VoIP service quality monitoring using active and passive probes

Shipra Agrawal, P. P. S. Narayan, Jeyashankher Ramamirtham, Rajeev Rastogi,
Mark Smith, Ken Swanson, Marina Thottan
{shipra, ppsnarayan, jai, rastogi, massmith, gks, marinat @ lucent.com}

***Abstract*— Service providers and enterprises all over the world are rapidly deploying Voice over IP (VoIP) networks because of reduced capital and operational expenditure, and easy creation of new services. Voice traffic has stringement requirements on the quality of service, like strict delay and loss requirements, and 99.999% network availability. However, IP networks have not been designed to easily meet the above requirements. Thus, service providers need service quality management tools that can proactively detect and mitigate service quality degradation of VoIP traffic.**

In this paper, we present *active* and *passive* probes that enable service providers to detect service impairments. We use the probes to compute the network parameters (delay, loss and jitter) that can be used to compute the call quality as a Mean Opinion Score using a voice quality metric, E-model. These tools can be used by service providers and enterprises to identify network impairments that cause service quality degradation and take corrective measures in real time so that the impact on the degradation perceived by end-users is minimal.

***Index Terms*—VoIP, Passive probe, Active probe, Service quality monitoring, Network measurements**

I. Introduction

Voice over IP networks are attractive to service providers and enterprises because they reduces expenditure and makes creation of new revenue generating services easy. By using the same network for voice and data services, service providers and enterprises save on equipment, operation and maintenance costs. Also, creation of new services like "bundled" voice and data service, web-enabled call centres, Click-to-dial, etc. is made easy by migrating to IP-based converged networks. Moreover, Internet based voice applications like Skype [1], Dialpad [2] and Net2Phone [3] and VoIP services by Vonage [4] are very popular because of the lower cost per call.

IP networks, on the other hand, are not designed to carry voice traffic. Voice demands stringement delay and loss requirements from the network for acceptable voice quality. Also, voice services require network availability of 99.999% for an always-on experience, and for emergency services. Thus, enterprises and service providers need to constantly monitor their networks to detect sevice quality degradation and take corrective action in real-time to ensure that the degradation perceived by end-users is minimal.

Traditionally, IP networks have been managed by measuring aggregate parameters over interfaces of routers or other network elements; for example, link utilization and packet losses. While this is sufficient to manage best-effort services, managing new services based on voice and video that have diverse requirement need measurements of finer granularity. In this paper, we describe two mechanisms that provide fine-grained measurements in the network that allow service providers to determine the service quality at a per-service, per-user granularity.

Active probes are used to measure the end-to-end service quality provided by the network for an application. Active probes send emulated VoIP traffic through the network and measure the service quality of the network. The measured service quality is indicative of the quality that users would see when using the network. Service providers can program the active probes to continuously and automatically monitor the network and report degradations in the network. Active probes provide a "black box" measurement mechanism of the network, and can be used to detect service degradations. It cannot however help pin-point the root cause of failure or congestion. For example, while an active probe can detect an excessive delay through the network, it cannot identify which link's congestion is causing the delay. Similarly, if there is a link or a server failure, the active probes can detect the failure, but cannot help identify the cause of the failure.

Passive probes provide a different perspective of the network as compared to active probes. Passive probes are installed on links within the network and they *snoop* on all the traffic that flows through the link being monitored. They "sieve" through the packets on the link to identify individual VoIP calls and compute the service quality received by each one of them. Passive probes can be used to continuously monitor the performance of the network for the actual application traffic, as opposed to the measurements for synthetic traffic using active probes. They also can be used to segment the network for source of failures or congestion. However, they do not give an end-to-end perspective of the network performance. They can compute the performance of the network only between the points of installation of the probes.

Active and passive probes, when used in combination to monitor network performance, give service providers the capability of effectively monitoring their network for VoIP performance.

Shipra Agrawal, Jeyashankher Ramamirtham, and Rajeev Rastogi are with Bell Labs Research India, Lucent technologies India, Bangalore, India.

P. P. S. Narayan, Mark Smith, Ken Swanson, and Marina Thottan are with Bell Labs Research, Lucent Technologies, Murray Hill, NJ, U. S. A.

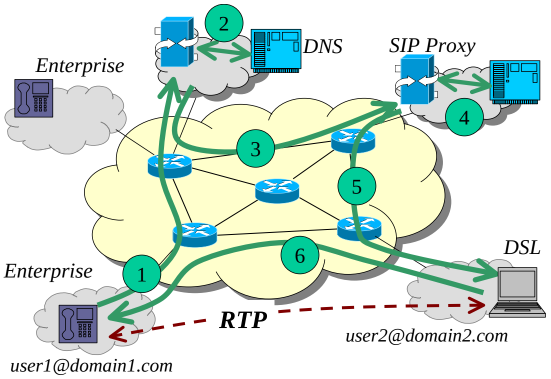


Fig. 1. Simplified VoIP network architecture

They provide fine grained measurements that can be used for real-time monitoring and immediate detection of faults.

In this paper, we discuss the implementation of active and passive probes, and the various issues that arise. We also discuss how they can be used by service providers and enterprises for effective VoIP service quality management. In Section II, we give a brief overview of the VoIP network architecture and its various network components, and describe service quality monitoring mechanisms and the various metrics used in measuring voice service quality. In Section III and Section IV, we describe the implementation of active and passive probes respectively, and discuss the various issues involved. We present an example set of reports generated by the active and passive probes for an experiment in Section V. Finally, we conclude and discuss future work in Section VI.

II. Service quality monitoring in VoIP networks

A. *VoIP network architecture*

Figure 1 shows a simplified architecture of a VoIP network that uses Session Initiation Protocol [6] [1]. The key network components of a VoIP network are a SIP Proxy and a location service. Typically, a client is assigned a SIP proxy in its domain and the client registers its IP address and other attributes with a registrar. For simplicity of discussion, we assume that the registrar is colocated with the proxy and we refer to the registrar as the proxy also. The SIP proxy handles requests from clients to initiate VoIP calls and uses the location service to locate the user who is being called. The location service can be implemented using protocols like DNS.

Figure 1 illustrates an example call setup using a signaling protocol like Session Initiation Protocol, where *user1@domain1.com* wants to place a VoIP call to a user in a different domain, *user2@domain2.com*.

1) *user1* sends a SIP INVITE to *user2*, which is forwarded to the SIP proxy in *user1*'s domain.
2) The SIP proxy looks up the location of the proxy for *user2*'s domain using DNS or another location service.
3) The SIP proxy forwards the INVITE to *user2*'s proxy.
4) *user2*'s proxy looks up the location of *user2*.
5) The proxy fowards the INVITE to *user2*.
6) Finally, *user2* accepts *user1*'s INVITE by sending a 200 OK message, thus establishing the call.

SIP also requires *user1* to respond with an ACK for a three-way handshake. The two users now know the identity of each other and the call is accepted. They then communicate with each other using VoIP codecs to generate packets and transmit the packets over the network using RTP.

B. *Voice service quality metrics*

The quality of a voice call is affected by three sets of impairments; signal processing impairments, network impairments, and environmetal impairments. Signal processing impairments are caused by the voice filters, quantizers, and codecs employed in the system. Network impairments like delay, loss and jitter determine the quality of a voice call. Environmental impairments include factors like the ambient noise levels in the user's location.

Measurements through monitoring mechanisms are mapped by voice quality metrics to the actual quality of the end-user experience, referred to as *Mean Opinion Score* (MOS). At present, there are no standardized voice quality metrics for VoIP. ITU has defined two voice quality metric standards for the circuit-switched PSTN, E model [7] and PESQ [8]. Both these models were originally designed to do transmission planning and attempts to address all the different impairments. The E model, best addresses impairments that occur as a result of ambient noise and delay (echoes) by explicitly accounting for these in an additive impairment model. However, from a signal processing perspective the E model is not as accurate as PESQ. The E model takes into account average delays experienced in the network. It assumes that packet losses that may occur in the network are accounted for by the individual codecs. Thus the extent of impairments due to packet loss is fixed for the specific codec used. The PESQ algorithm predicts the subjective MOS values by comparing the received signal distortion with respect to a standard reference signal. It assumes that the distorted signal should take into account the network impairments and therefore network performance does not need to be specified explicitly. PESQ does not take into account frequency responses and loudness which are two important factors affecting the perceived quality.

In our work for estimating the end user call experience in real time it is necessary to develop a model that maps per call and network performance metrics to the users perception of call quality. Furthermore for service assurance purposes it is necessary to build a voice quality model that can be used to enable root cause analysis in the event of performance degradations. Therefore the model must explicitly capture both the signal processing as well as the networking impairments. The additive nature of the E model makes it more amenable to explicitly account for such impairments on a per call basis and so we use a metric based on E model that has been adapted for VoIP as the voice quality metric to determine the MOS. The E model defines R factor as the measure of voice quality, where an R factor of 100 is the best

[1] Though we consider networks that use SIP for signaling, the same discussion can be extended to other signaling protocols like H.323

and 0 the worst.

$$R = R_0 - I_s + I_d + I_e + A \qquad (1)$$

where, R_0 is the signal to noise ratio that includes the end-to-end signal attenuation and ambient noise, I_s is the impairment factor that comprises of outbound volume, side tone and quantizing distortion, I_d is the impairment factor due to network delay and echo, I_e is the equipment impairments including codec impairments like signal distortion, and A is the user tolerance to degraded user quality (for instance, wireless users can be expected to be more tolerant to service quality degradation than wireline users). The R factor directly maps into MOS using the following expression.

$$MOS = 1 + 0.035R + 7 \times 10^{-6} R(R - 60)(100 - R) \qquad (2)$$

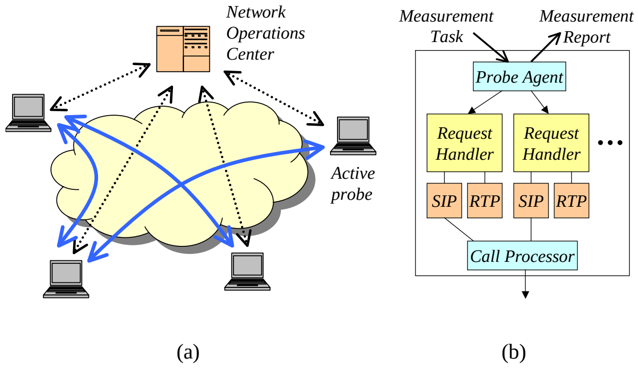
When the R factor is 100, the MOS is 4.5 and when the R factor is 0, the MOS is 1. Reference [9] and [10] describe E model in more detail.

C. Monitoring requirements

We now describe the key network factors that affect the quality of a voice call.

1) Call setup time is the time between when a client sends an INVITE and receives a final 200 OK response. For acceptable end-user experience, this has to be in the range of a few seconds.
2) Call tear-down time is the time taken to receive a final 200 OK response after a BYE request is initiated by a client to end a call.
3) One-way delay is the delay experienced by RTP packets that carry the voice traffic through the network. This needs to be measured for both directions of a call between the clients. Maximum acceptable one-way delays for reasonable call quality are in the range of 150 to 180 ms.
4) Packet losses in the network cause degradation of voice quality. There are two kinds of losses that one needs to account for, isolated packet losses and burst losses. An isolated loss, where one packet among many gets dropped by the network, does not have a big impact on the voice quality. Moreover, packet loss concealment algorithms exist that can mitigate the effect of isolated packet losses. Burst losses occur when the network drops a set of consecutive packets and this is difficult to mitigate using concealment techniques, and hence has a very adverse impact on the call quality.
5) Jitter is introduced by the network due to variability in queueing delays. This is normally handled at the receiver by introducing a playout buffer (also called jitter buffer). The receiver delays the playout of the packet by a fixed time and this absorbs the jitter introduced by the network. If the jitter of a packet is larger than the playout buffer, the packet is dropped and it effectively shows up as a loss.

To perform service quality monitoring in VoIP networks, we need mechanisms that measure the above network performance parameters for each call. These measurements are used to compute the call quality percecived by the end user and this can be used by service providers to detect quality degradation. We now discuss our approach to measure the performance of the network using active and passive probes.



(a) (b)

Fig. 2. (a) System architecture, (b) Active probe components

III. Monitoring using Active Probes

The architecture of the active probing system is shown in Figure 2(a). The system has a set of active probes and a Network Operations Center (NOC). The NOC conrols the monitoring operation in the network by initiating measurement tasks between pairs of active probes. Active probes are software agents that run on designated machines in the network and they establish VoIP calls based on requests from the NOC and measure parameters like delay, loss, and jitter, that affect call quality and compute the perceived call quality using this information. This information is then reported to the NOC, which tracks the service quality degradation across the network.

The active probing system uses two basic operations to perform monitoring, *measurement task* and *measurement report*. A measurement task is used by the NOC to get information about the network's performance for VoIP. The NOC identifies two active probes in the network to be a *caller* and a *callee* respectively. It then assigns a task to be performed by this pair of probes. The task consists of establishing a sequence of VoIP calls from the caller to the callee for a certain duration and measuring the network performance. A measurement task is composed using XML and sent to the probes. An example measurement task is shown in Fig. 3. The task specifies the caller's and callee's address and port information to which the task needs to the sent to. Also, the task configuration includes a Call Profile that describes the parameters of the calls of the task. The *numberOfCalls* is the total number of calls that need to be made between the caller and the callee as part of the measurement task, *callDuration* is the duration of each call in seconds, *interCallWait* is the time in seconds that the probes wait after completing a call before initiating the next call, and the *codec* is the type of Codec that is used for the task.

Once the probes receive the measurement task from the NOC, they execute the task by placing VoIP calls between each other

```
<?xml version="1.0" encoding="UTF-8"?>
<tasks xmlns="http://…">
    <taskconfig id="task1">
            <caller name = "caller“
                    ipaddress = "1.2.3.4“
                    probeAgentPort = "11000"/>
            <callee name = "callee“
                    ipaddress = "255.254.253.252“
                    probeAgentPort = "10999"/>
            <callProfile callDuration = "60”
                         interCallWait = "5“
                         numberOfCalls = "25“
                         codec = "G711"/>
    </taskconfig>
</tasks>
```

Fig. 3. Example measurement task

and measuring the network performance. The probes use SIP to establish the calls and they emulate VoIP calls by sending synthetic traffic that have characteristics similar to the specified codec's output. The probes send the measured results to the NOC as a measurement report. The report has the following measurements for the forward (caller to callee) and reverse (callee to caller) paths.

- Average one-way delay (forward and reverse)
- Average round trip delay
- Jitter (forward and reverse)
- Packet loss percentage (forward and reverse)
- Jitter buffer loss (forward and reverse)
- R-factor and MOS (forward and reverse)
- Call setup time
- Call teardown time
- Number of INVITEs per call

A. *Active probe operation*

An active probe (Fig. 2(b)) is a software agent that is installed on systems in the network, and is implemented as a multi-threaded process. Each active probe supports multiple callers and callees to allow for flexible monitoring configurations in the network. It is initialized with two threads, a *probe agent* and a *call processor*. The probe agent listens on a port of the system for measurement tasks from the NOC and manages the callers and callees in the system. The call processor handles the processing of all SIP messages and uses a port in the system for exchange of the SIP messages. It provides multiplexing and demultiplexing functionality for SIP messages, when multiple callers and callees are present in the system.

The operation of the system has three phases, initialization, call handling, and reporting phases. Once the NOC defines a measurement task, the initialization phase involves establishing the caller and callee at the probes. Also, the caller and callee exchange their addresses that is necessary for call establishment. During the call handling phase, the probes establish VoIP calls between the two sites using SIP, exchange VoIP packets, and measure the network performance. The measurement reporting phase involves sending the measurement report to the NOC.

When the NOC identifies a caller and a callee for a measurement task, the NOC sends the measurement task to the caller's probe agent. The probe agent spawns a *request handler* thread that acts as the caller and handles the task of measuring the call quality between the caller-callee pair. When the request handler is initiated, the probe agent assigns a SIP URI to it. The request handler then transmits the measurement task to the callee's probe agent. The callee's probe agent then spawns a callee request handler and assigns a SIP URI to it. The callee's request handler then communicates the callee's URI to the caller using the same socket that was used by the caller request handler for transmitting the measurement task. Thus, the caller now knows the identity of the callee. By using this initialization procedure, the NOC needs to know only the IP addresses of the probes and the probe agent port. It doesn't need to handle the assignment of SIP URIs for every measurement task, hence reducing the load on the NOC. Also, since each task is shipped to the caller and the caller and callee perform the task and return the measurement report, there is no central coordination of the task by the NOC.

During the call handling phase, the request handlers establish calls between themselves and measure the network performance. Each request handler has a SIP handler and an RTP handler. The SIP handler handles the connection establishment and teardown using SIP for each call as described in Section II-A, and computes the SIP related statistics such as call setup time and teardown time. It uses the Call Processor to send and receive SIP messages. The caller always initiates the calls and uses the callee's SIP URI to send the SIP INVITE. The request handlers also maintain failure statistics like the number of timeouts and eventual failure to connect. These can be used by the NOC to detect failures and in troubleshooting.

Once a connecion has been established, the caller and callee exchange VoIP packets using the RTP handler. The RTP handler generates RTP packets that emulate a codec as specified by the measurement task and transmits them over the network for the specified duration. It also receives RTP packets from its peer and records the measurements such as delay, loss, and jitter. Once the RTP handler completes the transmission of packets, the SIP handler tears down the call and records the teardown statistics. The request handlers setup multiple such calls and perform measurements according to the measurement task's specifications.

Once the task is completed, the callee's request handler transmits the measured statistics to the caller and the caller performs the call quality computation using this information. The caller then reports the measured statistics to the NOC, and the caller and callee request handlers then shut down.

B. *Call metrics computation*

We use the local clocks at the probes for our measurements and the probes use NTP [11][2] to synchronize their clocks. Once

[2] For more accurate measurements, we can deploy clock sychronization mechanisms using GPS or CDMA.

a call is initiated, both the caller and callee sent VoIP packets to each other using RTP. While sending a packet, the sender puts in the local time as a timestamp in the RTP payload. This is used by the receiver to derive the delay and jitter of the network.

1) Call setup and teardown time: Since the caller initiates and tears down calls, the call setup and tear down times are measured at the caller. For call setup, we measure the time between when the INVITE is sent and when the OK is received. For tear down, we measure the time between when the BYE is sent and when the OK is received. We also keep track of the number of timeouts at the caller for the INVITE.

2) One-way and round trip delay: The average one-way delay includes a packetization delay, which depends on the codec being used, and a jitter buffer delay, which depends on the size of the jitter buffer. Fixed packetization delay values are defined for each codec. The packetization delay value for each codec includes the time to encode and decode the packet. After the packet is received, the timestamp of the sender is subtracted from the current time at the receiver to get the propagation delay. The packetization delay is added at the receiver to get the total delay.

If the jitter buffer size is greater than 0, this will also add to the delay. The purpose of the jitter buffer is to smooth out jitter in the underlying network so that playback at the receiver can mirror the play rate at the sender. This is done by waiting until the jitter buffer is full to start the playback of the first packet. To determine the playback time at each subsequent packet that will maintain the play rate, we add a value we call the playback delay, D_{pb}, to the time stamp of each incoming packet, because this value is the same for all received packets, the playback rate will be the same as the sender play rate. Let the timestamp on the first received packet be t_0 , the arrival time of this packet at the receiver be t_1 and the time when the jitter buffer is first full be t_3. Note that t_3 is also the playback time of this first packet. The playback delay is

$$D_{pb} = (t_1 - t_0) + (t_3 - t_1) = t_3 - t_0$$

Thus, for each packet that arrives at time t_x with timestamp t_y, the jitter buffer adds an additional delay of $t_y + D_{pb} - t_x$. Packets that arrive after their scheduled playback time are discarded. At the end of the call, the average is taken over all packets received.

The average round trip delay is sum of the one-way delay from the caller to callee and the one-way delay from the callee to the caller. The callee sends the average delay for packets it receives to the caller at the end of a call, where the caller sums the values.

3) Jitter: RFC 3550 [12] defines interarrival jitter as ”the mean deviation (smoothed absolute value) of the difference D in packet spacing at the receiver compared to the sender for a pair of packets.” Also, the interarrival jitter is calculated continuously for each data packet i using the difference D for that packet and the previous packet $i-1$ in order of arrival (not necessarily in sequence), according to the formula

$$J_i = J_{i-1} + (|D(i-1,i)| - J_{i-1})/16$$

4) Packet loss percentage: At the end of a call, the callee sends the number of packets it sent and the number of packets it received. The caller uses these numbers along with the number of packets it sent and received to calculate the packet loss percentage for both the forward and reverse links. Loss can be determined by looking at the RTP sequence numbers of successive packets received. If at any point during a call, the difference in sequence numbers between two successive packets is greater than four, then the loss is defined as bursty. Otherwise, the loss of packets is termed random.

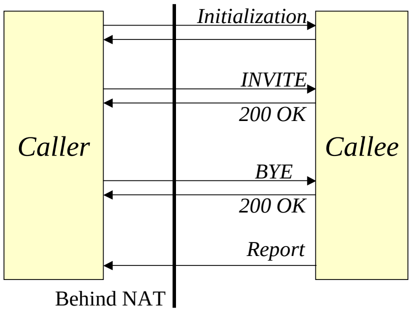


Fig. 4. Operations when the caller is behind a NAT

The other form of loss is due to the jitter buffer. If the playback time is ahead of the current time, the packet is discarded and this counts towards a jitter buffer loss. Also, if a packet arrives when the jitter buffer is full, then the packet is discarded. This is accounted as a jitter buffer loss too.

5) R-factor and MOS: The parameters one-way delays, round trip delay and jitter buffer loss are used to compute the R-factor and MOS using the E-model as describes in Section II-B.

C. Registration and SIP proxy based calling

The caller and callee need to register with the SIP proxies in their respective domains if necessary to place calls to each other. The measurement task from the NOC (Fig. 3) now provides the probes with the SIP registrar's address (name or IP address), and a user name and a password that are necessary for authentication. The caller and callee register themselves with the registrar before establishing calls in the call handling phase. If registration is not necessary, this can be bypassed by the probes.

The NOC also provides an outbound SIP proxy address to the caller in the measurement task and this is used by the caller to place the calls to the callee. This is handled by adding a *route* header to the SIP INVITE and the SIP proxy forwards the INVITE to the callee.

D. Handling NATs

In the presence of a NAT, the addressing information exchanged by the caller and callee in the initialization phase can be unusable, as they need the addresses as translated by the NAT. We have designed the system to handle a particular case when the caller is behind a NAT and the callee is not. The caller always initiates communication with the callee. Since the callee does not reside behind a NAT, the address used by the caller is the true address of the callee.

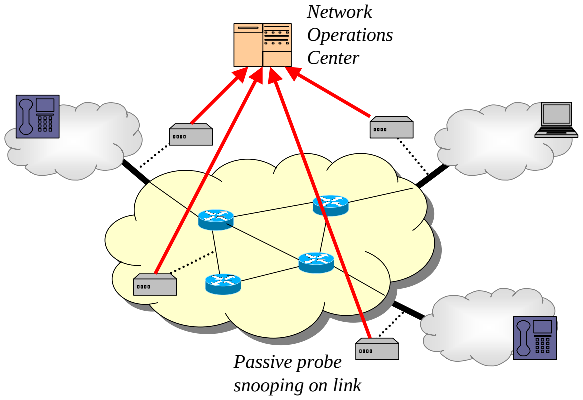


Fig. 5. Passive probing architecture

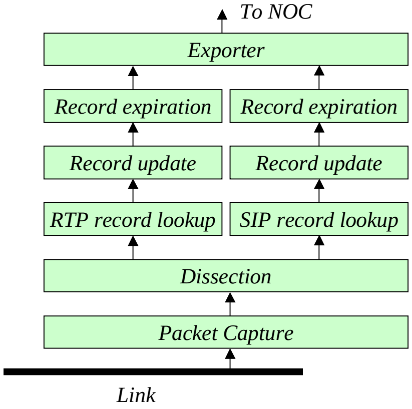


Fig. 6. Passive probe components

In the initialization phase, the caller transmits the measurement task received from the NOC to the callee using its true address. When the callee receives the task, the callee responds with its SIP URI to the caller on the same translated caller address instead of its true address.

In the call handling phase, the caller registers its URI with the proxy using its true IP address. This registration is not useful as the proxy will not be able to communicate to the caller using that address. However, the callee's registration is valid. We handle this case by ensuring that the caller always initiates SIP transactions (INVITE and BYE) and the callee uses the same channel to respond, thus ensuring that the translated address is used.

Finally, the callee reports its measurements to the caller by using the same channel as the one used in the initialization. This channel uses the translated address and thus, the report is forwarded to the caller by the NAT.

IV. Monitoring using Passive Probes

We now present the design and implementation of passive probes that can be used to monitor VoIP networks. A passive probe is used to snoop on a network link and observe the VoIP traffic traversing that link. In combination with other passive probes, it can be used to monitor the network between them for service quality degradation. In addition, they can be used to pinpoint cause of failure or degradation in the network by strategically placing them.

Fig. 5 shows the architecture of the passive probing system. Passive probes are setup to snoop traffic on links in the network. They dissect the protocol fields of the packets, determine if they belong to a VoIP session, and if they do, collect and store statistics of the session as *records*. The probes then send the records to the NOC. The NOC correlates the measurements from probes by determining the probes through which the same VoIP session flows and then computes the call quality of the flow between the probes.

A. Measurement records

A passive probe maintains two sets of records, one corresponding to SIP and the other corresponding to RTP. Each record contains three fields, protocol, key, and statistics. The protocol corresponds to the protocol being measured, SIP or RTP. The key corresponds to a unique identifier of the connection to which the record corresponds to. The measurements from the probes corresponding each protocol connection are stored in the statistics field, and this field is different for each protocol.

1) SIP Record: The key for a SIP connection is defined as the 3-tuple, ($to, from, call\_id$), where to, $from$ and $call\_id$ are fields in the SIP header [6]. For each SIP connection, we store the time when the probe sees a SIP message with the key corresponding to the connection. The messages we are interested in are INVITE, TRYING, RINGING, 200 OK, and BYE. Using these times, the NOC can determine the call setup and tear down times for each VoIP session observed by the probes.

For any SIP message, we also store the source IP addresses of the UDP or TCP packet. A SIP host sending an INVITE corresponds to one participant of the VoIP session and the host sending the 200 OK corresponds to the other. The IP address of these messages is used to correlate the SIP records with the corresponding RTP records by the NOC.

2) RTP Record: The key for an RTP connection is defined as the 5-tuple, ($src\_ip, src\_port, dst\_ip, dst\_port, ssrc$), where src_ip and dst_ip correspond to the source and destination IP addresses respectively, src_port and dst_port correspond to the source and destination UDP ports respectively, and $ssrc$ is the synchronization source field in the RTP header [12].

Each record contains the measures, the average estimated time stamp, the number of packets observed, the minimum and maximum RTP sequence number, and the arrival times of the first and the last RTP packet. The average estimated timestamp is the average value of the timestamps of each RTP packet received. When losses occur in the network, this average can get skewed. Hence, we adjust for losses and store an estimate of the average timestamp value. This is used to compute the delay through the network.

B. Passive probe components

Our implementation of passive probes uses off-the-shelf desktop computers or servers running Linux. Similar to active probes, passive probes synchronize their local clocks using NTP or more accurate mechanisms. Fig. 6 shows the components of the passive probe.

1) Packet capture and dissection: For links that have low traffic volumes ($50 - 100$ Mbps), we use commercial NICs to perform the snooping. For higher rates, we use high-end DAG [13] capture cards. To snoop the traffic on links, there are a few methods that are available. For 10-100 Mbps Ethernet links, we can use a hub to connect the passive probe and snoop on all the traffic on the hub. Some switches and routers support mirroring of one port's traffic on another and this can be used by passive probes to monitor the traffic. Another method is to install a passive tap (optical splitter) and use the tap for monitoring.

We use the *libpcap* library to perform the packet capture from the links. We capture all packets that are of type UDP and packets that are destined to the SIP port (5060). We capture all UDP packets because any UDP packet can be an RTP packet, and we identify packets that are RTP packets through heuristics. To determine if a packet is an RTP packet, Reference [12] prescribes certain tests. The version number of the packet needs to be 2, the length of the RTP packet should be the same as the length header, and we should receive three consecutive sequence numbers at some point in the flow's duration. We use these tests to validate the packets as an RTP stream. If a stream does not satisfy the tests, we remove the RTP record from memory. For SIP, we parse the header of the packet and extract the relevant fields such as to, $from$, $call\_id$, $cseq$, $method$, and $status$.

2) Record lookup: A Gigabit Ethernet link can carry approximately 16,000 simultaneous VoIP connections. Thus, our passive probes need to be able to process as many measurement records. We use two hash tables to store the records, one for RTP and the other for SIP. Since the number of records is small, the entire hash table fits in memory, for a system with 256 MB main memory.

We use the UNIX ELF hash function for hashing, and the hash function uses the record key to perform the hashing. The ELF function provides a simple software implementation of a hash function that uses minimal computation and gives good performance. Stronger hash functions such as cryptographically secure functions, MD5 and SHA-1, are expensive to implement in software in terms of their computational resource requirements.

3) Record update: For each packet we receive, we perform a lookup to determine if a record for that key already exists. If so, we update the record. For SIP, this update is straight-forward. We store the type of header received and store the current local time of arrival of the packet in the record. For RTP, the update involves incrementing the number of packets observed, computing the average timestamp value, updating the minimum and maximum observed sequence numbers, and update the reception time of the last packet. If a record does not exist, we create a new record and update it.

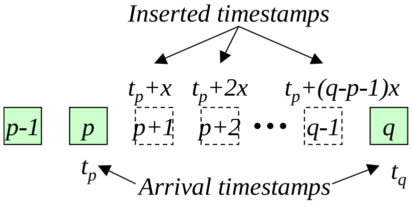


Fig. 7. Compensating for losses by inserting timestamps

While maintaining the average timestamp, packet losses can result in an inaccurate measurement. To compensate for losses, we smoothen the average. Let T represent the estimate of the average timestamp of some RTP connection, N be the number of packet received until now, p represent the last sequence number received, and t_p be the time when the last packet was received. Suppose we receive an RTP packet belonging to this connection with sequence number q and at time t_q, we update the average normally if the packet is the next packet in the sequence ($q = p + 1$).

$$T = (T.N + t_q)/(N + 1)$$

If the sequence number of the received packet is less than the last received sequence number ($q \leq p$), this corresponds to an our-of-order arrival, and we do not update the timestamp.

When we see a jump in the sequence number ($q > p + 1$), we assume that the packets with sequence numbers $p + 1$ to $q - 1$ are lost and we smoothen out the average timestamp by assuming that we received the timestamps of lost packets with equal spacing in the duration t_p to t_q. This is illustrated in Fig. 7. The spacing of the timestamps is $x = \frac{t_q - t_p}{q-p}$ and the estimated arrival times of the lost packets are $t_p + x$, $t_p + 2x$,..., $t_p + (q - p - 1)x$ respectively. We also increment the number of packets N by $q - p$ for the purpose of computation of the average timestamp. However, this does not correspond to the actual number of packets received. Thus, we maintain a separate count of the actual count of the number of packets received. We update the timestamp using the expression,

$$\begin{aligned} T &= [T.N + (t_p + \frac{t_q - t_p}{q-p}) + (t_p + 2\frac{t_q - t_p}{q-p}) \\ &\quad + \ldots + (t_p + (q-p)\frac{t_q - t_p}{q-p})]/(N + q - p) \\ &= [T.N + \frac{t_p}{2}(q - p - 1) + \frac{t_q}{2}(q - p + 1)]/(N + q - p) \end{aligned}$$

4) Record expiration: This module determines when a record needs to be cleared from memory and sent to the NOC. A SIP record is expired using two criteria. If we receive a 200 OK for a BYE request, this signals the termination of the connection, and we expire the record and sent the record to the NOC. We also expire the record if we do not receive any message in the session for a timeout period.

An RTP record expiration uses two criteria, an inactive timeout and an active timeout. An inactive timeout is triggered when we do not receive any RTP packet in the session. This typically signals the termination of the VoIP session, and the record is sent to the NOC. An active timeout is used when during a period (longer than the inactive timeout period), we do not expire the record using the inactive timeout expiration. This is used for sending periodic reports to the NOC for VoIP sessions that last for a large duration.

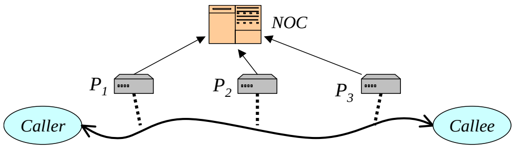


Fig. 8. Example showing probes observing a VoIP session

5) *Exporter:* The exporter is a thread running on the probe that handles the transmission of expired records to the NOC. The exporter formats the records such that each record consists the address of the probe, the time of export, and the record. The exporter maintains a socket connection to the NOC and uses this to transmit the records.

C. Call metrics computation

We now discuss how we compute the call metrics at the NOC using the records from the probes.

1) *Call setup and tear down times:* The NOC can receive SIP records with the same key from multiple probes because a VoIP session can be observed at multiple probes as shown in Fig. 8. We compute call setup time as the time between when a probe gets a SIP INVITE and when it receives a 200 OK. It is easy to see that the probe closest to the caller will have the largest call setup time value. Thus, the NOC computes the call setup times at the probes that observe the call session and reports the maximum of these values as the call setup time. Since, the records from the probes need not arrive at the same time, we wait for a period of time at the NOC for all records to arrive. This waiting period typically needs to be at least as much as the maximum delay from a probe to the NOC. Note that the reported call setup time does not account for the delay from the caller to the closest probe, and this is not the end-to-end value.

Consider the scenario shown in Fig. 8. A VoIP session is observed by three probes, P_1, P_2, and P_3, and the three probes report records to the NOC. Since the caller initiates the call by sending the SIP INVITE, the delay between when the probes observe an INVITE and the 200 OK response from the callee is the largest for P_1. Thus, the NOC reports the call setup time using P_1's measurements. The delay from the caller to P_1 for the INVITE and the delay from P_1 back to the caller for the 200 OK message is not accounted for in this setup time.

The call tear down time is computed similar to the setup time computation as the time between when the probes receive a BYE and the time when they receive a 200 OK.

2) *Average one way delay:* The NOC maintains one-way delay estimates for each RTP connection for which it receives a record from some probe. The forward and reverse RTP connections of a VoIP session have different RTP records, and the source and destination IP addresses of the forward connection are interchanged to get the addresses of the reverse connection. When the NOC first receives a record for an RTP connection, it stores the record and waits for records from other probes to arrive. It waits for a time period that is greater than the sum of the active timeout period and the maximum delay between probes and the NOC. This ensures that the NOC receives records from all probes that observe the RTP connection.

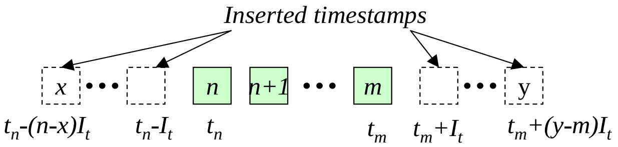


Fig. 9. Compensating for losses by inserting timestamps at the NOC

Consider two probes that observe some RTP connection. Let the number of packets they see be N, assuming no losses in the network and let the arrival times at the first and the second probe be x_i and y_i, $\forall i \in [0, N-1]$ respectively. The average one way delay, D, is given by

$$\begin{aligned} D &= \frac{1}{N}\sum_{i=0}^{N-1}(y_i - x_i) = \frac{1}{N}\sum_{i=0}^{N-1} y_i - \frac{1}{N}\sum_{i=0}^{N-1} x_i \\ &= T_2 - T_1 \end{aligned}$$

where T_2 and T_1 are the average timestamp values at the second and the first probe respectively. Note that the second probe can be identified by a larger average timestamp value than the first's value when the clocks at the probes are synchronized. Thus, in the example in Fig. 8, for the RTP flow from the caller to the callee, P_3's average timestamp estimate will be greater than P_2's estimate, which in turn will be greater than P_1's estimate.

When there are losses in the network and the second probe receives lesser number of packets than the first, the delay estimate will not be not accurate because of the skew introduced by the losses. The losses at the second probe can be either at the beginning, or at the end, or at in the middle of the flow's duration. Each probe compensates for losses in the middle through the procedure described in Section IV-B.3. The losses at the beginning and at the end, however, cannot be accounted at the probes as the probes do not know the actual number of packets in the connection. These losses are compensated for at the NOC by assuming that the lost packets arrive at a constant interarrival time. The interarrival time is determined using the total time of observation (given by the time of last packet arrival and the time of first arrival) and the total number of packets observed.

Consider the case when the first and last sequence numbers of some flow seen at a probe, P_3, in Fig. 8 are n and m respectively. Suppose another probe, P_1 has first and last sequence number x and y for the same flow and $x \leq n$ and $y \geq m$ (that is, this probe has observed this flow somewhere before in the path and hence, has seen more sequence numbers). Then we add $n - x$ extra timestamps at the beginning and $y - m$ extra timestamps at the end of the flow seen at P_3 as shown in Fig. 9. The interarrival time, I_t, is given by $(t_m - t_n)/(m - n)$. We thus update the timestamp as

$$\begin{aligned} T &= [T.(m-n) + t_n(n-x) - I_t(1 + \ldots + n - x) + \\ &\quad t_m(y-m) + I_t(1 + \ldots + y - m)]/(y-x) \end{aligned}$$

Using these estimated average timestamp values, we compute the average one way delay approximately.

3) Round trip delay: The round trip delay is calculated as the sum of the one way delays of the forward and the reverse RTP connections, which can be determined by the source and destination IP addresses and ports. It is possible that the NOC has only one direction's records because the other direction's connection uses a different path through the network and does not pass through the passive probes. In this case, we report round trip delay as twice the one way delay.

4) Packet loss percentage: The first probe that observes traffic of an RTP connection receives the maximum number of packets. For instance, in Fig. 8, P_1 receives more packets than P_2 or P_3 for the RTP flow from the caller to the callee. Consider an RTP record from the first probe, and if the minimum sequence number seen be x and the maximum sequence number be y, then $y - x$ is the maximum number of RTP packets that will be received by all the probes if there are no losses in the network. The fraction of lost packets at some probe is given by $(y - x - N)/(y - x)$, where N is the actual number of packets received by the probe, as reported in the RTP record from that probe.

5) Jitter: Consider an RTP packet stream with N packets that is observed at two probes. Let the packet timestamps at the probes be t_i^1 and t_i^2, $\forall i \in [0, N-1]$ respectively. Jitter is defined as the mean deviation of the absolute difference of the interarrival time at the receiver as compared to the sender. The absolute difference, D, of interarrival times of some pair of packets is given by

$$D = |(t_{i+1}^2 - t_i^2) - (t_{i+1}^1 - t_i^1)|$$

The timestamps at the first probe are not available at the second and vice versa. Thus, this absolute value of the interarrival time differences cannot be computed at the probe. If we need to compute it at the NOC, we need the timestamp of every packet and this is too expensive to perform due to the size of each RTP record that needs to be exported to the NOC.

The VoIP source inserts a timestamp value at the source. However, this timestamp does not correspond to the actual time at the source, and is defined by the codec that is used. For example, the timestamps of a source that is using G.711 codec correspond to the number of voice samples in the packet. To use this timestamp information to calculate jitter, we need to identify the codec being used at the source. Also, the jitter calculation becomes codec specific.

The jitter value is used to identify losses at the receiver due to the playout buffer. Thus, we also need to know the size of the playout buffer at the receiver to compute this information. We do not implement jitter calculation in our probes currently and are investigating methods of implementing it. Currently, we assume infinite playout buffer sizes at the receivers and equivalently, there are no losses due to jitter at the receivers.

6) R-factor and MOS: Using the delay values, one way as well as round trip, and the loss percentages, we calculate the R-factor and MOS using the E-model as described in Section II-B. When multiple probes observe some some VoIP connection, we can compute the network performance between each pair of probes, thus enabling us to segment the network's performance. This can be used to pinpoint the actual cause of service degradation more effectively than end-to-end measurements. The ability to pinpoint causes of service degradation depends crucially on the placement of probes in the network. Probe placement algorithms are discussed in Reference [15]. The R-factor and MOS values do not reflect the end-to-end network performance since we account for delays between probes only and not end-to-end delays.

V. Reports

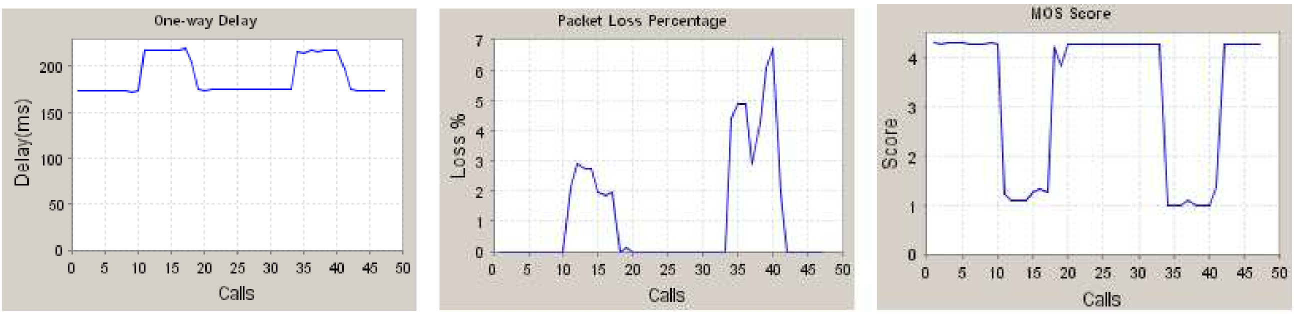
We now present the reports generated by the probes for a particular test scenario. We installed the active probe software on two machines at two enterprise locations, one in Bangalore, India and the other in Murray Hill, NJ, U. S. A.. We denote the two machines as BA and MH respectively. We performed a measurement test between the two sites and setup 50 VoIP calls using the G.711 codec and measured the performance of the network between the two sites. We also used passive probes to snoop on the calls made by the active probes at the same two locations. While the calls were in progress, we setup UDP sources to send continuous traffic from MH to BA. This causes the routers buffers to get filled and results in packet drops. We start one source at around the 10th call and stop it at the 18th call. We then start two sources at around the 33rd call and stop the sources at the 42nd call.

Fig. 10 shows the results from the active probes and Fig. 11 shows the results from the passive probes. The delay measurements from the active probes include the packetization delay introduced at the source and is hence, around 25 ms higher than the delays measured by the passive probes. The passive probes measure losses accurately as compared to the losses observed by the active probes.

VI. Conclusions and Future Work

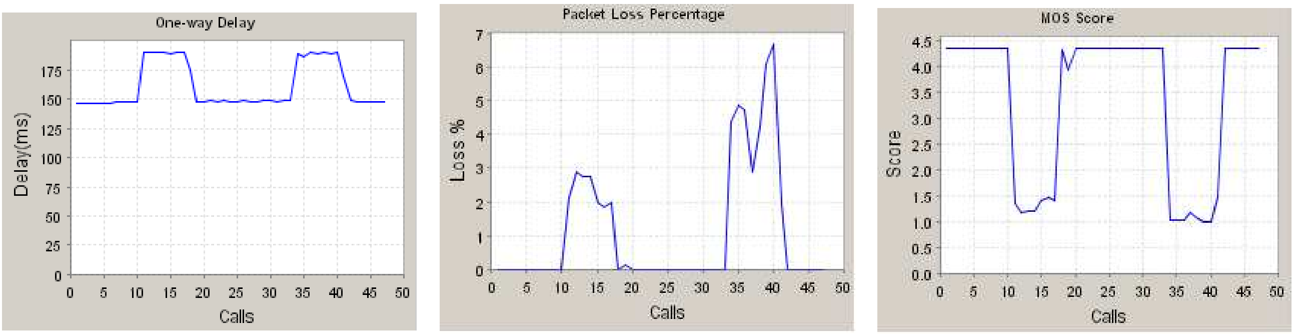
Since IP networks have not been designed to provide the quality requirements for VoIP networks, monitoring the network to identify performance degradations and to provide quality assurance is crucial. In this paper, we have discussed the design and implementation of active and passive probes that can be used to monitor the network performance for VoIP effectively. We present the computation of call quality using E-model, which uses delay, loss and jitter measurements from the network, and probes are used to perform the necessary measurements in the network. This can be used to identify service quality degradation and take remedial actions to ensure that the impact of the degradation on the end-user is minimal.

Active probes are software agents running on end systems in the network, and can be used to establish VoIP calls and measure the parameters that affect call quality. We discussed the architecture of our active probing system and the implementation of the probes to perform the measurements. The system does not require central coordination and thus, places minimal burden on the NOC and scales to a large number of probes. The probes can



(a) Delay (b) Loss (c) MOS

Fig. 10. Measurements using active probes for the call stream from MH to BA



(a) Delay (b) Loss (c) MOS

Fig. 11. Measurements using passive probes for the call stream from MH to BA

also handle NATs for a particular case when the caller is behind a NAT and the callee is outside.

Passive probes are used to snoop on network traffic to perform measurements. In this paper, we presented the design of passive probes and discussed the implementation of the various components. We identified issues in performing accurate delay measurements in the presence of losses and presented a mechanism that compensates for losses by smoothening the measurements. Finally, we discussed how the measurements at the probes can be correlated at the NOC to compute the network performance metrics.

Once probes detect service quality degradation in the network, service providers need automated tools to identify root cause of the degradation. This requires the integration of other network management tools with probing tools, like autodiscovery mechanisms that discover the topology of the network, and automatically detect users and applications in the network, and other tools that monitor the performance of network elements.

The performance of the SIP proxy plays a crucial role in the service quality of a VoIP network, in terms of call setup and teardown times. The SIP proxy also uses backend transactions (e.g. DNS lookups, database transactions for user identification and authentication) to perform the proxying functions. Passive probes can be used to monitor the load on SIP proxies, and the delays of the various transactions. This needs to be expored further.

References

[1] Skype, http://www.skype.com.
[2] Dialpad, http://www.dialpad.com.
[3] Net2Phone, http://www.net2phone.com.
[4] Vonage, http://www.vonage.com.
[5] G. Prabhakar, R. Rastogi, and M. Thottan, "OSS architecture and requirements for VoIP networks," *Bell Labs Technical Journal*, May 2005.
[6] J. Rosenberg *et al*, "SIP: Session Initiation Protocol," *RFC 3261*, June 2002.
[7] ITU-T Rec. G. 107, "The E-model, a computational model for use in transmission planning," March 2003.
[8] PESQ, "Perceptual evaluation of speech quality," http://www.pesq.org
[9] R. G. Cole and J. H. Rosenbluth, "Voice over IP performance monitoring," *ACM SIGCOMM Computer Communication Review*, April 2001.
[10] M. E. Perkins, C. A. Dvorak, B. H. Lerich, and J. A. Zebarth, "Speech transmission performance planning in hybrid IP/SCN networks," *IEEE Communications Magazine*, July 1999.
[11] D. L. Mills, "Network time protocol (Version 3) specification, implementation and analysis," *RFC 1305*, March 1992.
[12] H. Schulzrinne *et al* "RTP: A transport protocol for real-time applications," *RFC 3550*, July 2003.
[13] Endace http://www.endace.com
[14] Libpcap http://www.tcpdump.org
[15] S. Agrawal, K. V. M. Naidu, and R. Rastogi, "Anomaly detection and diagnosis using passive monitoring," *Paper under submission*