SIPstone - Benchmarking SIP Server Performance[*]

Henning Schulzrinne Sankaran Narayanan Jonathan Lennox
Michael Doyle
Columbia University, Ubiquity
{hgs,sankaran,lennox}@cs.columbia.edu, mdoyle@ubiquity.net

April 12, 2002

SIPSTONE
real world real time

Abstract

SIP-based Internet telephony systems need to be appropriately dimensioned, as the call and registration rate can reach several thousand requests a second. This draft proposes an initial simple set of metrics for evaluating and benchmarking the performance of SIP proxy, redirect and registrar servers. The benchmark SIPstone-A expresses a weighted average of these metrics.

1 Introduction

SIPstone is a benchmark for SIP (Session Initiation Protocol [1]) proxy and redirect Servers (SIP Server). The benchmark attempts to measure the request handling capacity of a SIP server or a cluster of SIP servers.

SIPstone is designed to be a benchmark with a single standardized implementation and workload. The implementation performs a series of tests that generate the pre-configured workload. This workload is performed in a controlled IP telephony test bed that simulates the activities of multiple users initiating SIP calls. The workload is intended to be simple and repeatable, and exercises a breadth of system components associated with this environment such as concurrent initiation of calls by multiple users, random call arrivals and the forwarding of requests.

The benchmark currently is limited to evaluating the sustainable rate of what we believe to be a representative workload, consisting of registrations, successful forwarding and unsuccessful call

[*]This work was supported by grants from SIPcomm, Clarent Communications and Lucent as well as equipment grants from Sun Microsystems and Compaq.

attempts. The relative importance of these operations will differ significantly between different installations. For example, in a call center, each registered user will register once an hour, but probably receive on the order of 20 calls during that time. On the other hand, for a residential or office phone, the call rate will be less than one per hour, while maintaining the same registration interval. Unsuccessful calls, where the user is temporarily unreachable, will play a role for networks with primarily wireless end systems that are frequently turned off.

This benchmark is an initial attempt at characterizing server performance in a way that is useful for dimensioning and provisioning. However, it intentionally omits a number of more complicated scenarios, such as parallel or sequential forking or the processing of sip-cgi [2] or CPL [3] services. Given lack of operational experience, it is hard to predict what type of programmable services will find widespread use or how common forking proxies will be in the core of the network. (For many enterprise and residential proxy servers, performance issues are of secondary concern.)

Note that the benchmark has a number of self-imposed limitations. The benchmark does not evaluate protocol compliance or robustness. There are separate tests for parser robustness [4], while testing behavior under signaling overload is the subject of on-going work. To limit the time it takes to perform the test suite, individual tests run 15 minutes, making it possible that a server that leaks memory, for example, could evade detection.

Finally, a benchmark is only one metric out of many in evaluating servers and does not reflect functionality. For example, a server could be fast, but not support SNMP-based management and request counting.

The remainder of this document is organized as follows: Section 2 presents an overview of SIP servers, and illustrates SIP call setup with several examples. Issues involved in designing a benchmark for an Internet telephony system are discussed in Section 3. Section 4 presents the architectural features of the benchmark such as the test bed components and load generation mechanisms. Performance metrics for SIP Servers are defined in Section 5. Section 5 also details the measurement methodology used for benchmarking. Section 6 discusses issues that are to be addressed in the next revision. Section 8 presents conclusions.

2 SIP Servers

Before we look at benchmarking mechanisms, it is necessary to be familiar with the categories of SIP servers and their operation. Readers familiar with SIP may skip this section. SIP supports user mobility by allowing addresses in the form of *user@domain* to be used when making Internet calls. For instance, *alice@home.com* can call *bob@office.com* no matter what communication device, IP address or phone number bob is using currently. The locations where users could be reached are maintained by the SIP location or registration servers.

When Alice, with address *sip:alice@home.com*, wants to call Bob, *sip:bob@office.com*, her SIP phone contacts the server at *office.com*. The server knows where Bob can be reached and can either return Bob's location to Alice's phone (so-called redirect mode) or can itself try to contact Bob at his current location (proxy mode). In the former case, Alice's phone retries the new location, while in the latter case the server proxies the request, transparent to the caller. It is possible to encounter multiple SIP servers (either in redirect or proxy mode) in a given call attempt. A *forking proxy* can fork the call request to more than one locations, so that the first phone that is picked-up gets the call, while all the other phones stop ringing.

The list of audio/video algorithms supported and the transport addresses to receive them, are described using Session Description Protocol (SDP [5]) messages, carried as the body of SIP requests and responses.

3 Issues in Designing a SIP Benchmark

The goal of any benchmarking system is to help understand how a system will perform, and to help compare different implementations. The primary SIP performance metric is the number of requests that a system can process successfully within a given delay constraint, while the request processing delay itself is of somewhat secondary importance as long as it remains below the acceptable "post-selection delay" and "answer signal delay" [6]. For informational responses, the delay needs to be sufficiently short to avoid triggering UDP retransmissions, i.e., about 500 ms minus any network delays. The performance of a SIP server depends not only on the implementation but also the size of the user population serviced, the number of DNS lookups required, the transport protocol used (UDP or TCP), and the type and statistical arrival process of requests submitted. We discuss some of these considerations in more detail in the sections below.

3.1 User Population

The performance of a SIP server that has per-user information is likely to depend on the size of the user population being reached by the server, since a typical SIP transaction involves looking up the contact addresses of a given user in a database of some kind. There are at least three different approaches for implementing such a database, including an in-memory table without persistent storage, in-memory caching with disk write-through and a database located on a networked server. Depending on the cache size and the number of users registering, the performance will differ dramatically and will be determined largely by the back-end database, less by the server. Since it appears likely that the user population served by carrier-scale SIP servers is likely to be large, measured in thousands, our benchmark attempts to model this scenario. However, the evaluation of a server needs to ascertain whether the user population in the benchmark is a fair representation of the actual user population size and whether the server performs in-memory caching that may decrease in effectiveness for a larger population.

However, readers should be aware that different database architectures have trade-offs other than performance. For example, web-based user management may require the use of networked databases. Also, in-memory databases, while faster, generally cannot maintain registration information across system reboots.

Finally, for outbound proxy servers, database interactions are limited to authentication lookups, rather than registration lookups. However, we believe that outbound proxy servers are typically installed in corporate networks and thus have a lower call volume. (For example, if we estimate that during the busy hour, one fourth of the IP phones are in use and that a business call generally lasts three minutes, we would expect a call volume of about 5 BHCA (busy hour call attempts) per user, or about 100,000 outbound calls during an hour for a large campus with 20,000 phones. This is about 28 call attempts per second, which is likely to be well within the capacity of most servers.) We thus treat outbound proxies, with authentication, as a separate case.

3.2 Request Modeling

Traditionally, a Poisson process[1] is used to characterize PSTN call arrivals [7]. For system sizing purposes, generally the busy hour call volume is used as a requirement, indicating the time-average arrival rate during the busiest hour of the busiest day or an average of the n busiest days of the year [8].

The time between setup and teardown is the conversation time or the call holding time. The exponential distribution is the traditional method of approximating call holding times in a switch, although it is only modestly accurate [9]. Since SIP proxy servers mostly maintain transaction state, not call state, we can largely ignore the call duration statistics. More important for SIP servers is the time between call arrival and answer, as it determines the number of transaction records that the server needs to maintain. Using the statistics from [10], about 70% of the calls are answered, in roughly 8.5 seconds, while unanswered calls ring for 38 seconds. Thus, an average INVITE transaction duration of 20 seconds is chosen.

For call stateful (user agent) servers, the shape of the distribution of call durations may have some influence on the number of calls present in the system, but is not likely to affect the call throughput.

3.3 Standard Call Transaction

Some Internet services, such as HTTP, DNS and LDAP, have a very simple request-response structure, where measuring the response time is fairly straightforward. However, SIP requests often generate multiple provisional responses and may even generate, when forking, multiple final responses. The "100 Trying" response needs to be returned soon enough to avoid retransmissions of the request, as excessive retransmissions will quickly lead to server congestion collapse. Hence, in the request mix used to benchmark a server, we need to consider the response time of a full transaction such as INVITE, "100 Trying" or "180 Ringing" provisional response and the "200 OK" final response.

3.4 Other SIP Issues

Other SIP specific issues include the use of different transport protocols such as TCP and UDP. When using TCP, issues such as the ability of the server to handle a large number of open connections and its use of persistent connections need to be considered. We currently do not have a good estimate as to the typical number of upstream servers reaching a high-volume proxy server. If the number is low, almost all such servers will use persistent connnections, while, for example, an outbound proxy server serving a typical office may see a new TCP connection for every request. Until we have more operational experience, we have each thread in our load generator maintain a persistent connection to the server under test.

When UDP is used, we can ignore the effect of network packet losses for requests, since they do not affect server load, but need to consider retransmissions of requests and final responses from downstream servers if the initial response or the ACK request are excessively delayed by the server under test. For simplicity, we ignore all network packet losses, since it appears likely that operational networks will have to have very low losses, typically below 5%, to achieve good voice quality and acceptable call setup delays.

Mechanisms such as forking and Via-header processing impose higher processing demands on the proxy server. Forking in SIP Proxy Servers was discussed in Section2. When a chain of proxy servers

[1]A Poisson arrival process is equivalent to an exponential interarrival time.

are used, handling of the Via header becomes critical in processing the message. The workload should be able to make the proxy server perform such computations. In this version of this document, we do not consider forking and Via-header processing. In addition to the above, size of the SDP used with SIP requests has the potential to affect performance. A large SDP may result in several datagrams when UDP is used.

3.5 General Benchmarking Considerations

Other relevant issues to consider when designing a benchmark include repeatability, measurability, and scalability. Repeatability refers to the ability to replicate the results by a third-party when running the test under the specified environment. Measurability refers to the ease with which the measured test results can be quantified in a way to present a picture of the system being measured. The ability of the benchmark to run on different configurations such as simulating a large number of users, different arrival distributions refers to scalability. We elaborate on these issues further in the next section.

4 SIPstone

SIPstone describes a workload for SIP requests in proxies using a set of tests that exercise various components of typical SIP servers.

4.1 Architecture

The "server under test" (SUT) is a SIP proxy, redirect or registrar server whose performance is to be estimated. The benchmark consists of a set of SIPstone load generators that create the SIP request load, a call handler that simulates a user agent server and a central benchmark manager ("coordinator") that coordinates the execution of the benchmark, and the SUT. Actual measurements are based on the controlled workload profile described in Section 5. The call handlers may run along with the load generators or on different systems.

4.1.1 Server under Test

The SUT consists of the host system(s), including hardware and software, required to support the SIP telephony server and any other components including database(s), if applicable. All network components (hardware and software) between host machines which handle intra-SUT application communications are part of the SUT. The software consists of the SIP redirect and/or proxy server. This software is referred to as the SIP server in the discussion below. For simplicity, this document considers the registrar server to be co-located with the SIP server.

For the test, file-based logging must be enabled, so that each request is written to a log file. If the system does not support file-based logging, an equivalent logging mechanism must be enabled and this fact be noted in the description.

4.1.2 User Agent Client (UAC)

SIP requests for the benchmark are generated by user agent clients (UACs). Depending on the request rate needed, one or more clients may be needed to generate the requests. The requests are directed to the UAS (Section 4.1.4). If only one UAC is used, it is configured with the following information:

- the interarrival time for requests for each host;
- the type of request (INVITE (implies BYE) or REGISTER),
- the transport protocol (UDP or TCP),
- the number of requests to generate.

The user names are chosen as counting from A000000.

4.1.3 Coordinator

If more than one test UAC is needed, a coordinator reads the input benchmark configuration file. The benchmark configuration file contains

- the names of the UAC hosts;
- the names of the UAS hosts;
- the interarrival time for requests for each host;
- the type of request (INVITE (implies BYE) or REGISTER),
- the transport protocol (UDP or TCP),
- the number of requests to generate,
- the starting user name (e.g., A013878).

The coordinator invokes the UACs and UASs on each of the participating computers using mechanisms such as rsh/rcmd or Java RMI. Each UAC generates SIP requests with a Poisson interarrival distribution and records the response delay, i.e., the time between sending the request, the arrival time of the 1xx response, if any, and the arrival time of the 2xx response.

4.1.4 User Agent Server (UAS) – Call Handler

The UAS is needed for the "Proxy 200", "Outbound proxy" and "Redirect" tests (see Section 4.2 below). For the "Outbound proxy" and "Proxy 200" tests, the UAS answers incoming INVITE and BYE requests. One or more UAS are used. Upon receipt of an INVITE request, they answer immediately with a "180 Ringing" response, followed instantly by "200 OK". For the "Redirect" test, the UAS only registers its location.

The test UAS MUST be able to respond, under load, to an incoming INVITE or BYE within 100 ms.

Prior to the benchmarking run, the UAS registers with the registrar. For reproducibility, the UAS registers the names A000000, A000001, A000002, etc., with the number of registrations large enough that each user name will be used only once.

4.2 Description of Tests

The benchmark currently consists of five tests. Each test is performed using both UDP and TCP, with results reported separately for each transport protocol. Thus, a total of ten results is to be reported for servers that support both transport protocols. The individual tests are:

Registration: Registrations of the form shown in Fig. 4 and Fig. 5 are sent by the load generator, using digest authentication, corresponding to call flow 2.1.1 [4]. The user accounts have been established previously in a manner depending on the SUT. For simplicity, account name and user secret are typically chosen to be the same. The message flow is shown in Fig. 1. The transaction delay is measured from sending the first REGISTER request to receiving the final 200 response, including the 401 "Unauthorized" message.

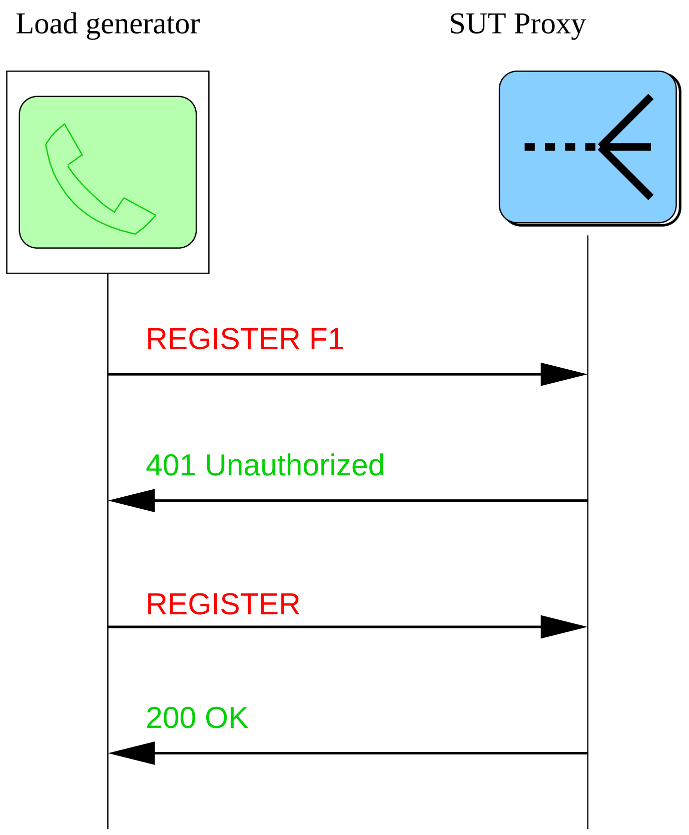

Figure 1: Registration test

Outbound proxy: The load generator sends an INVITE request to the SUT acting as an outbound proxy. The message has the format shown in Fig. 6 and is destined to the UAS. The message flow is the same as for the "Proxy 200" test below.

Redirect: The load generator sends an INVITE request to the SUT acting as a redirect server, using the message format in Fig. 6. It is assumed that all call destinations have registered with that server. The message flow is shown in Fig. 2. The transaction delay is measured from sending the INVITE request to receiving the 3xx response.

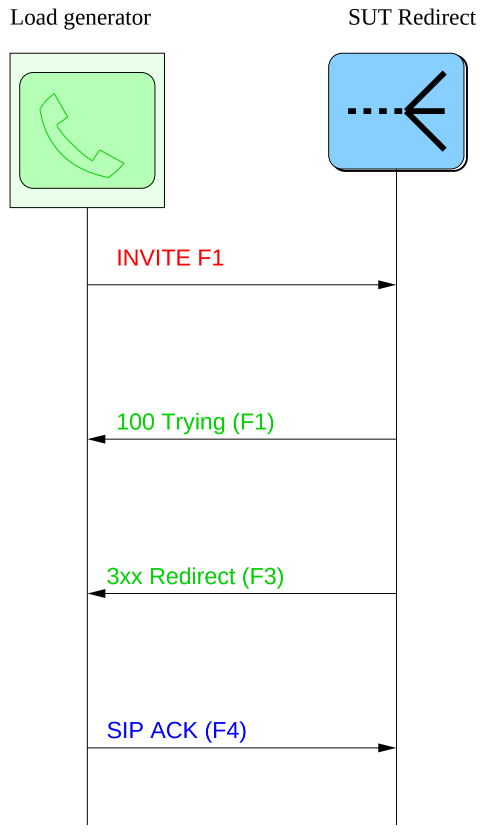

Figure 2: Redirect test

Proxy 480: The load generator sends an INVITE request to the SUT acting as a redirect server, using the message format in Fig. 6. The call destinations are known to the server, but have not registered, so that the server returns a 480 (Temporarily unavailable) response. The request is not authenticated.

Proxy 200: The load generator alternates between INVITE and BYE transactions to the SUT acting as a non-forking proxy server. The BYE transaction is sent immediately after the INVITE transaction completes. The message flow is shown in Fig. 3. The requests are not authenticated.

The server should insert a Record-Route to reflect real operational conditions, however, the operation of the system does not depend on this since the caller load generator always sends the request to the SUT.

The TRT is measured only for the INVITE request.

The first REGISTER request is shown in Fig. 4, the second, authenticated REGISTER in Fig. 5 while the INVITE request is shown in Fig. 6.

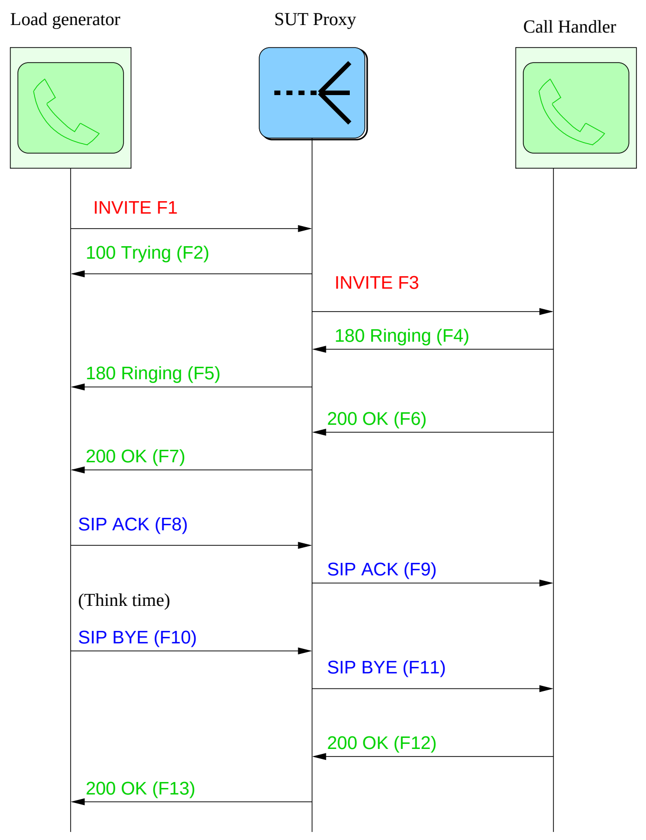

Figure 3: Proxy 200 test

```
REGISTER sip:registrar SIP/2.0
Via:  SIP/2.0/UDP origin
From:  <sip:>;tag=1
To:  sip:registrand
Call-ID: call-id value
CSeq:  1 REGISTER
Contact:  <sip:IP address>
Expires:  7200 Content-Length:  0
```

Figure 4: Initial REGISTER request

5 Benchmarking Methodology

5.1 Environment

SIPstone benchmarking consists of a series of test runs, with increasing load levels generated by the load generators, and targeted at the SIP server being tested.

```
REGISTER sip:registrar SIP/2.0
Via:  SIP/2.0/UDP origin
From:  <sip:registrand>;tag=1
To:  sip:registrand
Call-ID: call-id value
CSeq:  1 REGISTER
Contact:  <sip:IP address>
Expires:  7200 Authorization:Digest username="registrand",
realm="SIPstone", nonce="ea9c8e88df84f1cec4341ae6cbe5a359",
opaque="", uri="registrar", response="dfe56131d1958046689cd83306477ecc"
Content-Length:  0
```

Figure 5: Authenticated REGISTER request

```
INVITE sip:destination SIP/2.0
Via:  SIP/2.0/UDP caller address
From:  Alice <sip:destination>;tag=17
To:  Bob <sip:origin>
Call-ID: 602214199@mouse.wonderland.com
CSeq:  1 INVITE
Contact:  Alice <origin>
Subject:  SIP will be discussed, too
Content-Type:  application/sdp
Content-Length:  187

v=0
o=user1 53655765 2353687637 IN IP4 128.3.4.5
s=Mbone Audio
t=3149328700 0
i=Discussion of Mbone Engineering Issues
e=mbone@somewhere.com
c=IN IP4 caller IP address
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 6: INVITE request

5.2 Definition of Terms

Measurement Interval (MI): The **measurement interval** is defined as the steady state period during the execution of the benchmark from which the reported performance metric is derived. During the measurement interval, the UACs generate SIP requests.

Transaction Response Time (TRT): The transaction response time is defined as the time elapsed from the first byte sent by a UAC to initiate a transaction until the last byte received by the UAC that ends the transaction. For the “proxy 200” test case, a transaction can be either the INVITE or the corresponding BYE transaction, but only the INVITE transaction is measured.

Registrations per second (RPS): **Registrations per second** is defined as the average number of successful registrations per second during the measurement interval.

Calls per second (CPS): **Calls per second** is defined as the average number of calls per second completed with a 2xx or 4xx response during a measurement interval. For the “proxy 200” test case, a single call includes both the INVITE and corresponding BYE transaction.

Transaction failure probability (TFP): The transaction failure probability is the fraction of transactions that fail, i.e, where the server does not return a provisional or final response within the time limit or where the response is a 5xx response.

In the future, we may define a cost metric, \$/CPS or \$/RPS, which measures the cost, in U.S. dollars, per call or registration per second, including all server hardware and software. (A similar metric is commonly used for transaction processing systems. SIP server functionality can be spread relatively easily across a number of network hosts, so that this metric is useful for network planning.)

5.3 Measurement Methodology

To determine the RPS and CPS values, the request rate is increased until the TFP increases to 5%, evaluated across a measurement interval of 15 minutes. The highest sustained throughput is reported as the benchmark number. To speed up the benchmarking process, the measurements that probe for the capacity can be shorter than 15 minutes, with five minutes generally sufficient. For more detailed benchmarks, the TRT as a function of the transaction rate should be plotted, with at least four measurements recommended, including one at a load below 10% of the maximum RPS or CPS. The latter value is useful to estimate the idle-server TRT.

Prior to each measurement interval, the SUT is initialized, as necessary, with the user database and registrations.

The test operates in an “open loop” mode, where the arrival of the $n+1$st request does not depend on the completion of the nth request for the same UAC[2]

5.4 Scaling Requirements

The size of the user population used for the tests should scale with the request handling capacity of the server. Assuming an average of one hour for registration intervals and inter-call intervals yields a user population of 3.6 million for a server capable of handling 1000 calls or registrations per second. However, for reasonable measurement intervals, we cannot reach this number of destinations, and are likely to spend an inordinate amount of time priming the system with user data. Thus, instead we simply require that each request during the measurement interval has a different user name, except that the BYE request in the “Proxy 200” measurement corresponds to a previous INVITE request.

[2]In a closed-loop test, each new request is only issued after the completion of the previous one and a suitable waiting time.

5.5 Response Time Constraints

During each measurement interval, at least 95% of call setup requests of each type must have a TRT of less than the constraint specified (in milliseconds) in Table 1. The values except for the time-to-200 threshold are designed such that they are less than the SIP retransmission default ($T1$) timer of 500 ms.

The time-to-200 threshold depends on the call model. E.721 [6] suggests a 95th percentile value of 6.0 seconds for local ISDN calls under normal load (8.0 seconds for toll and 11.0 seconds for international connections), assuming between one and 10 nodes. One model is that SIP calls will traverse at least two proxy servers (outbound and remote) and possibly more (e.g., in 3GPP), so that a conservative upper bound of four servers appears reasonable, for both "toll" (i.e., inter-provider) and international calls. This leaves a delay budget of 2.0 seconds per server.

Instead of triggering a retransmission when the timer elapses, the worker marks it as failure and continues with the next iteration. The average TRT of a measurement interval is computed by averaging over the successful (timely) responses.

Scenario	Response	Response Time (ms)	Description
Figure 1	400	200	Final response before retransmission occurs
Figure 3	1xx	100	Initial 1xx (if applicable) in all scenarios
Figure 2	non-1xx	400	For redirect servers, before retransmission occurs
Figure 3	200	2,000	For a simple SIP call through the proxy.

Table 1: Allowed Response Times

5.6 Metrics and Parameters to be reported

When presenting SIPstone results, the following data has to be provided. The configuration parameters are constant over a test run.

- Description of the clustering configuration, including the number of hosts and the load balancing mechanism used (e.g., DNS SRV or stateless proxy);
- All aspects of the server hardware, in particular
  - the CPU count, type and speed,
  - the memory configuration,
  - the network interface type and speed,
  - the disk and disk controller configuration;
- the server operating system and version and any non-standard tunings or settings, e.g., for network parameters;
- the server software configuration, including
  - the number of servers used,
  - the type of location server (e.g., a local database, in-memory, or a network server) and whether it was located on the same host as the proxy server,

  - request logging used (note that basic file-based logging is required by default),
  - network management features in use;

- the number and type of SIPstone load generators that participate in the run, and the number of worker threads created in each generator;
- the number and type of network segments connecting the load generators to the SIP server;
- the arrangement of load generators on the network segments;
- the workload parameters except the load level, which may be increased across several test runs.

To remove one variable from the measurement results, the server and load generators should be connected by a switched 100 Mb/s Ethernet that only carries measurement-related traffic and no other LAN traffic. This ensures that network loading does not become a bottleneck. (At 100 calls a second, with 1500 bytes each, the data rate for signaling is about 1.2 Mb/s.)

The following benchmark results should be reported:

- The number of connections requested by the clients and accepted by the SUT per second. The intent is to count only the number of new connections made successfully by the clients in generating the load for the benchmark.
- CPU and memory utilization of server at various loads;
- A curve plotting the TRT as a function of request arrival rates, with at least four plot points and one value at approximately 10% of the capacity;
- CPS and RPS.

When reporting results for server clusters, both the total, aggregate CPS/RPS and the per-server rate should be reported. For clusters, it is suggested to include a reasonable sample of cluster sizes, e.g., in geometric progression (1, 2, 4, 8, . . .).

We also define an initial composite benchmark called **SIPstone-A** that weighs the ten processing rate metrics as follows:

Test	weight UDP	weight TCP
Registration	0.20	0.05
Outbound Proxy	0.10	0.05
Redirect	0.10	0.05
Proxy 480	0.10	0.05
Proxy 200	0.20	0.10

The number is to be reported as a unit-less number, for example, “1234 SIPstone-A”.

6 Limitations

Given the large number of test results, it may be worthwhile to create two composite numbers, namely a “call center” number and a “mobile users” throughput. For the call center metric, we assume that there are 20 calls per registrations, while mobile users have one registration per call. Separate metrics are to be reported for redirect and proxy mode, since many servers are operating as one or the other.

It remains to be evaluated whether call metrics can be combined linearly. For example, a server that can handle 1000 registrations/second or 200 redirections/second should also be able to handle a mixed load of 500 registrations/second and 100 redirections/second or 800 registrations/second and redirections/second.

More complicated scenarios, such as invocations of SIP servlets, sip-cgi and CPL scripts, need to be considered. However, given that processing requirements for each script can be effectively unbounded, depending on the internal computation and external references needed, a set of simple services will need to be proscribed. For example, a simple time-of-day call forwarding mechanism may be indicative of the efficiency in script invocation and processing.

Also, future tests will need to incorporate SSL and IPsec scenarios if these become practically important. Currently, few systems appear to use these security mechanisms.

7 Summary of Requirements

Below, R is the request handling rate.

Requirements	value	unit
User population	900 max(RPS, CPS)	
Call holding time	0	s
Maximum “100” response time	0.1	s
Maximum “200” response time	2	s
Run time	900	s
Transaction failure prob.	1	%

8 Conclusions

This draft summarizes an early benchmarking scheme for SIP telephony servers. We discussed several issues to be considered while designing benchmarks for SIP servers, and presented an architecture and a technique to measure server performance.

References

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, “SIP: session initiation protocol,” Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[2] J. Lennox, H. Schulzrinne, and J. Rosenberg, “Common gateway interface for SIP,” Request for Comments 3050, Internet Engineering Task Force, Jan. 2001.

[3] J. Lennox and H. Schulzrinne, “CPL: a language for user control of internet telephony services,” Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[4] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, D. Willis, J. Rosenberg, K. Summers, and H. Schulzrinne, “SIP telephony call flow examples,” Internet Draft, Internet Engineering Task Force, Apr. 2001. Work in progress.

[5] M. Handley and V. Jacobson, “SDP: session description protocol,” Request for Comments 2327, Internet Engineering Task Force, Apr. 1998.

[6] International Telecommunication Union, “Network grade of service parameters and target values for circuit-switched services in the evolving isdn,” Recommendation E.721, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1999.

[7] J. Bellamy, *Digital Telephony*. New York: John Wiley & Sons, 1991.

[8] F. Schmidt, F. G. Lpez, K. Hackbarth, and A. Cuadra, “An analytical cost model for the national core network,” consultative document, Wissenschaftliches Institut fr Kommunikationsdienste, Apr. 1999.

[9] Common Carrier Bureau, “Trends in telephone service,” tech. rep., Federal Communications Commission, Washington, D.C., Dec. 2000.

[10] F. P. Duffy and R. A. Mercer, “A study of network performance and customer behavior during-direct-distance-dialing call attempts in the USA,” *Bell System Technical Journal*, vol. 57, no. 1, pp. 1–33, 1978.

9 Acknowledgements

Bertrand Farabet (Sun) provided valuable comments.