Measurements of SIP Signaling over 802.11b Links

Cristian Hesselman Henk Eertink
Telematica Instituut
The Netherlands
{cristian.hesselman, henk.eertink}@telin.nl

Ing Widya
University of Twente
The Netherlands
widya@cs.utwente.nl

Erik Huizer
Utrecht University
The Netherlands
huizer@cs.uu.nl

ABSTRACT

The Session Initiation Protocol (SIP) is a popular application-level signaling protocol that is used for a wide variety of applications such as session control and mobility handling. In some of these applications, the exchange of SIP messages is time-critical, for instance when SIP is used to handle mobility for voice over IP sessions. SIP may however introduce significant delays when it runs on top of UDP over lossy (wireless) links. These delays are the result of the exponential back-off retransmission scheme that SIP uses to recover from packet loss, which has a default back-off time of half a second.

In this paper, we empirically investigate the delay introduced by SIP when it runs on top of UDP over IEEE 802.11b links. We focus on the operation of SIP at the edge of an 802.11b cell (e.g., to update a mobile host's IP address after a handoff) as this is where SIP's retransmissions scheme is most likely to come into play. We experiment with a few 802.11 parameters that influence packet loss on the wireless link, specifically with different link-level retransmission thresholds, signal-to-noise-ratios (SNRs), and amounts of background traffic. We conduct these experiments in a controlled environment that is free from interfering 802.11 sources.

Our results indicate that (1) SIP usually introduces little delay except for an SNR range of a few dBs at the very edge of an 802.11 cell in which the delay increases sharply, and (2) that a maximum of four 802.11 retransmissions suffices to limit the delay introduced by SIP retransmissions. The first result is of interest to developers of SIP applications who have to decide at which SNR to initiate a handoff to another network. The second result allows network providers to optimize their 802.11b networks for delay sensitive SIP applications.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design – *network communications, wireless communication;* C.2.2 [**Computer Communication Networks**]: Network Protocols – *applications*; C.4 [**Performance of Systems**]: Performance attributes; Reliability, availability, and serviceability.

General Terms

Measurement, Performance, Experimentation.

Keywords

Measurements, 802.11 networks, signaling, SIP

1. INTRODUCTION

The Session Initiation Protocol (SIP) [1] is an application-level signaling protocol that was originally developed for establishing, modifying, and releasing multimedia sessions (e.g., voice over IP calls). Over the last few years, SIP has evolved beyond this scope into a multi-purpose protocol that is also being used for applications like mobility handling [2], instant messaging [3], event notification [4], capability negotiation [5], and for switching mobile hosts between application-level redistributors of live multimedia content [6].

In some of these applications, the exchange of SIP messages is time-critical. For example, to handle mobility for voice over IP sessions, SIP has to inform the correspondent node (cf. Mobile IP [7]) of the mobile host's new IP address (i.e., execute a handoff) within a few tens of milliseconds to avoid perceptual glitches. Other applications (e.g., one-way streaming [6, 8]) are more delay tolerant, but may still want to set an upper bound to the handoff delay (e.g., because of the size of their playout buffer is limited).

A potential problem of SIP-based applications is that SIP may introduce significant delays when it runs on top of UDP over lossy (wireless) links. These delays are the result of the exponential back-off retransmission scheme that SIP uses to recover from packet loss. The default back-off time of this scheme is half a second second [1], which means that a single packet loss results in 0.5 seconds of delay, two consecutive losses in 1.5 seconds of delay, three consecutive losses in a delay of 3.5 seconds, and so on.

In this paper, we empirically investigate the delay introduced by SIP when it runs on top of UDP over IEEE 802.11b links. IEEE 802.11 is a popular wireless LAN technology that is for instance used in hotspot environments [6, 9, 10]. The delay that SIP applications incur on 802.11b links has not been empirically studied before.

We concentrate on the operation of SIP at the edge of an 802.11b cell (e.g., to update a mobile host's IP address after an 802.11 handoff) as this is where SIP's retransmissions scheme is most likely to come into play. We experiment with a few 802.11 parameters that influence packet loss on the wireless link, in particular with (a) the maximum number of 802.11

retransmissions (the so-called *retry limit*), (b) the signal-to-noise-ratio (SNR), and (c) the amount of background traffic on the network. Other parameters that influence packet loss (e.g., multi-path interference, natural background noise [11], and motion [12]) are outside the scope of this paper. We conduct our experiments in a controlled environment without interfering 802.11 sources.

In the rest of this paper, we first consider the notion of a SIP transaction, which is SIP's basic form of interaction. In Section 3 we present our measurement set-up, and in Sections 4 and 5 we discuss our results.

2. SIP TRANSACTIONS

SIP interactions are organized in so-called *transactions*. A transaction is a sequence of SIP messages (i.e., protocol data units) that begins with a SIP client transmitting a request to a SIP server. The SIP server responds with zero or more so-called provisional responses followed by one final response. A provisional response informs the client that the server is handling the request, whereas a final response indicates that the server has executed the client's request. Depending on the type of request, clients confirm the receipt of a final response by transmitting an acknowledgement (an 'ACK') to the server. Figure 1 shows this basic behavior.

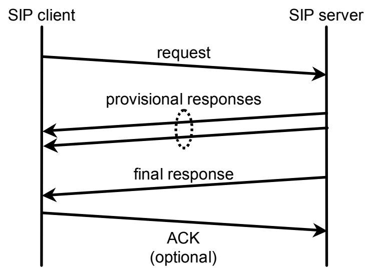

Figure 1. Basic SIP transaction.

In this paper, we concentrate on so-called *invite transactions*. Invite transactions begin with an INVITE request message and are for instance used to establish a multimedia session or to query a SIP server for its capabilities [5]. In mobility handling, mobile hosts use invite transactions to update their IP address at correspondent hosts (to execute a handoff) or at their home SIP proxy (to change the mobile host's contact information) [2]. SIP also distinguishes re-invite transactions, which are a special type of invite transactions [1]. The details of re-invite transactions are outside the scope of this paper, but they follow the same message sequence as regular invite transactions, including retransmissions.

The other types of transactions that SIP supports are collectively referred to as non-invite transactions. They start with any other type of request message than an INVITE (e.g., a BYE message, which releases a multimedia session). Non-invite transactions are not explicitly discussed in this paper, but their (retransmission) behavior is similar to that of invite transactions.

SIP supports different types of provisional and final responses. The type of a response is determined by its status code, which is an integer number between 100 and 699. A provisional response has a status code between 100 and 199, whereas the status code of a final response falls in the range 200-699. The most important status code is 200 (OK). A response with this status code signals that the server has successfully executed the request of the client (e.g., an INVITE request). An example of a provisional status code is 100 (Trying), which informs the client that the server is trying to execute its request. All other status codes indicate the client needs to redirect its request to another SIP server (300-399), or that the SIP server could not serve the client's request (400-699). Without losing generality, we only consider here the transactions that involve a 100 (Trying) provisional response and a 200 (OK) final response. SIP clients confirm the receipt of a final response in an invite transaction by transmitting an acknowledgement (an 'ACK') to the server.

When SIP runs on top of UDP, SIP transactions retransmit their messages following an exponential back-off scheme with a default back-off timer of 0.5 seconds [1]. Figure 2 shows an example of the retransmission behavior of a SIP transaction.

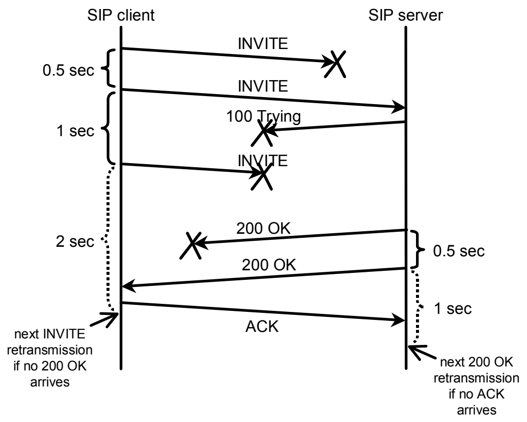

Figure 2. Retransmission behavior of a SIP transaction.

After it sent the original INVITE, a SIP client retransmits the INVITE for at most 32 seconds or until it receives a response (100 Trying or 200 OK), whichever happens first. As a result, a SIP client can retransmit an INVITE at most 6 times (i.e., transmit it 7 times), specifically at 0.5, 1.5, 3.5, 7.5, 15.5, and 31.5 seconds after the transmission of the original INVITE.

When a SIP server receives an INVITE and has transmitted a 200 OK, it retransmits the 200 OK for at most 32 seconds or until it receives an ACK, whichever happens first. A SIP server follows the same retransmission scheme as a SIP client, except that it caps the back-off time at 4 seconds. As a result, a SIP server can retransmit a 200 OK at most 10 times (i.e., 11 transmissions) after the transmission of the original 200 OK (at 0.5, 1.5, 3.5, 7.5, 11.5, 15.5, 19.5, 23.5, 27.5, and 31.5 seconds after the transmission of the first 200 OK).

We consider a SIP transaction failed if (1) the client did not receive a 200 OK 32 seconds after it sent the original INVITE, or (2) when the server did not receive an ACK 32 seconds after it sent the original 200 OK. We note that the SIP RFC does not explicitly call the first case a transaction failure. In addition, if a 200 OK reaches the client but the ACK never gets back to the server, then the SIP RFC does not consider that a transaction failure either because an ACK is not part of a transaction.

A SIP server transmits a provisional response when it receives an INVITE, either the first INVITE or a retransmitted one. It does however not retransmit the provisional responses on its own initiative. Similarly, SIP clients only transmit an ACK when they receive a 200 OK (original or retransmission) and do not retransmit ACKs either.

3. MEASUREMENT SET-UP

Figure 3 shows the high-level organization of our measurement set-up. The main components are a SIP client and a SIP server that execute a series of SIP transactions over an 802.11b link. We refer to such a series of transactions as a *(transaction) run*. The other component in the set-up is a traffic generator.

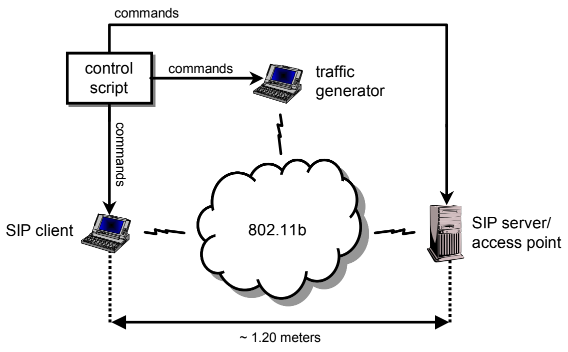

Figure 3. Measurement set-up.

The SIP client and SIP server are based on the Open SIP stack (version 1) [13], which we enhanced to make measurements. The SIP client is a laptop with a Prism2 802.11b card. The SIP server is a PC with the same type of card, but with adjustable transmission power and configured to act as an access point. The SIP client and the SIP server (access point) both use the hostap device driver [14]. The traffic generator is a separate laptop that uses the tool jtg [15] to generate traffic. It connects to the 802.11b network through an Orinoco gold card. The PC runs Debian Linux, the laptops Redhat Linux. The laptops in the set-up run on AC power to exclude influences from battery drain (e.g., lower processor speeds).

Figure 4 shows how we physically arranged the machines in the set-up. The whole set-up is located in a computer lab, with a distance of about 1.20 meters between the SIP client and the SIP server. This relatively short distance minimizes multi-path effects and effectively creates a free space environment. The traffic generator is located about halfway between the SIP client and the SIP server. The third laptop in Figure 4 (not shown in Figure 3) runs the 802.11 sniffing tool Kismet [16]. We used it during several transaction runs to check if the environment was free from interfering 802.11 sources. This was indeed the case: Kismet showed no other networks on the channel that the access point was using (channel 11) and its logs did not contain any foreign beacons.

Since the SNR at the SIP client is still quite good at a distance of 1.20 meters from the access point, we had to wrap the access point's network card in protective foil (see the enlargement in Figure 4) to create SNR levels of a few dBs at the SIP client, thus 'putting' the SIP client at the edge of the cell.

3.1 Experiment Control

The whole set up in Figure 3 is controlled by a script. This script reconfigures the 802.11 network, restarts the SIP client, the SIP server and the traffic generator before each transaction run. The control script runs on the SIP client and transmits commands to the remote devices (the SIP server and the traffic generator)

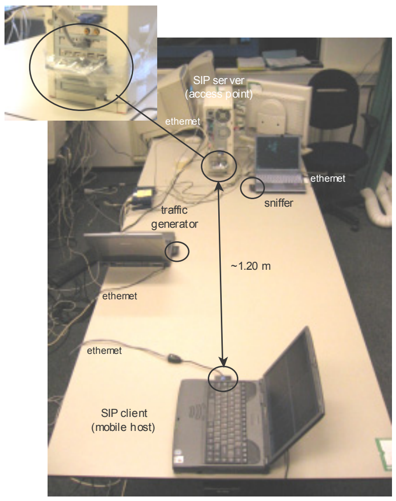

Figure 4. Physical arrangement.

through a fixed network (Ethernet). These commands also contain the new 802.11 network parameters. To avoid ARP delays, the control script inserts a manual entry for the access point in the SIP client's ARP cache when the script initializes.

To reconfigure the 802.11 network, the script changes the retry limit and the transmission rate at the SIP client and the SIP server as well as the amount of background traffic injected into the network by the traffic generator. The script also modifies the SNR at the client by adjusting the transmission power of the access point (the SIP server). Due to hardware/firmware limitations, we could not change the transmission power of the SIP client, which means that it is always transmitting at the default transmission power (-3 dBm in our specific case). As a result, our set-up is *asymmetric*. This means that the INVITEs and ACKs of a transaction will generally arrive at the SIP server, but that the 200 OKs and 100 Trying messages might be lost as a result of a low SNR (our measurements confirm this, see sections 4 and 5). An advantage of this set-up is that it enabled us to study one path (i.e., from the SIP server to the SIP client) in isolation.

3.2 Measurement Points

During a transaction, the SIP client and server log the number of 200 OKs that were transmitted, if the transaction succeeded or failed, and the time it takes to complete the transaction. In this paper, we refer to the latter as the *SIP transaction delay* and define it as the delay between the transmission of the first INVITE of a transaction and the arrival of the first 200 OK (i.e., the client-perceived transaction delay). Figure 5 shows where we made the measurements in the SIP stack.

Using the information provided by the 802.11 card, the SIP client also measures the signal-to-noise ratio (SNR) and logs the average SNR during a transaction. The SIP client calculates the average SNR for a transaction by measuring the instantaneous SNR when it transmits or receives certain messages (e.g., INVITEs and 200 OKs). The maximum size of the SIP messages we used is around 900 bytes, which means that they fit in a single UDP packet (i.e., no IP or 802.11 fragmentation).

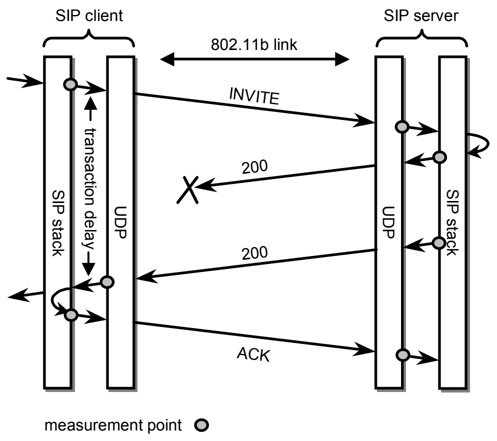

Figure 5. Measurement points.

During a transaction run, the SIP client and server store their transaction logs in memory in order not to affect the measurements through I/O operations. At the end of a transaction run, the SIP client transmits a stop command to the SIP server. At that point, both of them dump their logs to file and exit. This terminates the current iteration of the control script. The SIP client initiates the transactions in a run at random intervals.

3.3 Experiments

The goal of our experiments is to investigate the SIP transaction delay (see Figure 5), in particular at the edge of an 802.11 cell. To accomplish this, we first ran a few initial experiments to check and calibrate our measurement set-up. We for instance checked that the processing delays of the SIP stacks remained stable and determined a suitable number of transactions per run such that the experiments would not involve transient effects (see below) [17]. Next, we used the control script described in Section 3.1 to conduct two actual experiments: one in which the network was unloaded (i.e., it did not carry any competing traffic), and one in which the network carried 1 Mbps of constant bit rate background traffic (experiments with other amounts of background traffic are discussed in [17]). The second experiment represents a situation in which other audio-video streams are present on the network. The packet size of the background traffic was 1000 bytes.

In both experiments, we set the transmission rate of the SIP client, the SIP server, and the traffic generator to 1 Mbps as this is the typical rate at the edge of a cell. As a result, the network was saturated in the second experiment, which means that the SIP transactions in this experiment were executed under worst-case traffic conditions.

Aside from the background traffic, we also varied the SNR at the SIP client and the maximum number of 802.11 retransmissions (i.e., the retry limit) in both experiments. In the first experiment, we used 29 different SNR values (between approximately 0 and 17 dB) and five retry limits (0, 2, 4, 6, and 8). In the second experiment, we used 28 SNR levels (between approximately 0 and 14 dB) and the same retry limits as in the first experiment.

Experiments one and two involved 145 and 140 transaction runs, respectively. Each of these transaction runs consisted of 500 transactions, which is sufficient to obtain stable results [17]. Because an 802.11 access point will generally serve only a few mobile hosts, we configured the SIP client such that it uses a random delay of either 1 or 2 seconds between two consecutive SIP transactions.

4. UNLOADED NETWORK

We analyze the results we obtained from our two experiments (see Section 3.3) in three steps. First, we consider the number of 200 OK transmissions, which largely determine the SIP transaction delay in our asymmetrical set-up. Using this information, we define a metric for the edge of an 802.11b cell as perceived by a (SIP) application, which we refer to as the fall-off region (cf. [18]). Next, we use the average number of transmitted 200 OKs required in the fall-off region to determine an optimal retry limit. Finally, we use the fall-off region and the optimal retry limit to estimate the SIP transaction delay at the edge of an 802.11b cell. We use the data we obtained from our asymmetrical set-up to estimate the SIP transaction delay in a symmetrical set-up. We go through these three steps for the experiment with the unloaded network as well as for the experiment in which the network is saturated with background traffic (Section 5).

4.1 200 OK Transmissions

Figure 6 plots the number of 200 OK transmissions in an unloaded 802.11b network as a function *F(t,r,s)*, which represents the percentage of successful transactions that required t 200 OK transmissions in the transaction run at SNR level s (in dB) and at retry limit r. Figure 6a through e show F(t,r,s) for each of the five retry limits we used (i.e., for r=0, r=2, r=4, r=6, and r=8). While the number of 200 OK transmissions in a transaction can be between 1 and 11 (see Section 2), we only plot F(t,r,s) for the range t=1 through t=5 to increase the readability of Figure 6. The graphs in Figure 6 also show *E(r,s)*, which is the number of transactions that fail at a particular retry limit r and SNR s.

Observe that F(1,r,s) (no 200 OK retransmissions) is an application-neutral curve, whereas F(t,r,s) with $t \geq 2$ (at least one 200 OK retransmission) is specific for SIP applications. The delay associated with F(1,r,s) (see Section 4.3) is mainly determined by the delay across the 802.11b link, whereas the delay associated with F(t,r,s) for $t \geq 2$ also includes an extra delay introduced by SIP retransmissions.

As expected, Figure 6 shows that an increased 802.11 retry limit increases the percentage of SIP transactions that only need to transmit one 200 OK. For example, without any retries, 65% of the transactions requires only one 200 OK transmission at 5 dB (i.e., F(1,0,5) = 65%). With a retry limit of 2 or more, almost 100% of the transactions require only one 200 OK transmission at 5 dB (e.g., F(1,2,5) equals around 95%). Figure 6 also shows that an increase in the retry limit enables SIP transactions to deliver 200 OKs in one transmission at lower SNRs. For example, without retries, the SNR must be around 7.4 dB for 90% of the SIP transactions to deliver a 200 OK in one shot (i.e., F(1,0,7.4) = 90%). For a retry limit of 2 this is at approximately 4.4 dB and for the retry limits of 4, 6, and 8 it is around 3.5 dB.

The dotted rectangles in Figure 6 represent what we call *fall-off regions*, which are the SNR ranges in which F(1,r,s) falls from 90% to 10% (cf. [18]). A fall-off region is basically a metric for the edge of an 802.11b cell as perceived by a (SIP) application. As we will see below, the width of a fall-off region turns out to depend on the configuration of the network (e.g., on the retry limit). Note that a fall-off region might also occur in the middle of a cell (i.e., at a relatively high SNR), for instance as a result of interference. We will however not consider this case in this paper and consider it an item of future work

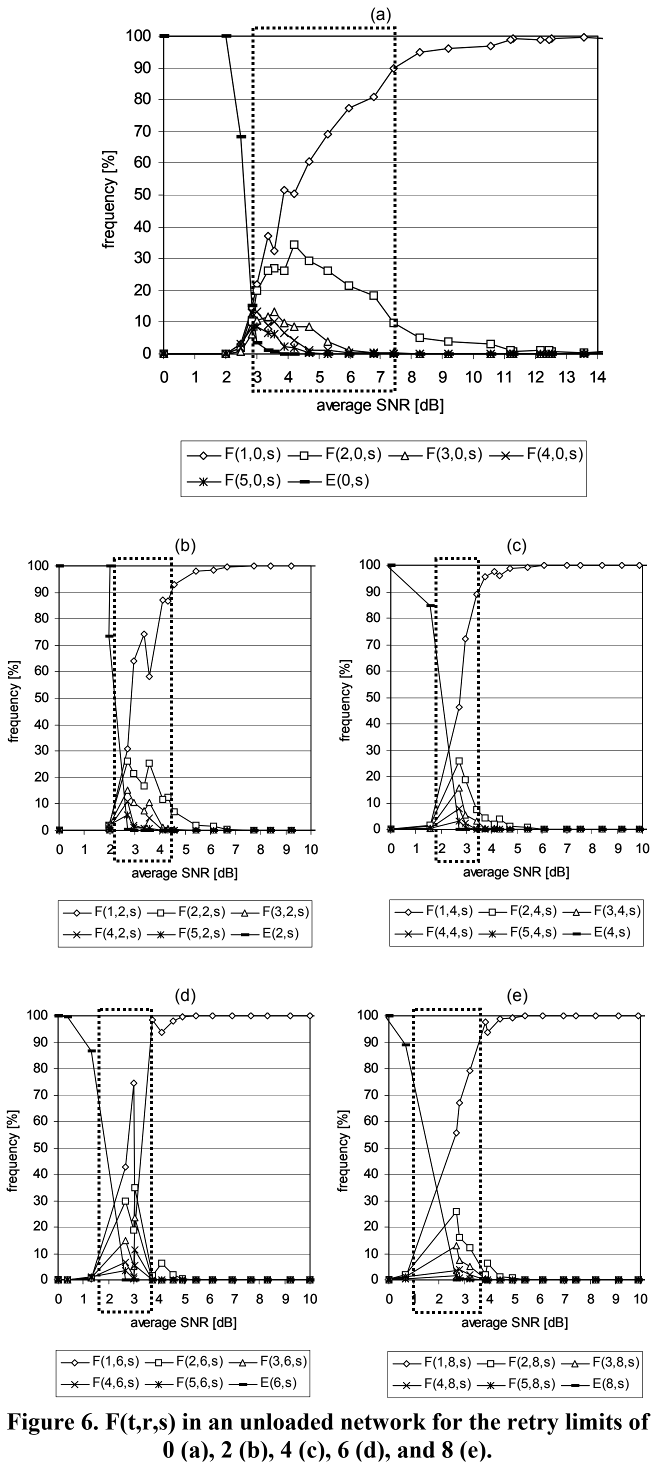

Figure 6. F(t,r,s) in an unloaded network for the retry limits of 0 (a), 2 (b), 4 (c), 6 (d), and 8 (e).

Figure 6 shows that 802.11 retransmissions decrease the width of the fall-off region. For example, the fall-off region is 4.7 dB wide without any 802.11 retries (Figure 6a) and around 1.7 to 2.5 dB wide with retries (Figure 6b-e). The advantage of a small fall-off region is that the number of SIP retransmissions (if any) will remain relatively constant in the interior of the cell (i.e., at a relatively high SNR). Only in the fall-off region (i.e., at the very edge of the network) will the number of retransmissions increase sharply and will the delay introduced by the SIP transactions increase significantly.

4.2 Optimal Retry Limit

Although we only used five retry limits, Figure 6 suggests that the width of the fall-off region does not significantly decrease beyond a retry limit of 2. For 2, 4, 6, and 8 retries, the width of the fall-off regions is around 2.2 dB, 1.7 dB, 2.1 dB, and 2.5 dB, respectively.

To see in more detail if a retry limit higher than 2 would be beneficial inside the fall-off region, we calculated the average percentage of 200 OK transmissions (i.e., the average of the values of F(t,r,s)) inside the fall-off regions. Figure 7 shows the result. The y-axis of Figure 7 indicates the average of F(t,r,s) in the fall-off region, whereas the x-axis specifies the number of 200 OK transmissions, t.

Figure 7 suggests that an increase in the retry limit to 4, 6, or 8 only marginally improves the average percentage of transactions in the fall-off region that manage to deliver their 200 OK in one shot (i.e., the average of F(t,r,s) at t=1 does not significantly increase). Equivalently, increasing the retry limit to 4, 6, or 8 does not significantly reduce the percentage of transactions that requires one or more retransmissions to deliver a 200 OK (i.e., the average of F(t,r,s) at a particular t>1 does not significantly decrease). The reason is probably that the quality of the link in the fall-off region is so bad that the additional 802.11 retries are lost as well.

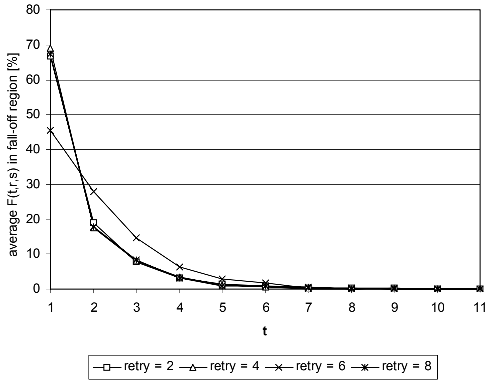

Figure 7. Average percentage of successful transactions that required t 200 OK transmissions (1≤t≤11) inside the fall-off region on an unloaded network.

Based on this information, we draw the preliminary conclusion that a retry limit of 2 suffices for SIP-based applications, at least in an environment without interference and background traffic. Such a small retry limit may be advantageous for packets carrying multimedia data because a smaller retry limit can reduce the time these packets spend in the MAC queue waiting for the (re)transmission of packets at the head of the queue [19]. ISPs might use this information to dimension their 802.11b networks to optimally support delay sensitive SIP applications (e.g., voice over IP applications).

Observe that the transaction run at 3.03 dB with a retry limit of 6 (Figure 6d) experienced an unusual amount of packet loss, thus causing the average number of 200 OK transmissions inside the fall-off region to deviate from those of the other retry limits (see Figure 7). Unfortunately, we do not know what caused this anomaly.

4.3 SIP Transaction Delay

The SIP transaction delay is the delay between the moment of transmitting the first INVITE of a transaction and the arrival of the first 200 OK (see Figure 5). Since our measurement set-up is asymmetrical (see Section 3.1), we will have to estimate the SIP transaction delay for a typical symmetrical situation. To accomplish this, we use the optimal retry limit of Section 4.2 (2) and estimate the symmetrical SIP transaction delay to twice the average one-way delay from the SIP server to the SIP client in our asymmetrical set-up. The latter is possible because the retransmission schemes for INVITEs and 200 OKs are the same until the 4th retransmission (see Section 2) and because four or more 200 OK retransmissions occur very infrequently when the retry limit is 2 (F(5,2,s) is at most 5.8% at around 2.7 dB, see Figure 6b).

The average one-way delay from the SIP server to the SIP client is equal to the average SIP transaction delays we measured (i.e., in an asymmetrical set-up) minus the average one-way delay from the SIP client to the SIP server. Since the SIP client transmits at full power, the messages on the path from the SIP client to the SIP server will usually not be subject to SIP retransmissions. As a result, the average one-way delay from the SIP client to the SIP server equals half the delay of SIP transactions that do not involve any SIP retransmissions.

In summary, the SIP transaction delay in a symmetrical situation d_{sym} is defined by

$$d_{sym} = 2*(d_{asym} - (t_{asym}/2)) \qquad (1)$$

where d_{asym} is the average SIP transaction delay we measured in our asymmetrical set-up and t_{asym} the average delay of SIP transactions that do not involve any SIP retransmissions, also measured in our set-up.

Figure 8 shows the information we need to solve equation (1). Figure 8a shows the average transaction delay over all SIP transactions (i.e., d_{asym}) executed with a retry limit of 2. As expected from Figure 6b, the average increases sharply within the fall-off region. Figure 8b is a blow-up of Figure 8a.

Figure 8c shows the average delays for the transactions that do not involve any SIP retransmissions (i.e., t_{asym}). These delays are close to the average round-trip delay to traverse the wireless link. The average in Figure 8c only increases as a result of 802.11 retransmissions, which happens as of around 8 dB.

Figure 8d shows the estimated transaction delay in a symmetrical situation, which is the combination of Figure 8a and Figure 8c using equation (1).

The dashed rectangles in Figure 8 represent the fall-off region when the retry limit is two (see Figure 6b).

Figures 8a through 8c also show the standard deviations of the average transaction delays. The standard deviation of the average transaction delay over all SIP transactions (figures 8a and 8b) increases rapidly inside and near the fall-off region as a result of SIP's exponential back-off mechanism.

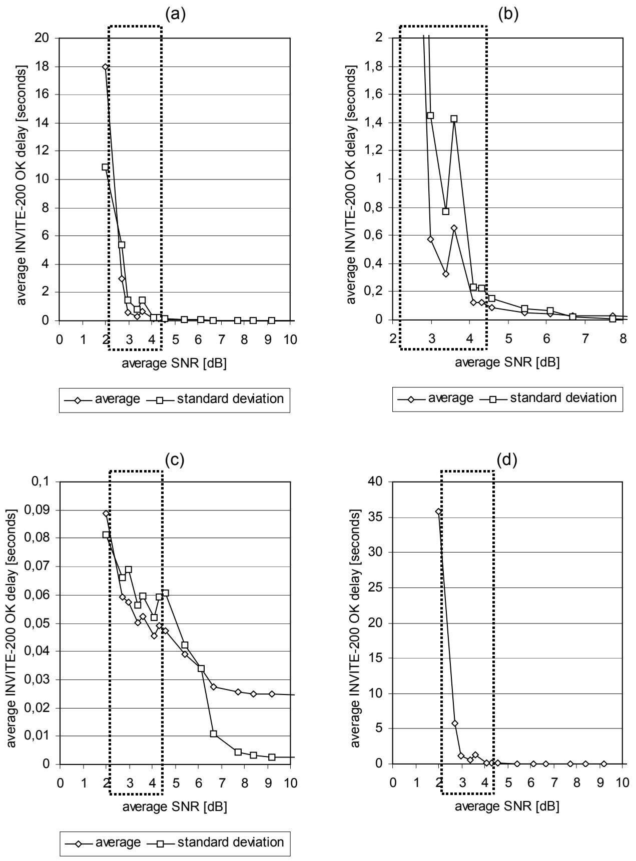

Figure 8. Average SIP transaction delay in an unloaded network using a retry limit of 2. Graph (a) shows the average delays for all SIP transactions (d_{asym}); (c) shows the average delays for transactions without any SIP retransmissions (t_{asym}); and (d) shows the estimated delay in a symmetrical situation (d_{sym}). Graph (b) is a blow-up of graph (a).

Since a fall-off region is a metric for the edge of an 802.11b cell as perceived by a (SIP) application, we define the average SIP transaction delay at the onset of the fall-off region (i.e., at the dB value where F(1,r,s) equals 90%) as SIP's *edge delay* (for 802.11b). In general, SIP applications should avoid transmitting messages inside the fall-off region because this will usually result in delays larger than the edge delay. This is for instance important for developers of SIP applications who need to decide at which dB value to initiate a handoff to another network (e.g., using handoff policies [20]).

In our experiments, the edge delay over all SIP transactions is approximately 100 milliseconds at around 4.4 dB (Figure 8b). Figure 9 shows that the two closest transaction runs (at 4.58 and 4.33 dB) contain transactions that involved SIP retransmissions (at 0.5 and 1.5 seconds), which somewhat increased the average delay. Notice that we could have used the delay median instead of the delay average to reduce the influence of such large but rare delay instances. The y-axis of Figure 9 denotes the frequency at which a certain SIP transaction delay on the x-axis occurred. We sampled the delay values using an interval of 250 milliseconds, which means that a bar at transaction delay x represents the number of SIP transactions that experienced a transaction delay in the range of [x, x+0,25) seconds.

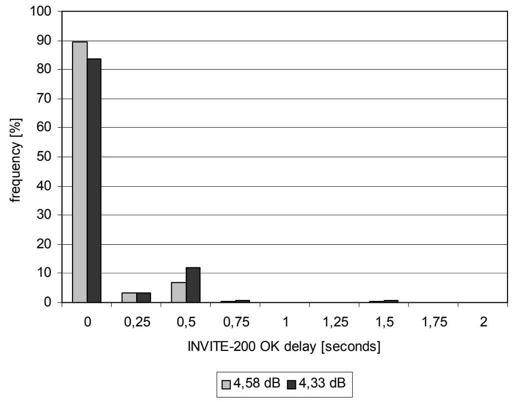

Figure 9. Frequency charts for SIP transaction delays in the asymmetrical set-up, unloaded network.

The edge delay for transactions that do not involve any SIP retransmissions equals around 48 milliseconds (Figure 8c), which means that we can estimate the edge delay for a symmetrical set-up to be 2*(100 – (48/2)) = 152 milliseconds (see equation (1)).

5. LOADED NETWORK

We use the same three steps to analyze the results of the second experiment (network loaded with 1 Mbps of constant bit rate traffic) as we used to analyze the experiment in which the network did not contain any competing traffic (see Section 4).

5.1 200 OK Transmissions

Using the same rationale as in Section 4.1, Figure 10 shows F(t,r,s) in a network loaded with 1 Mbps background traffic. Figure 10 shows the same pattern as Figure 6, specifically that the width of the fall-off region decreases as the 802.11 retry limit increases. Figure 10a (no 802.11 retries) shows that the 1 Mbps background traffic increases the fall-off region to about 11 dB (was 4.7 dB in the unloaded network of Figure 6a), probably as a result of mid-air collisions. Figures 10b through 10e show that a retry limit of at least two alleviates this problem. The width of the fall-off region is about 3 dB for a retry limit of 2 (Figure 10b), 2 dB with a retry limit of 4 (Figure 10c), 2.3 dB for a retry limit of 6 (Figure 10d), and 1.8 dB for a retry limit of 8 (Figure 10e). These values are similar to those in the unloaded network, which were 2.2 dB, 1.7 dB, 2.1 dB, and 2.5 dB for a retry limit of 2, 4, 6, and 8, respectively (see Figures 6b through 6e).

Notice that the width of the fall-off region in Figure 10a is an estimate because F(1,0,s) did not reach 90%. Similarly, the width of the fall-off region in Figure 10b is also an estimation because F(1,2,s) did not cross the 10% threshold.

5.2 Optimal Retry Limit

Figure 11 plots the average of F(t,r,s) inside the fall-off regions of Figure 10. It shows that the average percentage of transactions that require only one 200 OK (t=1) increases as the retry limit increases and that the difference between a retry limit of 2 and a retry limit of 8 is around 22%. The difference between the average percentage of transactions that transmit two or more 200

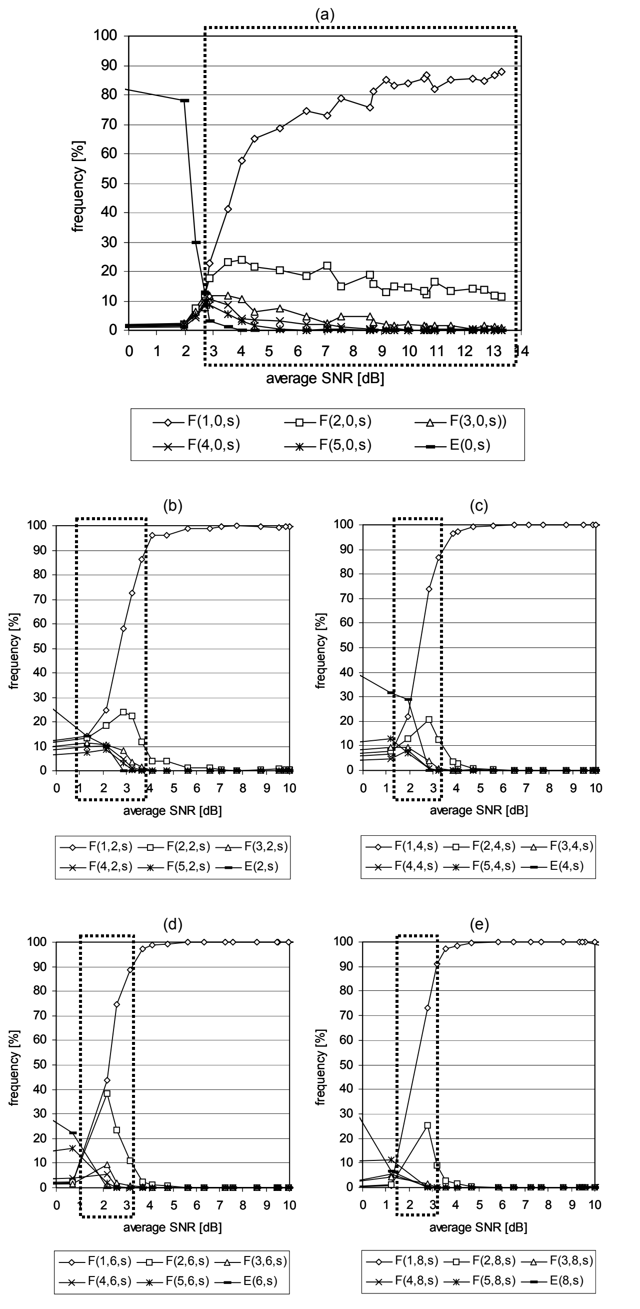

Figure 10. F(t,r,s) in a fully loaded network for the retry limits of 0 (a), 2 (b), 4 (c), 6 (d), and 8 (e).

OKs gets smaller when the retry limit increases. The difference is at most 10% (at t = 2).

From Figure 11 we observe that a retry limit higher than 2 slightly improves the percentage of transactions that deliver their 200 OK in one shot, but that it does not significantly reduce the percentage of transactions that require two or more 200 OK transmissions. A retry limit of 4 or perhaps 6 therefore seems appropriate compared to the case where the network is unloaded. As before, ISPs can use this information to dimension their 802.11b networks.

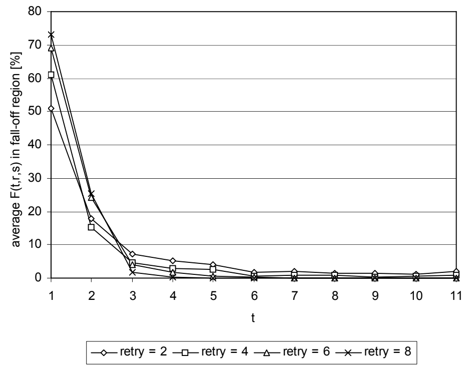

Figure 11. Average percentage of 200 OK transmissions inside the fall-off region on a fully loaded network.

We note that the number of measurements inside the fall-off region decreases as the retry limit increases. As a result, the averages in Figure 11 are based on only few transaction runs, in one case even on only one (retry limit of 8).

5.3 Transaction Delay

We reuse equation (1) to estimate the average transaction delay in a network loaded with 1 Mbps of background traffic. Equation (1) continues to hold because 1 Mbps of constant bit rate background traffic does not cause any SIP retransmissions using an 802.11 retry limit of 4 and a 'good' SNR value (around 32 dB) [17]. Since the SIP client transmits at full power in our set-up and provides a 'good' SNR at the access point, there will be no SIP retransmissions as a result of packet loss on the path from the SIP client to the SIP server. This means that the average one-way delay from the SIP client to the SIP server is still half the average delay introduced by transactions that do not involve any SIP retransmissions (i.e., half of t_{asym}).

Figure 12 shows the same type of information as Figure 8, but for a retry limit of 4 and with the network carrying 1 Mbps of constant bit rate background traffic. Figure 12a shows the average transaction delay over all SIP transactions in a run, with a blow-up of Figure 12a in Figure 12b. Figure 12c shows the average delay for the transactions without SIP retransmissions and Figure 12d shows the estimated transaction delay in a symmetrical situation using equation (1). The dashed rectangles represent the fall-off region of Figure 12c. As in Figure 8a, the average transaction delay and its standard deviation increase sharply inside the fall-off region of Figure 12a, again as a result of SIP's exponential back-off mechanism. The average delay of SIP transactions that do not involve any SIP retransmissions (Figure 12c) start at a higher value than in the unloaded network of Figure 8c (around 50 milliseconds instead of around 24 milliseconds),

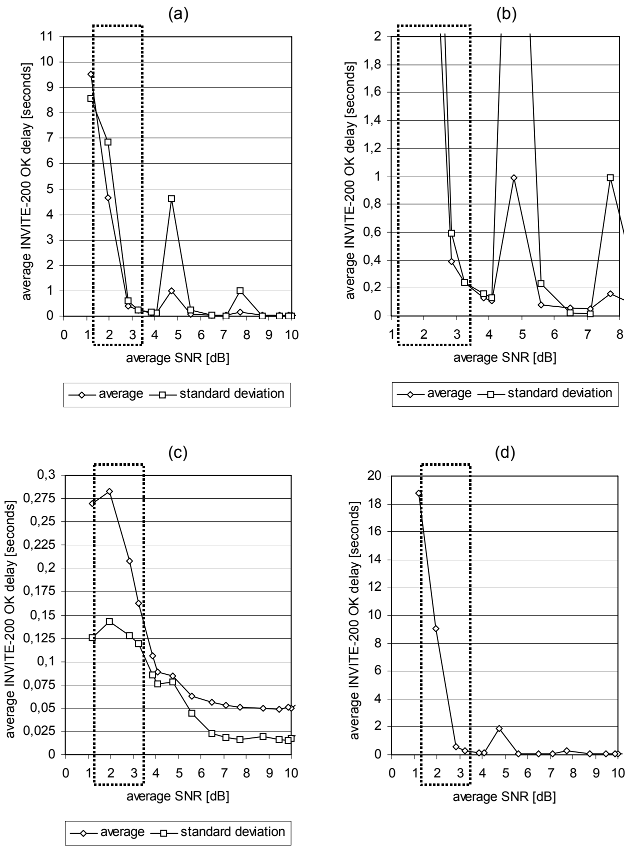

Figure 12. Average SIP transaction delay in a network loaded with 1 Mbps of constant bit rate traffic and using a retry limit of 4. Graph (a) shows the average delays for all SIP transactions (d_{asym}); (c) shows the average delays for transactions without any SIP retransmissions (t_{asym}); and (d) shows the estimated delay in a symmetrical situation (d_{sym}). Graph (b) is a blow-up of graph (a).

which is the result of the competing traffic on the network. Figure 12b shows that the edge delay over all SIP transactions (i.e., d_{asym}) is approximately 200 milliseconds at around 3.4 dB.

Similar to Figure 9, Figure 13 shows that the two closest transaction runs took place at approximately 3,85 and 3,25 dB and that they involved a few SIP retransmissions (at 0.5 and 1.5 seconds). As in Figure 9, these large but rare delay instances somewhat increased the average delay.

The edge delay for transactions without SIP retransmissions (i.e., t_{asym}) is around 140 milliseconds at 3.4 dB. Using equation (1), this means that the edge delay is in a symmetrical situation equals 260 milliseconds.

6. RELATED WORK

As far as we know, a thorough empirical investigation into the relation between SIP retransmissions and 802.11 packet loss parameters has not been reported before. Simulation and analytical approaches do however exist.

Curcio et al. [21] used a SIP emulator to measure the delay introduced by SIP transaction over lossy wireless LAN links. They focus on telephony applications and measure the delay

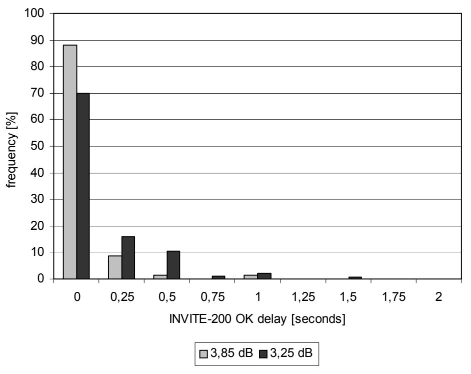

Figure 13. Frequency charts for SIP transaction delays in the asymmetrical set-up, network loaded with 1 Mbps of constant bit rate traffic.

between an INVITE and a 180 Ringing (the post dialing delay). They vary radio coverage levels (percentages) and the packet loss rate. The exact meaning of a radio coverage level is however unclear plus that they do not consider the 802.11 parameters that influence their packet loss rate. Banerjee et al. [22] take an analytical approach and use queuing models to analyze the one-way delay to transmit an INVITE from a mobile host to a correspondent host over an 802.11 link. They vary the frame error rate and the transmission rate of an 802.11 link as well as the rate at which a mobile host transmits INVITEs.

The delay introduced by SIP transactions has also been analyzed for UMTS networks [22, 23, 24] (analytically and through simulations), for UMTS satellite links [25] (simulations), and for the fixed portion of the Internet [26] (simulations).

The influence of 802.11 packet loss parameters has been empirically studied in two other (i.e., non-SIP) application areas. Aguayo et al. [18] empirically investigated the influence of 802.11 packet loss parameters in the context of multi-hop ad-hoc networks. They consider the influence of the SNR on 802.11 packet loss, but do not investigate the influence of 802.11 retransmissions (the retries) and of different levels of background traffic (i.e., they focused on F(1,0,s) in an unloaded network). Instead, they look into other packet loss parameters such as multi-path effects. Hoene et al. [12] performed an empirical study on the effects of 802.11 packet loss on the actual streaming of multimedia packets. They consider two different 802.11 transmission rates (1 and 11 Mbps) and two 802.11 retransmission thresholds (0 and 8). They use the distance between a mobile host and the access point (in meters) instead of the SNR, which is information that is not always available to applications. Our measurements are in line with those of Aguayo et al. and Hoene et al. in that they confirm that the quality of an 802.11b link is usually good and that the link only looses packets at the very edge of a cell.

7. CONCLUSIONS AND FUTURE WORK

The Session Initiation Protocol (SIP) uses an exponential back-off retransmission scheme when it runs on top of UDP over lossy (wireless) links. The default back-off time of this scheme is half a second, which means the loss of a few consecutive packets results in significant delays (e.g., the loss of two consecutive SIP packets results in a delay of 1.5 seconds). These delays are often unacceptable for applications in which the execution of a SIP transaction is time-critical (e.g., when SIP handles mobility for voice over IP sessions).

We investigated the delay introduced by SIP when it runs on top of UDP over 802.11b links. We focussed on the operation of SIP at the edge of an 802.11b cell (e.g., to update the IP address of a mobile host at a correspondent host) and experimented with different 802.11 retransmission limits, different signal-to-noise-ratios (SNRs), and different amounts of constant bit rate background traffic to determine the effects of these parameters on the delay introduced by SIP transactions. We conducted our experiments in a controlled free space environment.

Our results indicate that SIP typically introduces little delay inside an 802.11b cell, but that this delay increases sharply within a few dBs at the very edge of a cell. We refer to this SNR range as the fall-off region, which is effectively a metric for the edge of an 802.11b cell from the perspective of a (SIP) application. The width of a fall-off region depends on the configuration of the network, for instance on the number of 802.11 retransmissions and on the amount of constant bit rate background traffic. At the onset of a fall-off region, the average round-trip delay of a SIP transaction is about 152 milliseconds in an unloaded network and approximately 260 milliseconds in a network saturated with constant bit rate traffic.

In general, SIP applications should avoid transmitting messages inside the fall-off region because this will often result in delays larger than the edge delay. This is for instance important for developers of SIP applications who need to decide at which SNR value to initiate a handoff to another network (e.g., when a mobile host leaves a hotspot).

Another result of our experiments is that a maximum of four 802.11 retransmissions seems to suffice to limit the delay introduced by SIP retransmissions. This sort of cross-layer information might be of interest to ISPs to optimally dimension their 802.11b networks for delay sensitive SIP applications (e.g., voice over IP).

Future work includes conducting similar measurements in a less controlled environment (e.g., in an office space with obstacles, which may introduce fall-off regions in the interior of an 802.11b cell) and to use different types of background traffic (e.g., smaller packets or variable bit rate sources).

8. ACKNOWLEDGMENTS

Hans Zandbelt, Arjan Peddemors, Remco Poortinga, and Roy Arends helped out with various parts of the measurement set-up. The anonymous reviewers provided constructive comments on this paper.

9. REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol”, RFC 3261, June 2002

[2] E. Wedlund, H. Schulzrinne, “Mobility Support Using SIP”, 2nd ACM/IEEE Int. Conf. on Wireless and Mobile Multimedia (WoWMoM’99), Seattle, USA, August 1999

[3] J. Rosenberg, “A Presence Event Package for the Session Initiation Protocol (SIP)”, RFC 3856, Agust 2004

[4] A. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification”, RFC 3265, June 2002

[5] J. Rosenberg, H. Schulzrinne, “An Offer/Answer Model with the Session Description Protocol (SDP)”, RFC 3264, June 2002

[6] C. Hesselman, H. Eertink, I. Widya, and E. Huizer, “Delivering Live Multimedia Streams to Mobile Hosts in a Wireless Internet with Multiple Content Aggregators”, Mobile Networks and Applications (MONET), Special Issue on Wireless Mobile Applications and Services on WLAN Hotspots, Vol. 10, No. 3, June 2005, pp. 327-339

[7] J. Solomon, “Mobile IP — The Internet Unplugged”, Prentice Hall, 1998

[8] M. Li, M. Claypool, M. Rinicki, “MediaPlayer^{TM} versus RealPlayer^{TM} – A Comparison of Network Turbulence”, Proceedings of the ACM SIGCOMM Internet Measurements Workshop, Marseille, France, November 2002

[9] G. Køien and T. Haslestad, “Security Aspects of 3G-WLAN Interworking”, IEEE Communications Magazine, November 2003, pp. 82-88

[10] A. Doufexi, E. Tameh, A. Nix, S. Armour, and A. Molina, “Hotspot Wireless LANs to Enhance the Performance of 3G and Beyond Cellular Networks”, IEEE Communications Magazine, July 2003, pp. 58-65

[11] D. Eckhardt and P. Steenkiste, “Measurement and Analysis of the Error Characteristics of an In-building Wireless Network”, Computer Communications Review, October 1996, pp. 243-254

[12] C. Hoene, A. Günther, and A. Wolisz, “Measuring the Impact of Slow User Motion on Packet Loss and Delay over IEEE 802.11b Wireless Links”, In Proc. of Workshop on Wireless Local Networks (WLN) 2003, Bonn, Germany, October 2003

[13] Open SIP webpage, http://www.gnu.org/software/osip/

[14] Hostap driver, http://hostap.epitest.fi/

[15] jtg v1.69, http://www.cs.helsinki.fi/u/jmanner

[16] Kismet, http://www.kismetwireless.net/

[17] C. Hesselman, “Distribution of Multimedia Streams to Mobile Internet Users”, Ph.D. thesis, University of Twente, the Netherlands, May 2005, http://www.lab.telin.nl/~hesselma/thesis/thesis_cr_final.pdf

[18] D. Aguayo, J. Bicket, S. Biswas, G. Judd, R. Morris, “Link-level Measurements from an 802.11b Mesh Network”, SIGCOMM’04, Portland, Oregon, USA, Aug-Sept 2004

[19] R. Punnoose, R. Tseng, and D. Stancil, “Experimental Results for Interference between Bluetooth and IEEE 802.11b DSSS Systems”, IEEE Vehicular Society Conference, October 2001

[20] H. Wang, R. Katz, and J. Giese, “Policy-Enabled Handoffs Across Heterogeneous Wireless Networks”, 2nd IEEE Workshop on Mobile Computing and Applications (WMCSA 1999), New Orleans, USA, February 1999

[21] I. Curcio and M. Lundan, “Study of Call Setup in SIP-Based Videotelephony”, 5th World Multi-Conference on Systemics, Cybernetics, and Informatics (SCI 2001), Orlando, Florida, USA, July 2001

[22] N. Banerjee, W. Wu, K. Basu, and S. K. Das, “Analysis of SIP-Based Mobility Management in 4G Wireless Networks”, Computer Communications, Vol 27, No. 8, pp 697-707, 2004

[23] N. Banerjee and S. K. Das, “Hand-off Delay Analysis in SIP-based Mobility Management in Wireless Networks”, Proceedings of IEEE International Workshop on Wireless, Mobile Ad hoc Networks (WMAN'03), Nice, France, April 2003

[24] I. Curcio and M. Lundan, “SIP Call Setup Delay in 3G Networks”, 7th International Symposium on Computers and Communications (ISCC’02), Taormina-Giardini Naxos, Italy, July 2002

[25] V. Kueh, R. Tafazolli, and B. Evans, “Performance of VoIP Call Set-up Over Satellite-UMTS Using Session Initiation Protocol”, European Wireless Conference 2004, Barcelona, Spain, February 2004

[26] T. Eyers and H. Schulzrinne, “Predicting Internet Telephony Call Setup Delay,” Proc. 1st IP Telephony Wksp., Berlin, Germany, Apr. 2000