ACME: An automated tool for generating and evaluating the quality of VoIP calls

Leandro C. G. Lustosa, André A. D. P. Souza, Paulo H. de A. Rodrigues, Douglas G. Quinellato[1]

Laboratório de Voz Sobre IP – Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro (NCE/UFRJ)*
Caixa Postal 2324 – 20001-970 – Rio de Janeiro – RJ – Brasil
leandro@instant.com.br, {andreabrantes,aguiar}@nce.ufrj.br, douglas@las.ic.unicamp.br

Abstract. ACME, an automated tool for generating and evaluating the quality of VoIP calls, is presented. ACME examines the availability of bandwidth and processing capacity to determine the largest number of simultaneous calls that a node can operate in a given topology, besides supporting pre-programmed schedule of VoIP experiments. Demos of ACME usage as a capacity estimation tool and as an instrument for monitoring of an IP telephony production service show its effectiveness and versatility.

Keywords: monitoring tool, VoIP call generation, capacity determination.

1 Introduction

ACME (Automatic Call Measurement Environment) is a tool capable of performing two basic activities: generation of VoIP calls and measurement of the quality of the generated calls. The tool uses the potential of these two activities to build an automated environment supporting two classes of experiments. The first class, called stress experiment, is primarily used for determining the ability (in number of simultaneous calls) that a medium (wired or wireless) or VoIP system (service or set of servers) can support without compromising voice quality below a previously established level. The second class, called periodic experiment, is used to check the change in call quality over time, even allowing VoIP service availability monitoring.

ACME development was based on a voice quality evaluation infrastructure proposed by [13] which assumes that VoIP clients (IP phones or gateways) are able to assess the quality of the incoming voice flow and provide a specific call detailed record named VQCDR (Voice Quality Call Detail Record) . VQCDR contains several voice quality indicators, besides parameters for call and involved terminals identification. Additionally, it may include a report with the history of quality

* Partially funded by the Brazilian National Education and Research Network (RNP). Paulo Rodrigues is also a professor at DCC/UFRJ. Douglas Quinellato is a member of LASS at IC/UNICAMP. Leandro Lustosa is presently at Instant Solutions (www.instant.com.br).

indicators status throughout the call. VQCDR quality indicators are computed according to the E-Model and its extensions [9, 6, 7].

Voice degradation can be caused by several factors [12] such as network packet losses, losses due to late arrival in jitter compensation buffer, extreme large end-to-end round trip delay (influencing interactivity), factors related to codec (inherent quantization errors, robustness to single or multiple losses, loss compensating algorithms, voice activity detectors, etc), among others. E-model output is a scalar factor R, which can be correlated to MOS (Mean Opinion Score) [10]. MOS is a voice quality score representing the average opinion of a group of unbiased testing listeners and varies from 1 (poor) to 5 (excellent).

E-model computation and MOS determination is performed by a library called VQuality (Voice Quality Library) [14]. VQuality was developed in C + + and has been integrated to IP clients to enable VQCDR generation. The use of VQCDR allows ACME to analyze a call in detail and plot timely graphics of results.

Evaluating a system or transmission technology for voice capacity is a more complex task than the evaluation of a simple call. To determine the capacity of a link in carrying good quality VoIP calls, for example, several rounds of calls are needed. ACME uses an automatic mechanism for dynamic call generation and achieves an estimate of VoIP call capacity when overall MOS for the increasing number of simultaneous calls falls below an acceptable level. Sizes of confidence intervals are used as stopping times for the call generation process.

Several commercial tools enable the evaluation of VoIP call quality [17, 5, 2, 16], but none can be found that enables automatic capacity determination of links and servers. With the cited tools it is possible to get somehow the capacity of a medium, but it would require that all calculations and call executions be carried out manually, what would be laborious and prone to human error when large numbers are involved.

This paper is organized into eight sections. In Section 2, the infrastructure and architecture for evaluation of VoIP calls is described. In Section 3, the architecture of ACME and its components are presented. In Sections 4 and 5, the concepts of stress and periodic experiments are discussed. Section 6 shows how experiments are carried out concurrently. In Section 7, the use of ACME is demonstrated and, finally, Section 8 presents the conclusions and future work

2 Architecture for Evaluating VoIP Calls

According to the architecture proposed in [13], VQCDRs are collected by an entity called VQCDR Server, which is in charge of interpreting, authenticating and forwarding VQCDRs to a data base storage. The VQCDR server has three modules:

Collector Module (CM): responsible for collecting and interpreting the VQCDR, besides acting on the Authenticator Module (AM) and the Storage Module (SM).

Authenticator Module (AM): responsible for validating VQCDRs received by CM. For testing and measurements in basic experiments or controlled environments, VQCDR server offers a simple access list based on IP addresses. However, in

production environments, where a more sophisticated and flexible validation mechanism is required, the AM module is the solution.

Storage Module (SM): responsible for storing VQCDRs collected by CM. The storage can be deployed in a SQL data base or Radius Server.

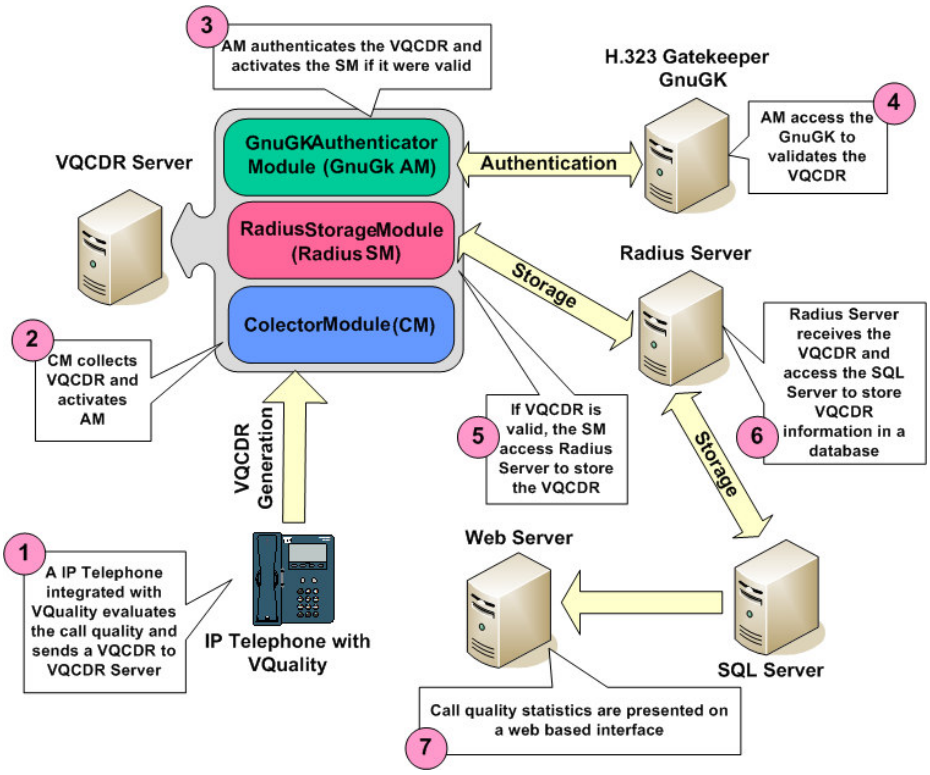


Fig. 1. UFRJ VoIP Call Quality Evaluation Architecture.

Fig. 1 illustrates general operation of the architecture: At the end of a call, an IP phone integrated to VQuality evaluates the received voice quality and then generates a VQCDR, which contains voice quality indicators and call identifiers (Fig. 1-1). The CM module of the VQCDR Server collects the VQCDR (Fig. 1-2) and instructs the AM module to check its legitimacy (Fig. 1-3). In case of H.323 architecture using GnuGK [8], authentication is performed by a specific module (Fig. 1-4). If VQCDR validation is successful, module SM is instructed to store the VQCDR. When using RADIUS, VQCDR is sent to a RADIUS Server (Fig.1-5). The RADIUS server stores VQCDR data in a database, for example an SQL server (Fig. 1-6). A Web interface then allows displaying graphics and statistics reports (Fig. 1-7). ACME uses the whole VQCDR collecting architecture and adds new elements described in section 3.

3 ACME Architecture

ACME has a master-slave type architecture. A central element called ACME Master (or simply master) coordinates a series of peripheral ACME Slaves (or simply slaves), instructing them to initiate (said an active slave) or get ready to receive calls (said a passive slave). An active slave can start a call to a non slave destiny as an IP phone to check VoIP system availability and operational status.

A passive slave has two elements: a VoIP client softphone (VC) with VQuality integration, and an interpreter for interacting with the master through the ACMEp protocol (to be described in section 3.2). It is expected a slave being able to generate multiple simultaneous calls. However, slave hardware has limited memory and processing power what limits the maximum number of calls that can be generated without impairing voice quality, which is named slave capacity. Over a threshold, longer delays are incurred in the coding/decoding process and discards in jitter buffers start to happen. How to determine a slave capacity is addressed in section 3.1.

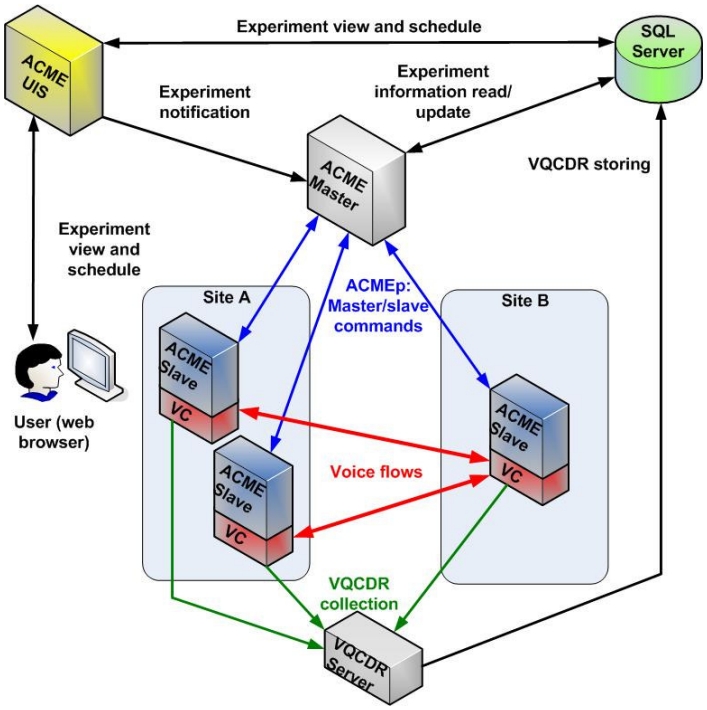


Fig. 2: ACME Architecture.

To determine the capacity of a system or slow speed communication link, at least one active and one passive slave are needed. The site concept allows using a set of slaves of inferior hardware to generate a larger number of simultaneous calls to stress a superior hardware system or a very high speed link. Experiments are always realized between two sites hosting one or more slaves.

ACME has a Web interface running in an HTTP server called User Interface Server (UIS), allowing user to schedule experiments and visualization of outputs.

ACME architecture and its components are shown in Fig. 2. When a user schedules a new experiment via Web interface, the experiment data is stored in an SQL database (PostgreSQL is used) and the master is informed that a new experiment has been filed. When starting the experiment at the programmed time, the master reads experiment configuration parameters and orders slaves in the two sites involved in the action to perform calls and collect VQCDRs.

UIS enables the user to receive via e-mail status reports about its experiment, when certain events occur: an error is detected; there is an unreachable slave; or measured voice quality is in an alarming level, possibly indicating that a system being monitored by a periodic experiment (fired in pre-scheduled times) is in critical condition.

ACME core was developed in C++ and UIS in PHP. ACME is aimed toward Linux, but any other Unix systems may be used. Voice streams are prerecorded and have three minute duration, long enough for detecting voice degradation in all situations. Nevertheless, prerecorded messages of any length can be used.

3.1. Slave Capacity Determination

Stress experiments must be performed to determine slave capacity and avoid getting imprecise results. In capacity determination, the number of simultaneous calls is gradually increased while call quality is kept at maximum optimal value. The stopping condition is the start of call degradation. There are three possible scenarios: *two slaves with equivalent computational power hardware* - quality degradation occurs symmetrically, and capacity is valid for both slaves; o*ne superior slave and one inferior slave* - quality degradation occurs asymmetrically and only the inferior slave capacity is determined; *one slave S with much superior capacity and a set of slaves with overall capacity higher than S* - degradation occurs asymmetrically and only S has its capacity determined. In capacity determination experiments, available network bandwidth has to be enough to force slave hardware to be the limiting factor.

3.2. ACMEp Protocol

ACMEp is the protocol used in the communication between master and slave. Master commands passive slaves to get them ready to receive a certain number of calls, or order active slaves to start calls to specific destinations. In this process, call configuration data is also informed: codec, duration and destination (IP address and port, E.164 number or user alias). Master can also instruct slave to register in a specific server, making it possible to test a server or an environment associated to a server. In this last case, server address and account/password for registration are also passed to the slave. The protocol has only three messages:

Start: From master to slave. It commands slave to get ready for receiving calls, initiating calls or registering to a server.

Status: From slave to master. It allows a slave to inform its master if a call was successful or not. Passive slaves also use status message to inform when they are ready to receive calls.

Cancel: From master to slave. It allows a passive slave to inform that an active slave call is canceled. Basically, *Cancel* cancels a *Start* message previously sent.

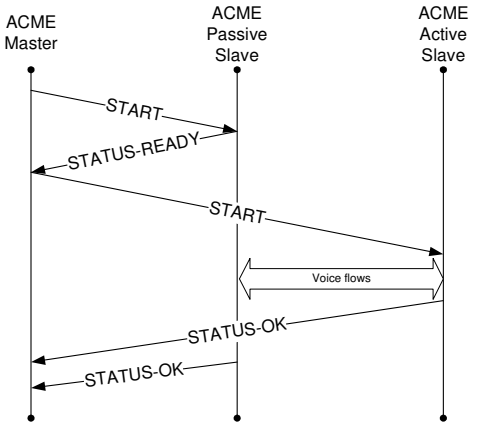


Fig. 3. Successful slave to slave exchange.

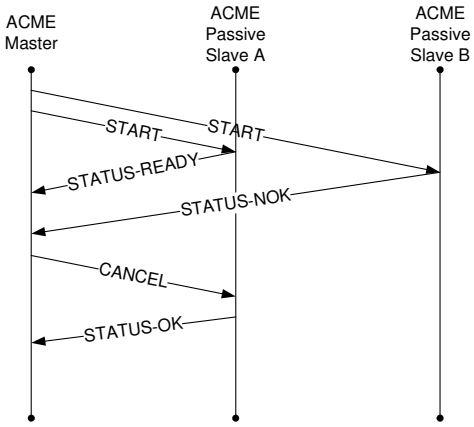


Fig. 4. Failed slave to slave exchange.

Fig. 3 shows an experiment between two slaves. Master commands passive slave to get ready to receive calls (START). Passive slave informs master it is ready to receive calls (STATUS-READY, STATUS message with ready indicator). Master commands active slave to start calls (START). Calls are established and media flows start. If everything goes right, both slaves send to master a STATUS with an OK indicator, otherwise a STATUS with NOK (NOT OK) indicator is sent. It is important to mention that if any call fails, the whole experiment fails and must be restarted, because it is essential that the correct number of simultaneous calls is executed.

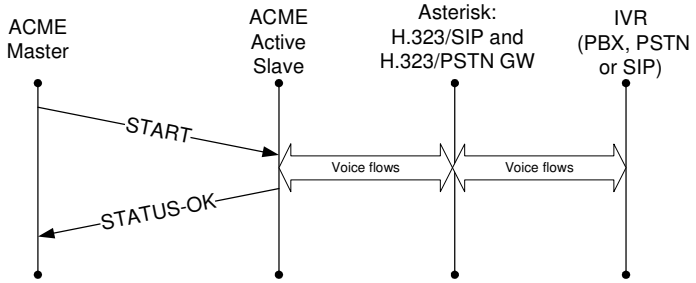


Fig. 5. Slave to gateway**.**

Fig. 4 also shows an experiment between two slaves. However, in this example an error occurs in the active slave: Master commands passive slave to get ready to receive calls (START). Passive slave informs master it is ready to receive calls (STATUS-READY). Master commands active slave to start calls (START), but, for

some reason, active slave cannot initiate calls and send master a STATUS-NOK. Master then sends CANCEL to passive slave, informing that for some reason slave will not get the calls it was waiting for.

Fig.5 shows an experiment between an active slave and a gateway. The gateway can forward calls to PBX or PSTN. In this case, master just needs to send START to the active slave and wait for a STATUS-OK. This operation is useful when monitoring gateway availability and its capability to establish calls to a certain network or destination.

ACMEp runs over TCP and uses port 8000 by default (configurable). No cryptography is available, but security aspects are being considered for the future. To avoid a non authorized master to command a slave, access lists can be configured in each slave. However, the START message is critical, since it carries a non ciphered password. TLS [4] must be used for secure operations.

4 Stress Experiments

The goal of a stress experiment is capacity determination of a target. Capacity is the number of concurrent VoIP calls that can be performed (using a specific codec, e.g., G.711), without getting the average MOS (see section 1) over all calls to fall below a minimum value. Target can be a link, a network, a system or even a VoIP service.

Slave capacity determination evaluates hardware via a stress experiment which demands that the average MOS is not less than the maximum MOS in ideal conditions. It is important to say that the maximum MOS depends on type of codec. For example, G.711 reaches a 4.41 maximum MOS score.

A stress experiment is made of tests. Each test has a certain number of concurrent calls. So, if test 1 which handles N simultaneous calls obtains an average MOS equal or above the established average MOS, a new test, test 2, is executed. Test 2 on its hand will handle N + X simultaneous calls. If test 2 reaches equal or better result compared to the established average MOS, a new test (test 3) is performed now with N + X + X simultaneous calls. And so on. The initial number N of concurrent calls and the number X of calls increment are configurable parameters.

When the user configures a required average MOS for an experiment, he also configures a target percentage, as, e.g., 10%, to be greater or equal the size of the confidence interval divided by the center value of the interval. Any test is required to achieve a confidence interval of size equal or less than the target percentage of the center of the interval. The center of the confidence interval is the average MOS estimator.

To reach the target confidence interval, a test has to be repeated many times. Each repetition is called a round and may require a minimum number of rounds per test, before considering the restriction on the size of the confidence interval. This procedure avoids situations where a few number of rounds with similar MOS values gives an average which satisfies the target percentage but does not guarantee MOS convergence indeed. It is important to say that the average MOS and associated confidence interval at each round is calculated over the MOS values of all calls of all rounds (current and previous) of a test.

Table 1 shows the structure of an experiment. Three tests are presented, where test 1 (with three concurrent calls) is expanded to show details of the executed rounds. The target percentage for this test is 10%. It can be noted that the target percentage is met after five rounds, when the size of the interval is 0.32 and the average MOS is 4.08. Round 5 gives a percentage around 8% (0.32/4.08), which is less than 10%. Round 4 gives a percentage that is slightly greater than 10%. After test 1 reaches the target percentage, test 2 with 4 concurrent calls is started.

Table 1. Experiment structure**.**

Test 1 → 3 concurrent calls
• *Round 1 – average MOS = 4.10, conf. int. size = 0.81, target = 0.41 (10%)*
• *Round 2 – average MOS = 4.03, conf. int. size = 0.71, target = 0.40 (10%)*
• *Round 3 – average MOS = 4.08, conf. int. size = 0.58, target = 0.41 (10%)*
• *Round 4 – average MOS = 4.07 conf. int. size = 0.42, target = 0.41 (10%)*
• *Round 5 – average MOS = 4.08 conf. int. size = 0.32, target = 0.41 (10%)*
Test 2 → 4 concurrent calls
Test 3 → 5 concurrent calls

5 Periodic experiments

There are three basic types of periodic experiments:

Availability Experiment: In regular time intervals, a destination is called to monitor its reachability and availability. Presently, only H.323 protocol is directed by ACME, but using a gateway H.323/SIP allows monitoring of SIP destinations. Similarly, a voice gateway allows monitoring a PBX system and PSTN destinations reached through the PBX. In this kind of experiment, destinations are answering machine or IVR (*Interactive Voice Response*), as calls are answered automatically on these systems.

Quality Experiment: By default, all availability experiments also evaluate the received call quality. The difference with quality experiments is that alerts can be set. Then, as MOS values are obtained below a certain level, the user is notified. Availability and authentication experiments only emit alert in case of error.

Authentication Experiment: In regular time intervals, a slave tries to authenticate in a given server to monitor the correctness of the authentication operation. At present, only H.323 is supported.

6 Concurrency of Experiments

Slaves are experiment resources. After registered in ACME, slaves can be included in sites and be used by any experiment. Slaves can be included in more than one site at the same time, but the sites of an experiment cannot share the same slave.

A test is the granularity in experiment concurrency. Thus, two experiments A and B can be executed concurrently and have slaves and even sites in common. However, tests never occur simultaneously, as ACME executes one test at a time.

Fig. 6 shows the finite state machine (FSM) for a quality periodic experiment. When the experiment is scheduled, it enters the Scheduled state and remains in this state until the programmed starting time is reached and there is not other experiment ahead waiting for execution. When the first test starts execution, FSM changes to Running. At the end of a test, Ready-OK state is entered, in case the MOS is above the required value. If MOS is below required value, the state Ready – Low MOS is entered. Another possibility is the test not finishing because of an error. If an expected error occurs, as destination unreachability, the FSM enters Ready- Can't connect to host, or another state which represents a defined error. If an undefined and irreversible error occurs, Undefined error state is entered. The state of an experiment always returns to Running when a next test is run. Although, Undefined error state does not allows transition out of the state.

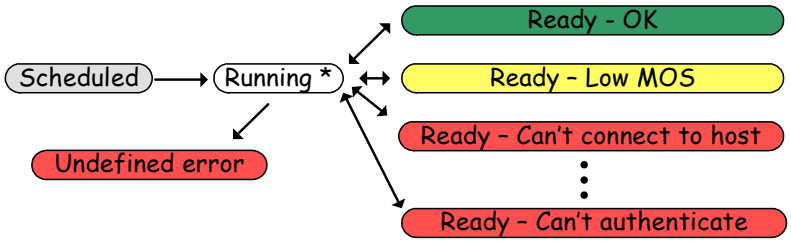


Fig. 6. Finite State Machine for a Quality Experiment.

All experiments have similar state machines. A periodic experiment only ends if its FSM enters Undefined error. On the other hand, stress experiments end when the required target MOS is reached, when available slaves are unable to support more increment in the number of calls, or, finally, when slave goes to Undefined error.

Experiment execution is performed as a kind of circular round-robin scheme. Two lists for experiments in Ready state are kept, one for periodic and another for stress experiments. The scheduler (ACME module for concurrency management) always dispatches the next experiment in the periodic experiment ready list. If this list is empty, next experiment in the stress experiment ready list is taken. Periodic experiments are prioritized because they have time constraints.

7 Demonstrations

Fig. 7 shows a 500 kbps serial PPP link interconnecting two Cisco routers (not pictured) for the stress experiment. UDP generator [11] was configured to generate 200 kbps constant rate large packet background traffic towards the reflector. Slaves 1 and 2 send G.711 VoIP media packets marked with DSCP EF [1], generating around 80 kbps at the link level per call. Two router queue disciplines were tested: WFQ

(weighed fair queue); and a 320 kbps guaranteed rate priority queue (PQ) [3] for voice flow.

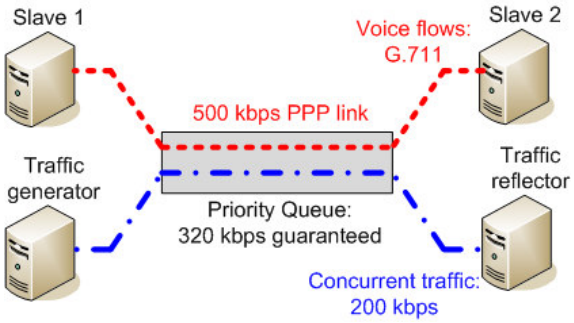


Fig. 7. Scenario for Stress Experiment.

Fig. 8 shows the stress experiment output. The target percentage for confidence interval was set at 10%. Confidence intervals are automatically plotted. With PQ, up to 4 concurrent calls cause no degradation and MOS is maintained at 4.41 (maximum value for G.711). From the fifth simultaneous call on, as PQ only supports four calls with no degradation, MOS degrades abruptly and confidence intervals also increases, indicating variability in call quality. If minimum acceptable MOS is 3.0, the maximum number of simultaneous VoIP calls should be 5 with PQ. On the other hand, when a WFQ queue is used, up to 6 simultaneous calls can be accepted. WFQ inherently prioritizes small packets (like voice) over large packets. However, large confidence intervals with increasing number of calls show that MOS begins to vary randomly among voice calls.

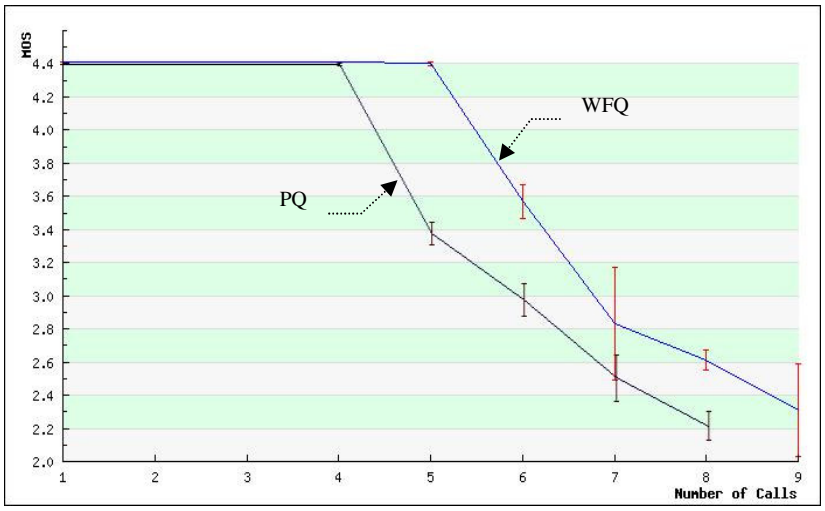


Fig. 8. Stress Experiment Output. The vertical axis represents the average MOS and the horizontal axis is the number of simultaneous calls per test.

The fone@RNP VoIP service [15] was used as a scenario for periodic experiments. Fone@RNP uses a heterogeneous H.323 and SIP architecture which, besides interconnecting academic institutions PBXs, it allows IP softphone usage and PSTN interconnection. The objective of the demo was to monitor the service at a local institution (UFRJ) and fictitious institution established in the same manner as a real institution (called Institution 1). As shown in Fig. 9, each monitored institution represents a site and has a local slave. The central slave in the figure represents a third site that will interact with the other two sites. According to fone@RNP, each institution has a gatekeeper to attend H.323 clients, an OpenSER proxy for SIP clients registration, and a gateway Asterisk for H.323/SIP interoperability and PBX interconnection. Six periodic experiments were designed for each institution:

SIP Phone: Monitors SIP service availability via regular calls from central slave to SIP softphone with auto-answer. Local gateway converts from H.323 to SIP.
H.323 Phone: Monitors H.323 service availability via regular calls to local slave;
PBX Phone: Monitors call routing to PBX via regular calls from central slave to an IVR PBX extension PBX;
PSTN Phone: Monitors PSTN routing via regular calls from central slave to an 800 IVR service;
H.323 Authentication: Monitors H.323 authentication via periodic authentications of local slave to the gatekeeper;
Quality: Monitors quality of periodic calls between central and local slaves.

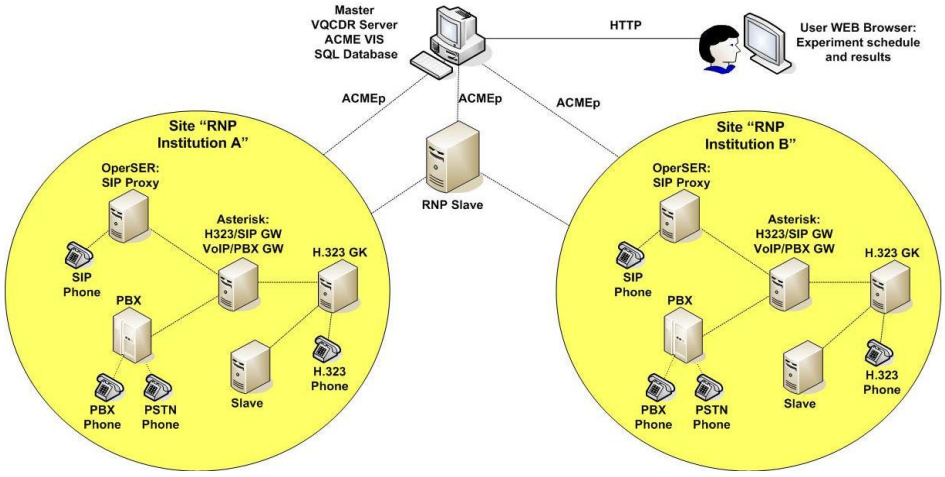
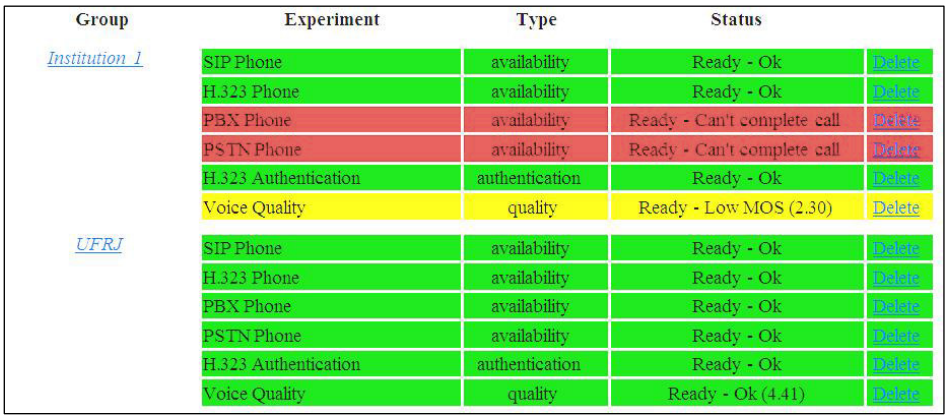


Fig. 9. Scenario for Periodic Experiments



Group	Experiment	Type	Status	
Institution_1	SIP Phone	availability	Ready - Ok	Delete
	H.323 Phone	availability	Ready - Ok	Delete
	PBX Phone	availability	Ready - Can't complete call	Delete
	PSTN Phone	availability	Ready - Can't complete call	Delete
	H.323 Authentication	authentication	Ready - Ok	Delete
	Voice Quality	quality	Ready - Low MOS (2.30)	Delete
UFRJ	SIP Phone	availability	Ready - Ok	Delete
	H.323 Phone	availability	Ready - Ok	Delete
	PBX Phone	availability	Ready - Ok	Delete
	PSTN Phone	availability	Ready - Ok	Delete
	H.323 Authentication	authentication	Ready - Ok	Delete
	Voice Quality	quality	Ready - Ok (4.41)	Delete

Fig. 10. ACME Monitoring Panel.

Fig. 10 shows ACME monitoring panel: experiments with positive status are shown in green; experiments with problems are shown in red, indicating failure; experiments in critical condition, as with MOS below some acceptable level, are in yellow. Panel configuration allows yellow and red indicators to trigger alerts via e-mail.

8 Conclusions and Future Work

An automated tool for the generation and quality evaluation of VoIP calls is presented and its effectiveness and versatility are demonstrated in real environments. Dynamic automated experiment execution and graphics output with confidence intervals is an advantage when comparing to other tools. As a demo of stress experiment, a link configured with priority queuing and WFQ is analyzed in terms of its capacity to handle VoIP concurrent traffic. For periodic experiments demonstration, ACME was shown to perform as a monitoring tool for fone@RNP, a heterogeneous and complex SIP and H.323 environment. ACME has also been used with success as a tool in VoIP courses. In research, the tool is planned to be used to evaluate VoIP capacity in complex wireless environments mixing priority disciplines and choice of configuration parameters. Detailed performance studies in slow access link can be done to determine the best selection of router parameters and disciplines to maximize VoIP calls.

TLS will be supported in ACMEp for the future, as secure transport will eliminate vulnerabilities in master-slave communication. SIP native support is being developed, what will permit SIP availability monitoring without any signaling gateway in place.

References

1. Babiarz, J., Chan, K., Baker, F: Configuration Guidelines for DiffServ Service Classes, IETF. RFC 4594 (2006)
2. Brix: BrixCall. <http://www.brixnet.com/products/ >. Access in Dec. 2007
3. Cisco Systems: VoIP over PPP Links with Quality of Service. <http://www.cisco.com/warp/public/788/voice-qos/voip-mlppp.html>. Access in Dec. 2007
4. Dierks, T., Rescorla E.: The Transport Layer Security (TLS) Protocol Version 1.1. IETF. RFC 4346 (2006)
5. Empirix: VoIP and IMS Hammer Call Analyzer. <http://www.empirix.com/products-services/v-hca.asp>. Access in Dec. 2007.
6. ETSI TS 101 329-5 v1.1.2: Telecommunications and Internet Protocol Harmonization Over Networks Release 3, End-to-end Quality of Service in TIPHON systems; Part 5: Quality of Service (QoS) measurement methodologies (2002)
7. ETSI TS 102 024-5 v4.1.1. (2003). Telecommunications and Internet Protocol Harmonization Over Networks Release 4, End-to-end Quality of Service in TIPHON systems; Part 5: Quality of Service (QoS) measurement methodologies.
8. GnuGK: The GNU Gatekeeper. <http://www.gnugk.org/>. Access in Dec. 2007.
9. ITU-T G.107: The E-Model, a computational model for use in transmission planning. (2003)
10. ITU-T P.800: Methods for subjective determination of transmission quality. (1996)
11. Leocadio, M. A. P., Rodrigues, P. H. A.:Uma Ferramenta para Geração de Tráfego e Medição em Ambiente de Alto Desempenho. In: Proceedings of XVIII. SBRC (2000)
12. Lustosa, L.C.G., Carvalho, L.S.G., Rodrigues, P.H.A., Mota, S. E.: Utilização do Modelo E para avaliação da qualidade da fala em sistemas de comunicação baseados em voz sobre IP. In: Proceedings of XXII SBRC. pp. 03-616 (2004)
13. Lustosa, L.C.G., Rodrigues P.H.A., David , F., Quinellato, D. G.: A Voice over IP Quality Monitoring Architecture. In: Proceedings of MMNS 2005. Berlin : Springer Verlag, 2005. v. 1. pp. 168-178. (2005)
14. Lustosa, L.C.G., Rodrigues, P.H.A., David, F., Quinelato, D. G.: VQuality: Uma Biblioteca Multiplataforma para Avaliação de Qualidade de Chamadas Telefônicas IP, In: Proceedings of XXIII SBRC. p. 1185-1188 (2005)
15. RNP: fone@RNP. <http://www.rnp.br/voip/>. Access in Dec. 2007.
16. Solarwinds: VoIP Monitor. <http://www.solarwinds.com/>. Access In Dec. 2007.
17. Telchemy: SQmon. <http://www.telchemy.com/products.html>. Access in Dec. 2007.