

Reliable IP SoftPhones with Rapid Network Failure Detection

Mark Karol

Avaya Labs Research
307 Middletown Lincroft Road
Lincroft, NJ 07738-1526
Email: mk@avaya.com

P. Krishnan

Avaya Labs Research
233 Mount Airy Road
Basking Ridge, NJ 07920-2311
Email: pk@avaya.com

J. Jenny Li

Avaya Labs Research
233 Mount Airy Road
Basking Ridge, NJ 07920-2311
Email: jjli@avaya.com

Abstract - In this paper, we propose an enhancement to IP SoftPhones that rapidly detects network failures and informs users of network problems during VoIP sessions, which will allow users, for example, to decide whether to terminate or reroute their affected calls. The failure detection and notification may be specific to certain traffic sessions. A “keep-alive” signal is used for rapid failure detection; e.g., the sender IP SoftPhone increases the number of RTCP packets when the number of RTP packet decreases (during silent periods), and the receiver SoftPhone declares that a failure has occurred if neither RTP nor RTCP packets are received within a certain time window (e.g., 1 – 2 seconds).

Internet Protocol (IP) networks are inherently unreliable. Network applications such as Voice over IP (VoIP) must take this into consideration to improve service quality. The deployment of VoIP applications includes devices such as media gateways, and software such as communication managers and IP SoftPhones. SoftPhones are stand-alone software applications for PCs; they are end stations in VoIP communication.

In this paper, we propose an enhancement to IP SoftPhones that rapidly detects network failures and informs users of network problems during VoIP sessions, which will allow users, for example, to decide whether to terminate their affected calls. The failure detection and notification may be specific to certain traffic sessions. Feedback to the user can take many forms: e.g., a voice or text message, or a button on the SoftPhone that lights up saying, in effect, that a failure has occurred and “Network Restoration is in Progress.” In other words, the network itself has put the user “On Hold.” The key point is that once the user (or network manager) is made aware of the network problem, they can decide what to do (or what not to do) long before the network recovers.

Currently, it is possible to provide voice quality feedback to a user participating in a VoIP session. Also, various metrics related to RTP (Real-Time Transport Protocol) traffic might be computed by an endpoint application such as IP SoftPhone and then presented to the user. However, current network devices do not automatically inform applications or users about network problems. These are mostly gleaned via concepts like timeouts. The emphasis in VoIP and other applications has been to deliberately hide the network and its problems from the user.

In this paper, we propose that a “keep-alive” signal be used

for rapid failure detection. In our method, the sender IP SoftPhone increases the number of RTCP packets when the number of RTP packet decreases (during silent periods), making sure that, for example, at least one RTP or RTCP packet is sent every T seconds. Specifically, additional (short) RTCP packets are injected at rate $R = 1/T$ during silent periods in the RTP stream. The receiver SoftPhone declares that a failure has occurred if neither RTP nor RTCP packets are received within a window of kT seconds ($k = 2, 3, \dots$). The receiver then notifies the users who are involved in the sessions, plus, perhaps, users that might be affected if they were to attempt to set up a new call. Since the RTP/RTCP packets experience network jitter and loss, there is a chance that a failure will be wrongly declared (i.e., a “false alarm”). However, as soon as either an RTP or an RTCP packet is once again received, the receiver cancels its “failure announcement.” This could either (i) correct a false alarm or (ii) announce restoration after an actual failure.

We implemented a version of our new failure detection and notification method on an IP SoftPhone application. Figure 1 shows the setting of a general network with two IP SoftPhone endpoints.

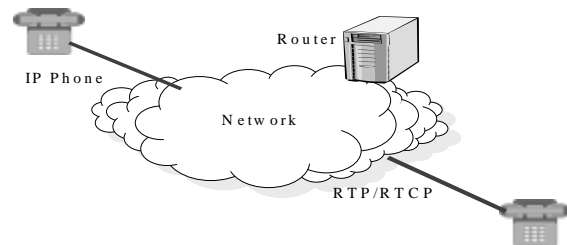


Figure 1. A general setting of SoftPhones

Instead of using an actual network environment where the network performance is out of our control, we used an emulated network with two actual IP SoftPhones. Figure 2 shows the setting and the components of our implementation. It includes two IP SoftPhones and one router, all of which ran on Linux machines. Each SoftPhone includes four components: RTP/RTCP stack at the bottom, sender and receiver in the middle, and the phone application (with detector and notifier) built on top of the sender and receiver.

We installed *NIST Net* (see <http://snad.ncsl.nist.gov/itg/nistnet>) on the router to emulate network failure and QoS (delay, loss, jitter) impairments.

By operating at the IP level, *NIST Net* can emulate the critical end-to-end performance characteristics imposed by various wide area network situations (e.g., congestion loss). The tool allows an inexpensive PC-based router to emulate numerous complex performance scenarios, including: tunable packet delay distributions, congestion and background loss, bandwidth limitation, and packet reordering/duplication. It allows users to apply selected performance "effects" to the stream's IP packets.

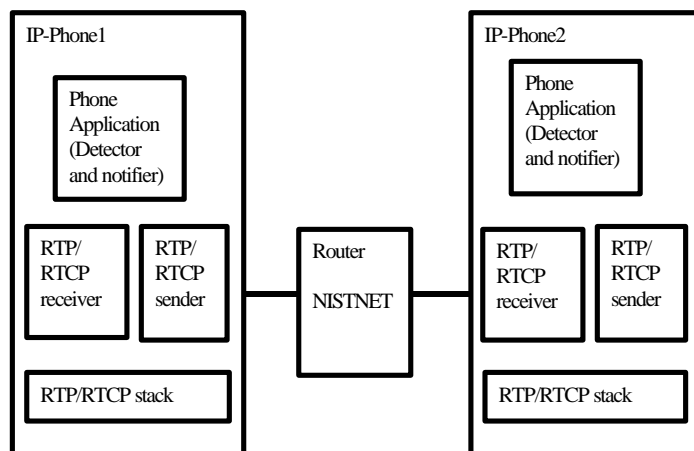


Figure 2. Experimental setting of failure detection and notification method

We implemented the failure detection and user notification components on the phone application. The sender of the application was modified to insert RTCP packets during silent periods, and the receiver was modified to wait for additional RTCP packets during silent periods. A new component, detector and notifier, was added to the application to decide when failure occurs and to signal the failure to the user. The various components were synchronized so that accurate measurements could be made of the detection times.

Our experimental setup uses one router between the two end-phones. However, for all practical purposes, it models general network scenarios including the cases when there are various routers (hops) between the two IP-phones, since we can introduce delay and jitter with the *NIST Net* network emulator. The number of hops between the two phones will not affect our experimental results because the delay caused by geographical distance has an impact on all packets and our method waits for a period of length kT with no packet reception before declaring a fault.

On our implementation platform, we carried out several sets of failure detection and notification experiments. The number of RTP packets per talkspurt was a random number uniformly selected between 1 and 20, and the RTP silent periods were uniformly distributed between zero and two seconds. We designed two kinds of experiments to study the failure detection times and the false-alarm probabilities: 1) a Detection time experiment, and 2) a False-alarm experiment.

We observed that faults were discovered quickly (e.g., in less than one second if the keep-alive RTCP packets are transmitted, e.g., once every $T = 250$ msec) and we didn't declare faults incorrectly (i.e., "false alarms") in 10-minute calls when k was greater than or equal to three. These experimental results show the effectiveness of the proposed failure detection and notification method. The time from failure occurrence to detection is small and the probability of false alarm is small.

Packet loss is the major contributor to false alarms. The impact of jitter will be negligible since, typically, the failure detection intervals (k/R) will be much greater than the delay jitter experienced by RTP and RTCP packets. It is unlikely that jitter alone will cause the detection interval to pass without reception of any RTP or RTCP packets.

The proposed enhancement to IP SoftPhones reduces failure detection time by coordinating the sending frequency of RTP and RTCP packets. There are three key aspects to this new design. First, it exploits the fact that detection times today are often faster than network restoration times. Second, it recognizes that there typically is a significant difference between the information available at network devices and the information that is actually conveyed to users/applications. Although this difference is often for good reason (e.g., not overburdening users/applications with unimportant information), there sometimes are advantageous ways for users/applications to exploit information about the network conditions (which is the third key aspect of our proposal). Third, it exploits the notion that the applications that are using the network currently are most interested in knowing about its state, and so the application can deduce this information from packets flowing in the session and report it to the user.

Our ongoing research includes the following topics:

1. Other than RTCP SDES for keep-alive, we may use other methods such as special mechanisms during RTP silent periods, or using occasional RTP packets, or specialized RTP or RTCP packets as keep-alive messages.
2. The detection can also be used in some cases for error recovery such as re-routing calls.
3. IP SoftPhones can be used in two modes: i) where the SoftPhone only does signaling and a traditional PSTN phone is the media terminator, and ii) where the SoftPhone receives and plays the media. If the SoftPhone sends special packets every T seconds, the SoftPhone could potentially start out in mode ii) and the media gateway could switch to mode i) if it notices a fault. The uniqueness of this design is that rather than check from our side if the connection is up (as with pings), we help the other person know for sure if he is receiving our packets.

In the future, we plan to experimentally determine various failure detection speeds and capacities. Also, it would be interesting to measure the restoration times in real networks and compare with the detection and reaction times attainable with our design.