Efficient, Scalable Migration of IP Telephony Calls for Enhanced Fault-tolerance

Manish Marwah
Department of Computer Science,
University of Colorado,
Campus Box 0430, Boulder, CO 80309-0430
marwah@colorado.edu

David Chavez Don Gillespie Vivek Velamala
Avaya Labs,
1300 West 120th Ave.,
Westminster, CO 80234
{davidchavez,dongillespie,vivek}@avaya.com

Abstract

IP telephony or VoIP has seen tremendous growth in the past few years and is predicted to grow rapidly in the coming years. Hence, IP telephony components must provide service with high-availability, comparable to traditional telephony systems. In this paper, we present an efficient and scalable fault-tolerance mechanism for migrating calls to an alternate IP telephony call controller in the event of the failure of the call controller or network connectivity to it. We also present an efficient algorithm for merging components of migrated calls, so that the same call features are available on these calls as the original calls. Some of these techniques have been incorporated in commercial products, resulting in improved fault-tolerance.

Keywords: *IP Telephony, VoIP, fault-tolerance, dependability, high-availability, H.248, H.323.*

1 Introduction

IP telephony or VoIP (Voice over IP) has seen tremendous growth in the past few years and is predicted to grow rapidly in the coming years. Both business and residential customers are switching to IP telephony, attracted not only by cost savings associated with convergence of data, voice and video, but also by the numerous compelling services and applications that arise from the integration of voice and Internet-based applications such as email and instant messaging. Furthermore, for businesses, IP telephony leads to simplified network administration and management due to convergence of voice and data networks.

To be universally accepted, IP telephony must provide telephony services with the same degree of reliability and availability as circuit switched based telephony systems. A large body of research has focused on QoS and reliability of transporting voice over the Internet [6, 3], however, not much attention has been given to efficient fault tolerance techniques for other components of a VoIP system. Specifically, it is challenging to provide an efficient and scalable mechanism for call-preserving recovery of IP telephony components on failure of their call controller. In this paper, we describe such a call-preserving mechanism for migrating calls to an alternate call controller. Furthermore, we present an efficient algorithm for merging call components of migrated calls.

In IP telephony systems, clients (such as IP endpoints and VoIP gateways e.g. Avaya G700 Media Gateway) usually connect to a call controller (such as Avaya Communication Manager) over an IP network for basic call services and various call features. Failure of the IP network or the call controller (CC) leads to service outage at the gateways and IP endpoints. This problem is usually addressed by installing alternate CCs that can provide service in the event of such a failure. However, when gateways or IP endpoints migrate to a new controller, existing calls on them may get torn down. Even if they are not torn down, no CC-implemented call features are available on these calls since the new controller has no knowledge of the state information associated with them. To be able to preserve not only the call bearer connections but also call features on such existing calls requires that the call state information be made available to the new controller. We refer to the ability of moving a call from one controller to another, while preserving the bearer connections and CC-implemented call features, as call migration. This ability is very useful in providing seamless service in not only failure situations but also during controller maintenance (e.g. upgrade of HW/SW on a CC) and potentially for load balancing, moving calls to a preferred controller, etc.

Call migration requires sharing or transferring of call session state information between call controllers. We propose an efficient and scalable mechanism to share call session state for implementing call migration. This technique is similar in concept to the HTTP cookie mechanism [4]. The basic idea is that a CC saves call session state information on IP telephony entities such as VoIP gateways and IP endpoints; on failover, the alternate CC retrieves the call session state information from the failed-over entity and uses this information to reconstruct calls.

"Client side session cache" mechanism [7] is another conceptually similar technique. It can be used by a server to save TLS session state information on a client; this information can be used later to restart a TLS session, saving time since a new session does not have to be negotiated. The objective of saving the session information at the client, however, is not fault-tolerance, but to ease storage requirements at the server. The alternative – saving session state at the server – may be impractical in situations where the server interacts with a large number of clients.

The next section provides a functional overview of an IP telephony system. This is followed by goals for the fault-tolerance mechanism described in this paper. Section 4 describes the call migration technique being proposed here. Merging components of a migrated call, spread over multiple VoIP gateways, is discussed in Section 5. Finally, the conclusions are presented in Section 6.

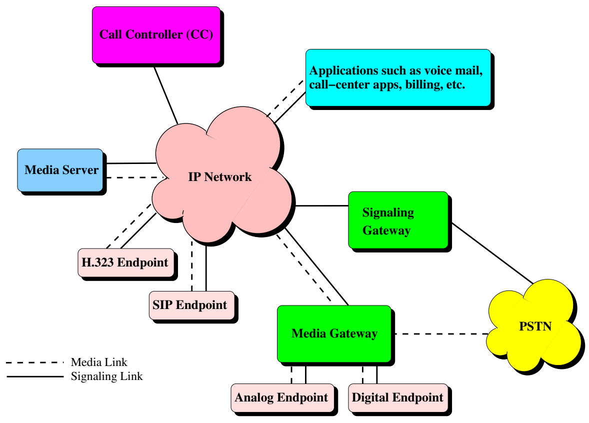

Figure 1: Functional architecture of an IP telephony system

2 Architecture overview of a IP telephony system

A high-level view of an IP telephony system is shown in Figure 1. The figure identifies the main functional components of such a system; however, it does not show how such a system may be implemented. Indeed, in an actual instantiation of such a system many of the functional components may be distributed or implemented together or co-located. Furthermore, they may be geographically separated and connected by wide area network links. The main functional components, shown in the figure, are briefly discussed below.

Call Controller (CC). The CC functional component aggregates together a number of functionalities such as that of a H.248 [1] media gateway controller (MGC), H.323 [2] gatekeeper, feature server, SIP registrar and proxy, etc. It provides call logic and call control functions, and also typically maintains call state information.

An excellent description of the H.248 and H.323 protocols is provided in [8] and [5], respectively.

VoIP Gateways. Considering the huge amount of existing investment in traditional TDM-based telephony, it is evident that IP and TDM-based telephony systems will coexist and will need to inter-operate for a long time to come. Thus, an important part of the system is gateways which allow communication between IP and TDM-based circuit switched systems. A media gateway (MG) terminates media streams from more than one kind of network, e.g. IP and TDM, and thus allows those different networks to inter-operate. Similarly, a signaling gateway (SG) converts signaling protocols from one kind of network to another. Specifically, MGs and SGs are essential for inter-working between IP and TDM based public switched telephone network (PSTN). For example, if an analog or digital phone needs to call an IP phone, both voice and signaling for the call will pass through such gateways. In this paper, we use the terms gateway and VoIP gateway interchangeably.

Media Server. A media server provides a number of media services such as announcements (e.g. *the number you have dialed has been disconnected...*), music on hold, etc. It also provides DTMF tone detection (which is, e.g, used for recognizing tones of the digits pressed while making an automated credit card payment on the phone), and, call classification (which involves appropriately routing a call based on caller input, including using automatic speech recognition techniques). A media server may also provide resources for hosting conference calls.

Endpoints. Endpoints are the devices used for making and receiving calls. These can be traditional (that is, analog or digital, e.g., Avaya DCP phones) or IP endpoints. An IP endpoint typically uses SIP or H.323 protocols for call signaling. It is usually also available as a softphone, an IP endpoint application that runs on a generic computer. In this paper, the term endpoint without qualification refers to an IP endpoint.

3 Requirements/Goals

We were guided by the following goals while designing the failure-recovery mechanism described in this paper.

- **Failover to an alternate CC.** Gateways and endpoints failover to an alternate CC in the event of failure of their current CC or failure of network connectivity to it.

- **Call session preservation.** Call session state of stable calls is preserved on failover of gateways and endpoints to an alternate CC. A call is considered stable if it is fully established, that is, all signaling messages required to setup the call have been exchanged and the voice paths already established. Also, we distinguish between preserving just the connection, i.e. the voice path, of a call (connection preservation) and preserving the connection as well as the call session state of the call (call session preservation). In case of connection preservation, it may not be possible to make changes to the state of the call. Specifically, call features implemented at the CC are not accessible to the call, e.g. the user may not be able to conference in another party to the call. On the other hand, in case of call session preservation, the call session state is available at the alternate CC which allows the users on the call to continue to access call features offered by the CC.

- **Seamless failover and recovery.** Failover to an alternate CC and recovery are seamless from the users' point of view, that is, they do not experience an outage.
- **Scalability.** The fault-tolerance mechanism scales well, preferably linearly, with increase in the number of gateways and endpoints.
- **Efficiency and minimal overhead.** The fault-tolerance mechanism is efficient such that there is minimal overhead during normal operation (no failures). Furthermore, failover and recovery is fast so that there is minimal disruption in the continuity of service to the user.

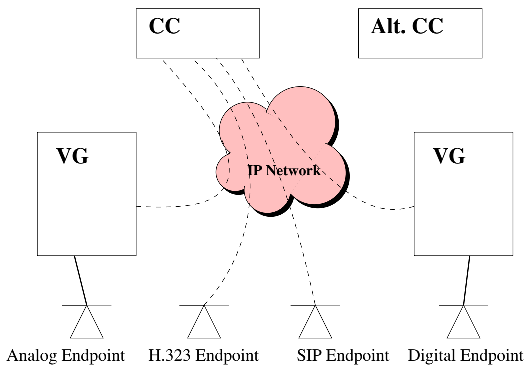

Figure 2: An IP telephony system.

4 Call Migration

Without loss of generality, we will consider the IP telephony system elements shown in Figure 2 for discussing call migration. During normal operation (failure-free periods), the gateways and endpoints receive service from the main call controller. The VoIP gateways (VGs), shown in the figure, combine the functionality of both media and signaling gateways. In addition, they also act as media servers. Depending on the needs of the organization deploying the IP telephony system, the VGs may be distributed over a wide area and connect to the main CC over WAN links.

If the main CC, or network connectivity to it, fails, the gateways and endpoints seek service from an alternate CC. Based on the degree of service availability required, there may be one or more alternate CCs. Usually, VGs and endpoints that are remotely located are provided with a local alternate CC for failure scenarios where the remote location may lose connectivity with the main location.

In order to provide seamless service, calls on a particular VG are migrated to an alternate CC when that VG fails over. To accomplish this migration, the alternate CC needs state information related to the existing calls. There are various ways to synchronize this information to alternate CC(s). However, the mechanism we describe below is more scalable, robust and efficient as compared to others. We compare our methodology with the alternatives in Section 4.3.

Note that call migration can also be used in non-failure conditions for moving gateways and IP endpoints, and, migrating the associated calls from one controller to another for reasons other than failures such as maintenance, load balancing, etc.

4.1 Saving State on Clients

As calls are made, a CC saves the call state information on the MGs and IP endpoints, similar to how an HTTP server saves client state information on the client machines as HTTP cookies [4]. When a CC fully establishes a call, it generates one or more messages with call state information and sends them to the VG and/or IP endpoint(s) involved in that call. If the state of a call changes (e.g. another party is added to the call), the controller sends out updated message(s). When a call ends, the CC sends out a delete message to purge the call state information related to that call. For performance efficiency, the cookie messages can be piggy-backed on regular signaling messages exchanged between the CC and VGs/endpoints.

Like an HTTP cookie, the call state information is opaque to the client, that is, a VG or an IP endpoint has no knowledge of its content; only call controllers are aware of its structure and semantic content. This is very advantageous as changes can be made to the structure or content of a call state information message without requiring any code changes at all on the VGs or IP endpoints.

4.2 Reconstructing Calls

When a VG or an IP endpoint fails over to an alternate call controller, as part of initial negotiation, all the existing calls on the VG or IP endpoint are migrated to the new call controller. This new CC queries [1] the VG or IP endpoint

[1]For a H.248 gateway, the H.248 audit mechanism [1] can be used for this purpose.

for its call state information. This state information is then used to reconstruct (as far as possible) the existing calls by recreating data structures and state that represented the calls in the original controller. This allows the new controller to provide call features on those calls.

4.3 Comparison to alternatives

Beside saving the call session state at the clients (gateways and IP endpoints) of the CC, there are other ways of synchronizing the state information between the CCs. We compare our approach with two such approaches.

- The call controller and the alternate call controllers can use a common, redundant backend database where all pertinent call state information can be stored. This allows all the call controllers to have access to the same call state data. This approach has the following shortcomings.
  - Such a common, backend database can be expensive to build and maintain, considering that call session state data is very dynamic and voluminous (up to hundreds of thousands of calls per hour)
  - The database can become a bottleneck, especially as the number of controllers accessing the database increase.
  - This solution is not very robust if the gateways are geographically distributed as they often are for big enterprises, since WAN network failures can prevent alternate controllers from accessing the common database. For example, consider an enterprise IP telephony system with a main and multiple remote locations. During normal operation all the remote locations are served by the CC and the redundant database at the main location; all the remote locations have alternate CCs in case they lose connectivity to the main location. Here, if the WAN link to the main location were to fail, the alternate CC at this location will not be able to access the database.
- The call controller dynamically updates the call session state information on the alternate call controllers. Conceptually, this is similar to having a database at the main location which is replicated in real time at all the alternate CC locations. This approach has a number of drawbacks.
  - It adds a lot of overhead and complexity as the number of call controllers increase.
  - The alternate controllers have to be updated with the call state information of all the calls on the main controller. This can lead to network congestion if the link to a particular alternate controller does not have sufficient bandwidth.
  - This solution is not very robust, since network failures can prevent controllers from communicating with each other.

4.4 Advantages

We now summarize the main advantages of our approach.

1. **Enhanced Reliability.** Our solution is highly robust to network failures due to its highly distributed nature.
2. **Enhanced Scalability.** Since VGs and IP endpoints carry their own call state information, this solution easily scales to any number of alternate call controllers. In contrast to the two approaches described above, increase in the number of alternate call controllers does not require additional processing, nor does it introduce any bottlenecks.
3. **Efficiency.** Only state information that is needed is transferred to an alternate controller and it is done only when needed.
4. **Simplicity.** Requires less resources and is simpler to implement than other approaches.

5 Merging call components during reconstruction

As described earlier, when a VoIP gateway migrates to an alternate call controller, it queries the gateway for call session state information (saved by the call controller that originally created the call) in order to reconstruct the calls on that gateway.

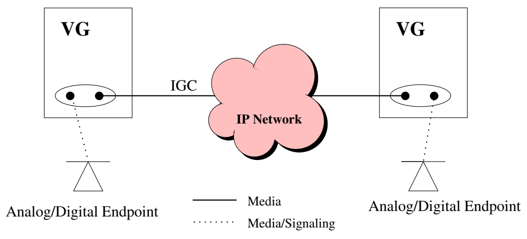

Figure 3: An inter-gateway connection.

A call may span more than one gateway. For instance, if an analog/digital endpoint on one VG calls another on another VG, an IP bearer path is created between the two VGs to setup the call. Such an IP bearer connection between two VGs is called an inter-gateway connection (IGC) and connects two parts of a call as shown in Figure 3. A conference call may involve multiple VGs and contain multiple IGCs.

Since VGs (and endpoints) may detect failure at different times, a call spanning multiple VGs will not failover in its entirety at one instance. Rather, it will failover in parts as the VGs and endpoints containing those parts failover.

As a CC processes each entity that fails over, it reconstructs the calls residing on that entity. Calls that span multiple entities have to be treated specially, since when a part of the call is being processed, the CC has to determine whether to reconstruct it into a new call or merge it into an existing call. We present an algorithm here that, during call reconstruction, correctly and efficiently merges call components on different entities, which migrate to a controller, into the same call.

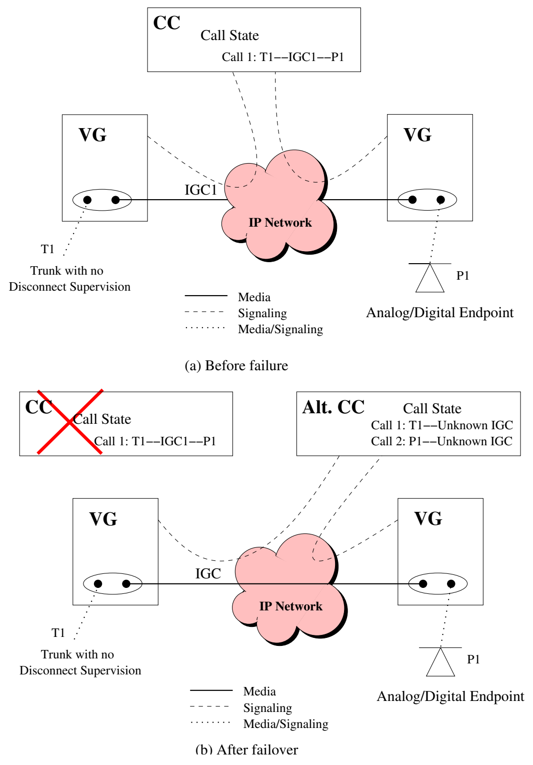

Figure 4: (a) A call between a trunk, T1 and a phone, P1 is created. (b) After failure of the CC, the VGs migrate to the alternate CC. The original call gets reconstructed as two different calls.

5.1 Advantages of merging a call

The simplest solution to merging a call is, in fact, to leave the components of a call unmerged, that is, let each component of the call be reconstructed as a separate call. This preserves the talk path, however, some of the call features will not work correctly since the structure of the call is different from the original. In other words, it is connection preserving, but may not preserve the call session faithfully.

For example, it is possible that a component reconstructed separately as a call has no disconnect supervision (ability to signal a disconnect). In Figure 4, we show a call between a phone (P1) and a trunk (T1). We assume that the trunk does not have disconnect supervision. When P1 hangs up, the entire call is torn down. However, this is not the case if the VGs migrate to another CC and the two components of the call are reconstructed as separate calls. The call with the trunk termination, Call 1 in Figure 4 (b), does not have disconnection supervision. Note that the alternate CC does not the know the identity of the remote ends of IGCs in both Calls 1 and 2.

Thus, to summarize, the main advantage of merging the components of a call is that the reconstructed call would behave exactly the same with respect to all the call features as the original call.

5.2 Call components merging algorithm

One strategy to merging components of a call is to assign an identifier to uniquely identify each call. In addition to be unique for each call generated by a CC, this identifier would also have to be unique across different CCs and across reboots of the same CC. In our algorithm, presented below, we do not use such a unique identifier and thus save the resources required in its generation and management.

Instead, to identify components of a call, at each IGC endpoint the identity of the remote end of that IGC is saved. This identity consists of two pieces of information which are saved by the CC at the gateway when a call is created:

1. Far end gateway ID (FEGI)
2. Far end context ID (FECI) [2]

(Here, we assume that the VGs are controlled by the CC using H.248 protocol. However, the merging algorithm is applicable to any VoIP gateway.)

This information is saved as a property of the ephemeral terminations involved in the IGC. During reconstruction on the alternate CC this information is retrieved from the VG and is used to merge IGCs. Whenever a call containing an IGC is to be reconstructed, it is first checked if the IGC in the incoming call matches an IGC in an already reconstructed call. If there is a match, the incoming call is merged with the matched call, else, it is reconstructed into a new call. The IGC merging algorithm can be described through the following two basic scenarios:

- **Scenario 1: Call involving one IGC.** This scenario is depicted in Figure 5. This call has two parts – C1 on VG1 and C2 on VG2. Here, on failover, the alt. CC can uniquely match the two parts of the call as the FEGI and the FECI of the two ends of the IGC point to each other.

[2]This is the H.248 context ID. However, any other gateway local ID that represents an association between the terminations in a connection can also be used.

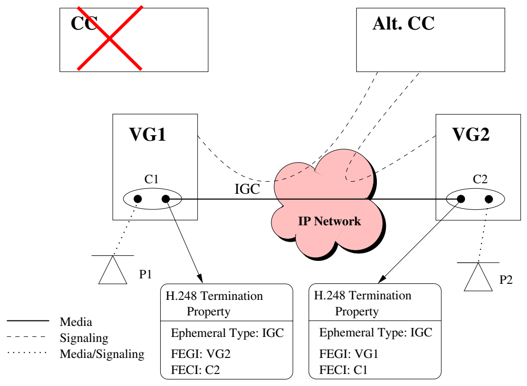

Figure 5: A call between P1 and P2 is shown. After failure of the CC, VG1 and VG2 migrate to the alt. CC, which uses the H.248 properties saved at the IGC terminations to merge the two ends of the call.

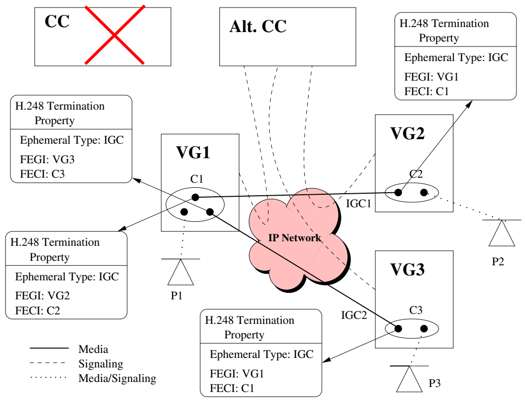

Figure 6: This shows a three party call. After the failure of the CC, the order in which the VGs migrate to the alt. CC is significant. The alt. CC uses the merge algorithm described here to merge the three parts of the call.

- **Scenario 2: Call involving multiple IGCs.** This scenario is described in terms of the example in Figure 6, where three VGs are involved in a conference call which is hosted on VG1. There are two IGCs. Unlike in the previous scenario, here the order in which the VGs migrate to the alt. CC is significant. Two cases are possible depending on the order of VG failover (all other cases can be reduced to these two, even when more than two IGCs are involved):

  – **Case 1:** VG2 first migrates to the alternate CC, followed by VG3. Here, parts of the call (C2 and C3) can be uniquely matched as FEGI and FECI of the two ends of the IGC point to the same gateway (VG1, here) and context (C1, here) respectively.

  – **Case 2:** VG2 first migrates to the alternate CC, followed by VG1. This case is identical to the scenario 1 above.

6 Conclusions

IP systems must provide toll-quality telephony services and maintain continuity of service. In this paper, we presented an efficient and scalable fault-tolerance mechanism to migrate calls from one CC to another. CCs save call session state on the IP telephony entities that host the call. This allows these entities to carry call session state information with them and seek service from an alternate CC during failures. Since the entities supply call state information, the alternate CC is able to provide call features for the session created on the original CC. The alternate CC uses an efficient algorithm to merge components of calls so that the calls retain their original structure when reconstructed.

These mechanisms have been implemented in Avaya IP telephony products resulting in preserved calls and better user experience during failures.

Acknowledgments: We would like to thank Kurt Haserodt and Jeff Meis for numerous stimulating discussions and ideas during the course of this project.

References

[1] ITU-T Recommendation. H.248: Gateway control protocol, June 2000.

[2] ITU-T Recommendation. H.323: Packet based multimedia communications systems, Feb. 1998.

[3] M. Karol, P. Krishnan, and J. J. Li. VoIP network failure detection and user notification. In *Proceedings of the IEEE Int. Conf. on Computer Communications and Networks*, Oct. 2003.

[4] D. Kristol and L. Montulli. HTTP state management mechanism. Request For Comments 2109, Internet Engineering Task Force, 1997.

[5] H. Liu and P. Mouchtaris. Voice over IP signaling: H.323 and beyond. *IEEE Communications Magazine*, Oct. 2000.

[6] A. Markopoulou, F. Tobagi, and M. Karam. Assessment of VoIP quality over internet backbones. In *Proceedings of IEEE Infocom*, June 2002.

[7] H. Shacham and D. Boneh. Fast-track session establishment for TLS. In *Proceedings of the Network and Distributed System Security Symposium*, 2002.

[8] T. Taylor. Megaco/H.248: A new standard for media gateway control. *IEEE Communications Magazine*, Oct. 2000.