Telecommunication Systems 28:2, 211–230, 2005

Efficient De-Jitter Control for Voice Applications over Wireless Ad Hoc Networks

MOUNA BENAISSA, VINCENT LECUIRE and FRANCIS LEPAGE
{mouna.benaissa;vincent.lecuire;francis.lepage}@cran.uhp-nancy.fr
CRAN, CNRS UMR 7039, University of Nancy 1-Henri Poincaré, Faculté des Sciences et Techniques, BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France

ANDRÉ SCHAFF Andre.Schaff@loria.fr
LORIA, CNRS UMR 7503, University of Nancy 1-Henri Poincaré, Faculté des Sciences et Techniques, BP 239, 54506 Vandœuvre-lès-Nancy Cedex, France

Abstract. Voice over IP applications require a playout buffer at the receiver side to smooth network delay variations. Unfortunately, existing algorithms for dynamic playout adjustment designed for wireline networks do not operate correctly in wireless ad hoc networks. These algorithms estimate the end-to-end delay on the set of previous received audio packets. Indeed, such a delay estimation based on past history is not appropriate due to mobility which leads to random changes of the network topology. In this paper, we highlight this delay estimation problem. We show that *route request* AODV control messages provide more accurate delay estimation. Then, we propose a new algorithm for playout delay adjustment based on these control messages. The performance evaluation is performed by simulation using ns-2. We show that this algorithm outperforms existing playout delay adjustment algorithms. Performance criteria are loss late percentage (reliability criterion), averaged playout delay (interactivity criterion) and playout delay variation (stability criterion).

Keywords: wireless ad hoc network, AODV, voice packets, playout delay, jitter control, speech quality

1. Introduction

One of the challenges of transmitting real-time voice on packet networks is how to overcome the variable inter-packet delay -the jitter- encountered as packets move on the transmission path through the network. In order to compensate these variable delays, packets are buffered at the receiver side and their *playout* is delayed for a period of time. Thus, most of the packets will be received before their scheduled *playout* times [Moon et al., 17; Clark et al., 4]. The playout delay adjustment must take into account three constraints. The first one is the interactivity constraint which requires playout delay below a certain value considered to be quite acceptable in human conversation, less than 300 ms but 100 ms is recommended to obtain excellent interactivity [9; Kitawaki and Itoh, 13]. Second, the reliability constraint which requires low packet loss rate, generally less than 5%. Third, the stability constraint requires no large playout delay variation (this constraint is effective when playout delay is adjusted dynamically).

Adaptive strategies adjust dynamically the playout delay through the duration of an audio session. Existing adaptive algorithms designed for wireline infrastructure network operate by adjusting the playout delay either per-talkspurt or per-packet. They try to reduce the tradeoff between loss percentage and averaged playout delay. Playout delay is based on the end-to-end delay estimation computed on a set of previously received audio packet delays. The number of packets used in this computation varies for each algorithm. It can concern packets included in the more recent talkspurt as well as a few thousand previous packets. We observe that these algorithms do not operate correctly in wireless ad hoc networks: when the topology changes, end-to-end delay estimation based on past history (delays of audio packets which arrived by an obsolete route) becomes inappropriate.

This paper highlights this problem and then presents a new algorithm for playout delay adjustment which is designed to operate well in ad hoc network environment. End-to-end delay estimation is based on an ad hoc routing event: the *route request* control message (RREQ) of the AODV routing protocol. Of course, we suggest that AODV is adopted for the deployment of voice applications over ad hoc networks. However, it is not a drawback since it is more appropriate for such applications [Benaissa et al., 2].

The paper is organized as follows: In section 2, we present specific problems encountered when running voice applications over mobile ad hoc networks, compared to wireline networks. In section 3, we give a general overview of existing playout delay adjustment algorithms. In section 4, we study the reasons why these algorithms do not operate correctly. This is verified using a reference audio trace obtained by simulation for two autoregressive estimation based algorithms. In section 5, we show that RREQ-AODV message provides an accurate estimation for end-to-end delay. Then, we describe our algorithm and give its pseudo code. Section 6 provides performance evaluation and comparison results obtained by simulation using ns-2 [Greis, 6; 7]). The performance criteria are loss late percentage (reliability criterion), averaged playout delay (interactivity criterion) and playout delay variation (stability criterion). Section 7 concludes the paper.

Note that results given in this paper are obtained using the ns-2 simulation model described in the appendix.

2. Ad hoc reconfiguration phases: a typical disturbing event for VoIP

In ad hoc networks, mobile nodes communicate with others using multi-hop wireless links. There is no stationary infrastructure such as base stations. Each node in the network also acts as a router to forward data packet to other nodes [Perkins, 18]. There are two approaches used by existing routing protocols in mobile ad hoc networks: the proactive approach such as OLSR [Jaquet et al., 10] and the reactive approach such as AODV [Perkins, 18]. The proactive approach consists of every node emitting hellos messages periodically in order to learn the network topology. Reactive protocols invoke a route determination procedure on demand only.

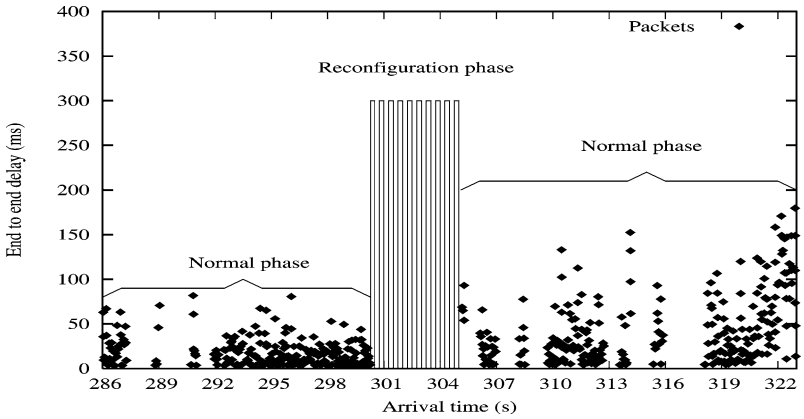

Figure 1. Example of reconfiguration phase.

In [Benaissa et al., 2], we showed that packets audio streaming over mobile ad hoc networks have clearly distinguished communication phases and network reconfiguration phases. During a reconfiguration phase, the audio flow transfer is disrupted because of a delay caused by the routing protocol to establish a new route towards the destination. Packets waiting for a new route are dropped or delayed in intermediate nodes. The receiver identifies this phase when a sudden interruption occurs on packets arriving, generally followed by series of packets arriving with high end-to-end delays. Such an interruption disturbs the played out audio speech at the receiver side during a significant time, generally of some seconds.

An example of a reconfiguration phase is illustrated in figure 1. Each point on the figure represents a packet arriving at the time indicated by its x-axis value, having experienced an end-to-end delay equal to its y-axis value. Two communication phases span two intervals: [286 s–300.29 s] and [305.10 s–323.81 s] and a reconfiguration phase occurs at 300.29 s lasting 4.81 s on this example.

After a reconfiguration phase, the route for the new communication phase can present different or similar network conditions compared to the previous one, due to different traffic load and hop number on the route. We define *strong* to be a reconfiguration which leads to different network conditions and *light* to be a reconfiguration which leads to similar network conditions. Reconfiguration phases occur more frequently when node mobility increases. This is a problematic event for voice applications which cannot be avoided. Results from [Benaissa et al., 2] show that OLSR leads more often than AODV to reconfiguration phases. Thus, AODV is more adapted to the deployment of such application over ad hoc networks, although AODV is more sensitive to increasing load traffic [Jaquet and Viennot, 11]. In the following, we give an overview of existing algorithms of playout delay adjustment and show that they cannot operate correctly in ad hoc networks since they do not take into account reconfiguration events. This will motivate the design of a new algorithm.

3. Related work on playout delay algorithms

Extensive research work has been done in the area of adaptive playout mechanisms for voice over IP. These algorithms differ generally in the way they estimate the end-to-end delay. They use collected delay measures of the most recently received audio packets. These measures can involve one packet [Ramjee et al., 21], L packets [Moon et al., 17; Leon and Sreenan, 15; Sreenan et al., 23; Agrawal et al., 1; Pinto and Christensen, 20; Liang et al., 16], M talkspurts [Ramos et al., 22] or all received packets [Fujimoto et al., 5].

3.1. Reference algorithms

The algorithms proposed in [Ramjee et al., 21] are the reference playout delay algorithms which are implemented in existing audio tools, and to which all proposed algorithms are compared. The authors propose four algorithms where the playout delay estimation is updated based on the most recent delay observation using an autoregressive linear filter. All the four algorithms compute averaged estimate of the end-to-end delay $\hat{d}_i$ and averaged delay variation $\hat{v}_i$ for each received packet i. These values are computed for each packet but the playout delay is changed only at the beginning of a talkspurt. Let n_i be the end-to-end delay achieved by the packet i ($n_i = a_i - t_i$), where t_i and a_i are the sender and receiver time, respectively.

If packet i is the first of a talkspurt, its playout time, noted p_i is computed as

$$p_i = t_i + \hat{d}_i + 4\hat{v}_i. \tag{1}$$

For any subsequent packet j in a talkspurt, the playout time, p_j is computed as an offset from the point in time when the first packet is the talkspurt was played out, given by

$$p_j = p_i + (t_j - t_i). \tag{2}$$

The algorithms that perform better in [Ramjee et al., 21] are 1 and 4. In this paper, we refer to these algorithms as *mean delay* and *spike* algorithms, respectively. They differ in the way they compute $\hat{d}_i$.

The *mean delay* algorithm computes $\hat{d}_i$ and $\hat{v}_i$ autoregressively as given in the following equations:

$$\hat{d}_i = \alpha \hat{d}_{i-1} + (1 - \alpha) n_i, \tag{3}$$

$$\hat{v}_i = \alpha \hat{v}_{i-1} + (1 - \alpha) \left| \hat{d}_i - n_i \right|. \tag{4}$$

The weighing factor of linear filter, α controls the rate convergence of this algorithm. When $(1 - \alpha)$ is very small, the playout delay is maintained at a steady value. The NetVot audio tool uses $\alpha = 0.998002$.

The *spike* algorithm has two modes of operation, depending on whether or not a *spike* has been detected. A *spike* is defined as a sudden and large increase in the network delay which is followed by a series of packets arriving simultaneously. The estimated delay $\hat{d}_i$ depends on the current mode. In normal mode, it operates like the *mean delay*

algorithm but the α weighting factor is set to 0.875. In impulse mode, which begins upon the detection of a *spike*, $\hat{d}_i$ is given by the following equation to catch up quickly with the delay spike:

$$\hat{d}_i = \hat{d}_{i-1} + (n_i - n_{i-1}). \tag{5}$$

The beginning of a *spike* is detected if the delay between consecutive packets is large enough, i.e. higher than a threshold. In a similar way, the end of *spike* is detected when the variation in the delay of the three most recent packets become small enough, i.e. lower than another threshold. See [Ramjee et al., 21] for details. Note that when a *spike* spans multiple talkspurts, it is possible to react quickly to the delay *spike*. However, when a *spike* is fully contained within a talkspurt, the next opportunity to change the playout delay occurs after the delay *spike* terminates.

A variant of the *spike* algorithm is proposed in [Kansal and Karandikar, 12]. The weighting factor α is dynamically adjusted in equation (3) throughout the duration of the audio session. Note that the *mean delay* and *spike* algorithms are the reference playout delay algorithms which are implemented in existing audio tools, and to which all proposed algorithms are compared.

3.2. Playout delay estimation problem in ad hoc networks

The common issue is that *mean delay* and *spike* algorithms as well as other existing algorithms use previous audio packet delay observations to estimate the future delay. We think that this is not appropriate in wireless ad hoc network due to the unpredictable events which happen in such an environment. When the topology changes, we must consider a new network whose conditions are not yet known. The delay estimation in this new network is not appropriate when it is based on delays observed on the old topology. Of course, such an assumption needs to be checked.

Accordingly, we study the behavior of *mean delay* and *spike* algorithms on a reference audio trace no. 1. Let us recall that these algorithms use only the most recent delay observation to update the end-to-end delay estimation. Clearly, long past statistics based algorithms are not appropriate and thus are not tested here.

Our trace no. 1 has been obtained with the *ns*-2 simulator and is shown in figure 2. The audio session runs from 110 s to 600 s using the PCM audio codec and the AODV routing protocol (see the appendix for details of simulation). Each point in this figure represents a packet arriving at the time indicated by its x-axis value, having achieved an end-to-end delay equal to its y-axis value.

The estimated playout delays when running the *mean delay* and *spike* algorithms are plotted with a dashed line in figure 2.

It shows that the *mean delay* algorithm reacts slowly to the delay variance and gives a stable playout delay. This behavior is due to the choice of the α value in equation (3) ($\alpha = 0.998002$) and the absence of *spike* detection. This factor controls the weigh of delay measures in the estimated playout delay. When delays increase or decrease with large values, the estimated playout delay does not change significantly.

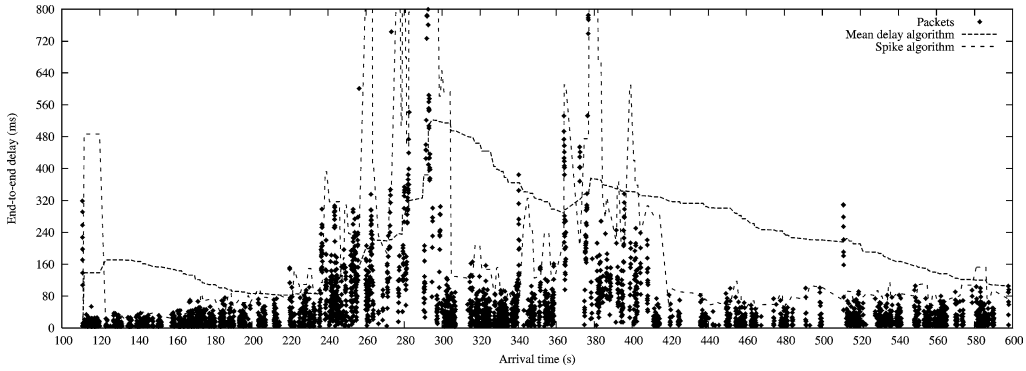

Figure 2. Mean delay algorithm vs spike algorithm applied to trace no. 1.

The estimated playout delay when using the *spike* algorithm shows that this algorithm is reactive to the delay variance due to the choice of the α value ($\alpha = 0.8752$) and *spike* processing. The playout delay follows delay tendency but it is not stable. This appears, for example, at 256.34 s where playout delay is adjusted from 235 ms to 855 ms after receiving an audio packet with an achieved delay of 600 ms. Then, at the next talkspurt, the playout delay decreases from 855 ms to 369 ms. This strong playout delay variance is not desirable in human conversation. The *spike* algorithm switches several times between normal and impulse mode: it considers a large increase in delays due to reconfiguration event as if it was a *spike*.

3.3. Synthesis

From the study presented in section 3.2, we conclude that existing playout delay adjustment algorithms which are based only on past delay history and designed for wireline large scale networks lose their effectiveness in ad hoc networks. Topology changes lead to significant and sudden end-to-end delay variance. When estimated playout delay considers a long past delay history as is the case with the *mean delay* algorithm, the playout delay adjustment is performed too slowly to react correctly to such delay variation. If α is set to a smaller value as is the case with the *spike* algorithm, the playout delay adjustment becomes more accurate but that can lead to degraded stability. We remark that using the *spike* algorithm, the large delays observed just after a reconfiguration phase can affect significantly the playout delay estimation while they do not provide a correct indication on new network conditions. From these observations, we propose a new playout delay adjustment algorithm in the next section, designed for voice applications over ad hoc networks.

4. RREQ-AODV algorithm

Existing strategies used to estimate future delay are not efficient in mobile ad hoc networks because they are based on past delay measures, as shown in the previous section.

Our approach is different: it uses AODV routing information to predict network delays. Particularly, the RREQ message generated by AODV protocol during a route discovery process provides pertinent information about future voice packet delay.

4.1. Delay indication using RREQ-AODV messages

The AODV routing protocol generates control messages to establish a new route towards the destination. The source node initializes a route discovery process just before sending data. It is achieved using an RREQ message which is broadcasted across the network. When the destination node receives the first RREQ message, it carries back the route in an RREP message to the source and ignores the next received RREQ messages for this route. The established route is the network path built by the RREQ message. Thus, RREQ messages and voice packets use the same path from the source to the destination. This path is known by the receiver because the AODV source uses the ones built by the first RREQ message which reaches the destination. So, the end-to-end delay achieved by the RREQ message presents for the receiver a pertinent indicator about the delay for voice packets to be received through this route.

At the receiver side, this indicator is available and updated dynamically before receiving voice packets from each communication phase. A new RREQ message is received during the reconfiguration phase of a new route discovery cycle, since the source maintains at most one route per destination. This can happen in several cases:

- *The beginning of an audio session*. As with any reactive routing protocol, AODV initiates a route discovery process to start an audio session. This involves the sending of the first RREQ message which provides to the receiver a delay indication from the beginning of the session.
- *Mobility*. When a node moves from an active audio path, a new reconfiguration phase begins. The node which detects the link failure sends a route error (ERR) message to the source. If the audio source node still desires the route, it reinitiates the route discovery process. Consequently, the destination receives a new RREQ message which provides a new delay indication appropriate to the new topology.
- *Long silence period*. AODV maintains a timer-based states in each node, about the usage of individual routing table entries. A routing table entry expires if not used recently. Thus, the audio route expires during a silence period which is longer than a route expiry time (*active_route_timeout*). Due to this long silence period, the source needs to initiate a route discovery process at the next talkspurt. This provides a new RREQ message to the audio destination and updates the delay indicator.
- *High traffic load*. AODV maintains topology information via *HELLO* messages. If a node does not receive any *HELLO* message from its known neighbor, the link is considered broken. This can occur when network traffic load is high, even if there is no mobility. So, a new route discovery process starts and presents a new RREQ message to the receiver. This message provides a new delay indication appropriate to the network load conditions.

In our work, we assume that the receiver is always notified about route changing by RREQ messages. This is achieved in the following way. Firstly, the procedure of RREQ message broadcasting is modified. In the initial procedure, the intermediate node receiving the RREQ message may send a RREP message if it has a route to the destination and stop broadcasting. When this happens, the audio destination does not receive any *RREQ* message and can not detect this reconfiguration. In the modified procedure, the intermediate node having a route to the destination unicasts the received RREQ message to the final destination and does not reply any RREP message to the source. So, broadcasting is stopped and the receiver is notified about this reconfiguration. Secondly, the local repair procedure is not used. Using this procedure, AODV attempts to repair locally a failed link instead of informing the source immediately in order to initiate a new route discovery process. Note that in 50 node networks, the use of a local repair procedure does not have any significant performance advantage while it is recommended in larger networks to increase scalability [Lee et al., 14].

4.2. RREQ-AODV algorithm description

We attach greater importance to the audio interactivity constraint than to the reliability constraint. Therefore, the proposed algorithm aims to avoid playing out late arrival voice packets to satisfy reliability if interactivity becomes impossible. Indeed, a packet arriving with delay larger than 260 ms is considered lost by the audio application in regard to the interactivity constraint, thus there is no additional advantage to increasing the playout delay to absolutely save such packets. A second objective is to get a stable and accurate playout delay. Our algorithm performs playout delay adjustments in two cases at the beginning of a new talkspurt (as existing algorithms), and at the beginning of a new communication phase. Indeed, since a reconfiguration phase causes disruption of the audio speech over a long period, it is advantageous to adjust the playout delay at this time in order to limit disruption to the user.

4.2.1. Playout delay estimation

The playout delay estimation is based on delay indication provided by RREQ messages. Let $Drreq_n$ be the end-to-end delay achieved by the RREQ message received during the nth reconfiguration phase. The playout delay estimation $\hat{d}_{n,k}$ for packets of the kth talkspurt belonging to the nth communication phase is computed as follows:

$$\hat{d}_{n,k} = Drreq_n + \beta_{n,k}, \tag{6}$$

where $\beta_{n,k}$ is a safety factor, added to ensure that the estimated end-to-end delay is greater than the actual network delay.

4.2.2. Playout delay adjustment

The delay indication $Drreq_n$ is updated at every RREQ message arrival and used at the beginning of a new communication phase or a new talkspurt: if arriving packet i is the first packet of talkspurt k or a communication phase n, the playout delay is computed as

given in equation (6). This delay is preserved for each subsequent packet j. The playout times p_i and p_j are computed as below:

$$p_i = t_i + \hat{d}_{n,k}, \tag{7}$$

$$p_j = p_i + (t_j - t_i), \tag{8}$$

where t_i and t_j are respectively times at which packets i and j are generated at the sender.

In order to get more accurate playout delay estimation, it is interesting to adjust dynamically the safety factor $\beta_{n,k}$ instead of setting it to a constant value. In the following section, we discuss in which manner it is appropriate to adjust dynamically the safety factor $\beta_{n,k}$ used in equation (6).

4.3. Safety factor adjustment

Our algorithm distinguishes two events for adaptation of the safety factor $\beta_{n,k}$: the beginning of a new communication phase (a RREQ message is received) and the beginning of a new talkspurt (a new talkspurt begins in the same communication phase while no new RREQ message is received). The adaptation of $\beta_{n,k}$ is performed in the following way for each case:

– *RREQ message is received.* The reception of a new RREQ message indicates that a new network topology is established. Then, the new delay indication $Drreq_n$ is updated in equation (6). To identify the type of the occurred reconfiguration, the algorithm computes the difference δ between the current delay indication $Drreq_n$ and the previous one $Drreq_{n-1}$ as follows:

$$\delta = |Drreq_n - Drreq_{n-1}|. \tag{9}$$

According to a certain threshold *threshold_req*, $\beta_{n,k}$ is adjusted as below:

1. δ *is large enough* ($\delta >$ *threshold_req*). This case indicates that the network conditions of the new topology have changed significantly; it was a strong reconfiguration. For the new communication phase, the delay estimation is based on the new delay indication $Drreq_n$ and $\beta_{n,k}$ is reset to its initial value $\beta_{\min}$:

$$\begin{aligned} &\text{if} \quad |Drreq_n - Drreq_{n-1}| > threshold\_req \\ &\text{then} \quad \beta_{n,k} \leftarrow \beta_{\min}. \end{aligned} \tag{10}$$

2. δ *is small* ($\delta \leqslant$ *threshold_req*). In this case, we consider that the network conditions are similar between the old and the new topology, thus it was a light reconfiguration. The safety factor $\beta_{n,k}$ preserves its previous value $\beta_{n-1,k}$:

$$\begin{aligned} &\text{if} \quad |Drreq_n - Drreq_{n-1}| \leqslant threshold\_req \\ &\text{then} \quad \beta_{n,k} \leftarrow \beta_{n-1,k}. \end{aligned} \tag{11}$$

– *No RREQ message is received while a new talkspurt begins.* The absence of RREQ message indicates that no changes happen on the ad hoc network topology. In this

case, the delay estimation for this talkspurt can be based on the recent delay past history. We propose to adapt $\beta_{n,k}$ as a function of loss percentage q_{k-1} achieved on the more recent talkspurt $(k-1)$:

$$\beta_{n,k} = f(\beta_{n,k-1}, q_{k-1}). \tag{12}$$

In order to keep a certain stability of the estimated playout delay, the adjustment of $\beta_{n,k}$ in equation (12) is performed in a gradual way as follows:

1. *No loss observed on the previous talkspurt* ($q_{k-1} = 0$). This means that the safety factor is large and it can be decreased to improve interactivity without degrading reliability. Then, $\beta_{n,k}$ is decreased by a factor r:

$$\text{if } q_{k-1} = 0 \quad \text{then } \beta_{n,k} = f(\beta_{n,k-1}, 0) = (1-r) \times \beta_{n,k-1}. \tag{13}$$

2. *Loss percentage is less or equal than the user tolerable limit* q_{ref} ($q_{k-1} \leqslant q_{\text{ref}}\%$). This means that the algorithm gets a good reliability but there is no margin on the safety factor to improve interactivity. Then, $\beta_{n,k}$ preserves its previous value for the next talkspurt in order to maintain the same level of reliability and interactivity:

$$\text{if } q_{k-1} \leqslant q_{\text{ref}} \quad \text{then } \beta_{n,k} = \beta_{n,k-1}. \tag{14}$$

3. *Loss percentage exceeds the user tolerable limit* ($q_{k-1} > q_{\text{ref}}\%$). This means that the safety factor is small. It must be enlarged at the next talkspurt to increase reliability; $\beta_{n,k}$ is increased by a multiple of factor r as function of observed loss percentage q_{k-1}, as follows:

$$\begin{array}{ll} \text{if } q_{\text{ref}} < q_{k-1} < 10\% & \text{then } \beta_{n,k} = (1+2r) \times \beta_{n,k-1}, \\ \text{if } 10 < q_{k-1} < 20\% & \text{then } \beta_{n,k} = (1+4r) \times \beta_{n,k-1}, \\ \text{if } 20 < q_{k-1} < 30\% & \text{then } \beta_{n,k} = (1+6r) \times \beta_{n,k-1}, \\ \text{if } q_{k-1} > 30\% & \text{then } \beta_{n,k} = 2 \times \beta_{n,k-1}. \end{array} \tag{15}$$

The equation (15) is defined as to avoid adjusting $\beta_{n,k}$ with strong value from talkspurt to another. Indeed, we aim to keep a certain stability of the playout delay. For larger value of r, the playout delay adjustment is larger. Note that q_{ref} is set to 3% in our work.

To keep acceptable interactivity (end-to-end delay less than 300 ms including delay required to collect audio samples), the adjustment of $\beta_{n,k}$ is set to be in the interval of $[\beta_{\min}, \beta_{\max}]$. Assuming that voice packet is a sample of 40 ms, playout delay must avoid exceeding 260 ms as much as possible. We set the minimum safety value $\beta_{\min}$ to 40 ms and the maximum value $\beta_{\max}$ to 200. $\beta_{\min}$ value corresponds to delay required to collect 4 voice samples of 10 ms used to gather one voice packet in our simulations. $\beta_{\max}$ value is chosen considering the averaged delay achieved by RREQ messages (60 ms) on reference trace no. 1 (from equation (6), we obtain $\beta_{\max} = 260 - 60 = 200$ ms where 260 ms is the tolerable limit of playout delay to preserve interactivity). The value of parameters *threshold_req* and r must be chosen as to give the better tradeoff between interactivity, reliability and stability. We applied

Table 1
Parameters values used in algorithm RREQ-AODV with adapted safety factor.

Parameter	$\beta_{\min}$	$\beta_{\max}$	*threshold_req*	q_{ref}	r
Value	40	200	80	3%	0.05

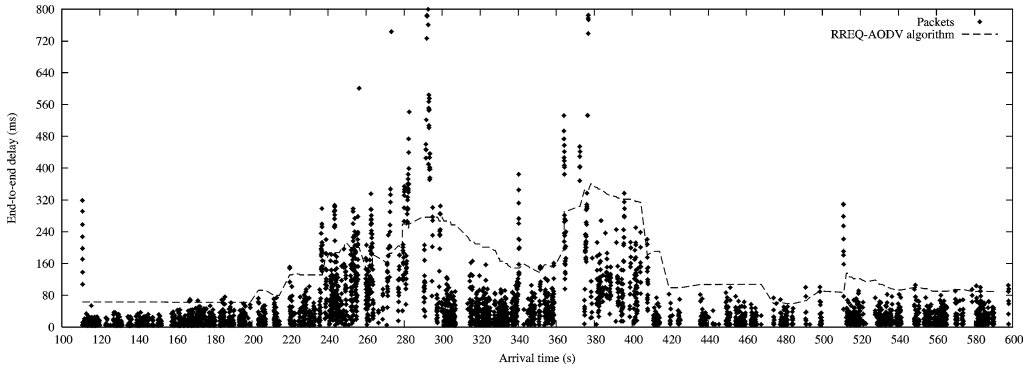

Figure 3. RREQ-AODV algorithm applied to trace no. 1 (Comparison can be made to figure 2).

RREQ-AODV algorithm on different traces varying *threshold_req* from 20 to 100 and r from 2.5 to 15. The better performances are obtained for $r = 5$ and *threshold_req* $= 80$. Table 1 gives the parameter values used in AODV-RREQ algorithm. Figure 3 presents the playout delay adjustment obtained when AODV-RREQ algorithm is applied on trace no. 1.

The pseudo-code of RREQ-AODV algorithm is given in algorithm 1.

5. Performance evaluation

In this section, we evaluate and compare the RREQ-AODV algorithm performances to that of the *mean delay* and the *spike* algorithms. The results shown in this section are obtained from six audio reference traces: the trace no. 1 showed in section 4.1 and five other traces described in this section.

Algorithm 1 (RREQ-AODV algorithm).

Data: q_{ref}; // tolerable loss percentage
$\beta_{\min}$; $\beta_{\max}$; *threshold_req*; r;
Result: Estimated playout delay $\hat{d}_i$ of packet i
Initialization: $Drreq_{n-1} \leftarrow 0$;
$N_{\text{talkspurt}} \leftarrow 0$; // Talkspurt during which RREQ message is received
$\beta_{\text{old}} \leftarrow \beta_{\min}$;
$Nb_k \leftarrow q_{\text{ref}}$; // Number of lost packet during the kth talkspurt
$N_k \leftarrow 100$; // Number of received packets during the kth talkspurt

```
begin
    if (RREQ message is received during the kth talkspurt) then
        Drreq_n ← Recv_timestamp_n − Send_timestamp_n;
        N_talkspurt ← k;
        if abs(Drreq_n − Drreq_{n−1}) > threshold_req then
            β_current ← β_min;
        else
            β_current ← β_old;
        Drreq_{n−1} ← Drreq_n;
        β_old ← β_current;
    if (audio packet i is received) then
        d_i ← Recv_timestamp_i − Send_timestamp_i;
        if (packet i is the first packet of the kth talkspurt) and (k = N_talkspurt + 2) then
            Nb_{k−1} ← Nb_k; N_{k−1} ← N_k; Nb_k ← 0; N_{k−1} ← 0;
            q_{k−1} ← (Nb_{k−1} × 100)/N_{k−1};
            if q_{k−1} = 0 then β_current = max((1 − r) × β_old, β_min);
            if 0 < q_{k−1} ⩽ q_ref then β_current = β_old;
            if q_ref < q_{k−1} ⩽ 10% then β_current = min(β_max, (1 + 2r) × β_old);
            if 10% < q_{k−1} ⩽ 20% then β_current = min(β_max, (1 + 4r) × β_old);
            if 20% < q_{k−1} ⩽ 30% then β_current = min(β_max, (1 + 6r) × β_old);
            if q_{k−1} > 30% then β_current = min(β_max, 2 × β_old);
        d̂_i ← Drreq_n + β_current;
        if d_i > d̂_i then
            Nb_k + +; N_k + +;
end
```

5.1. Performance metrics

To measure the obtained speech quality Q at the receiver when applying a playout delay algorithm, we take into account three criteria: Interactivity (averaged playout delay I), reliability (percentage of loss due to late arrivals F) and stability (averaged playout delay jitter S). The E-model given in [9] predicts the subjective quality Q of a telephone call based on its characterizing transmission parameters. It combines impairment caused by these parameters into a single rating Q. According to the ITU-T recommendations, the rating value range of Q corresponds to a speech transmission category, as follows: best for range of [90, 100], high for range of [80, 90], medium for range of [70, 80], low for range of [60, 70], poor for range of [0, 60]. The rating Q is given by

$$Q = Q_0 - E, \tag{16}$$

where Q_0 takes into account the effects of noise. The default value of Q_0 is 94.2. E combines impairment of different transmission parameters. In our work, E groups the impairment relative to interactivity $E(I)$, the impairment relative to reliability $E(F)$ and the impairment relative to stability $E(S)$. The function $Q(I, F, S) : \mathcal{R}^+ \times \mathcal{R}^+ \times \mathcal{R}^+ \rightarrow$

Table 2
Parameter values of function $E(I)$.

Parameters	γ_1	β	γ_2	δ	b_1	b_2	b_3
Values	0.001	0.02	0.01	32	18.89	185	17.1

[0, 100] is given by

$$Q(I, F, S) = 94.2 - E(I) - E(F) - E(S). \tag{17}$$

Let p_i be the playout delay of packet i, N be the total sent voice packets, A be the total received voice packets, L be the total played out voice packets during the audio session. A packet i, sent at time t_i and received at time a_i, is played out if it arrives before its playout time tp_i, i.e. $a_i \leqslant tp_i$ (where $tp_i = t_i + p_i$).

5.1.1. Interactivity metric

The averaged playout delay I provides an indication of the interactivity level. I is given by

$$I = \frac{1}{L}\sum_{i=1}^{L} p_i. \tag{18}$$

In human conversation, end-to-end delay must not exceed 110 ms for a good interactivity but tolerates degraded speech quality for end-to-end delay between 110 ms and 260 ms (in our work 40 ms are required to collect samples of one audio packet). When end-to-end delay exceeds 260 ms, speech quality is poor. Considering these bounds, $E(I)$ is given by

$$E(I) = \begin{cases} \gamma_1 I & \text{for } I \leqslant 110, \\ b_1 \tanh\big(\beta(I - b_2)\big) + b_3 & \text{for } 110 < I \leqslant 260, \\ \gamma_2 I + \delta & \text{for } I > 260, \end{cases} \tag{19}$$

where parameters γ_1, β, γ_2, δ, b_1, b_2 and b_3 are constants selected to ensure the continuity of function $E(I)$. Values of these parameters are given in table 2 and obtained as recommended in [Boutremans and Le Boudec, 3].

5.1.2. Reliability metric

The loss late percentage F indicates the reliability level. F is given by

$$F = \frac{A - L}{A}100. \tag{20}$$

Independently of the codec in use, $E(F)$ is given by

$$E(F) = 34.3 \ln(1 + 12.8F). \tag{21}$$

When the percentage of loss is less than 3%, the speech quality is good but tolerates degraded audio quality for loss percentage between 3% and 15%. When loss percentage exceeds 15%, speech quality is poor.

5.1.3. Stability metric

The jitter S on the playout delay during the audio session provides indication about the stability of the playout delay. S is given by

$$S = \frac{\sum_{i=2}^{L} |p_i - p_{i-1}|}{L - 1}. \tag{22}$$

In our work, $E(S)$ is given as follows:

$$E(S) = 2S. \tag{23}$$

As S increases, $E(S)$ increases and playout delay is less stable.

5.2. Reference traces

To evaluate the performance of our algorithm and compare them with those of *mean delay* and *spike* algorithms, we selected six reference traces which characterize different ad hoc network conditions in terms of load traffic and mobility speed. For each trace, we record sending and receiving time of all audio packets and all RREQ messages transfered from the audio source to the audio destination. Principals characteristics of these traces are:

- Reference trace 1, 2 and 3. They present normal node mobility (1 m/s to 2 m/s) and normal load traffic conditions. We consider that network conditions on these three traces are favorable to the deployment of VoIP applications.
- Reference trace 4. This trace presents high mobility (8 m/s) and high load traffic (especially from 300 s to 580 s). We consider that network conditions on this trace are difficult for the deployment of VoIP applications. End-to-end delays on this trace increase a lot due to high load traffic and particularly after reconfiguration phases. This trace is used to study the behavior of the playout delay algorithm in difficult situations.
- Reference trace 5. This trace presents light load traffic and high mobility (6 m/s). End-to-end delays are very small during communication phases and very high after reconfiguration phases. When route reconfiguration occurs, packets waiting for a new route arrive with very high delays because network queues are not loaded and thus these packets are not discarded. Let us recall that routing messages are given higher priority than data packets in the interface queue. This trace is used to show particularly the effect of mobility on packet end-to-end delays. Note that delays increase a lot after reconfiguration phases.
- Reference trace 6. This trace presents normal load traffic conditions without any mobility. We consider that this trace is appropriate for *mean delay* and *spike* algorithms.

Table 3 gives a summary of the characteristics of these six reference traces.

Table 3
A summary of the characteristics of the six reference audio traces.

Reference trace	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5	Trace 6
Duration (seconds)	490	590	600	700	600	600
Number of talkspurts	174	215	214	223	170	245
Mobility speed (m/s)	2	1	1	8	6	0
Network loss (%)	10.4	4.7	7.6	15.9	26.1	0

Table 4
Result obtained on the six audio traces.

Trace	Algorithm	I	F	S	$Q(I, F, S)$	Quality
1	Mean delay	241.46	6.01	0.32	41.57	poor
	Spike	194.29	6.72	3.63	45.08	poor
	RREQ-AODV	131.04	5.33	0.32	73.91	medium
2	Mean delay	143.27	4.40	0.21	74.23	medium
	Spike	72.66	6.74	1.45	69.89	low
	RREQ-AODV	86.31	3.45	0.24	81.09	good
3	Mean delay	181.35	5.30	0.21	60.29	low
	Spike	117.08	8.73	1.88	64.16	low
	RREQ-AODV	124.66	6.43	0.34	71.65	medium
4	Mean delay	1467.28	4.40	2.43	27.33	poor
	Spike	361.42	13.50	11.21	1.74	poor
	RREQ-AODV	148.80	20.91	1.83	40.48	poor
5	Mean delay	958.15	5.35	2.74	29.24	poor
	Spike	208.63	9.53	9.01	23.41	poor
	RREQ-AODV	84.91	6.74	0.28	72.22	medium
6	Mean delay	44.32	3.8	0.04	80.48	good
	Spike	37.55	6.70	0.66	71.60	medium
	RREQ-AODV	61.91	1.3	0.19	88.53	good

5.3. Performance comparison

In this section, we evaluate and compare the RREQ-AODV algorithm to *mean delay* and *spike* algorithms on the six reference audio traces. For each trace, we give a table summarizing the obtained results: I, F, S, $Q(I, 0, 0)$, $Q(0, F, 0)$, $Q(I, F, 0)$ and $Q(I, F, S)$.

5.3.1. Trace 1,2,3: normal mobility with normal load conditions

Results obtained on trace 1, 2 and 3, given in table 4, show that the RREQ-AODV algorithm obtains the better tradeoff between interactivity, reliability and stability.

Results for trace 1 show that the RREQ-AODV algorithm obtains medium speech quality while the other algorithms give a poor speech quality. Impairment relative to reliability is approximately the same with the three algorithms ($70 < Q(0, F, 0) < 80$) while impairment relative to interactivity and stability are different. The *mean delay*

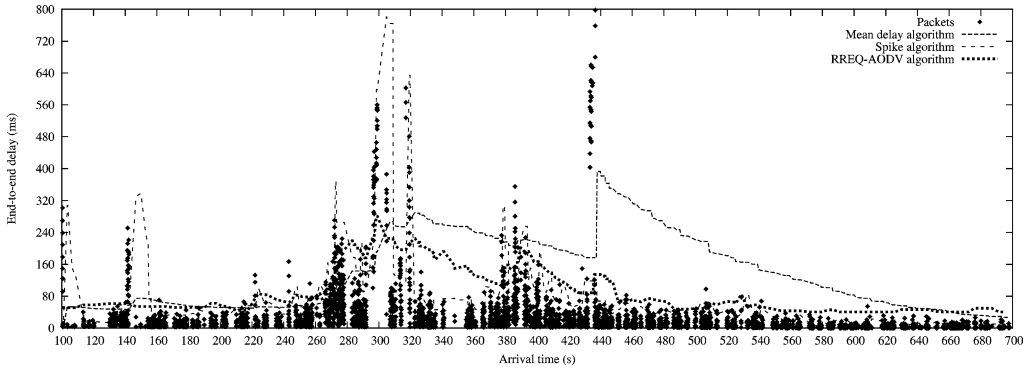

Figure 4. Mean delay algorithm, spike algorithm and RREQ-AODV applied to trace no. 2.

algorithm preserves a good stability but obtains low audio interactivity ($60 < Q(I, 0, 0) < 70$). The *spike* algorithm provides medium audio interactivity ($70 < Q(I, 0, 0) < 80$) but does not give good stability ($S = 3.63$). By comparison, algorithm RREQ-AODV gives excellent interactivity and good stability ($S = 0.32$). When observing results in detail, we notice that the RREQ-AODV algorithm follows more suitably the delays evolution than the others. For example, when delays increase at 230 s, the playout delay increases with a suitable value using the RREQ-AODV algorithm while it oscillates with the *spike* algorithm and increases strongly with the *mean delay* algorithm.

Results for trace 2, plotted in figure 4, show that we obtain good speech quality when using algorithm RREQ-AODV, medium quality with the *mean delay* algorithm and low quality with the *spike* algorithm. We remark that impairment is clearly related to reliability and stability ($90 < Q(I, 0, 0) < 100$). The *spike* algorithm loses more packets and gives less stable playout delay than the others ($F = 6.74\%$ and $S = 1.45$). This is due to the fact that this algorithm reacts late to delay variance and often switches to *spike* mode. The RREQ-AODV algorithm gives better interactivity than the *mean delay* algorithm and thus leads to good speech quality ($Q(I, F, S) = 81.09$).

On trace 3, where average delay and jitter are higher than on previous traces, we obtain medium audio quality when applying the RREQ-AODV algorithm while the others lead to low audio quality. Compared to results obtained with the *mean delay* algorithm, the other algorithms provide excellent interactivity ($90 < Q(I, 0, 0) < 100$) but they lose more audio packets, especially when applying the *spike* algorithm. In addition, the *spike* algorithm does not give a good stability ($S = 1.88$). Impairment relative to interactivity is more important with the *mean delay* algorithm ($Q(I, 0, 0) < 80$). Observing results in more detail, we see that the playout delay obtained with the RREQ-AODV algorithm does not increase strongly after a large delay variance as is observed with the *spike* algorithm, while playout delay obtained with the *mean delay* algorithm remains high for a long time after this event.

5.3.2. Trace 4: high mobility with high traffic load
Network conditions on trace 4 are difficult for the deployment of VoIP applications, due to both high mobility and traffic load. The three algorithms lead to poor speech quality. The *mean delay* algorithm outperforms others when considering reliability ($70 < Q(0, F, 0) < 80$) but it obtains poor interactivity ($Q(I, 0, 0) < 60$). The averaged playout delay obtained with this algorithm is 10 times higher than with the RREQ-AODV algorithm. Indeed, the RREQ-AODV algorithm provides good interactivity but at the cost of poor reliability ($F = 20.91\%$). The *spike* algorithm leads to poor interactivity ($Q(I, 0, 0) < 60$), poor reliability ($Q(0, F, 0) < 60$) and very high stability factor ($S = 11.21$ ms). When observing results in more detail, we note that most lost packets are those arriving with great delays, which are not useful to the VoIP application due to the interactivity constraint. These packets are played out when using *mean delay* algorithm which leads to an excessive playout delay of 1450 ms.

In difficult network conditions, playout delay adjustment algorithms can not give a good tradeoff between interactivity and reliability. Of course, the use of some mechanisms for the quality of service may improve the results.

5.3.3. Trace 5: high mobility with light traffic load
On trace 5, which presents high mobility (6 m/s), the RREQ-AODV algorithm provides medium speech quality and then outperforms other algorithms which lead to poor speech quality. Contrary to the results observed on the previous traces, the *mean delay* algorithm and the *spike* algorithm do not give good stability (high stability factor $S = 2.74$). This is due to the fact that end-to-end delays on this trace are very short during communication phases while they are very long during reconfiguration phases. These long delays are considered when computing playout delay estimation using the *mean delay* algorithm and the *spike* algorithm while they do not give an appropriate indication about future delays. In this case, the *mean delay* algorithm gives medium reliability ($70 < Q(0, F, 0) < 80$) at the cost of poor interactivity ($Q(I, 0, 0) < 60$). The *spike* algorithm gives poor quality when considering both criteria of interactivity and reliability. However, the RREQ-AODV algorithm leads to more stable playout delay ($S = 0.28$) and excellent interactivity ($Q(I, 0, 0) = 94.12$). This is due to the fact that this algorithm uses an appropriate delay indication based on RREQ message delays. However, it gives a degraded reliability ($70 < Q(0, F, 0) < 80$). A more careful analysis of the results reveals that most of the lost packets are those arriving with great delays, much higher to the bound of interactivity. Thus, the algorithm RREQ-AODV does not increase playout delay because this does not improve the speech quality (cf. figure 5).

These results confirm that algorithms which are based on delay past history are not appropriate to adjust playout delay in the presence of mobility even with light load traffic. In these conditions, the RREQ-AODV algorithm reacts correctly.

5.3.4. Trace 6: no mobility with normal load conditions
On trace 6, there is no mobility. We suppose then that it presents good operating conditions for the *mean delay* algorithm and the *spike* algorithm. Results show that the *mean*

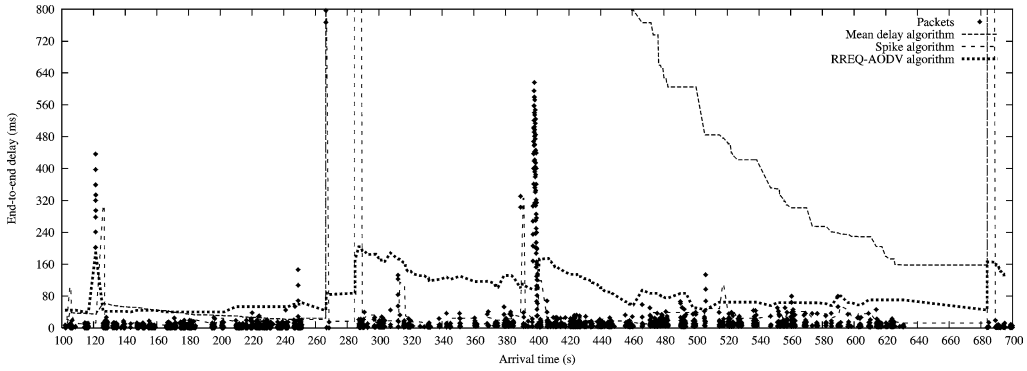

Figure 5. Mean delay algorithm, spike algorithm and RREQ-AODV applied to trace no. 5.

delay algorithm and RREQ-AODV lead to good speech quality and the *spike* algorithm leads to medium speech quality. All algorithms provide excellent interactivity and stability but lead to different reliability levels. When observing results in more detail, we note that the *mean delay* algorithm and the *spike* algorithm underestimate playout delay and thus lose packets arriving with acceptable delays. As algorithm RREQ-AODV considers a minimum bound for the safety factor ($\beta_{m,k} > 40$), it loses less packets than the *mean delay* algorithm and the *spike* algorithm.

6. Conclusion

In this paper, we have proposed a new playout delay algorithm, called the RREQ-AODV algorithm, specially designed for voice over ad hoc networks. Its first strength is in the way that it estimates the end-to-end delay in the presence of mobility which leads to route reconfiguration. The algorithm uses RREQ-AODV message delay as a delay indicator. This is appropriate because the receiver is sure that the audio packet will go through the same path. Its second strength is the adaptation strategy which gives the same importance to interactivity, reliability and stability constraints. The performance evaluation results show that our algorithm outperforms existing algorithms in all cases when considering simultaneously the interactivity, the reliability and the stability criteria, as well as when considering only interactivity and reliability criteria. A drawback of our solution is that the methodology is tied to AODV protocol. However, a general methodology for reactive protocols can perhaps be derived from the proposal.

In conclusion, our algorithm will contribute to improve the quality of voice applications running on ad hoc networks. Of course, other mechanisms, such as FEC and network differentiated services, should be also used to supply enough QoS for the user of voice applications.

Appendix

Our simulation modeled an ad hoc network of 50 mobile hosts placed randomly within a 1000 meter $\times$ 1000 meter area. Radio propagation range for each node was 250 meters and channel capacity was 11 Mb/s. The IEEE 802.11 MAC protocol [8] is used as the MAC layer in our simulations. The specific access scheme is Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) with acknowledgements.

Node mobility is based on random-way-point. Node mobility speeds vary from 2 m/s to 20 m/s with average pause time of 50 s. A node selects a destination randomly within the terrain range and moves towards that destination at a predefined speed.

A traffic generator was developed to simulate constant bit rate (CBR) and audio sources:

- An audio traffic is generated between a source and a destination randomly selected. The alternating periods of activity and silence are exponentially distributed with average duration of 1.004 s and 1.587 s, respectively [9]. We use the PCM (Pulse Codec Modulation – see recommendation G.711 in [9]) codec bit rates at 64 kbps to generate audio traffic. During activity periods, 320 byte voice packets were generated periodically representing samples of 40 ms.
- Background traffic generated by two kinds of Constant Bit Rate (CBR) data sessions with randomly selected sources and destinations were simulated. They transmit packets of 160 bytes every 0.2 s and 2048 bytes every 0.3 s, respectively. The traffic load varies during the audio session by varying the number of active data sources.

AODV protocol is used for ad hoc routing. The AODV parameter values were chosen as they minimize network congestion and allow the protocol to operate as quickly and as accurately as possible [Perkins and Royer, 19]: *HELLO_Interval* = 1.0, *Route_Reply_Wait_Time* = 1.0, *Reverse_Route_Life* = 3.0 and *Active_Route_Timeout* = 3.0. The routing protocol maintains a send buffer of 64 packets. It contains all data packets waiting for a route. All packets (both data and routing) sent by the routing layer are queued at the interface queue until the MAC layer can transmit them. The interface queue has a maximum size of 50 packets and is maintained as a priority queue with two priorities each served in FIFO order. Routing packets get higher priority than data packets.

References

[1] P. Agrawal, J. Chen and C. Sreenan, Use of statistical methods to reduce delays for media playback buffering, in: *Internat. Conf. on Multimedia Computing and Systems* (1998) pp. 259–263.

[2] M. Benaissa, V. Lecuire, F. Lepage and A. Schaff, Analysing end-to-end delay and loss in mobile ad hoc networks for interactive audio applications, in: *Workshop on Mobile Ad Hoc Networking and Computing MADNET'2003* (2003) pp. 27–33.

[3] C. Boutremans and J.Y. Le Boudec, Adaptive joint playout buffer and fec adjustment for Internet telephony, in: *Proc. of IEEE INFOCOM'2003*, San-Francisco, CA (2003).

[4] D. Clark, S. Shenker and L. Zhang, Supporting real-time applications in an integrated services packet network architecture and mechanism, in: *SIGCOMM'92* (1992) pp. 14–26.
[5] K. Fujimoto, S. Ata and M. Murata, Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications, in: *IEEE GLOBECOM* (2002).
[6] M. Greis, Tutorial for the network simulator ns-2, `http://www.isi.edu/nsnam/ns/tutorial/` (2001).
[7] ISI, The network simulator ns, //www.isi.edu/nsnam/ns/ (2001).
[8] ISO/IEC, IEEE, standard for wireless medium access control (MAC) and physical layer (PHY) specifications, 8802-11:1999(E) (1999).
[9] ITU-T, List of ITU-T recommendations, `http://www.itu.int/publications/itu-t/itutrec.htm` (2001).
[10] P. Jaquet, P. Muhletaler, A. Quayyum, A. Laouiti, T. Clausen, L. Viennot and P. Minet, Optimized link state routing protocol, Internet Engineering Task Force, Internet Draft draft-ietf-manet-olsr-06.tx (2002).
[11] P. Jaquet and L. Viennot, Overhead in mobile ad hoc network protocols, Technical Report, INRIA (2000).
[12] A. Kansal and A. Karandikar, Adaptive delay estimation for low jitter audio over Internet, in: *Proc. of IEEE GLOBECOM*, San Antonio, USA (2001) pp. 17–18.
[13] N. Kitawaki and K. Itoh, Pure delay effects on speech quality in telecommunications, IEEE Journal of Selected Areas in Communications 9 (1991) 586–593.
[14] S.-J. Lee, E.M. Royer and C.E. Perkins, Scalability study of the ad hoc on-demand distance vector routing protocol, International Journal of Network Management 13(2) (2003) 97–114.
[15] P.D. Leon and C. Sreenan, An adaptive predictor for media playout buffering, in: *Proc. of IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)* (1999) pp. 3097–3100.
[16] Y. Liang, N. Farber and B. Girod, Adaptive playout scheduling and loss concealment for voice communications over IP networks, IEEE Transactions on Multimedia (2001).
[17] S. Moon, J. Kurose and D. Towsley, Packet audio playout delay adjustment: Performance bounds and algorithms, Multimedia Systems 6 (1998) 17–28.
[18] C. Perkins, *Ad Hoc Networking* (Addison-Wesley/Longman, 2001).
[19] C.E. Perkins and E.M. Royer, Ad-hoc on-demand distance vector routing, in: *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA (1999) pp. 90–100.
[20] J. Pinto and K. Christensen, An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods, in: *Proc. of the IEEE 24th Conf. on Local Computer Networks* (1999) pp. 224–231.
[21] R. Ramjee, J. Kurose, D. Towsley and H. Schulzrinne, Adaptive playout mechanisms for packetized audio applications in wide-area networks, in: *Proc. of the IEEE INFOCOM* (1994) pp. 680–688.
[22] V. Ramos, C. Barakat and E. Altman, A moving average predictor for playout delay control in voip, Technical Report, Nice–Sophia Antipolis University, Mistral and Planète research team, INRIA (2003).
[23] C. Sreenan, J. Chen, P. Agrawal and B. Narendran, Delay reduction techniques for playout buffering, IEEE Transactions on Multimedia 2(2) (2000) 88–100.