Signaling Performance of SIP Based VoIP: A Measurement-Based Approach

Swapna S. Gokhale    Jijun Lu
Department of Computer Science and Engineering
University of Connecticut, Storrs, CT 06269
Email: {ssg, jijun.lu}@engr.uconn.edu

Abstract—VoIP must offer the same level of performance and reliability as PSTN, if it were to replace the traditional PSTN. An important indicator of the performance of VoIP is the signaling or call set up delay. Most of the prior research efforts concerning the signaling performance consider long distance calls routed over public wide area networks. Due to the many outstanding issues associated with using VoIP over public networks, however, VoIP is considered to be a viable option primarily in the case of corporate intranets and tunnels. When VoIP is used over corporate intranets and tunnels, the influence of factors such as workload of end user devices, and the number of proxies along the routing path may be prominent and this influence needs to be determined. Also, corporate VoIP service is likely to be offered across heterogeneous network infrastructures using solutions from multiple vendors. As a result, it is necessary to be able to measure the signaling performance of VoIP independent of the underlying networking technology and the vendor-specific implementation. In this paper we demonstrate the feasibility of open, standard JAIN (Java APIs for Integrated Networks) SIP APIs, which are originally developed to foster rapid service creation across heterogeneous networks, for the purpose of measuring signaling performance of VoIP. Since the performance measurement capability is based on open, standard APIs it can be used uniformly for VoIP implementations from multiple vendors operating on heterogeneous network infrastructures. We quantify the impact of workload on end-user devices on the signaling performance of VoIP via extensive experimentation. Our results can be used to estimate the signaling performance for a given level of load prior to deployment of VoIP.

I. INTRODUCTION

VoIP (also known as IP telephony) is the transport of voice traffic by using the Internet Protocol (IP), rather than the public switched telephone network (PSTN). Compared to traditional circuit-switching technology, VoIP is an attractive choice for voice transport for many reasons [3].

To be widely accepted however, VoIP must match or exceed PSTN in terms of QoS, if it were to replace the traditional PSTN. From a user’s perspective, the quality of service comprises of the following aspects: (i) the reliability of the service, (ii) the amount of time required to set up the call, and (iii) the audio and video quality of the actual call [4]. The voice quality of a VoIP call has been the focus of many research efforts, however, the call set up delay of VoIP has received little attention [4].

In the case of VoIP, call set up delay is referred to as the signaling delay, and may be defined as the period between the instant the caller initiates the session and the instant the caller receives the message that the other party has been alerted [6].

With very few exceptions, it is the voice quality of the call routed using an IP network rather than the call set up delay of the call that has received maximum attention. Eyers et al. [4] present a simulation study which targets the call set up delay based on UDP delay/loss traces for the SIP [15] and H.323 [10] protocols. Kist et al. [6] investigate the possible sources of SIP call set up delay in 3GPP [7] and focus on the delays caused by the DNS (Domain Name System) and the message propagation delay.

Most of the recent research efforts which address the performance issues associated with signaling in VoIP do so in the context of long distance calls routed over public wide area networks. Currently, however, VoIP is not really a viable option for long distance calls over general wide area networks, since there are still some outstanding issues with reliability and sound quality primarily due to the limitations in both Internet bandwidth and current compression technology [9]. As a result, most corporations today confine their VoIP applications to their intranets. With more predictable bandwidth available than the public Internet, intranets can support full-duplex, real-time voice communications [9].

When using VoIP across corporate intranets, issues such as the workload of the user devices (due to the number of simultaneous ongoing sessions and/or other applications that may be running), number of proxies that may be used to route the call and the message processing time at the user devices as well as at the proxies will have an impact on the call set up delay. When VoIP is used over the public Internet, the impact of these factors may be insignificant due to the dominant issue of limited bandwidth. These factors, however, may be significant when VoIP is being used across corporate LANs. The other aspect which may impact the signaling performance of VoIP is the architectural shift to intelligent terminals and dumb network in case of VoIP, as opposed to dumb terminals and intelligent network as in the case of traditional PSTN. In other words, in the case of VoIP it is expected that most of the service logic will be provided by application layer software running in the user devices rather than being closely integrated with the transport capabilities implemented on the switch as in the case of PSTN. The

This research was supported in part by National Science Foundation (NSF), Award # CNS-0406376.
logics for some services over which a service provider needs tighter control for security and pricing purposes may reside on the proxy servers. These proxy servers may be placed within the trusted perimeters of the service providers. In order to enable widespread acceptance by fostering portability, the functionality in the user devices as well as in the proxies is expected to run on general purpose hardware. This is unlike the conventional PSTN, where the switch software runs on special purpose hardware that is optimized for performance.

It is thus important to assess the impact of this architectural change on the signaling performance of VoIP. Corporate VoIP is also likely to be offered across heterogeneous network infrastructures, including wireless/wireline networks and circuit-switched and packet-switched networks. In addition, implementations from different vendors may be used in conjunction for providing VoIP service. As a result, it is necessary to be able to measure the signaling performance of VoIP in a manner that is independent of the underlying networking technology as well as the vendor-specific implementation.

In this research we demonstrate the feasibility of using open, standard JAIN (Java APIs for Integrated Networks) SIP APIs [2] for measuring the signaling performance of VoIP. JAIN APIs were originally developed to shield the developers from the peculiarities of the underlying network infrastructures, and to foster rapid creation of value-added services. This research thus establishes dual-use of JAIN SIP APIs. Since the performance measurement capability is based on open, standard APIs, it could be used uniformly across implementations from multiple vendors operating across heterogeneous networking technologies. We also seek to quantify the impact of the workload on end-user devices on the signaling performance. Our results could be valuable in obtaining an estimate of the call set up delay for a given level of load prior to deployment of VoIP.

The balance of this paper is organized as follows: Section II provides a general overview of the SIP protocol. We also discuss and motivate the use of Java APIs for Integrated Networks (JAIN) SIP API for our experiments. Section III describes the experimental test bed and the experimental scenarios. Section IV presents the results of our experiments along with an analysis of the results. It also highlights some anomalies and suggests areas which need further investigation. Conclusions and future research directions are presented in Section V.

II. SESSION INITIATION PROTOCOL (SIP)

In this section we provide an overview of the SIP protocol, which is expected to be the dominant protocol for VoIP. We also motivate the use of JAIN SIP APIs for creating VoIP service.

A. Overview

Like H.323 [10], SIP is also a signaling protocol. It is an application layer control protocol that can be used to establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls [15]. Compared with H.323, SIP is a more flexible solution, simpler and easier to implement, better suited to support intelligent user devices, as well as for the implementation of advanced features. Many believe that SIP, in conjunction with the MGCP [1], will be the dominant VoIP signaling architecture in the future [3].

SIP is a peer-to-peer protocol. The peers in a session are called user agents (UAs). Figure 1 shows a typical SIP message flow between two UAs. In Figure 1, UA1 initiates the session by sending an INVITE request to UA2. UA2 will be alerted (i.e., the phone is ringing) about the request and an interim response, “180 Ringing”, will be sent back to UA1. Subsequently, UA2 answers the phone, which generates an OK response back to UA1. UA1 acknowledges this response by sending an ACK message to UA2. After this INVITE/200/ACK three-way handshake [15], the session is established and the two UAs begin to exchange data. At the end of the data exchange, UA2 sends a BYE request to UA1. Upon receiving this request, UA1 will send a “200 OK” response back to UA2 and this session is closed bidirectionally.

Call set up delay in SIP depends on a variety of factors [6]. These factors include SIP connection delay, SIP processing/queuing delay and DNS delay. The impact of DNS delays should be considered when SIP is being used for VoIP in a wide area network. However, when the SIP protocol is being used across a LAN, the delays that can be attributed to the DNS system are negligible. SIP processing/queuing delays are rather small and believed to be of minor importance in large-scale networks [6]. However, when considering VoIP over corporate LANs, the processing and queuing delays must be considered. Thus in the experiments conducted during this research, the call set up delay is mainly composed of SIP connection delay and processing/queuing delay.

B. JAIN SIP APIs

Although SIP will be adopted as a standard [8] by IETF, implementations by different vendors may not be completely compliant with the specification of SIP. This non compliance could be in part deliberate to enhance competitiveness, in part due to the misinterpretation of the specification, and in part due to the implementation of SIP across different underlying networking technologies. In order to foster interoperability between solutions of multiple vendors it is necessary that a standard API to the various implementations of the protocol.
be available. Such a standard interface will also facilitate rapid deployment of value-added services by offering the application developers a uniform abstraction and shielding them from the peculiarities of the underlying implementations. Due to the dual advantage of enabling interoperability and facilitating rapid service creation, corporations adapting VoIP are likely to use a SIP stack which implements such a standard API, rather than being tied down to a single vendor’s native implementation. Java APIs for Integrated Networks (JAIN) [13] defines an API which offers a uniform abstraction to the SIP protocol. The JAIN APIs, however, add another layer of processing to the SIP messages, and the impact of this additional processing on the call set up delay must then be considered.

JAIN SIP is the standardized Java interface to the Session Initiation Protocol for desktop and server applications [11], [12]. It is based on a provider/listener event driven model and enables transaction stateless, transaction stateful and dialog stateful control over the protocol [14]. JAIN SIP uses the Java technology, and it ensures true interoperability that is, the JAIN SIP specification enables interoperability between stacks and the interoperability of applications across stacks, often referred to as application portability [14]. Since JAIN is the only known standard API defined for the SIP protocol, we use JAIN SIP APIs for the purpose of measuring signaling performance of VoIP in a technology and vendor-neutral manner.

III. EXPERIMENTAL TESTBED AND SCENARIOS

The experimental infrastructure consists of the public implementation of the JAIN SIP API available from NIST [12]. It contains Reference Implementation (RI), TCK, examples and some basic tools for JAIN-SIP-1.1 (JSR-32 maintenance release) [12]. JAIN-SIP RI is a full implementation of the SIP specification as given in RFC 3261.

The hardware platform hosting the infrastructure comprises of two machines. The two machines have the following configuration. The first one is a Dell OptiPlex GX260 (Intel Pentium 4 processor at 2.4GHz, 1GB of RAM, 40GB hard driver and Intel PRO 1000 MT network adapter) and the other one is an IBM ThinkPad T40 (Intel Pentium-M processor at 1.5GHz, 512MB of RAM, 40GB hard driver and Intel PRO 100 VE network adaptor). Both computers are installed with Windows XP professional SP1. The two computers are connected via a 100M Ethernet connection across a LAN.

In all the experiments conducted in this research, calls are set up between two user agents UA1 and UA2. We repeat experiments for two scenarios:

**Scenario I:** UA1 and UA2 are on the same machine and they communicate with each other directly without using the LAN. While this scenario may be considered to be equivalent to a user communicating with himself/herself, and is unlikely to occur in practice, it reflects a case where the call set up delay is expected to be lowest. The call set up delay measured from this scenario can then serve as a baseline to compare the delay measured in the case of other scenarios.

**Scenario II:** UA1 and UA2 are on different machines and communicate with each other via the LAN.

The SIP call flows in both scenarios are identical to the call flows described in Figure 1. Our previous research, which studied the impact of intermediate proxies on the signaling delay [5], indicated that the signaling performance was not influenced by whether UA1 or UA2 initiated and terminated the call. As a result, in the experiments reported in this paper we assume, without loss of generality, that UA1 initiates calls to UA2. UA1 accepts the following parameters as input: (i) Distribution and the parameters of the interarrival time of the calls, and (ii) Distribution and the parameters of the holding time distribution. Using the interarrival time distribution, UA1 generates a series of interarrival times \{t_i\}, i = 1, 2, .... Based upon \(t_i\), the arrival time of each call \(S_i\) can be computed by \(S_i = \sum_{j=1}^{i} t_j\). Using the information of the call holding time distribution, UA1 also generates the holding time of each call \(d_i\). Then the ending time of each call can be computed by \(T_i = S_i + d_i\). At each arrival time \(S_i\), UA1 automatically initiates a call to UA2 by sending an INVITE request to UA2. This call is held for a duration \(d_i\). At time \(T_i\), which is the ending time of this session, UA1 sends a BYE request to UA2, which causes the call to be terminated. The transport protocol used is UDP. A test media is exchanged between the two UAs during the session. The test media is a a QuickTime video clip (available from [11]), 3546KB in size and 85 seconds in duration (audio property: 8000 sample rate, 8 bits per sample; video property: 160*120 frame size, 7.5fps). This test media was sent over and over in the established session.

In order to emulate the loading of an end-user device using a number of simultaneous ongoing sessions, the mean call holding duration \(d_i\) was chosen to be higher than the mean interarrival time \(t_i\) in all the experiments. At the start of the experiment, there is a warm-up or a ramp-up period, when the number of ongoing calls between the two UAs continue to rise. Eventually, after a steady increase in the number of ongoing sessions, the arrival and departure of the sessions occurs at a rate which causes the system to reach a quasi steady state where the number of simultaneous ongoing sessions is approximately constant with minor fluctuations. The number of simultaneous ongoing sessions in the steady state will be a function of the mean interarrival time and the mean call holding time. The signaling delay for the calls that are set up during the ramp-up period is likely to be less than the calls that are set up after the system reaches steady state. As a result, in order to avoid biasing the results, only the signaling delay measurements for the calls that are set up after the system reaches the steady state are recorded and are used in the computation of the signaling delay statistics.

The JAIN SIP stack has a logging facility provided to capture all messages (inbound and outbound) and store them in a trace file. For each UA, a trace file, which logs the messages sent and received by that UA is generated during each experiment. A Java script was written to process the trace
TABLE I

EXPECTED NUMBER OF STABLE CALLS IN EACH EXPERIMENT

<table>
<thead>
<tr>
<th>Interarrival Time $t_i$ (s)</th>
<th>Mean Call Holding Duration $d_i$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Fig. 2. Signaling delay as a function of holding duration (on same PC).

Fig. 3. Signaling delay as a function of holding duration (on different PCs).

Fig. 4. Signaling delay as a function of interarrival time (on same PC).

Referring to Table I, when mean interarrival time is less than 6 seconds, some combinations of $t_i$ and $d_i$ did not yield any valuable results. The machines used in the experiments were unable to sustain the number of simultaneous ongoing sessions that resulted at low interarrival times. As a result, the system was unstable, and the set up delay measurements exhibited wide variations resulting in a large standard deviation. The hardware configuration used in our experiments was able to sustain less than 40 simultaneous calls without any problems. Hence the signaling delay results reported in the paper were obtained only when the average number of calls in the steady state was less than 40. Figures 6 and 7 illustrate the CPU utilization for the machine which hosts UA1 for each combination of $t_i$ and $d_i$, when both the UAs are on the same and different PCs respectively. In both the cases, it can be observed that the CPU usage increases rapidly as the number of simultaneous calls increases. As indicated in Figure 6, the CPU utilization reaches close to 100% when the number of calls between the two UAs is approximately 12 in the quasi

Fig. 5. Signaling delay as a function of interarrival time (on different PCs).

files and to consolidate the processing results to determine the call set up delay.

IV. RESULTS AND DISCUSSION

In this section we discuss the experiments and their results. We also offer additional insights into the factors that might influence the signaling delay.

The interarrival time and the call holding duration was assumed to follow an exponential distribution. For both the scenarios described in Section III, experiments were conducted along two dimensions. First, the mean interarrival time $t_i$ was held constant, and the mean call holding duration $d_i$ was increased. Second, $d_i$ was held constant, and $t_i$ was decreased. An increase in the mean call holding duration and a decrease in the mean interarrival time cause an increase in the number of simultaneous ongoing sessions which results in an increase in the workload for the end user device.

As indicated in Section III, the system reaches a quasi steady state with the number of simultaneous ongoing calls is on an average constant with minor fluctuations. In Table I we summarize the expected number of calls in the quasi steady state for each combination of the mean interarrival time and holding duration. For each cell, the value is computed by dividing the holding duration by the interarrival time. For each combination of mean holding duration and interarrival time, once the system reaches a quasi steady state, the signaling delay of more than 40 calls was measured and the average signaling delay of these 40 measurements was computed.

Figures 2 and 3 illustrate the effect of call holding duration on the call set up delay, when the two UAs are on the same PC and on different PCs respectively. Figure 4 and 5 illustrate the effect of mean interarrival time on the call set up delay, when the two UAs are on the same PC and on different PCs respectively.
steady state. In Figure 7, the CPU utilization reaches close to 100% when the number of calls between the two UAs is approximately 18 in quasi steady state. The CPU utilization approaches 100% with a fewer number of simultaneous calls in the former since both the UAs are on the same machine.

Figures 2 and 3 indicate that for a fixed mean interarrival time, the signaling delay increases as the call holding duration increases. In this case, the number of simultaneous sessions also increase, resulting in an increased workload on the machines which causes an increase in the signaling delay. On the other hand, when the call holding duration is fixed, the call set up time decreases as the interarrival time increases. As the interarrival time increases, the arrival rate decreases, and the number of simultaneous sessions also decrease, resulting in a lighter workload on the machines. As a result, the signaling delay for establishing a new session will decrease. This can also be observed from Figures 4 and 5. Although the general trend in the signaling delay as a function of the workload at the end-user devices (and a function of the mean interarrival times and mean call holding times) is intuitive and expected, the results reported in this paper can be used to obtain a quantitative estimate of the signaling delay for a given level of workload prior to deployment of a VoIP solution.

The signaling delay reported in Figure 2 is higher than the signaling delay reported in Figure 3. This is contrary to what might be expected, since the communication between the two UAs in the case of two machines will be through the LAN which is expected to result in a higher call set up delay. However, since both the machines are on the same LAN and have the same network configuration, the round trip time between them is very small (typically less than 1 ms). As a result, the impact of the communication overhead through the LAN on the call setup delay is negligible. Compared to the influence of the round trip time (RTT), however, the workload on the machines exhibits a higher impact on the signaling time. This is because when both the UAs are on the same machine, the load on this machine will be higher than the load on each of the individual machines when the UAs are on different machines. Similar observations can be made from Figure 4 and 5. The CPU utilizations shown in Figures 6 and 7 confirm this effect. Hence the mean call set up delay when UA1 and UA2 are on the same machine is higher than the case when UA1 and UA2 are on different machines. This further highlights the impact of workload on the signaling delay.

V. CONCLUSIONS AND FUTURE WORK

In this paper we demonstrate the feasibility of using open, standard JAIN SIP APIs which were originally developed to foster rapid creation of value-added services across heterogeneous network infrastructures, for the purpose of measuring signaling performance of VoIP. Since the performance measurement capability is based on open, standard APIs, it can be used uniformly across implementations from multiple vendors which operate on different underlying networking technologies. We also quantify the impact of the workload on end-user devices on the signaling performance of VoIP. Our results could be used to obtain an estimate of the signaling performance of VoIP prior to deployment.

REFERENCES