International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 2, Number 1, pages 18–40, January 2012

# A Packet Aggregation Mechanism For Real Time Applications Over Wireless Networks

Paulo H. Azevêdo Filho
Department of Computer Science
University of Brasilia (UnB), 70910–900, Brasilia, Brazil
Email: paulo@cic.unb.br

Marcos F. Caetano
Department of Electrical Engineering
University of Brasilia (UnB), 70910–900, Brasilia, Brazil
Email: caetano@cic.unb.br

and

Jacir L. Bordim
Department of Computer Science
University of Brasilia (UnB), 70910–900, Brasilia, Brazil
Email: bordim@cic.unb.br

Received: July 30, 2011
Revised: October 31, 2011
Accepted: December 15, 2011
Communicated by Akihiro Fujiwara

**Abstract**

This work presents a packet aggregation technique, named *Holding Time Aggregation - HTA*. HTA is tailored for real time applications whose data is carried over wireless network environments. At the center of HTA lies an elaborated packet holding time estimation, which makes HTA to be highly adaptable to the diverse link conditions of a wireless setting. Contrary to other proposals that consider fixed packet retention time, the proposed HTA uses an adaptable packet retention time to allow relay nodes to explore aggregation opportunities on a multi-hop path. The proposed mechanism was evaluated and compared to another prominent packet aggregation scheme. Simulation results have shown that the proposed mechanism is capable to keep jitter and total delay within application limits. Furthermore, HTA has shown to allow for substantial reduction on the number of packet transmissions as well as on the overall packet overhead. Savings in terms of packet transmissions reached nearly 80% in the evaluated scenarios. These results have shown that the proposed scheme is able to cope with varying network link capacity and strict application timing requirements. The empirical results have shown to be consistent with the analytical results.

## 1 Introduction

The broad usage of the Internet has boosted the race for new technologies and fostered the convergence of many different services in the use of Internet Protocol (IP). This movement has allowed

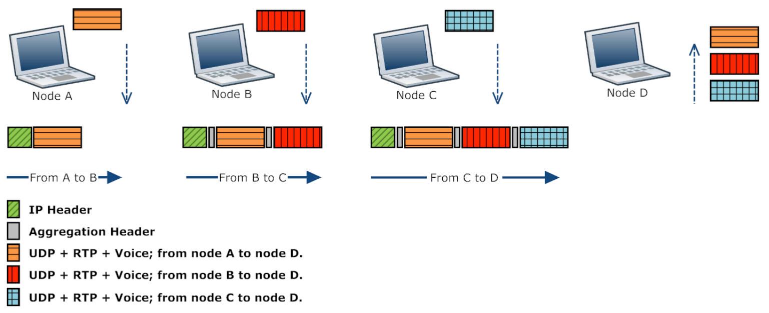
Node A
Node B
Node C
Node D
From A to B
From B to C
From C to D
IP Header
Aggregation Header
UDP + RTP + Voice; from node A to node D.
UDP + RTP + Voice; from node B to node D.
UDP + RTP + Voice; from node C to node D.

Figure 1: Graphical representation of an aggregation packet comprised of a number of data packets.

the birth of new trends and services, such as Voice over IP (VoIP) [22]. The private and public sector have been looking for alternative solutions to reduce their telephone costs and in this context, VoIP has gained considerable attention and is now commonplace [20]. At the same time in a different context, the adoption of wireless communications has increased with the advent of IEEE802.11 based networks (a.k.a WiFi) [4]. The IEEE802.11 standard became a natural alternative to bring mobility to VoIP applications, owing to its low cost and broad usage, particularly in the last mile [19]. However, wireless networks do not offer the same performance as wired networks do, as they are prone to interference, contention and packet collision [1], [6]. Also, the QoS constraints are severe when it comes to voice traffic demands as these packets are sent in a continuous flow requiring little end-to-end delay and packets arrival variation (jitter) [25].

When considering an IEEE802.11*b* network [4], its theoretical throughput would allow for nearly 341 VoIP streams, which corresponds to about 170 simultaneous dialogs. However, it turns out that such networks are capable of carrying only a few VoIP dialogs [23]. This poor performance is mainly due to the following problems: (*i*) the large packet headers added; (*ii*) the amount of fixed timers between frames; and (*iii*) the numerous control frames used by the IEEE802.11 standard. All these facts impose a significant overhead on the system, which leads to bandwidth wastage. In fact, when VoIP packets are considered, the overhead introduced can amount to more than 3 times the size of the voice packet [2]. The overhead introduced at the physical and data link layer of the WiFi standard poses a challenge for supporting real time applications. To reduce these problems, the scientific community has been seeking for alternatives in order to minimize the overhead and enable VoIP applications to be used over wireless networks [5]. As voice packets are only a few bytes long, it is possible to merge a number of voice packets into a single, larger packet as shown in Figure 1. The figure shows a number of nodes in a multihop setting. As the voice packets traverse the network, they are combined in order to reduce the overhead. This technique is called *packet aggregation* and has been shown to be a prominent approach, presenting reasonable results.

Packet aggregation has been shown as a feasible alternative to carry voice calls over wireless network. For these reasons, packet aggregation has been a popular subject and a number of aggregation protocols and techniques have been proposed in the literature. These proposals can be grouped according to the TCP/IP layer they are applied. The proposals in [3], [5], [9], [10], [11], [16], [18], [23] work at the network layer. Proposals in [13], [17], [24], [26], on the other hand, work at the data link layer. The network layer aggregation proposals can be also classified as *multi-hop* ([3], [5], [9], [10], [11], [16], [18]) or *single-hop* ([23]). When considering multi-hop aggregation algorithms, they can be further divided according to the point of aggregation in the network, which can be end-to-end ([5], [10], [11], [18]) or hop-by-hop ([3], [5], [9], [11], [16]). In what follows, we review some of the works which are closely related to the aggregation algorithm proposed in this work.

## 1.1 Related works

In [16], an algorithm tailored for sensor networks, which includes packet routing, compression and aggregation was proposed. Although packet aggregation is used in this work, the focus is on sensed data in which timing constraints are not as severe as in real time application such as VoIP. In [9] the authors offer an approach using linear combinations of the packets being transmitted, using the fact that network devices may store many different packets. This method requires deep changes in the TCP/IP stack, which makes implemention for broad usage difficult. In the IPAC [18], the authors define mechanisms to hold the data packets at the source node in order to aggregate as many packets as possible before transmitting them. As IPAC aggregates packets only at the source node, it fails to aggregate packet along the path, as in the scenario shown in Figure 1. A multihop aggregation scheme demands for a mechanism to estimate the amount of time during which a packet can be held at each relay node along the path towards the destination. This idea has been explored in [3], where the relay nodes may aggregate packets coming from difference sources and bound to other destinations. The main drawback of this proposal is that the holding time at each relay node is fixed, making it unsuitable for environments where the links have different characteristics, a common situation in wireless environments. In such scenarios, the proposed scheme may generate excessive delays. The protocol proposed by Kim *et al.* [11] also uses fixed timer, which is computed based on the topological information. Clearly, this protocol inherits the same problems found in [3]. In [5], the authors assessed the protocol proposed in [11] in a testbed.

Kekre *et al.* [10] focused on reviewing VoIP operations over IEEE802.11 WLANs. The authors focused on multi-hop scenarios. The aggregation algorithm uses the same approach presented by Kim [11]. However, a flow admission control is used before allowing a new flow, so as to ensure that QoS requirements are respected. Wang *et al.* [23] proposed an aggregation algorithm applied to a set of wireless infra-structured network topology, interconnected by a wired infra-structured network. The VoIP packets are aggregated by a Voice Gateway located at the edge of the wired network. The wireless nodes implement a demultiplexing algorithm, allowing them to extract the correct portion of the aggregated packets received via the wireless gateway node. Again, the aggregation time is static, and associated with the maximum number of nodes served by the gateway node.

A frame aggregation method was proposed by Zhang *et al.* [26]. The aggregated frame can be constructed in two different ways: either by packing the voice frames from the same call or from different calls. As in other approaches, the aggregation time is fixed for a given network topology. In the same context, Yun *et al.* [24] propose a zero-delay frame aggregation scheme for up stream/down stream asymmetric links. However, the frame aggregation operations are too complex and the adaptation method increases the load operations in the access point manager. Pentikousis *et al.* [13] present a frame aggregation method for a fixed WiMAX testbed, where single hop communications are considered. Pinola *et al.* [17] investigated the impact of the OFDMA and OFDM modulation techniques when frame aggregation is used over a WiMAX environment.

This work presents a packet aggregation protocol to support VoIP applications running on top of an ad hoc wireless networks. The proposed mechanism, termed *Holding Time Aggregation* (HTA) was designed to reduce the number of transmissions and overall transmission overhead. Unlike other works in this area, this proposal evaluates the network condition, regarding the behavior of the data flow, so that it is applicable to environments in which the network links have different capabilities. The time requirements of the application are respected and the aggregation is performed in an adaptive way. The results obtained through simulations show that the proposed mechanism, compared to other similar techniques, reduces the number of transmissions and overall network overhead. Savings in terms of packet transmissions reached nearly 80% in the evaluated scenarios. These results have shown that the proposed scheme is able to cope with varying network link capacity and strict application timing requirements.

The rest of this paper is organized as follows: Section 2 introduces concepts related to the packet aggregation process. Section 3 presents a mechanism to compute holding time of packets, so that aggregation decisions can be made in an adaptive way. These results are later used in designing a holding time aggregation protocol, which is presented in Section 4. The analytical model used to evaluate the proposed protocol is presented in Section 5 while the empirical and analytical results are presented in Section 6. Section 7 concludes this work and presents directions for further investigation.

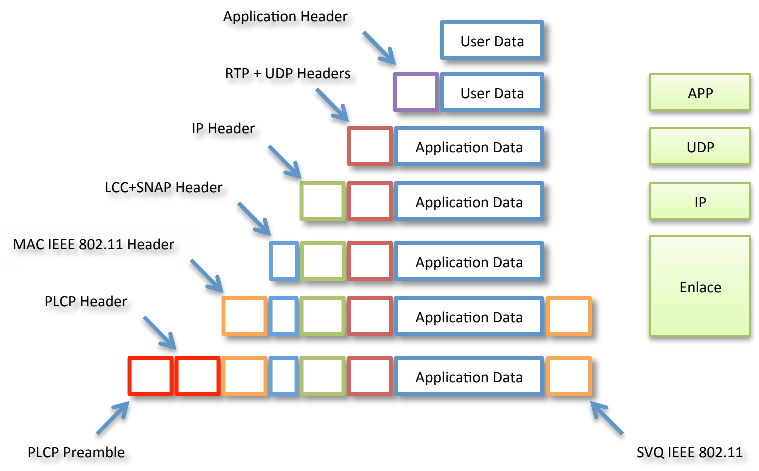

Figure 2: Data encapsulation process.

## 2 Packet Aggregation

The focus of packet aggregation in wireless networks is to reduce the overhead introduced by headers, fixed timers, and control packets which are used by the wireless protocols. As the user's data traverses the TCP/IP stack, from the application layer to the physical layer, a number of headers are added, as seen in Figure 2. Let us consider a voice packet received from the user application. At the transport layer, the Real Time Protocol (RTP) headers and the UDP headers are added to the VoIP packet. The RTP header comprises 12 bytes and the UDP header adds another 8 bytes. The segment is then passed to the network layer, where an additional 20 bytes are added. The packet is then passed to the data link layer, which is sub-divided into LLC (Logical Link Control) and MAC (medium access control) sub-layers. According to the IEEE802.11 standard [4], the LLC adds 8 bytes (3 for the LLC and 5 for SNAP). At the MAC sub-layer, an additional 34 bytes are added, and then the frame is passed to the physical layer, at which, another 24 bytes are appended, which comprises the PLPC preamble and PLPC header. Thus, altogether, the amount the headers added is 96 bytes. When considering the iLBC codec [12], the user application voice packet is about 38 bytes. Thus, the size of the headers added to the VoIP packets is over 2.5 times the size of the actual message.

IEEE802.11 defines a number of control packets which are used to reserve the medium and minimize collisions. These control packets include RTS (20 bytes), CTS (14 bytes) and ACK (14 bytes), which adds another 48 bytes. That is, to send 38 bytes from the application, 144 additional bytes are necessary. Also, as seen in Figure 3, the IEEE802.11 includes a number of fixed timers, such as Short Interframe Space (SIFS), DCF Interframe Space (DIFS) and the preamble, which are independent of the data size being sent. On top of that, the contention time before accessing the channel depends on various factors, including the number of transmitting nodes and channel conditions, which also have an impact on the application (further details about the timers, as well as the correlation between overhead and payload in wireless networks can be seen in [2]). Clearly, to allow voice data to be carried over an IEEE802.11 based network, one has to find means to overcome such problems.

In this context, packet aggregation unites the payload from many packets to send them as an aggregated packet. In this way, the impact of the aforementioned timers and the overall number of control packets can be greatly reduced. The related works presented in the previous section consider

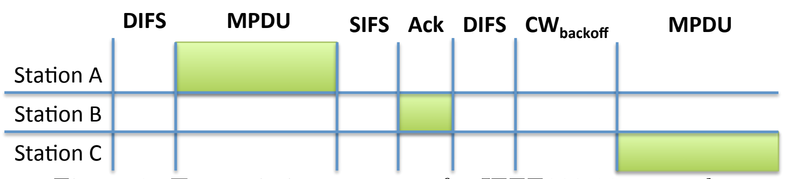

Figure 3: Transmission sequence for IEEE802.11 networks.

an environment in which the connections always have similar bandwidth and timing issues. However, due to the dynamic nature of mobile networks and the peculiarities of the medium, the conditions of the channel for a node $n_i$ can be very different from that in node $n_j$, even when $n_i$ and $n_j$ are adjacent (i.e. $i \neq j$, and $0 < i, j \leq N$, where $N$ is the number of nodes in the network). In other words, the conditions for a link between close nodes, $n_i$ and $n_j$ in a wireless network can vary with time and are subject to interference, collisions and contention. The latter, for instance, is directly related to the number of neighbors and their behavior, such as the number of packets sent, received or even routed. Therefore, in our packet aggregation mechanism we have to consider those issues to provide an acceptable QoS level. It is worth mentioning that timing is a critical factor to the quality of the service perceived by the user.

To better illustrate how to apply packet aggregation in an ad hoc network, let us consider the topology in Figure 4. The figure shows four nodes and three links connecting them. Note that the link between nodes $n_3$ and $n_4$ has a lower link capacity comparing to the other links. Now, suppose that nodes $n_1$ and $n_2$ have a number of constant bit rate (CBR) packets to transmit to node $n_4$. Considering the iLBC codec, 38 bytes would be generated at every $20ms$ in the application layer of each source node [12]. Those packets will be sent to node $n_3$ and then forwarded to the destination node $n_4$. It is noticeable that $n_3$ can combine the packets received from $n_1$ and $n_2$ before sending them to $n_4$. Packet aggregation allows $n_3$ to reduce the number of transmissions to $n_4$. Nevertheless, to make the aggregation viable, the time spent in this process has to be acceptable to the application. That is, node $n_3$ has to be able to estimate the time spent in receiving the packet from the source node as well as the time it will take to send it to the final destination node $n_4$. As link $n_3 \rightarrow n_4$ has a lower capacity than the link $n_1 \rightarrow n_2$, a packet may take longer to traverse the last link. Hence, a fixed holding time at each node may not be appropriate in this case. It is noteworthy that in a VoIP communication, the total time to transmit a packet from its source to its final destination cannot be above $150ms$, so as to meet the VoIP QoS constraints [7].

Until now, packet aggregation techniques define the time to transpose a link as constant, which simplifies the task of estimation. However, this assumption is not realistic as the links among nodes may have different characteristics, including channel capacity, as illustrated in Figure 4. Thus, using a single estimate for every link can lead to both waste of aggregation opportunities and packet discharge. In the next section the details of the proposed technique are presented. The proposed scheme aims to fulfill the aforementioned requirements, so that aggregation can be done even in scenarios where the links have different characteristics.

## 3 Holding Time Packet Aggregation

This section presents the key elements which will be later used in our packet aggregation scheme, termed Holding Time Aggregation (HTA). As discussed in the previous sections, other works suggest aggregation of packets by retaining them for a fixed amount of time at each hop. In HTA, in contrast, each packet is held for the maximum allowed time with respect to the application requirements. A packet aggregation protocol works by exploring aggregation opportunities. Clearly, the main difference among the existing aggregation protocols is the reliance on the techniques employed to identify and explore such opportunities. Suppose that node $n_r$ is a node on the path $P(n_s, n_d)$, that

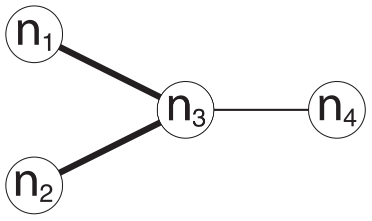

Figure 4: Network with links that have different capacities.

Table 1: Route-time traversal estimation.

| Time | $n_1$ | $n_3$ | $n_4$ |
|---|---|---|---|
| $t_0$ | $\rightarrow RQ_{1,4}$ | | |
| | Start $R_{1,4}$ | | |
| $t_1$ | | $\rightarrow RQ_{1,4}$ | |
| | | Start $R_{3,4}$ | |
| $t_2$ | | | $\rightarrow RQ_{1,4}$ |
| $t_3$ | | | $\leftarrow RP_{4,1}$ |
| $t_4$ | | $\leftarrow RP_{4,1}$ | |
| | | Stop $R_{3,4}$ | |
| $t_5$ | $\leftarrow RP_{4,1}$ | | |
| | Stop $R_{1,4}$ | | |

is, the path between the source and destination nodes, $n_s$ and $n_d$, respectively. We call any node on this path, except the destination node, a relay node, denoted by $n_r$. Upon receiving a packet $p(s,d)$, node $n_r$ needs to decide whether packet $p(s,d)$ can wait for aggregation opportunities or not. To answer this question, the relay node $n_r$ needs to know:

1. The amount of time the packet $p(s,d)$ took to reach the relay node; and
2. The amount of time the packet $p(s,d)$ will take complete its journey.

In other words, node $n_r$ needs to know the amount of time packet $p(s,d)$ took to traverse the path $P(s,r)$ and the amount of time it will take to reach node $n_d$ (*i.e.*, traverse the path $P(r,d)$). When such information is available, node $n_r$ can take proper action. Such action includes sending the packet immediately or holding the packet for an appropriate aggregation opportunity to arise. In what follows we will show how the above information can be obtained, first by showing how the path traversal time can be estimated.

## 3.1 Path Traversal Time Estimation

In order to estimate the maximum holding time for a given packet, each node needs to estimate the amount of time needed for this packet to travel from the source to the destination. The main goal of this subsection is to provide means to compute an estimated route traversal time between source and destination nodes. It is well known that many mobile routing protocols usually require the nodes to keep the necessary information about the path to the desired destination. This information can be the next hop through which the packet must be forwarded, as with the DSDV (Highly Dynamic Destination-Sequenced Distance-Vector Routing) [14] and AODV (Ad hoc On Demand Distance Vector) [15] protocols. Other protocols maintain the whole route, as with the DSR (Destination Source Routing) protocol, that can even record multiple routes for a given destination [8].

Let us consider the route discovery with the aid of the topology shown in Figure 4. Suppose that node $n_1$ needs to establish a route to node $n_4$. The first step taken by the source node is to find a valid route to the desired destination. Thus, node $n_1$ sends out a *route request*, ($RQ_{1,4}$), to verify whether a route to node $n_4$ exists or not, following the routing protocol rules. When the $RQ_{1,4}$ is issued, node $n_1$, at time $t_0$, starts the timer $R_{1,4}$ as shown in Table 1. At time $t_1$, the $RQ_{1,4}$ is received by node $n_3$, which starts the timer $R_{3,4}$ and forwards the received packet to node $n_4$. When node $n_4$ receives the $RQ_{1,4}$, it sends back to the source node a *route reply*, $RP_{4,1}$, informing that a valid route has been found. Upon receiving the $RP_{4,1}$ packet, node $n_3$ stops the timer $R_{3,4}$ and forwards the received route reply to node $n_1$. Note that node $n_3$ has been able to get an estimate round trip time to node $n_4$. When the $RP_{4,1}$ packet reaches source node $n_1$, node $n_1$ will learn that a route towards destination node $n_4$ exists and its estimated round trip time is $R_{1,4}$. Clearly the time to traverse the path $P(n_s, n_d)$ can be expressed as

$$T_{s,d} = \frac{R_{s,d}}{2}. \tag{1}$$

It should be noted that the estimated round trip can be updated constantly by using *keep-alive* messages to collect link variations and route changes. This practice is usually employed by many ad hoc routing protocols [8].

## 3.2 Elapsed Time

We now turn our attention to the time spent by a packet $p(s,d)$ to reach the relay node $n_r \in P(s,d)$. In order to estimate the elapsed time, each packet $p(s,d)$ is associated with the timer $E_{s,r}$. On receiving the packet $p(s,d)$, node $n_r$ receives, along with the packet, the elapsed time up to its predecessor $n_q$ on the path $P(s,d)$. The timer $E_{s,q}$ is updated with the amount of time the packet $p(s,d)$ spent at or before the node $n_q$, by $n_q$ itself or one of its predecessors. For this purpose, after receiving the packet $p(s,d)$, node $n_r$ starts a timer $T_r$. On sending packet $p(s,d)$ to the next node on the path $P(s,d)$, the timer $E_{s,r}$ is updated such that $E_{s,r} = E_{s,q} + T_r$. It is easy to see that, when the packet $p(s,d)$ is sent by node $n_j$ to the next hop node on the path $P(s,d)$, the $E_{s,j}$ is equivalent to

$$E_{s,j} = \sum_{i \in P(s,j)} T_i. \tag{2}$$

## 3.3 Holding Time Estimation

Since the path traversal time and the elapsed time estimation are available, we can now show that it is possible to compute the amount of time a packet can spend at a relay node. For that purpose, let $A_{max}$ be the maximum allowed time for a packet $p(s,d)$ to traverse the path $P(s,d)$. Recall that $n_r$ is a relay node on the path $P(s,d)$. Thus, from Equation 1, $n_r$ is able to compute $T_{r,d}$, that is, the amount of time to reach the destination. Then, from Equation 2 and Equation 1, $n_r$ is able to compute the *maximum holding time* $H(p)$ as follows:

$$H(p) = \frac{A_{max} - (E_{s,r} + T_{r,d})}{|P(r,d)|}, \tag{3}$$

where $|P(r,d)|$ is the number of hops in the $P(r,d)$ path, that is, the path from the relay node $n_r$ to the destination node. Note that the larger $H(p)$ is, the greater the aggregation opportunities will be, as a packet $p(s,d)$ can spend more time at node $n_r$. In this case, larger aggregated packets are likely to be produced. On the other hand, when the network conditions worsen, the path traversal time is likely to increase. This will have the effect of reducing aggregation opportunities, resulting in shorter aggregated packets. In this work we assume that $A_{max}$ is an application parameter, which is associated with the application requirements for a given class of packets.

# 4 A Packet Aggregation Protocol

This section presents the Holding Time Aggregation (HTA) protocol. The HTA protocol is a distributed protocol that uses, as a key ingredient, the holding time information, which was discussed previously. This information is used to decide whether, and for how long, a given packet can be retained to explore future aggregation opportunities. The HTA protocol uses a number of basic operations, described below.

Table 2: Priority queue entries example.

| $q_4$ | |
|---|---|
| $p(n_1, n_4)$ | $t_0$ |
| $p(n_2, n_5)$ | $t_1$ |
| $\vdots$ | $\vdots$ |

## 4.1 Basic Operations

The HTA protocol uses four basic operations regarding packet aggregation and queueing operation. We begin with some definitions. For this purpose, consider a network topology represented by the graph $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of edges. Let $N = |V|$ be the number of nodes and $p(i, j)$, $0 \leq i, j < N$ where $i \neq j$, denote a packet originated from source node $n_i$ bound for the destination node $n_j$. Also, let $A_p = \{p(i, j) \mid i, j \in |V|\}$ denote an aggregation packet. Furthermore, let $p(i, *)$ denote the packets whose source node is node $n_i$. Likewise, $p(*, j)$ denotes the set of packets whose destination node is $n_j$. Each node $n_i$ has $\Delta(n_i)$ priority queues, where $\Delta(n_i)$ represents the degree of node $n_i$. Then, a neighboring node of $n_i$, say $n_k$, is denoted as $n_i(k)$. Thus, at node $n_i$, the priority queue $q_k$ holds the packets whose next hop is node $n_i(k)$. The items in each priority queue are sorted in ascending order of $H(p)$, so that the head of the queue holds the packet with the least holding time. Four basic operations are used in the HTA protocol. These operations are defined below:

$(i)$ : The *unpacking* operation $U(A_p, n_k)$ allows retrieval of all the packets within $A_p$ whose next hop node is node $n_k$;

$(ii)$ : The *packing* operation $P(A_p, n_k)$ allows aggregation into $A_p$ of all packets whose next hop node is $n_k$;

$(iii)$ : The *enqueue* operation $E(U(A_p, n_k), q_k)$ retrieves all packets from the aggregation packet $A_p$ and place them into queue $q_k$;

$(iv)$ : The *dequeue* operation $D(q_k, t)$ retrieves all packets from queue $q_k$ whose holding time is at most $t$;

To better understand the above operations, let us consider the following example. Consider a string network topology consisting of five nodes $n_1, n_2, n_3, n_4, n_5$, as presented by the Figure 6. Note that node $n_2$ has two neighboring nodes, $n_1$ and $n_3$. Let $A_p = \{p(n_1, n_3), p(n_1, n_4), p(n_2, n_5)\}$ be an aggregation packet at node $n_3$. When node $n_3$ executes the unpacking operation $U(A_p, n_4)$, it retrieves the packets within $A_p$, in this case $\{p(n_1, n_4), p(n_2, n_5)\}$, whose next hop towards the destination nodes is $n_4$. Similarly, when node $n_3$ executes the packing operation $P(A_p, n_4)$, it would set $A_p = \{p(n_1, n_4), p(n_2, n_5)\}$. The enqueue operation $E(A_p, q_4)$ places the packets $\{p(n_1, n_4), p(n_2, n_5)\}$ into $q_4$. When a packet is inserted into the queue, its estimated holding time is computed and placed in the queue as well. In other words, each entry in a queue $q_k$ of node $n_i$, where $n_i(k)$, is a tuple $<p(s, d), t>$, where $t$ is computed using Equation 3 and $p(s, d)$ is a packet bound to the next hop node $n_k$. Table 2 shows the contents of queue $q_4$ at node $n_3$ after the enqueue operation. In this example, the dequeue operation $D(q_4, t_0)$ would retrieve the packet $p(n_1, n_4)$ from $q_4$. From the above, it should be clear that the unpacking operation is executed whenever a node receives an aggregated packet. The packing operation, on the other hand, is used to aggregate packets which are directed to the same next hop. The maximum size of an aggregated packet size is bounded by the network's maximum allowed transmission unit.

The packing operation allows a number of packets $p(s, d)$ going to the same next hop node to share the same structure of a larger packet $A_p$. Figure 5 shows the structure of an aggregation packet $A_p$. Note that the packet $A_p$ is just a common IP packet, where the payload part is used

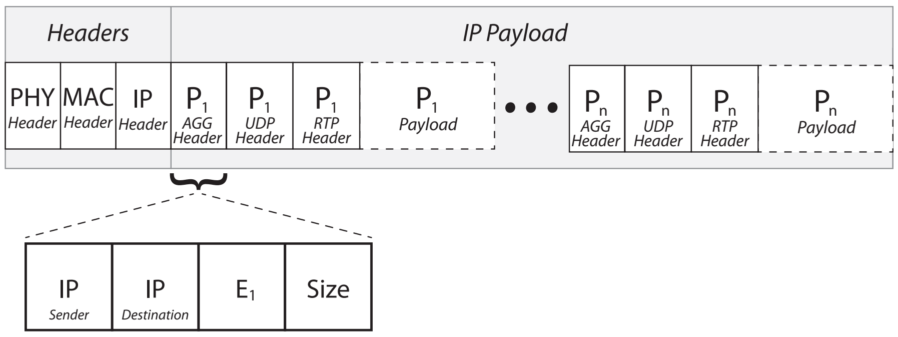

Figure 5: An IP packet comprising of a number of voice packets aggregated into the payload part.

to multiplex several voice packets. Each packet $p(s,d) \in A_p$ is inserted into the payload part with some information allowing relay nodes to correctly retrieve them. Thus, when a node $n_j$ receives a packet, it needs to know whether this is an aggregation packet or not. Here, the *protocol* field, in the IP header, is used to indicate whether a packet contains aggregated packets or not (similarly to the scheme proposed in [3]). When this is the case, the $E_{i,j}$ field of the aggregation header indicates the elapsed time since the aggregated packet was generated. As discussed in the previous section, every node is responsible for updating this value before forwarding the packet to the next hop on the path. The *size* field indicates the size of this aggregated packet. After *size* bytes there may be another aggregation header, up to the limit represented in the *total length* field in the IP header is reached. As can be seen, the aggregation header is used to multiplex several packets into an aggregation packet. Similarly, the aggregation header is used to demultiplex packets using the original IP header information. The structure of the $A_p$ packet can be created with few modifications to the IP packet. Indeed, only an 11 bytes header would be necessary for each packet $p(s,d)$ to be aggregated into a larger packet $A_p$. These 11 bytes correspond to the source and destination IP address, the elapsed time $E(s,r)$ and the packet size, as depicted in Figure 5.

## 4.2 HTA Protocol

In the HTA protocol, every packet is treated as an aggregation packet $A_p$, even when $A_p$ contains a single packet $p(s,d)$ within it. The HTA protocol is based on two main triggered events, the *receive* event and *timeout* event. We begin with the receive event. Upon receiving an aggregation packet $A_p$, node $n_i$ performs two main tasks as shown in Table 3. The first task of node $n_i$ is to retrieve all the packets $p(*,j) \in A_p$, where $i \neq j$, and place them into the corresponding next hop queue $q_j$. That is, those packets whose destination is the same next hop node are placed in the same priority queue. For this purpose, the unpacking operation is used. For each packet retrieved from the aggregation packet $A_p$, the maximum holding time $H(p)$ is computed using Equation 3. Then, the retrieved packets are inserted into the appropriate queue along with its holding time $H(p)$. These tasks are performed in Step 1. The packets whose destination is node $n_i$ are then retrieved and sent to the corresponding upper layers. Thus, after a *receive* event, the following conditions should hold for node $n_i$:

**CND1** : All packets being routed through $n_i$ are placed into the appropriated next hop queue; and

**CND2** : All packets bound to node $n_i$ are delivered to the upper layers of node $n_i$.

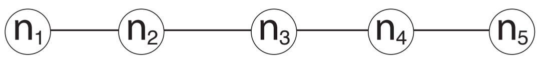

Figure 6: A linear network topology.

Table 4 describes the actions taken on a timeout event. Note that a timeout event occurs when the holding time $H(p)$ of a packet $p$ is below a given threshold $t$. Suppose that a node $n_i$ retrieves packets from a given timeout priority queue, say $q_j$. For this purpose, node $n_i$ uses the dequeue operation. The dequeued packets are placed into an aggregation packet $A_p$, as described in Step 1. Once the aggregation packet $A_p$ is built, node $n_i$ sends it to the appropriated next hop node $n_i(j)$. Thus, after a timeout event, the following condition is met:

**CND3** : All packets whose holding time is below threshold $t$ have been placed into an aggregation packet $A_p$ and sent to the corresponding next hop node.

Although our discussion here is focused on aggregation packets, it should be clear from the previous section that a node is able to determine whether a received packet is an aggregation packet or not. This information is inserted into the *protocol* field of the IP header. For each packet $p(s,d) \in A_p$, the time elapsed since the packet was created is associated with each packet, allowing intermediate nodes to take proper action based on the amount of time elapsed since its creation and the time necessary to reach its final destination. Once the inner workings of the HTA protocol have now been explained, our next task is to present an analytical model which will be later used in evaluating and comparing the proposed protocol with other related works.

## 5 Analytical Model

In what follows we will define an analytical model that will be later used to evaluate the HTA protocol, which has been presented in the previous section. The first task is obtaining the amount of packets that can be aggregated per transmission. For that purpose, we will consider the underlying network topology represented as a graph, according to the graph definition of Section 4.1. Let $AP = \{\bigcup P(s,d) | s,d \in V\}$ be the set of all active paths used by the existing flows on the network. Also, let $e$ be a link in $AP$. Clearly, the link $e$ may be shared by a number of active paths $P(s,d)$. Let $R(e)$ denote the set of paths $P(s,d)$ that share the link $e$. Then, the aggregation time $A_T^{R(e)}$ for all the flows that go through link $e$ is bound by the average aggregation time among all those flows. Then, a relay node can use the Equation 3 to compute the $A_T^{R(e)}$, as in

$$A_T^{R(e)} = Avg(H(p)). \tag{4}$$

To compute the amount of packets that can be aggregated per transmission, it is necessary to consider the average incoming packet frequency. Remember that $|R(e)|$ is the amount of active flows in the network that share the link $e$, between nodes $n_i$ and $n_j$, and let $I(i,j)$ be the average interval between packet arrivals incident to node $n_i$ heading towards $n_j$. Then, the number of aggregated packets from node $n_i$ towards node $n_j$ can be computed as in

$$N(i,j) = |R(e)| + \frac{A_T^{R(e)}}{I(i,j)}. \tag{5}$$

Table 3: Actions taken when an aggregated packet $A_p$ is received by $n_i$.

| **Trigger: Node $n_i$ receives $A_p$** | |
|---|---|
| **Step 1:** | Unpack all items in $A_p$, compute their holding time and enqueue them into their corresponding next hop queues. That is, $\forall j \mid j \in n_i(j), E(U(A_p, n_j), q_j)$, where the relay node for $p(s,d)$ is $n_i(j)$. |
| **Step 2:** | All packets in $A_p$ whose destination node is $n_i$ are sent to the upper layers $L_A$ of node $n_i$. That is $L_A \leftarrow U(A_p, n_i)$; |

Table 4: Actions taken when a queue $q_j$ on node $n_i$ has items whose holding time is about to expire.

| **Trigger:** | **Timeout on queue $q_j$.** |
|---|---|
| **Step 1:** | Node $n_i$ builds a packet $A_p$ such that $A_p \leftarrow D(q_j, t)$ by using the dequeue operation. Each packet has its holding time updated before the packing operation takes place; |
| **Step 2:** | Node $n_i$ sends aggregate packet $A_p$ to next hop node $n_j$; |

Once the aggregation time limits have been defined, the next task is to calculate how this translates into reducing the total amount of bytes transmitted. To do so, we start by calculating the size of each packet. For that purpose, some definitions are necessary, which are shown in Table 5.

Table 5: Headers added at different 802.11 layers with the iLBC Payload on top.

| Description | Symbol | Value |
|---|---|---|
| Physical layer headers | $H_p$ | 24 bytes (PLPC header and preamble) |
| MAC headers | $H_m$ | 42 bytes (LLC & SNAP) |
| IP headers | $H_i$ | 20 bytes |
| UDP headers | $H_u$ | 8 bytes |
| RTP headers | $H_r$ | 12 bytes |
| Aggregation headers | $H_g$ | 11 bytes |
| Packet payload | $P_l$ | 38 bytes (iLBC) |

The headers of a layer three packet $L_3$ are defined in Equation 6, whereas a layer four packet $L_4$, including the payload, is defined in Equation 7.

$$L_3 = H_p + H_m + H_i \tag{6}$$

$$L_4 = \begin{cases} H_u + H_r + P_l & \text{If no aggregation is used} \\ H_u + H_r + H_g + P_l & \text{If aggregation is used} \end{cases} \tag{7}$$

Then, the size of a packet $S_p$ can be computed by combining Equation 6 and Equation 7. Let $N_g$ denote the number of packets that are included in an aggregation packet $A_p$. When no aggregation is used, $N_g = 1$ and $N_g > 1$ otherwise. Then, the size of a packet $S_p$ can be defined as

$$S_p = L_3 + N_g \times L_4. \tag{8}$$

With that information at hand, we then go on to obtain the packet transmission amount issued per node, and then on the whole network. For a given node $n_i$, the total volume of data transferred $V_d(i, j)$ to a neighboring node $n_j$, over an interval of o time $I_t$, is given by Equation 9:

$$V_d(i, j) = \frac{I_t}{I(i, j)} \times \left( \frac{L_3}{N(i, j)} + L_4 \right). \tag{9}$$

Let $Adj(n_i)$ be the set of neighbors of node $n_i$. So, the total amount of data transmitted by node $n_i$, named $V_d(i, *)$, is given by the sum of the transmissions it made to all of its neighbors $n_i(*)$, and can be represented by Equation 10:

$$V_d(i, *) = \sum_{n_j \in Adj(n_i)} V_d(i, j). \tag{10}$$

Thus, the amount of data transmitted in the whole network, $V_d(*,*)$, is the sum of the amounts transmitted by each node in the network, as defined below:

$$V_d(*,*) = \sum_{i=1}^{N} V_d(i,*). \tag{11}$$

Using Equation 9, it is easy to obtain the total amount of payload $V_l$ transmitted from a node $n_i$ to its neighboring node $n_j$. This is done by Equation 12. Then, the total payload transmitted by a given node $n_i$ and the total amount of payload transmitted in the whole network can be obtained, as defined in Equation 13 and Equation 14 shown below:

$$V_l(i,j) = \frac{I_t}{I(i,j)} \times P_l, \tag{12}$$

$$V_l(i,*) = \sum_{n_j \in Adj(n_i)} V_l(i,j), \tag{13}$$

$$V_l(*,*) = \sum_{i=1}^{N} V_l(i,*). \tag{14}$$

The above equations allow us to compute both the total amount of data transmitted in the network and the total amount of payload transmitted. By using Equation 11 and Equation 14, it is possible to calculate the total amount of headers $V_h$ used to carry the network traffic over a period of time, which is defined in Equation 15:

$$V_h = V_d(*,*) - V_l(*,*). \tag{15}$$

The next section presents the empirical and analytical results for a number of different scenarios.

## 6 Empirical and Analytical Results

This section presents empirical and analytical results as well as those from a simulated environment. In order to evaluate the proposed method, a simulator was developed in C++. The simulator incorporates the MAC layer characteristics of the IEEE802.11$b/g$ standard protocol in mixed mode [4]. A shortest path routing is assumed in this work while the route request message ($RQ$) and route reply message ($RP$) are used to estimate the holding time. The protocol proposed in [3], hereafter denoted as HBH, has shown to attain reasonably good results while respecting the QoS requirements of the application. Thus, HBH is used as a benchmark in comparing the empirical results. Also, and the IEEE802.11 standard, which does not use packet aggregation, hereafter referred to as STD, has also been used for comparison. Altogether, three protocols have been implemented in the simulator: the proposed HTA, the HBH protocol proposed in [3], and the IEEE802.11 standard. The analytical results are shown along with the empirical results. The empirical results consider different topologies with a varying number of source/destination pairs. In the simulations, the $A_{max}$ value was set to $150ms$, corresponding to the maximum allowed time for a VoIP application with the iLBC codec. The iLBC codec generates VoIP packets (voice packets) of 38 bytes packets each, at regular intervals of $20ms$. The details of the simulations and their parameters will be discussed along with each scenario in the following subsections.

### 6.1 First Scenario: Single Constrained Link

The aim of this first scenario is to evaluate the effectiveness of the HTA timing mechanism in a network topology in which the links do not share the same characteristics. The topology considered here is the same as that shown in Figure 4. In the simulation, the nodes $n_1$ and $n_2$ are connected to $n_3$ via a $100kBps$ links. The communication link bandwidth connecting nodes $n_3$ and $n_4$ ranges from 10 to $100kBps$. Nodes $n_1$ and $n_2$ generate ten thousand VoIP packets each. The generated packets are bound to node $n_4$ and are forwarded through node $n_3$ before reaching their destination. The numerical results are also presented, following the approach used in the previous section, for the case in which the bandwidth between $n_3$ and $n_4$ equals $100kBps$, although the results for all bandwidth values are plotted on graphics.

Figure 7 shows the average number of transmissions. The $x$ axis shows the link capacity of the constrained link $n_3 \rightarrow n_4$. The figure shows that, when no aggregation is considered, node $n_3$ would receive 20000 packets (10000 for each source node), which are then forwarded to node $n_4$. Thus, altogether, 40000 packets are transmitted, as seen for the STD protocol. When packet aggregation is used, the volume of data transmitted is reduced, as seen in the case of the HBH and HTA protocols. The proposed method is able to aggregate nearly three times more packets than the HBH method, transmitting roughly one fourth of the amount of packets used by the STD method. The reason behind this difference is that the HBH approach uses a fixed time to hold packets at each node, losing aggregation opportunities. The value used for HBH is $5ms$, as suggested in [3]. Clearly, the reduction on the amount of transmissions generates significant economy in the total volume of headers transmitted, as seen in Figure 8. It is also notable that even with the usage of aggregation headers and timing information, the proposed approach still reduces significantly the total amount of control data (headers) transmitted. This efficiency means a reduction of more than 50% when compared to the mechanism in [3].

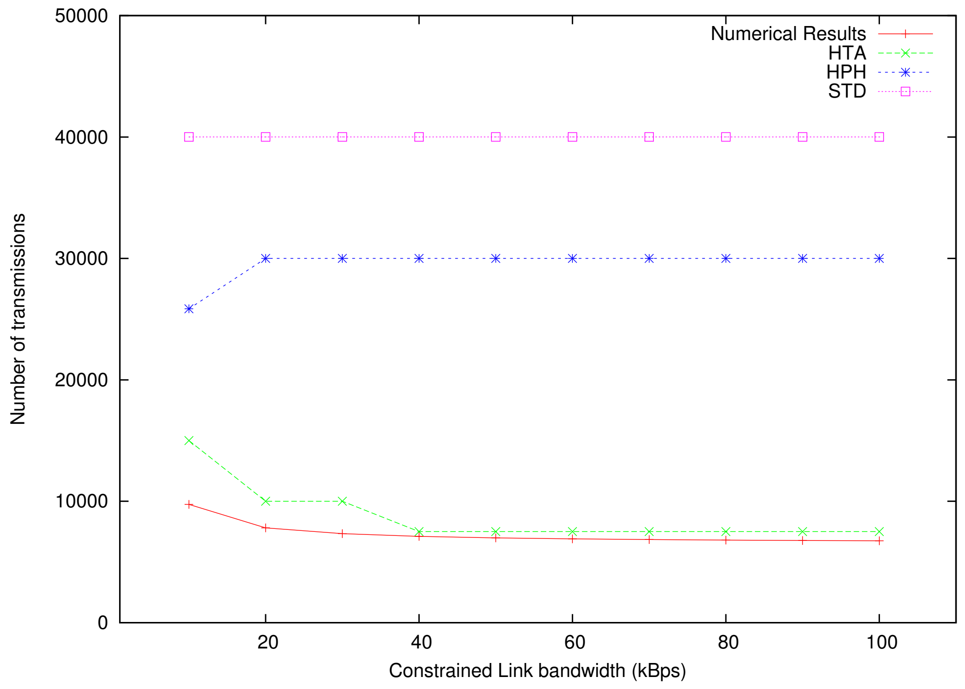

Figure 7: Overall number of packet transmissions.

From the simulation results, the estimated path traversal time $T_{1,4}$ was $\approx 16ms$ (see Equation 1). With $A_{max} = 150$, the holding time estimation at node $n_1$ can be computed with Equation 3, which is about $67ms$ per hop. As mentioned before, each node generates 10000 packets at a rate of $20ms$, then node $n_1$ and $n_2$ will be sending packets during $\approx 200000ms$. Recall that the iLBC codec generates 38 bytes VoIP packets every $20ms$. Thus, according to Equation 5, from both nodes $n_1$ and $n_2$ there will be an average of 4.36 packets aggregated per transmission, towards node $n_3$. Thus, $n_3$ will be able to aggregate, on average, 8.71 packets per transmission towards $n_4$. According to Equation 6 and the information on Table 5, the $L_3$ headers account for 86 bytes. Recall that an aggregation packet $A_p$ uses an additional 11 bytes for packing and unpacking operations. Considering the payload and additional layer four headers, a single aggregation packet, according to Equation 7 accounts for the 69 bytes. By replacing the values in Equation 9, it is possible to compute the total amount of data transmitted by nodes $n_1$ and $n_2$ without aggregation as shown below:

$$V_d(1,3) = V_d(2,3) = \frac{200000}{20} \times (\frac{86}{1} + 58) = 1,440,000 \; bytes \quad \text{without aggregation,}$$

$$V_d(1,3) = V_d(2,3) = \frac{200000}{20} \times (\frac{86}{4,36} + 69) = 887,384 \; bytes \quad \text{with aggregation.}$$

Thus, the amount of data sent to node $n_4$ equals to the amount of data issued by nodes $n_1$ and $n_2$, that is $V_d(1,3)+V_d(2,3)$. Using the aforementioned results and Equation 11, the total amount of bytes transmitted without aggregation accounts for $5,760,000$ bytes. With an average aggregation of 8.71 packet at $n_3$, the total volume of bytes transmitted is $3,352,151$ bytes. In other words, the throughput obtained is $\approx$ 28.8 and 16.76kBps, respectively, as can be seen in Figure 9 at 100kBps. In the simulations, the values obtained for STD and HTA were, respectively, $5,760,000$ bytes and $3,405,000$ bytes. These results represent a throughput of 28.8 and 17.02kBps as shown in Figure 9. Note that, in an ideal scenario, the simulation would last for 200 seconds. Figure 9 shows how the throughput behaves with the constrained link bandwidth variation. It is clear that the HTA approach considerably reduces the amount of strain put on the network, effectively freeing resources that can then be used by other traffic sharing the same links. It should be noted, however, that due to network traffic conditions, the transmissions may need to be scheduled at some points in order to wait for their turn. In such cases, the simulation may last longer than the expected, impacting the throughput. In this scenario, the HTA protocol was able to deliver packets in an efficient way, reducing the amount of resources used while carrying the same amount of traffic. Note that the higher throughput of STD and HBH, as compared to HTA, corresponds to the amount of headers used to carry the voice packets.

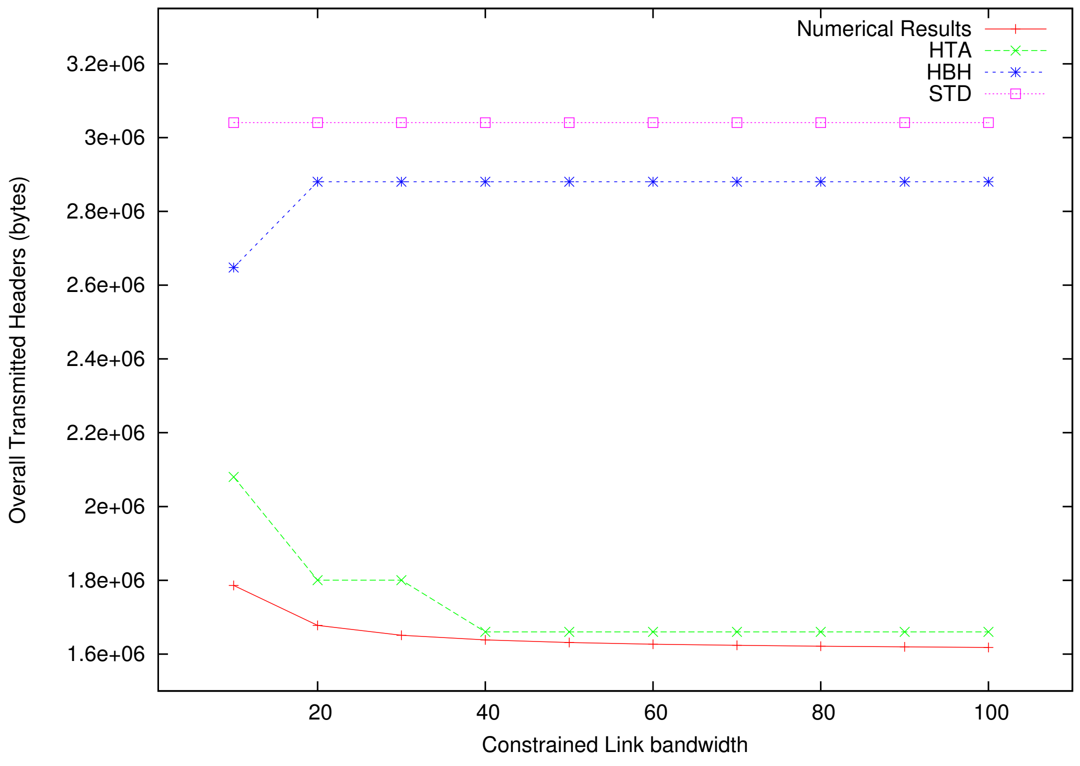

Figure 8: Total amount transmitted headers.

We will now focus on computing the amount of payload and headers that were transmitted during the simulation. Assuming VoIP packets of 38 bytes each and using Equation 13, then nodes

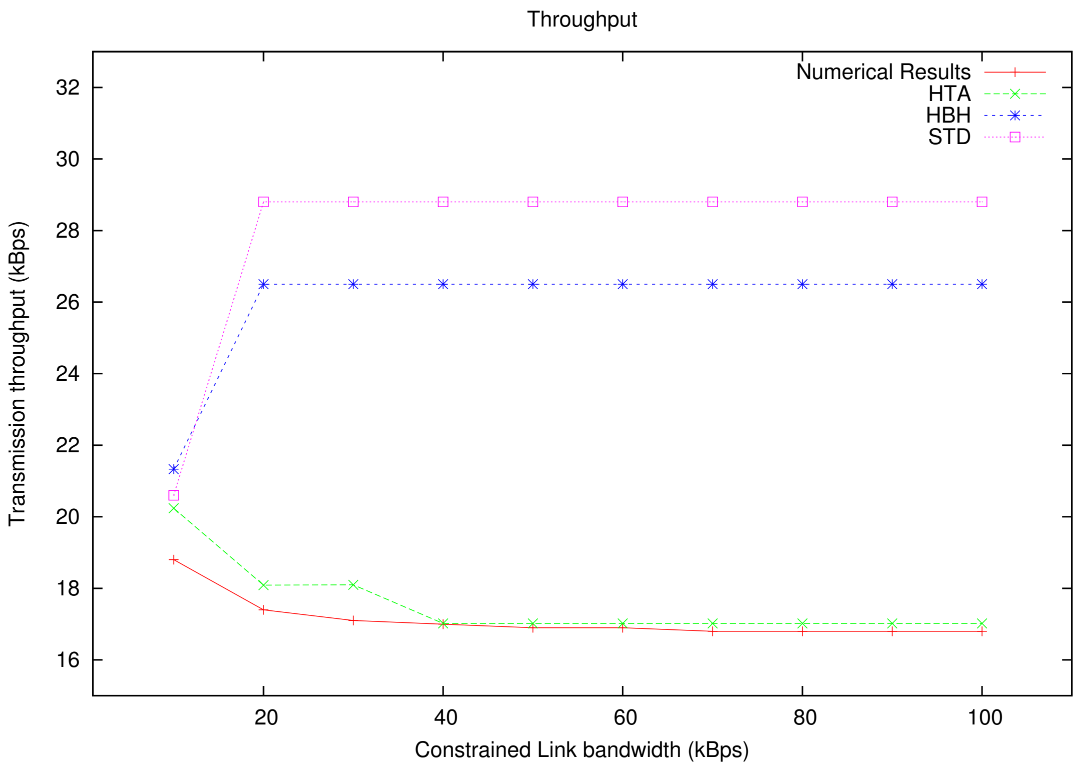

Figure 9: Overall throughput.

$n_1$ and $n_2$ would transmit 380, 000 bytes of payload each, and node $n_3$ transmitted 760, 000 bytes of payload, as computed by equations 16 and 17:

$$V_l(1,3) = V_l(2,3) = \frac{200000}{20} \times 38 = 380000 \ bytes, \tag{16}$$

$$V_l(3,4) = \frac{200000}{10} \times 38 = 760000 \ bytes. \tag{17}$$

Thus, according to Equation 14, the total payload transmitted in the network equals $V_l(*,*) = V_l(1,3) + V_l(2,3) + V_l(3,4) = 1,520,000$ bytes. Note that this holds true for both aggregating and non-aggregating environments, and also in simulated results, since the payload size and the number of hops is constant. Thus, by applying Equation 15, the total amount of headers without aggregation is 4, 240, 000 bytes, and 1, 832, 151 bytes with aggregation, as can be seen in Figure 8. The simulations were consistent with predictions when no aggregation was considered. On the other hand, when aggregation was used, the amount of headers weighted 1, 885, 000 bytes, which is slightly above the analytical model (see Figure 8).

In this work we say that a packet is *dropped* when it fails to meet the QoS constraints. Figure 11 shows the number of dropped packets. When the link between nodes $n_3$ and $n_4$ has the capacity of 10kBps, the proposed HTA protocol is still able to deliver all the packets. Note that both STD and HBH have similar performance in this case, dropping over 40% of the generated packets. Clearly, the lack of a holding time estimation makes the packet aggregation unfeasible for the HBH approach causing significant packet dropping due to excessive delays, especially when the link capacity fluctuates significantly. As for STD, the lack of an aggregation mechanism creates a larger volume of packet transmissions, which in turn, causes excessive packet delay and incurs packet discharge.

As packet aggregation mechanisms usually work by retaining packets at relay nodes, it is important to evaluate the amount of memory being used. Figure 12 shows the maximum memory size (in

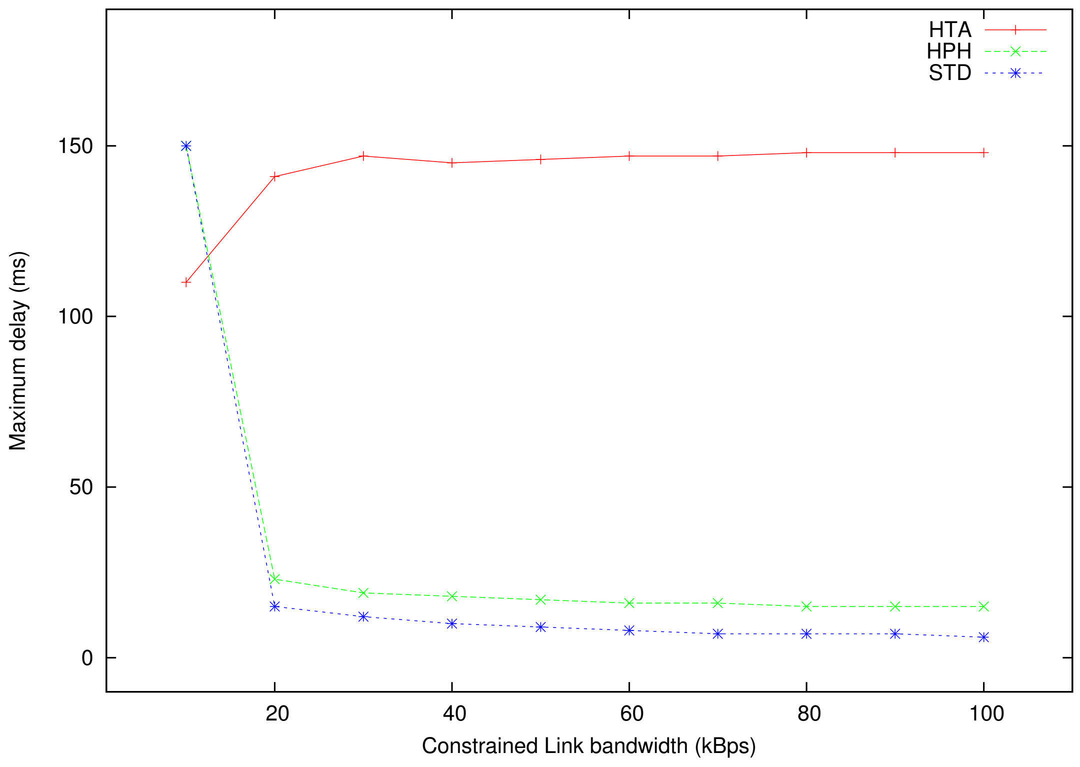

Figure 10: Overall packet delay.

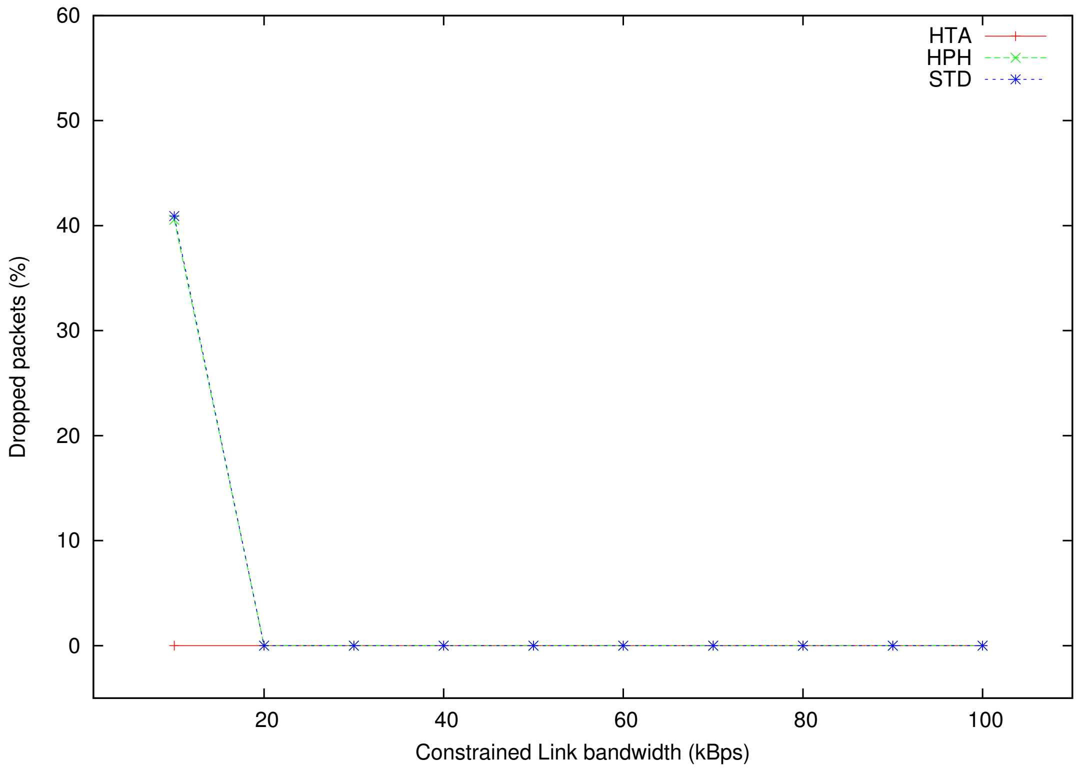

Figure 11: Overall packet drops.

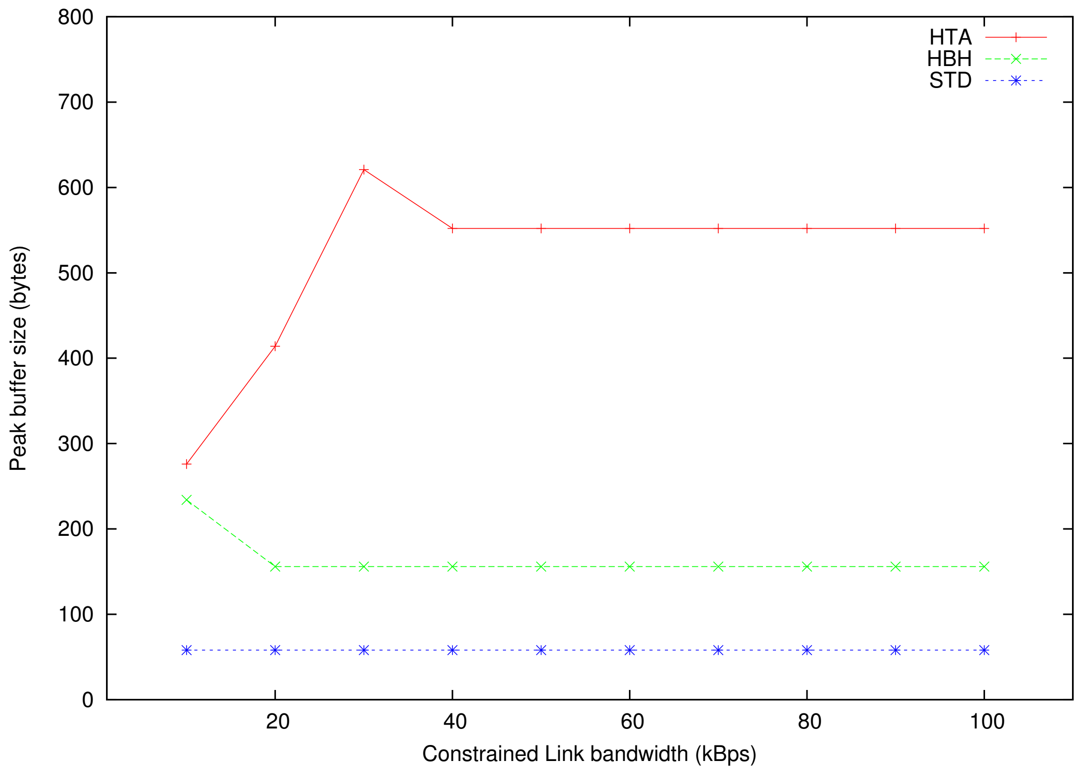

Figure 12: Buffer peak size.

bytes) demanded during the simulation for packet queues. As HTA has a better estimate, it is able to hold more packets, and hence demands more memory. Nevertheless, the memory requirement by HTA is below 650 bytes for this scenario. Note that a device is expected to be able to hold an IP layer maximum packet size, which can be as large as 65535 bytes [21]. Clearly, the memory requirements by HTA and HBH are acceptable for current technologies.

The retention time estimation mechanism in HTA allows for better management of the packet holding time, making it possible to transport the packets to their destinations within the constraints established by the application needs. It is important to note that the maximum delay experienced by the application meets its requirements. Figure 10 shows that all packets are below the maximum delay allowed for the application, which was set to be up to $A_{max} = 150ms$. As STD and HBH have a lower retention time, they experience a lower end-to-end delay. The reason for this is that HTA is adaptable. This can be seen in the figure when the bandwidth is set to 20kBps. In this case the HBH and STD end-to-end delays are above that experienced by HTA. Thus, when there is enough time for aggregation opportunities, while meeting the QoS constraints, HTA holds packets a little longer to better explore aggregation opportunities. When that is not the case, HTA forwards the packets as fast as it can to meet the QoS requirements.

## 6.2 Second Scenario: Multiplexed Network

This scenario consists of a network with a set of nodes having a single gateway node (node 9 in Figure 13). This figure reassembles a hot spot scenario in which a number of nodes are sending VoIP packets to an access point. The purpose of this test is to determine to what extent HTA can meet the application requirements when the underlining network presents a link shared by many nodes that need to route their data through the same relay node. In this scenario, each of the eight nodes is connected to the gateway with 100kBps links. Each of them will send ten thousand packets to node 10 at every $20ms$ interval, totalizing 80 thousand packets sent. Table 6 shows the empirical

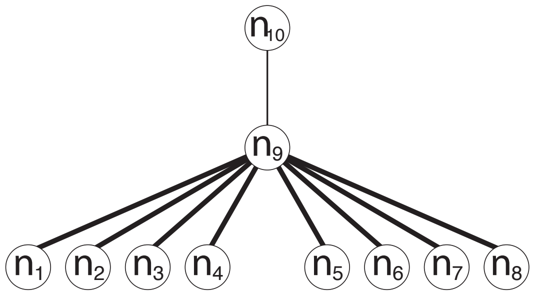

Figure 13: Network topology containing a single gateway for a set of nodes.

results for the case in which link capacity between nodes 9 and 10 is fixed at 51kBps, which was the minimum value of bandwidth that allowed at least one of the methods to deliver all the packets.

Table 6: Results for second scenario.

| Evaluation Parameters | Empirical Results | | | Analytical Results |
|---|---|---|---|---|
| | STD | HBH | HTA | |
| Transmissions | 160,000 | 110,000 | 31,672 | 24,439 |
| Headers (bytes) | 16,960,000 | 14,256,880 | 7,683,704 | 7,061,755 |
| Throughput (kBps) | 63.16 | 65.87 | 68.79 | 65.71 |
| Overhead (%) | 73.61 | 70.10 | 55.83 | 53.74 |
| Header economy (%) | 0 | 15.94 | 54.70 | 58.40 |
| Packet loss (%) | 41.12 | 40.88 | 0 | N/A |
| Max. delay (ms) | 150 | 150 | 146 | N/A |
| Jitter (ms) | 61 | 52 | 20 | N/A |
| Max. aggr. packets | 1 | 3 | 20 | N/A |
| Avg. aggr. packets | 1 | 1.19 | 4.44 | 6.55 |
| Max. buffer (bytes) | 58 | 234 | 1.656 | N/A |

As shown in Table 6, the results are similar to that in the first scenario. In this case HTA was also able to deliver the packets within the application limits, while not generating excessive delays and avoiding packet drop. The aggregation mechanism in HTA allows it to reduce the number of packets sent, and consequently, the volume of headers and control data. The average amount of packets within an aggregated packet in this scenario is about 4.44 packets. In contrast, HBH was slightly better than the STD mechanism. HTA aggregated nearly 4 times more packets than the HBH mechanism. It may come as a surprise that HTA attained a higher throughput in this scenario, as it carries less information. However, this is exactly the reason for the higher throughput. Because STD and HBH miss aggregation opportunities, they incurred a larger number of packet transmission. Clearly, such packets need to be scheduled at some point. This situation makes the simulation to last longer to carry the same amount of generated traffic. For this reason, the throughput obtained by HTA is higher than that experienced in the HBH and STD approaches in this scenario.

The economy obtained by HTA in terms of headers transmitted corresponds to approximately 55% when compared to the STD mechanism. The maximum delay created by HTA is less than that of other proposals and is within the application limits. Memory usage is this scenario reached 1.656 bytes with HTA. Furthermore, HTA suffered the smallest delay variation, or jitter, among the compared protocols. Indeed, the jitter is less than half of that experienced by the HBH approach. As in the first scenario, the empirical results for HTA are close to the analytical results.

## 6.3 Third Scenario: Hierarchical Network

The network topology used in this scenario is presented in Figure 14, which is analogous to a binary tree. In this network, every link has the same capacity (57kBps), that being the minimum capacity needed to transmit all the data without dropping packets for at least one of the methods tested. The purpose is to examine the performance of the protocols in which a number of potential aggregation

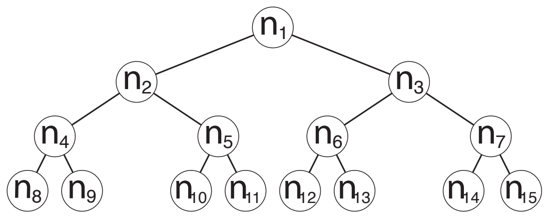

Figure 14: Hierarchical topology with 15 nodes.

points exists. In this scenario there are 15 nodes connected in a hierarchical way with each node at the base (leaf nodes) sending ten thousand packets each to the top node (root). Again, each packet is generated at a $20ms$ interval with a total of 80 thousand generated packets.

Table 7: Results for third scenario.

| Evaluation Parameters | Empirical Results | | | Analytical Results |
|---|---|---|---|---|
| | STD | HBH | HTA | |
| Transmissions | 240,000 | 145,106 | 107,693 | 62,637 |
| Headers (bytes) | 25,440,000 | 20,479,116 | 15,821,598 | 12,826,813 |
| Header economy (%) | 0 | 19.5 | 37.8 | 49.6 |
| Throughput (kBps) | 113.64 | 147.92 | 124.66 | 109.70 |
| Overhead (%) | 73.61 | 69.19 | 63.43 | 58.44 |
| Packet loss (%) | 20.44 | 17.27 | 0 | N/A |
| Max. delay (ms) | 150 | 150 | 142 | N/A |
| Jitter (ms) | 38 | 24 | 19 | N/A |
| Max. aggr. packets | 1 | 4 | 16 | N/A |
| Avg. aggr. packets | 1 | 1.60 | 3.49 | 3.83 |
| Max. buffer (bytes) | 58 | 312 | 1.104 | N/A |

In a scenario in which all nodes have the same link capacity, the proposed aggregation scheme still attained a better performance in terms of number of average aggregated packets and header transmissions, as verified in Table 7. HTA was able to reduce the overall number of transmissions by approximately 55%, and achieved an average of 3.49 data packets per aggregation packet, more than twice the HBH protocol, and very close to the prediction of the analytical model, that is 3.83 packets per transmission. As the proposed scheme incorporates a more sophisticated mechanism for packet retention, it creates better aggregation opportunities, giving HTA better packet transport efficiency than HBH's. HTA suffered lower jitter than its counterparts. In this scenario the throughput obtained by HTA was a little lower than that obtained by HBH. The reason for this behavior is that HTA transported significantly less headers. Even though HBH was not efficient in terms of packet aggregation in this scenario, it was able to carry the traffic without increasing the simulation time. Also, the memory usage in this scenario was below 1.200 bytes.

## 6.4 Fourth Scenario: Linear Network

In this scenario the nodes were arranged in a string topology, as depicted in Figure 6. Four network flows were set up, from node $n_1 \rightarrow n_4$, $n_1 \rightarrow n_5$, and two flows from $n_2 \rightarrow n_5$. This scenario not only considers nodes originating more than one flow but also having different destinations as well. The bandwidth was set to be 45kBps on every link. The results obtained are shown in Table 8.

Due to the high aggregation rates achieved by HTA, it was the only technique to withstand the traffic while delivering all the packets. HTA was able to transmit 80% less times than needed by the STD protocol. The reason for such an enormous difference is that HTA was able to aggregate, on average 6 packets per aggregation packet, whereas HBH was able to aggregate 4 packets on average. In this scenario, STD experienced a 20.5% packet loss and HBH 15%, while HTA managed to deliver

Table 8: Results for fourth scenario.

| Evaluation Parameters | Empirical Results | | | Analytical Results |
|---|---|---|---|---|
| | STD | HBH | HTA | |
| Transmissions | 130,000 | 42,808 | 25,002 | 16,823 |
| Headers (bytes) | 13,780,000 | 8,862,768 | 6,180,161 | 5,476,779 |
| Header economy (%) | 0 | 35.7 | 55.1 | 60.3 |
| Throughput (kBps) | 50.14 | 49.20 | 55.57 | 52.08 |
| Overhead (%) | 73.61 | 64.21 | 55.58 | 52.58 |
| Packet loss (%) | 20.50 | 14.96 | 0 | N/A |
| Max. delay (ms) | 150 | 150 | 147 | N/A |
| Jitter (ms) | 37 | 26 | 16 | N/A |
| Max. aggr. packets | 1 | 4 | 11 | N/A |
| Avg. aggr. packets | 1 | 4 | 6 | 7.8 |
| Max. buffer (bytes) | 116 | 312 | 759 | N/A |

all generated packets. Also, HTA was able to reduce by more than 55% the volume of headers transmitted. This can be attributed to the fact that HTA aggregated on average 50% more packets per transmission, when compared to HBH, and the savings provided by this feat made the difference by making the network resources available to transmit the packets that came afterwards. Memory usage by HTA was nearly twofold larger compared to HBH, which was expected. Nevertheless, memory requirement remained below 800 bytes. HTA achieved results which are quite close to the prediction of the analytical model in all the comparable parameters. As in the second scenario, here HTA attained higher throughput. That is, HTA was able to complete the task of moving the generated traffic from the source to the destination node in a more efficient way, completing its task in less time and with lower overhead.

### 6.5 Fifth scenario: Random networks

The fifth scenario has a different approach. Here, the underlying network topology was randomly generated. For each simulation run, 10 nodes were randomly placed in a $1000 \times 1000$ network area and connected in a way that the underlying graph was planar and connected. The link capacity for each edge was selected randomly, ranging from 30kBps to 110kBps. For each simulation run, six source/destination pairs were randomly selected for communication. Each source node generates ten thousand packets per dialog during a simulation run. Altogether, 25 random topologies were considered. The averaged simulation results are presented in Table 9. Note that in this scenario there was no guarantee that the bandwidth on the links would be sufficient to transmit all the generated conversations. The analytical results are not shown for this scenario as it would be necessary to compute them for each particular generated topology and simulation run.

Table 9: Results for fifth scenario.

| | STD | HBH | HTA |
|---|---|---|---|
| Transmissions | 106,808 | 89,809 | 25,889 |
| Headers (bytes) | 11,321,711 | 10,503,233 | 5,537,519 |
| Header economy (%) | 0 | 7.1 | 51.2 |
| Throughput (kBps) | 66.7 | 66.2 | 48.0 |
| Overhead (%) | 73.61 | 72.13 | 57.71 |
| Packet loss (%) | 26.23 | 21.40 | 5.65 |
| Max. delay (ms) | 113.2 | 96.8 | 149.1 |
| Jitter (ms) | 32.2 | 29.1 | 25.9 |
| Max. aggr. packets | 1 | 1.8 | 7.3 |
| Avg. aggr. packets | 1 | 1.2 | 4.6 |
| Max. buffer (bytes) | 125.3 | 268.3 | 1258.6 |

The results show that HTA was by far superior to the other approaches when it came to sparing network resources. HTA was able to transmit generated traffic in less than a quarter of the transmissions needed by the STD protocol and less than a third of what was needed by HBH. In this scenario, however, all the approaches failed to deliver all the packets to their destinations. This was expected as some of the links could have bandwidth lower than the minimum required to carry the generated traffic. Nevertheless, HTA was able to reduce the transmitted headers by nearly 48.8% as compared to the STD protocol. This was possible while still dropping approximately one fifth of the packets dropped by STD and one quarter of the packets dropped by HBH. Due to the high aggregation rates achieved by HTA, it sent almost four times as many packets as HBH per transmission. Also, although HTA had larger delays, it still respected the 150ms limit tolerated by iLBC, and the packets flowed more steadily, as jitter was lower than the one seen in other approaches. As in the second scenario, HTA achieved a higher throughput than the HBH and STD approaches, showing that HTA was able to deliver the packets in less time and more efficiently than the HBH approach, while the memory demand remained below 1.300 bytes.

## 7 Conclusions and Future Work

This work presents a packet aggregation mechanism tailored for real time applications over wireless networking environments. Although a number of protocols have been devised, they lack a more elaborate mechanism to estimate the packet holding time at which aggregation can take place. Indeed, most of the works presented in the literature only consider packet aggregation at the source nodes. On the other hand, those approaches that consider packet aggregation in multi-hop environments assume that the relay node knows the amount of time a given packet can be retained at its position. However, in a dynamic environment, assuming a fixed holding time may not be feasible nor desirable. Indeed, if the fixed holding time is too large, QoS constraints would not be met. On the other hand, if the fixed time is to short, aggregation opportunities would be missed. Thus, the core idea of this work is to present a new packet aggregation mechanism which incorporates an elaborate and adaptable packet holding time estimation. This estimation is used to allow for packet aggregation along the path a packet is traversing without degrading the quality of the service being provided.

The proposed mechanism was evaluated and compared with other schemes, one of which uses packet aggregation and another which does not use packet aggregation. Since HTA achieved better results consistently in all the scenarios, it is plausible to state that the technique is very efficient in using existing infrastructure and achieving better results than other current approaches. The results show that HTA is able to reduce the amount of resources needed to carry the generated traffic, both in terms of packet transmissions and the volume of bytes carried over the network. The proposed scheme uses a novel packet holding time estimation, which is adaptable to the network conditions and traffic fluctuations. Furthermore, the HTA results shown in all considered scenarios are consistent with the analytical results.

The empirical results have shown that the proposed mechanism is capable of keeping jitter and total delay under acceptable limits for the application. Furthermore, the proposed scheme allows substantial reduction of the number of packet transmissions as well as the overall packet overhead. The savings in terms of packet transmissions can be as high as 80% in the evaluated scenarios. These results have shown that the proposed scheme is able to cope with varying network link capacities and strict application timing requirements. As a future work we plan to evaluate the energy savings that the proposed HTA scheme is able to provide.

## Acknowledgements

This work was partially supported by FAP-DF and CNPq.

## References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM '04 Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, 34(4):121–132, October 2004.

[2] A. V. Barbosa, M. F. Caetano, J. L. Bordim, and P. S. Barreto. IEEE802.11b/g standard: Theoretical maximum throughput. In *ICNC'10: The First International Conference on Networking and Computing*, pages 197 – 201, November 2010.

[3] M. C. Castro, P. Dely, J. Karlsson, and A. Kassler. Capacity increase for voice over IP traffic through packet aggregation in wireless multihop mesh networks. *Future Generation Communication and Networking (FGCN 2007)*, 2(6):350 – 355, December, 2007.

[4] IEEE Standard for Information technology. Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements - part 11: Wireless lan medium access control (MAC) and physical layer (phy) specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pages C1–1184, December 1999.

[5] S. Ganguly, V. Navda, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. R. Das. Performance optimizations for deploying voip services in mesh networks. *IEEE Journal on Selected Areas in Communications*, 24(11):2147 – 2158, October 2006.

[6] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, August 2000.

[7] ITU-T. General characteristics of international telephone connections and international telephone circuits one-way transmission time (G.114), February, 1996.

[8] D. Johnson et al. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728, February, 2007.

[9] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510, June 2008.

[10] H. B. Kekre, V. A. Bharadi, R. S. Bansode, and Vikas Kaul. Performance problems of voip in 802.11 wireless mesh networks & their solutions. In *icwet '11 Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, pages 891–898, New York, NY, USA, 2011. ACM.

[11] K. Kim, S. Ganguly, R. Izmailov, and S. Hong. On packet aggregation mechanisms for improving voip quality in mesh networks. In IEEE, editor, *IEEE 63rd Vehicular Technology Conference*, pages 891 – 895, May 2006.

[12] W.B. Kleijn. Enhancement of coded speech by constrained optimization. In *IEEE Workshop Proceedings Speech Coding.*, pages 163–165, October 2002.

[13] K. Pentikousis, E. Piri, J. Pinola, F. Fitzek, T. Nissilä, and I. Harjula. Empirical evaluation of voip aggregation over a fixed wimax testbed. In *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, TridentCom '08, pages 19:1–19:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[14] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the conference on Communications architectures, protocols and applications*, volume 24 of *SIGCOMM '94*, pages 234–244, New York, NY, USA, October 1994. ACM.

[15] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications.*, pages 90–100, February 1999.

[16] D. Petrović, R. C. Shah, K. Ramchandran, and J. Rabaey. Data funneling: Routing with aggregation and compression for wireless sensor networks. In *Proceedings of the IEEE International Workshop on Sensor Network Protocols and Applications*, pages 156–162, May 2003.

[17] J. Pinola, E. Piri, and K. Pentikousis. On the performance gains of voip aggregation and rohc over a wirelessman-ofdma air interface. In IEEE, editor, *IEEE Global Telecommunications Conference*, pages 1–6, December 2009.

[18] R. Raghavendra, A. P. Jardosh, E. M. Belding-Royer, and H. Zheng. IPAC: IP-based adaptive packet concatenation for multihop wireless networks. In *40th Asilomar conference on signals, systems and computers*, pages 2147–2153, October 2006.

[19] K. P. Scheibe, L. W. Carstensen Jr., T. R. Rakes, and L. P. Rees. Going the last mile: a spatial decision support system for wireless broadband communications. *Decision Support Systems*, 42(2):557–570, November 2006.

[20] P. Sherburne and C. Fitzgerald. You don't know jack about voip. *Queue - VoIP*, 2(6):30–38, September 2004.

[21] W. R. Stevens. *TCP/IP Illustrated*, volume 1. Addison-Wesley Professional, January 1994.

[22] R. P. Swale. Voip – panacea or pig's ear? *BT Technology Journal*, 19(2):9–22, April 2001.

[23] W. Wang, S. C. Liew, and V. O. K. Li. Solutions to performance problems in voip over a 802.11 wireless lan. *IEEE Transactions on Vehicular Technology*, 54(1):366 – 384, January 2005.

[24] S. Yun, H. Kim, and I. Kang. 100+ voip calls on 802.11b: The power of combining voice frame aggregation and uplink-downlink bandwidth control in wireless lans. *IEEE Journal on Selected Areas in Communications*, 25(4):689–698, May 2007.

[25] H. Zhai, J. Wang, and Y. Fang. Providing statistical qos guarantee for voice over ip in the ieee 802.11 wireless lans. *IEEE Wireless Communications*, 13(1):36–43, February 2006.

[26] L. Zhang, K. Yu, Y. Du, and X. Wen. Improving capacity and qos of voip in ieee 802.11e wireless lans. In *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, Mobility '08, pages 12:1–12:8, New York, NY, USA, 2008. ACM.