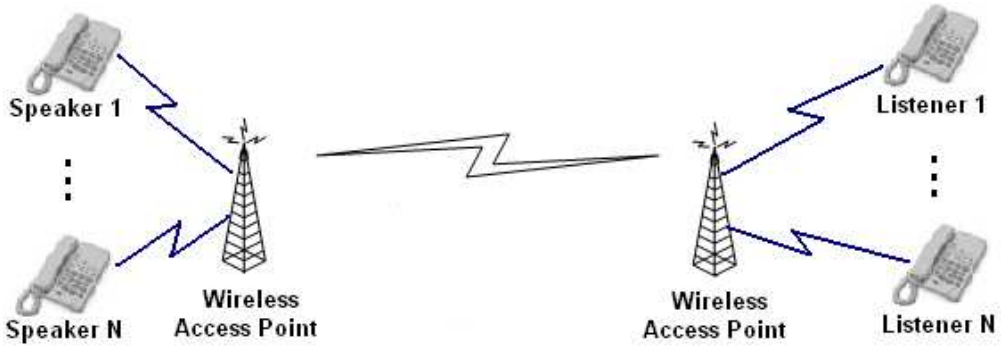

FIGURE 7.1: The experimental model

relationship between QoS and QoE (see [124] and [125]). The QoE is measured by MOS values. In a subjective test, the QoE is rated on a scale of 1 (bad) to 5 (excellent) by a significant number of people, and the average of the scores is called a MOS. Note that, the ITU-T Recommendation P.911 [126] provides the reference for carrying out subjective measurement of audiovisual materials.

In this chapter, we propose a QoE-aware POMDP-based congestion control algorithm, referred to as MOS-TCP, which exhibits an improved performance when transporting multimedia applications, specifically over a wireless path. Our algorithm is suited for networks containing wireless branches, like the model depicted in Figure 7.1. The goal of the MOS-TCP algorithm is to control the end-to-end congestion in order to maximize the QoE, where packets can be lost due to congestion or randomly due to errors encountered across the wireless path. Unlike the current TCP congestion control protocol that only adapts the congestion window to the network congestion (e.g. based on network congestion signals, such as the packet loss rate in TCP Reno, or the round-trip time in TCP Vegas), the proposed congestion control algorithm adopts a two-level congestion control mechanism. Indeed, it adapts over time the congestion window size according to the source rate and the QoE feedbacks. Moreover, we consider a set of updating policies composed of generic congestion control algorithms with general increase and decrease functions, such as AIMD, IIAD, SQRT, and EIMD. In order to capture dynamics of the network congestion and optimize the QoE, we formulate the congestion control using a POMDP framework. The proposed POMDP framework allows users to evaluate the network congestion variations over time, and determines an optimal threshold-based congestion window updating policy in order to maximize the long-term discounted reward. In this chapter, the QoE measured through the multimedia quality (MOS) defines the reward.

In summary, we address the following contributions:

QoE-aware congestion control: The proposed MOS-TCP adapts the AIMD-like congestion control policy to both varying network congestion and multimedia characteristics.

POMDP-based adaptation: We formulate the QoE-aware congestion control problem using a POMDP framework. The framework allows senders to optimize the congestion window updating policy that maximizes the long-term expected QoE. Furthermore, users have a partial knowledge about the bottleneck link status. In fact, the number of packets in the bottleneck link queue depends on the congestion windows of all users, which cannot be observed. Therefore, the long term prediction and adaptation of the POMDP framework is essential for optimizing the performances of multimedia applications.

Online learning for QoE-sensitive multimedia applications: Since the computation of an optimal policy is usually time/process consuming, and as wireless devices are capacity-limited, we propose practical learning method to solve the POMDP-based congestion control problem on-the-fly. The proposed model-free learning algorithm is based on TD-λ reinforcement learning, and is designed for QoE-sensitive multimedia applications.

This chapter is organized as follows. We introduce the QoE and explain the MOS calculation in Section 7.2. In Section 7.3, we model the QoE-aware congestion control problem that maximizes the performance of multimedia applications. Thereafter, in Section 7.4, we formulate the problem using a POMDP-based framework. We present a low-complexity algorithm to solve the POMDP in Section 7.5. Section 7.6 provides experimental results that validate the proposed congestion control method, and Section 7.7 concludes the chapter.

7.2 QoE-aware networking and MOS measurement

To overcome the limitation of QoS-based optimization, QoE-based approaches are introduced as a more effective way to optimize transmission algorithms and protocols with respect to user satisfaction. QoE metrics are defined as a set of quantitative measures to assess the perceived QoS of end users [127]. Moreover, a new approach, namely *QoE-aware networking*, is proposed to re-formalize the service optimization problem and to improve the user experience. Because the QoE metrics reflect the end user's experience, QoE-based approaches may improve the subjective service quality, optimize the use of network resources, and provide services to more users without noticeable degradation of users' experience. Recently, QoE metrics are used to optimize various types of network services. In cellular systems, authors of [128] used a QoE-based approach to allocate downlink wireless resources among different applications. They defined several QoE models for different types of applications such as file downloading, voice call and video

streaming, and adopt QoE-based utility maximization to improve the user perceived quality. In [129], authors applied QoE metrics to optimize IEEE 802.11 wireless LAN. They used a machine learning approach to generate real-time QoE measurements and used the QoE feedbacks to manage wireless network resources. In [130], authors used QoE metrics for packet scheduling in multi-hop wireless networks. The packet scheduler determines the packet drop pattern that minimizes the degradation of MOS values. In P2P networks [131], scalable video coding and QoE metrics are used to optimize the performance of P2P video streaming systems. In this chapter, we seek to enable QoE-awareness in a more general setting. We integrate the QoE metrics within the TCP protocol. Since TCP is a widely adopted building block in many network services, our approach is applicable to a much wider spectrum of applications.

Since QoE metrics are subjective, the standard QoE measuring process should involve human observers, e.g., when measuring VoIP quality, the MOS are often used as a subjective rating ranging from 1 (poor) to 5 (excellent). However, to enable QoE-awareness in multimedia services, it is infeasible to use subject human tests for real-time applications. Instead, some QoE online prediction methods should be used to estimate QoE from the service output. The QoE prediction methods are dependent on the types of content. A number of models are proposed for predicting QoE with different kinds of contents including web service[132], voice services [133], audio/video content [134], etc. Instead of proposing another new approach of QoE prediction, we base our experiments on the QoE prediction results produced by an existing real system, i.e. Microsoft Lync system [135] (previous known as Office Communication Server and Office Communicator [136]). In the Lync system, the VoIP software measures a set of variables, which may affect the QoE throughout the communication sessions. Based on the collected measurements, it can predict the subjective QoE metrics in real-time. Furthermore, the QoE metrics are normalized and represented in the standard MOS. Our considered Lync software provides several types of MOS values (NetworkMOS, ListeningMOS, conversationalMOS) in order to represent the degradation in different phases of the whole communication process (see Figure 7.2). The MOS prediction mechanism provides a quantitative approach to evaluate the communication quality that end users have experienced.

- NetworkMOS is calculated purely based on obtained network statistics (information), which include the packet loss, bit errors, packet delay, and jitter.
- ListeningMOS is not only decided by network parameters, but also by the choice of audio codec and audio devices, as well as the recording conditions such as echo, background noise level, talk-over, etc. It captures the perceived quality of an audio stream at the receiver side. Note that both NetworkMOS and ListeningMOS are only measured for unidirectional traffic.

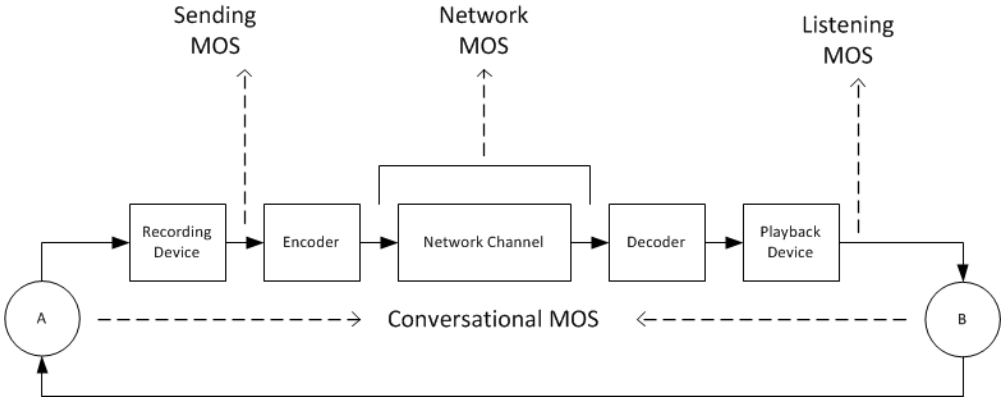

FIGURE 7.2: Different MOS measurements in Microsoft Lync system

- ConversationalMOS is measured for both sending and receiving streams. It takes into account the round-trip delay in addition to all the above-mentioned factors.

Observing these different MOS values gives us a clear perspective on the performance of the entire communication process. A congested network, for example, will cause degradation in NetworkMOS, while a bad recording device can be identified from low ListeningMOS values.

7.3 QoE-aware congestion control problem

We consider the same network model and congestion window adaptation defined in Chapter 6. In this section, we discuss the objective of the proposed QoE-aware congestion control. Denote by R_n^k the source rate of a multimedia application for user n in the k-th epoch. The source rate is the average number of packets that arrives at the transmission buffer per second at the transport layer. In fact, in a VoIP call, the source rate can be controlled and adapted to the network environment, since there are usually some rate control modules implemented in VoIP software.

We propose, in this chapter, a congestion control algorithm that dynamically changes the congestion window updating policy in order to maximize the QoE. Therefore, it is straightforward and somehow intuitive that each user has as objective to maximize its own QoE. As we can see, in Figure 7.3, the MOS is correlated with the listener satisfaction. The higher MOS, the greater the listener's satisfaction. Therefore, the objective of users is to maximize the expected future MOS starting from the current slot. A similar utility function was used in [137]. Each user tries to optimize, selfishly,

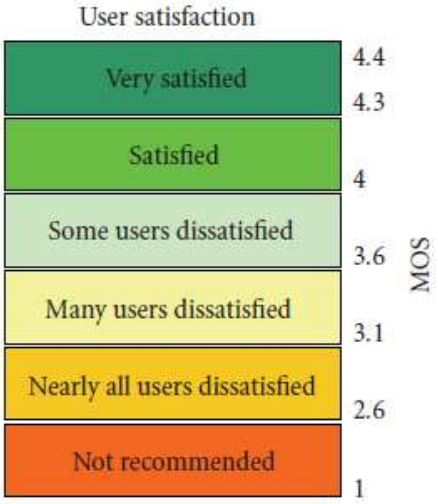

FIGURE 7.3: Relation between MOS and user satisfaction [134].

the following expected total discounted MOS:

$$U_n = \sum_{k=1}^{\infty} \gamma^k u_n^k(\mu_n^k, R_n^k), \tag{7.1}$$

where γ is a discount factor, and $u_n^k = MOS_n^k(\mu_n^k, R_n^k)$ is the received MOS by the user n at the k-th epoch, when the source rate at the k-th epoch is R_n^k, and the user n uses the congestion window updating policy μ_n^k. Since the network user do not take frequently decision (it chooses a congestion control every TRTT), we have used the expected discounted reward criterion instead of the average reward criterion. MOS-TCP mechanism allows the user to maximize its expected total discounted MOS. In fact, the QoE varies with the source rate, the congestion window updating policy, and the congestion status at the bottleneck links. The latter depends not only on the user n but also on other users. We show, in the next section, how the proposed POMDP-based congestion control algorithm determines the optimal updating policy given partial knowledge of bottleneck links status.

7.4 POMDP-based congestion control

The network user maximizes its expected discounted QoE expressed by Equation (7.1). We formulate our problem using a POMDP-based framework as the global system state is not well known for users. In fact, the user has a partial knowledge about the congestion status at the bottleneck links. The latter depends on the congestion windows of all users, which is unknown by the user n. Thus, the user n has to estimate solely the impact of all the other users based on the history of observations and actions. In

fact, the user n estimates the packet loss rate when it transmits data using the congestion window w_n. We define a POMDP-based congestion control of user n in a tuple $\{\mathcal{A}, \mathcal{X}_n, O_n, \Omega_n, P_n, U_n\}$ as follows:

Action: The user selects the congestion window updating policy $\mu_n = \{\mu_n^1, \mu_n^2, ...\} \in \mathcal{A}$, with μ_n^k is the updating policy of user n in the k-th epoch.

State: The state is defined as $X_n^k = \{p_n^k, R_n^k\} \in \mathcal{X}_n$. The source rate R_n^k is known by the user n. However, the packet loss rate p_n^k, which is impacted by other users' windows, cannot be directly observed. The user n has to infer the congestion status of the bottleneck links using congestion observations and QoE feedbacks. The belief of the packet loss rate is defined as $b : [0, 1] \rightarrow [0, 1]$. The function $b(.)$ represents the probability distribution of the packet loss rate.

Observed information and observation probability: The observed information is defined by congestion events $o_n \in O_n$. The observation probability is defined as a function $\Omega_n : \mathcal{T}_n \times O_n \rightarrow [0, 1]$. Let $\Omega_n^{k-1}(o_n = fail|w_n)$ represent the probability of packet loss when the congestion window size is w_n at the $(k-1)$-th epoch.

The conventional POMDP updates the belief function per time slot (RTT), but in the proposed POMDP framework, $b_n(p_n^k)$ is updated per epoch. In fact, the belief distribution is kept the same within the epoch, which reduces the computational complexity and also the memory requirement for calculating the optimal policy. Note that by updating the belief per epoch, we also tolerate delayed MOS feedbacks.

State transition: The average packet loss rate p_n^k when using the congestion window updating policy μ_n^k at the k-th epoch cannot be known by n until the end of the epoch. Instead, the user estimates it based on the following expression:

$$b(p_n^{k+1}|\mu_n^{k+1}) = \frac{Prob(p_n^{k+1}|p_n^k, \mu_n^{k+1})}{\sum_p Prob(p|p_n^k, \mu_n^{k+1})}, \tag{7.2}$$

where $Prob(p_n^{k+1}|p_n^k, \mu_n^{k+1})$ is the probability that the packet loss rate will be p_n^{k+1} at the $(k+1)$-th epoch when choosing the policy μ_n^{k+1}, given that the packet loss rate is p_n^k.

Based on the MOS feedbacks obtained at the end of every epoch, the user chooses the updating policy that maximizes the QoE, as illustrated in Figures 7.4 and 7.5.

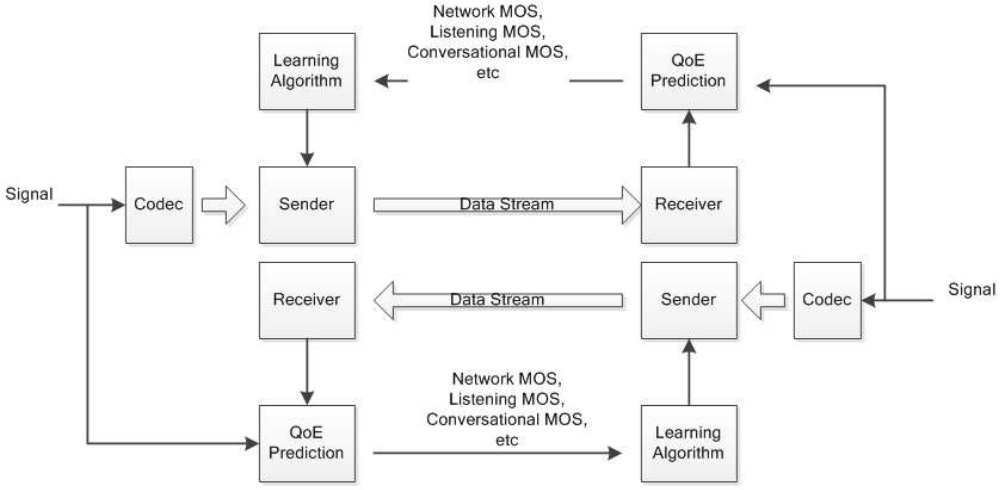

FIGURE 7.4: MOS exchange in bidirectional conversation.

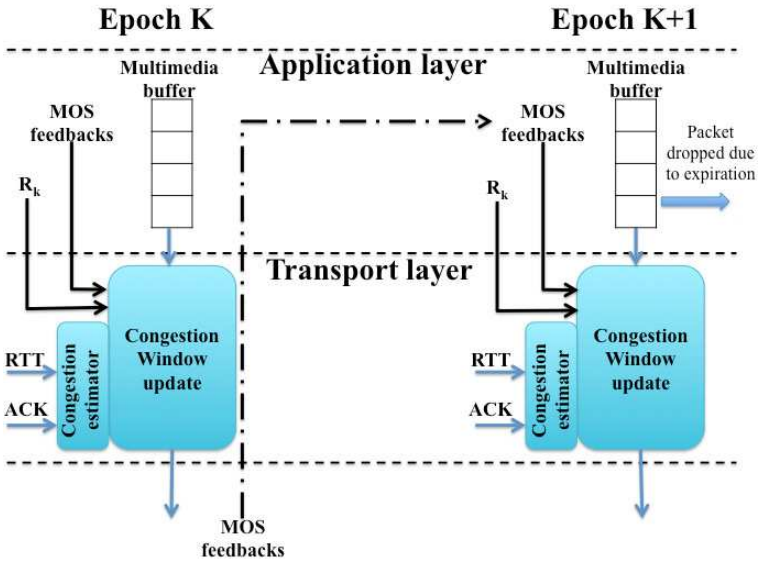

FIGURE 7.5: System diagram of MOS-TCP in time epoch k and $k+1$.

7.5 MOS-based POMDP algorithm

We propose, in this section, a POMDP-based algorithm in order to maximize the QoE for multimedia applications. Every epoch, MOS-TCP users receive three feedbacks: NetworkMOS, ListeningMOS and ConversationalMOS. These feedbacks reflect the listener's satisfaction, and the user has to choose the action that improves the total expected QoE. Therefore, based on these feedbacks, we propose a POMDP-based algorithm that maximizes the expected QoE. Furthermore, as solving POMDPs is an extremely difficult computational problem, we present a low computation complexity online learning algorithm in order to solve the proposed congestion control. Note that learning algorithms are very useful in wireless systems as they require low complexity.

7.5.1 Packet-loss differentiation

The main obstacle, that wireless networks have to face, is physical impairments. The fast recovery algorithm solves the single packet loss within one window. However, due to the nature of wireless networks, a fading channel may cause contiguous packet losses. Therefore, the key idea of designing a wireless TCP is to distinguish the cause of packet loss. Many schemes were proposed in the literature. For example, TCP Veno [32] estimates the backlogged packets in the buffer of the bottleneck link. It determines the optimal throughput the network can accommodate based on the minimal RTT. The difference between the optimal throughput and the actual throughput can be used to derive the amount of backlogged packets. TCP Veno suggests that the loss is said to be random if the number of backlogged data is below a threshold, otherwise the loss is considered as congestive. In our congestion control algorithm, we implement the same methodology, depicted in Algorithm 1, in order to distinguish random packet loss from congestive loss.

7.5.2 The objective function

Since our objective is to avoid the congestion at bottleneck links and improve the QoE, the MOS represents a consistent feedback that gives us information about the impact of the congestion status on the multimedia quality. MOS feedbacks vary with the packet loss rate and the jitter interval, which are related to the congestion status at the bottleneck links. The higher the MOS, the better the QoE and the lower the packet loss rate and the jitter interval. Note that our objective is to maximize the total expected received MOS. Depending on the multimedia application, the user maximizes the expected QoE using NetworkMOS, ListeningMOS or ConversationalMOS feedback. All these MOS feedbacks depend on the packet loss rate and on the jitter interval, both of which depend on the source rate and the congestion window updating policy.

7.5.3 The optimal policy

A policy that maximizes U_n is called an optimal policy $\mu_n^{opt} = \{\mu_n^{opt,1}, \mu_n^{opt,2}, ...\}$, it specifies for each epoch k the optimal updating policy $\mu_n^{opt,k}$. The optimal value function U_n^{opt} satisfies the following Bellman equation:

$$U_n^{opt,k}(p_n^k) = \max_{\mu_n^k \in \mathcal{A}} \{u_n^k(\mu_n^k, R_n^k) + \gamma \sum_{p'} b(p_n^k) T(p'|p_n^k) J_n^{k+1}(p')\}. \tag{7.3}$$

The optimal policy at the k-th epoch is expressed as follows:

$$\mu_n^{opt,k} = \arg\max_{\mu_n^k \in \mathcal{A}} \{u_n^k(\mu_n^k, R_n^k) + \gamma \sum_{p'} b(p_n^k) T(p'|p_n^k) J_n^{k+1}(p')\}. \quad (7.4)$$

7.5.4 Online learning

Solving the presented POMDP is expensive in terms of time (calculation) and space (memory) complexity. Then, it is not suitable for wireless systems with small capacity multimedia devices. In this section, we present a low-complexity online learning algorithm. Our online learning is an extension of the on-policy TD-λ algorithm Sarsa [121] for POMDPs.

Each MOS-TCP user estimates the state-values $Q(\mu_n^k, R_n^k, p_n^k)$, defined as the expected future reward starting from state (R_n^k, p_n^k) and taking the action μ_n^k. The MOS-TCP user chooses, at every epoch, the optimal policy based on Algorithm 3. Specifically, this algorithm supports delayed MOS feedbacks, as it changes the congestion window updating policy per epoch. As illustrated in Figure 7.5, the user gets some feedbacks at the end of each epoch, which reflects the impact of the network on the listening quality. Therefore, the user applies the online learning algorithm in order to choose the congestion window policy that maximizes the expected future MOS starting from the current slot. At the beginning of epoch k, the user receives the source rate R_n^k from the upper layer and selects the congestion window updating policy that maximizes its state-value function. Then, the user transmits its packets during the epoch using the chosen policy. At the end of the epoch, the user computes the packet loss rate and updates the state-value function $Q(\mu_n^k, R_n^k, p_n^k)$ based on the observed MOS feedback. Moreover, the user updates the belief probability of the packet loss rate. Depending on the MOS feedback considered in the objective function, we denote Network-CC the MOS-TCP algorithm that maximizes the NetworkMOS, Listening-CC the MOS-TCP that maximizes the ListeningMOS, and Conversational-CC the algorithm that maximizes the expected ConversationalMOS.

7.5.5 Implementation and complexity

Although the value iteration algorithms give exact solutions for the POMDP optimization problems (see [34]), they need expensive time and space complexities. In fact, the sender needs a large storage space, and spends an exponential time when seeking for the optimal policy. As we can see in Table 7.1, the complexity of the exact POMDP solutions grows exponentially with the number of epoch. Importantly, our online learning algorithm can be implemented on mobile devices that do not have a sophisticated

Algorithm 3 MOS-TCP online learning algorithm for POMDP-based congestion control

Initialize $Q(\mu_n^k, R_n^k, p_n^k) = 0$;
$k \leftarrow 1$;
Get application parameters R_n^1;
Choose arbitrarily the updating policy (μ_n^1);
while true **do**
  for $t = 1 \rightarrow T$ **do**
    Update the congestion window using policy μ_n^k ;
    Update the observation probability Ω_n^k based on congestion event observation;
  end for
  Evaluate the packet loss rate p_n^k;
  The user gets the QoE feedbacks: MOS;
  Update the beliefs based on Equation (7.2);
  Get application parameters R_n^{k+1};
  Choose updating policy

$$(\mu_n^{k+1}) = \arg\max_{\mu_n^{k+1}} Q(\mu_n^{k+1}, R_n^{k+1}, p_n^{k+1}) b_n(p_n^{k+1}),$$

  with probability $(1 - \epsilon)$;
  Else choose a random policy in $\mathcal{A}$;

$$Q(\mu_n^k, R_n^k, p_n^k) \leftarrow Q(\mu_n^k, R_n^k, p_n^k) + \alpha[MOS + \gamma \times \sum_{p_n^{k+1}} Q(\mu_n^{k+1}, R_n^{k+1}, p_n^{k+1}) b_n(p_n^{k+1}) - Q(\mu_n^k, R_n^k, p_n^k)];$$

  $k \leftarrow k + 1$;
end while

TABLE 7.1: Comparisons of exact POMDP solution and the proposed online learning algorithms

	Exact solution	**MOS-TCP**
Consumed Memory	$O(\lvert\mathcal{A}\rvert\lvert\mathcal{V}_{k-1}\rvert^{\lvert\mathcal{O}\rvert})$, with $\mathcal{V}_k$ is the solution of the $(k-1)$-th epoch	$O(\lvert\mathcal{A}\rvert\lvert\mathcal{X}\rvert)$
Time complexity	$O(\lvert\mathcal{X}\rvert^2\lvert\mathcal{A}\rvert\lvert\mathcal{V}_{k-1}\rvert^{\lvert\mathcal{O}\rvert})$	$O(\lvert\mathcal{A}\rvert\lvert\mathcal{X}\rvert)log(\lvert\mathcal{A}\rvert\lvert\mathcal{X}\rvert)$

calculation or a large memory space. Moreover, the proposed algorithm is implemented only on the transmitter side and is transparent to the receiver side. We do not even require any change at the routers. Interestingly, this algorithm supports the delay of MOS feedbacks as it updates the congestion window updating policy per epoch.

7.6 Numerical illustrations

7.6.1 Testbed experiments

Microsoft Lync is an integrated software-based communication and collaboration platform, which is mainly designed for enterprise users. It provides various real-time communication features such as instant messaging, software-based voip, and video/audio conferencing through the same user interface. The system includes a set of server components that can be deployed in the enterprise network. After installing the client-side component, enterprise users can initiate audio/video calls with others or set up a group conference through the IP network. Furthermore, it supports communications with traditional phone through some PSTN gateway.

The system supports the standard Session Initiate Protocol (SIP) for signaling and RTP/RTCP protocols for transmitting media packets. For the two-way communications, clients can directly connect with each other and transmit data in a peer-to-peer way. For multi-users conferencing sessions, a Multimedia Controller Unit (MCU) server can help to coordinate the session and to replicate data packets to all receivers. When users are behind some Network Address Translator (NAT) or firewalls, a mediation server allows clients to relay data packets. The MOS prediction module in Lync is implemented at the application layer and is independent on the transport protocol. The underlying transport protocol in Lync can be TCP, UDP, or even server-relayed tunnels (e.g. Traversal Using Relay NAT (TURN) protocol), depending on the connectivity of the Lync clients.

The proposed algorithm is implemented only at the sender side, and is transparent to routers and receiver. However, an end-to-end signaling mechanism needs to be implemented at the application layer on both the transmitter and the received side. Note that a library-based MOS feedback mechanism can be adopted to help developers of multimedia applications to design QoE-based multimedia applications without the need of run-time training and signaling.

MOS feedbacks need to be sent from the receiver side to the sender at every epoch. The MOS prediction and feedback are located at the application layer. Thus, there is no need to modify the receiver part of the TCP code. Meanwhile, the TCP sender part can be designed to be backward compatible, i.e. the sender works in normal mode when there is no MOS feedback and switches to the MOS-based congestion control mode only when the application layer has indicated it to do so. In this way, MOS-TCP clients can still interact with the old non-MOS version ones.

In our experiments, the QoE trace is captured and anonymized from a deployed Microsoft Lync 2010 Service in Microsoft Labs. The duration of the collected trace is about three

months. The average length of each session is 11 minutes. From the original trace, we extract only the PC-to-PC audio streams since it reflects the voice quality over pure IP networks. The extracted part contains 1,935,110 end-to-end audio streams in total. The audio codec used by clients is Microsoft RTAudio Speech codec with the clock rate 16KHz. We use the Gilbert model to model the wireless channel conditions. This approach was introduced in [138]. By generating synthetic traces that simulate the wireless network being tested, multiple users can access the network simultaneously and perform experiments.

We consider a set of policies $\mathcal{A}$ composed of AIMD, IIAD and SQRT, defined as follow:

$$\text{AIMD: } f(w) = \frac{3\beta}{2-\beta} \text{ and } g(w) = \beta;$$

$$\text{IIAD: } f(w) = \frac{3\beta}{2w-\beta} \text{ and } g(w) = \frac{\beta}{w};$$

$$\text{SQRT: } f(w) = \frac{3\beta}{2\sqrt{w+1}-\beta} \text{ and } g(w) = \frac{\beta}{\sqrt{w+1}};$$

where $\beta \in \{0.1, 0.2, ..., 0.9\}$. Note that the conventional TCP is AIMD(0.5). We compare our proposed algorithms with other congestion control algorithms for multimedia applications. We focus, especially, on AIMD and Binomial congestion control algorithms. In fact, authors of [113] proved that the AIMD-based Binomial congestion control algorithms IIAD and SQRT are well-suited for multimedia applications. We consider that the data is transmitted over an IEEE 802.11a wireless link and the playback delay is 200 ms. We use IEEE 802.11a in our numerical study only for illustrative purposes, and any kind of wireless device can be used instead.

7.6.2 Unidirectional communications

In this section, we focus on the unidirectional communications with a speaker and a listener in each session. We present a comparative study between Listening-CC, Network-CC and other congestion control algorithms. We compare, in different scenarios, the QoE (ListeningMOS) and we consider the following congestion control algorithms: Listening-CC, Network-CC, Binomial congestion control and AIMD algorithms. We do not compare with Conversational-CC as we are considering unidirectional communications. We consider that each pair is composed of a transmitter (speaker) and a receiver (listener). Let us focus on the first scenario of Table 7.2. We run audio transmissions with different source rates and we plot, in Figure 7.6, the obtained QoE for different type of users.

TABLE 7.2: Experimental scenarios in unidirectional communications

	AIMD	IIAD	SQRT	NetworkCC	ListeningCC
Scenario 1	2	2	2	2	2
Scenario 2	4	4	4	4	4
Scenario 3	8	8	8	8	8

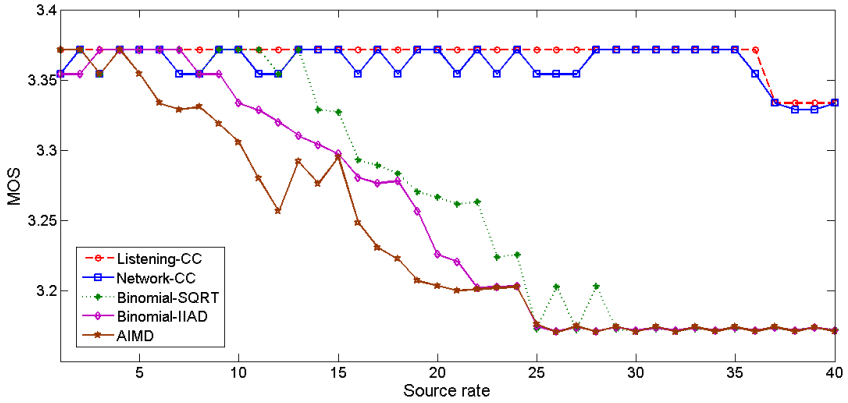

FIGURE 7.6: ListeningMOS with different source rates in the first scenario.

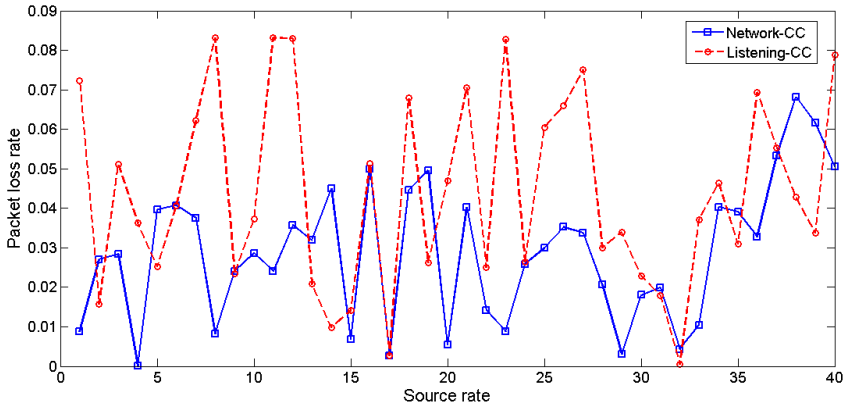

FIGURE 7.7: Packet loss rate depending on the source rate in the first scenario.

We observe that the Listening-CC and Network-CC algorithms improve significantly the QoE compared to AIMD and Binomial congestion control algorithms. Moreover, the MOS obtained with Listening-CC is slightly better than the MOS obtained by the Network-CC algorithm. Furthermore, as we can see in Figure 7.7, the packet loss rate for Listening-CC users is higher than Network-CC. In fact, as the NetworkMOS depends only on network factors, maximizing this MOS minimizes the packet loss rate and the jitter interval. However, Listening-CC bases on ListeningMOS, which depends on other factors than the network ones. Then, users can choose a policy that maximizes the ListeningMOS even with higher values of packet loss rate and jittering. Consider the second scenario of Table 7.2. We observe, in Figure 7.8, that Listening-CC and

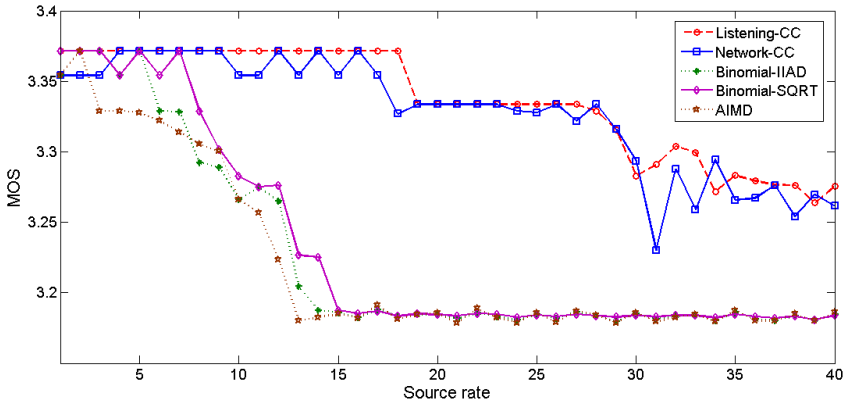

FIGURE 7.8: ListeningMOS with different source rates in the second scenario.

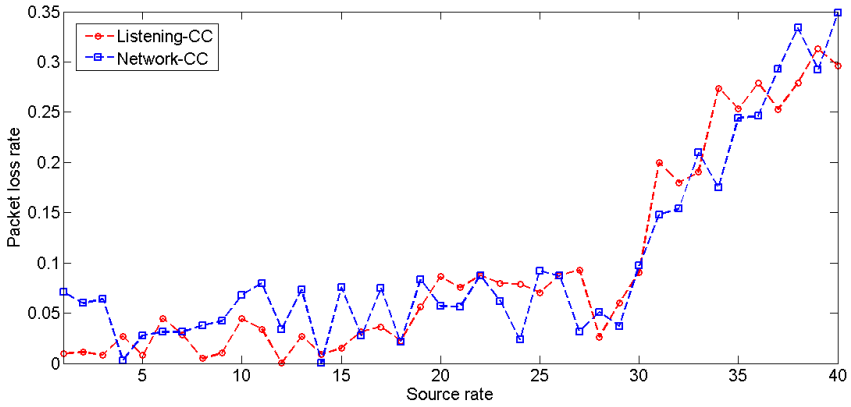

FIGURE 7.9: Packet loss rate depending on the source rate in the second scenario.

Network-CC algorithms lead to better QoE than Binomial and AIMD users. Moreover, Listening-CC leads to slightly better QoE than Network-CC. Figure 7.9 illustrates that the packet loss rate for both Listening-CC and Network-CC algorithms is increasing with the source rate as the bottleneck link become overloaded. The fluctuation of packet loss rate is due to the imperfect characteristics of the wireless link. In the third scenario of Table 7.2, we consider more load on the bottleneck link. Figures 7.10 and 7.11 illustrates the ListeningMOS and the packet loss rate for different congestion control algorithms. It is clear that the MOS-TCP frameworks lead to better QoE. However, the improvement decreases with the source rate, and all congestion control algorithms give the same QoE for high values of source rate. In fact, with such number of audio sessions and source rates, the wireless link is always overloaded and the source rates requested by users cannot be satisfied. Therefore, the packet loss rate increases for all the users, and the QoE decreases. In summarize, both Listening-CC and Network-CC algorithms improve the QoE compared to other AIMD-based congestion control algorithms for multimedia transmission. Moreover, Listening-CC is slightly better than Network-CC algorithm,

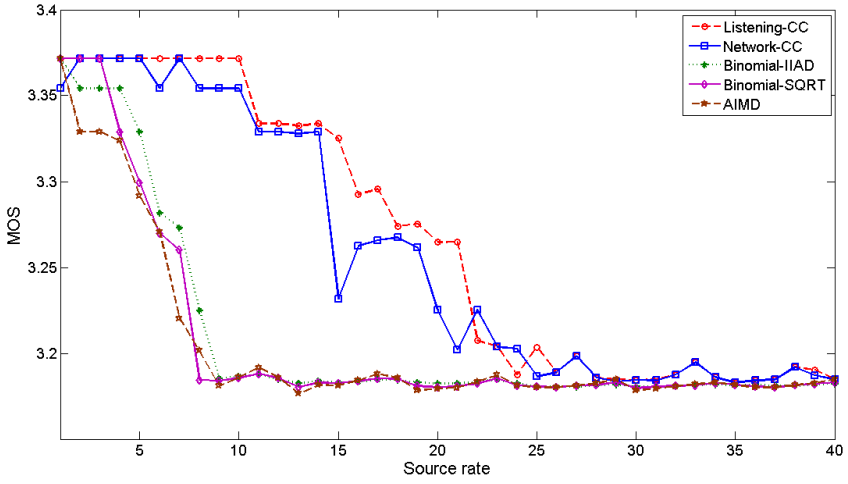

FIGURE 7.10: ListeningMOS with different source rates in the third scenario.

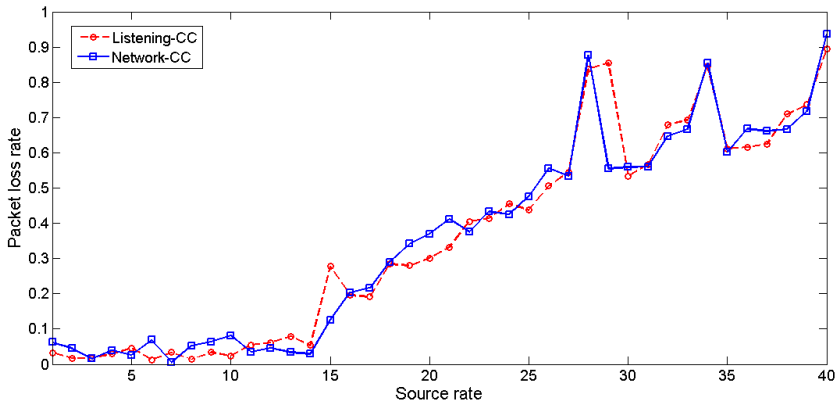

FIGURE 7.11: Packet loss rate depending on the source rate in the third scenario.

TABLE 7.3: Experimental scenarios in bidirectional communications

	AIMD	IIAD	SQRT	NetworkCC	ListeningCC	ConversationalCC
Scenario 1	2	2	2	2	2	2
Scenario 2	4	4	4	4	4	4
Scenario 3	8	8	8	8	8	8

as it considers not only packet loss rate and jitter but also the impact of non-network factors.

7.6.3 Bidirectional communications

We consider, in this section, bidirectional audio conversations. We run the three scenarios presented in Section 7.6.2 with a bidirectional communication.

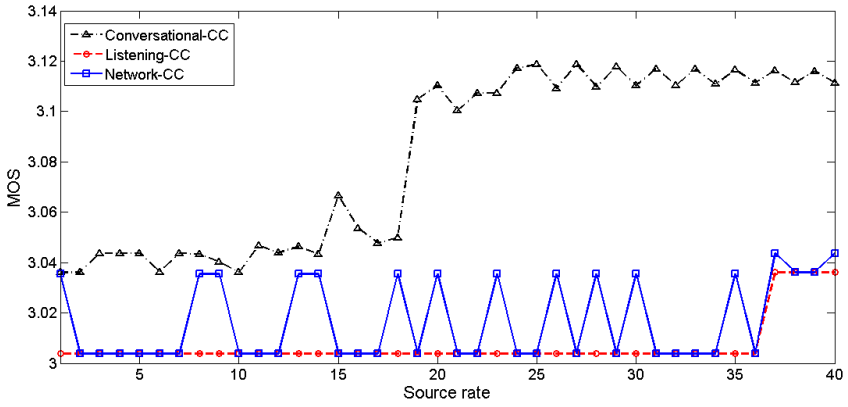

FIGURE 7.12: ConversationalMOS with different source rates in the first scenario.

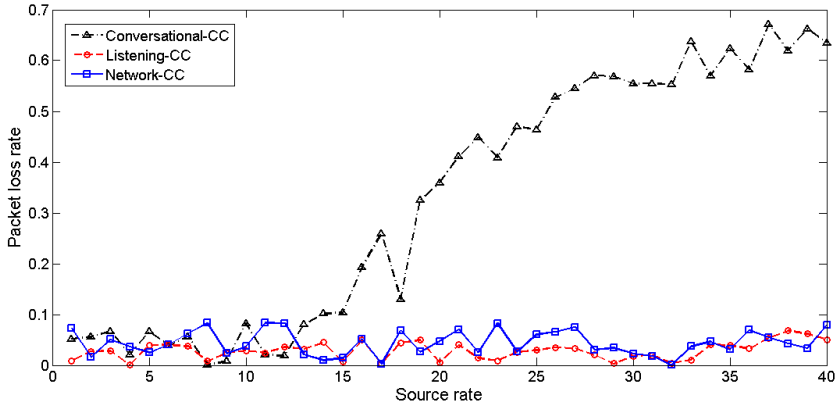

FIGURE 7.13: Packet loss rate depending on the source rate in the first scenario.

In the first scenario of Table 7.3, we run the conversations over the wireless link. Figure 7.12 illustrates the QoE (conversationalMOS) for Conversational-CC and other congestion control algorithms with different values of the source rate. We observe that the Conversational-CC leads to better QoE than Listening-CC and Network-CC algorithms. Surprisingly, the improvement of Conversational-CC compared to Listening-CC and Network-CC algorithms is more important for higher source rate. In fact, for high values of source rate, we observe, in Figure 7.13, that Conversational-CC algorithm is more aggressive than other congestion control algorithms as it leads to significantly higher packet loss rate.

In the second scenario of Table 7.3, we plot, in Figures 7.14 and 7.15, the Conversational-MOS and the packet loss rate for different congestion control algorithms depending on the source rate. We observe that the Conversational-CC algorithm outperforms other congestion control algorithms. Moreover, we remark that for some values of the source

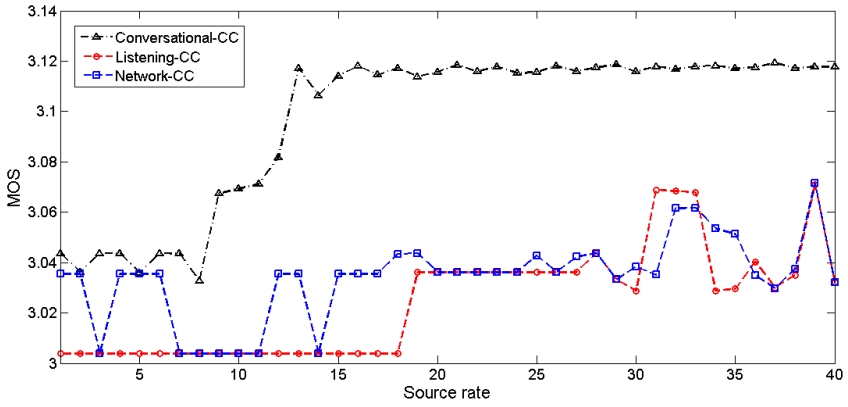

FIGURE 7.14: ConversationalMOS with the source rates in the second scenario.

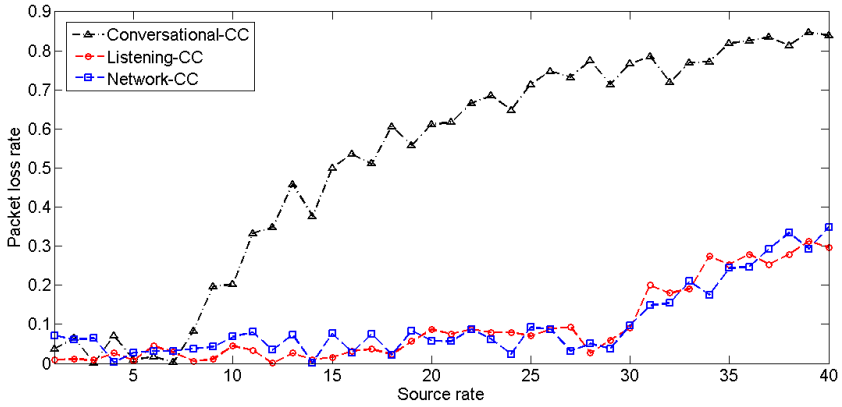

FIGURE 7.15: Packet loss rate depending on the source rate in the second scenario.

rate, the Listening-CC in better than Network-CC and for other values Network-CC is better.

Let us focus on the third scenario of Table 7.3. Figure 7.16 shows that the Conversational-CC leads to better QoE than other congestion control algorithms. In fact, it bases on the conversationalMOS feedback which takes into consideration both sent and received audio streams, and is less sensitive to the network factors, such as packet loss rate and jittering, than ListeningMOS and NetworkMOS. However, as we can see, in Figure 7.17, when the Conversational-CC algorithm is higher than Listening-CC and Network-CC, it leads to higher packet loss rate. Indeed, when the wireless link is overloaded, all congestion control algorithms lead to worst performances. Finally, the Conversational-CC is more suitable for bidirectional communications.

Although the improvements in MOS do not seem to be very large (0.1-0.3) in absolute values, the relative improvements are actually significant. In the practical system (e.g., Microsoft Lync), only few users have MOS values below 3 or above 4. The dynamic

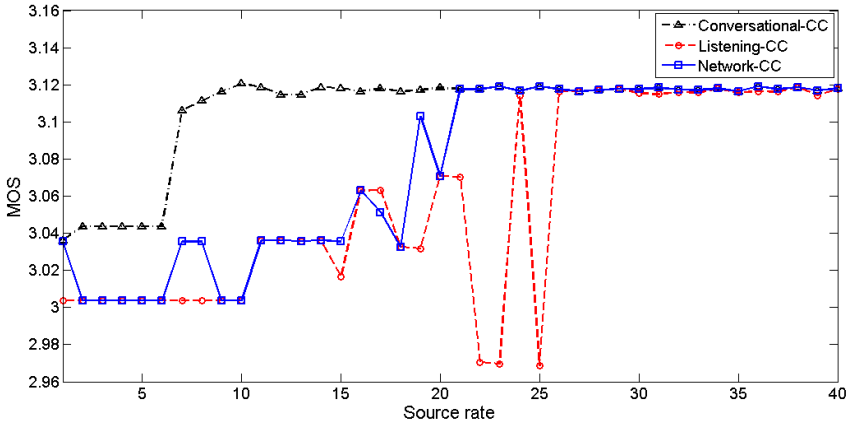

FIGURE 7.16: ConversationalMOS with different source rates in the third scenario.

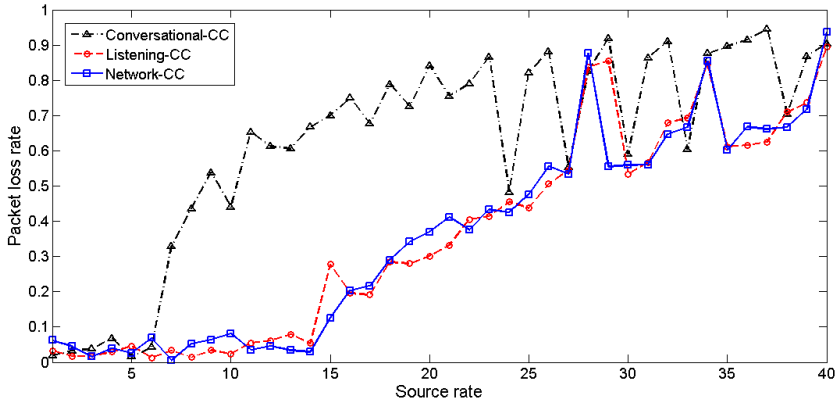

FIGURE 7.17: Packet loss rate depending on the source rate in the third scenario.

range of the MOS values is about 1.0. The region between 3.0 and 4.0 is a quite sensitive interval of MOS for users. Our improvement is about 10% to 30% in the range. Because the sessions in our traces are using the same audio codec and software version, this means that the actual degradation of ListeningMOS is relatively small. However, if we focus on the NetworkMOS, the improvements are significant. As we can see in Figure 7.18, the improvement of MOS-TCP user is about 1 in NetworkMOS.

7.7 Conclusion

We have formulated, in this chapter, the QoE-aware congestion control problem as a POMDP that maximizes the QoE for multimedia users. We have considered a set of generic AIMD-like updating functions for the congestion window. The optimal policy allows the sender to optimize the congestion window updating policy that maximizes the expected discounted QoE. We have also proposed an online learning method to solve

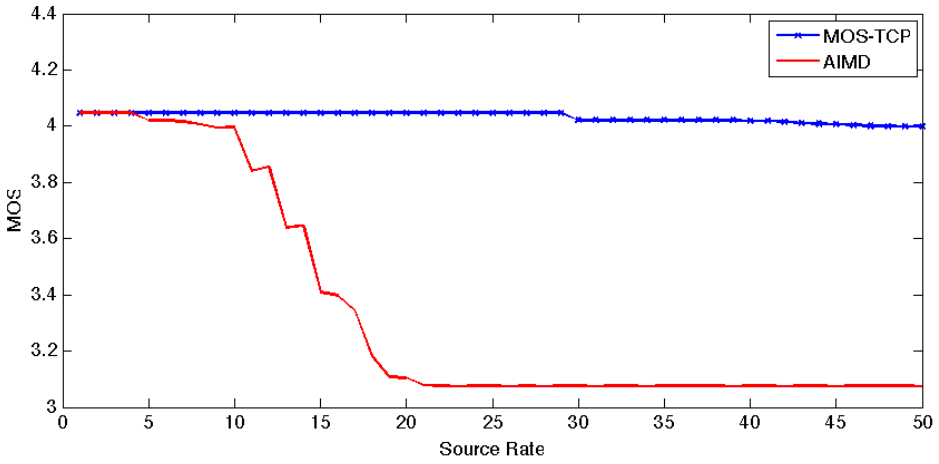

FIGURE 7.18: NetworkMOS for MOS-TCP user and a AIMD user.

the MOS-TCP on-the-fly. Experimental results have shown that the proposed algorithm outperforms other congestion control schemes in terms of QoE. Note that the friendliness of MOS-TCP can be studied similarly to the study of Learning-TCP in Chapter 6, and will be a part of our future work.