Integrating Enterprise Communications into Google Wave

C. Mohit, Xiaotao Wu, Venkatesh Krishnaswamy
mohitchavakula@gmail.com, xwu@avaya.com, venky@avaya.com
Avaya Labs Research, 233 Mt. Airy Road, Basking Ridge, NJ 07920, USA

This paper proposes our approach of integrating enterprise telecommunication functions into Google Wave. Our approach can monitor communication events and utilize Google Wave functions to enhance call handling.

***Index Terms*—enterprise telecommunication, Google wave**

I. Introduction

Google Wave [1] is a new tool developed by Google for communication and collaboration on the web. It combines different communication means, such as email, instant messaging, wikis, web chat, social networking, and project management in one in-browser communication platform. It was first announced in May 2009 and is expected to be released until later in 2009. It is open sourced and has many good features such as embeddability, extensibility, and drag and drop file sharing. But among all its features, it misses one important function that is critical for enterprise users: it cannot interact with enterprise voice communication networks. To overcome this drawback, we proposed a viable way of integrating enterprise voice communication functions into Google wave. The integration consists of three parts:

- monitoring voice communication events and presenting the events in Google Wave,
- bringing Google Wave functions to enhance call handling,
- and allowing users to control calls inside Google Wave.

Each step has its own challenges. We discuss these challenges and our solutions in detail in Section III, IV, and V. This paper is organized as below. In Section II, we discuss some related work. Section III discusses how to monitor voice communication events in Google Wave. Section IV discusses new call handling features we can bring in Google Wave. Section V addresses how to control calls from Google Wave. Finally, Section VI concludes this paper and discusses our future work. In this paper, we base our discussion on Session Initiation Protocol (SIP) [2] for enterprise communication.

II. Related Work

We first introduce some terminologies in Google Wave. In Google Wave, a threaded conversation can be referred as a 'wave'. A wave consists of one or several 'wavelets', which is like a single instant messaging (IM) conversation. Each message in the conversation is called a 'blip', and the content of the 'blip' is called a 'document'. There are two ways to extend Google Wave: gadgets and robots. A gadget is an application which users can participate with, for example, a chess game can be a Google Wave gadget. A robot is an automated participant within a wave. It can talk to wave users, check the content of a wave, and perform actions to alter a wave, such as inviting another user to join the wave. The idea of using robots allows developers to add arbitrary features to a particular wave. In our implementation, we use Wave robot to monitor communication events and control calls, and use Wave gadgets to provide additional call handling functions.

There is an existing Google Wave extension called Twiliobot [3] that provides click-to-call function in Google Wave. The communication capability of Twiliobot is very limited. It simply translates the click-to-call number into a URL pointing to Twilio server, which handles call control logic. Once the user clicks the URL, the communication logic is completely out of Google Wave domain. There is no communication events sent back to Google Wave from Twilio server. In our approach, our Wave robot can monitor communication events, update call states based on the events, and apply appropriate call control actions.

III. Monitoring voice communication events

In SIP networks, call states can be monitored by subscribing to SIP dialog events [4]. Ideally, we can run Google Wave on a communication application server and let a robot directly access call control functions. Enterprise Wave server can talk to Google Wave on Google App Engine through wave federation protocol. Figure 1 shows the ideal architecture. In the figure, the Wave Bot is the key element that brings communication functions into Google Wave. Upon receiving a SIP dialog events, e.g., an incoming call, the Wave Bot can post a blip to present the event.

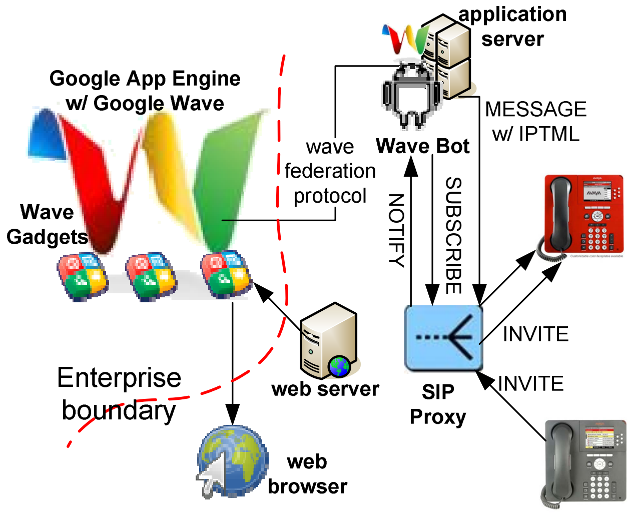

Figure 1. Ideal architecture to integrate voice communication functions into Google Wave

Currently, Google Wave can only run on Google App Engine. This makes it difficult for a Wave bot to cross enterprise boundary and access communication events. To prove the applicability of the ideal architecture, we use a border gateway to allow Wave bots to access our application server, as shown in Figure 2. In the figure, the Wave bot needs to poll events periodically. This is not scalable. But once Google Wave can reside inside enterprise boundary, we can discard the gateway. For safety reasons, a Wave bot cannot create a new thread. So we use a cron job shown below to fetch events.

```
<w:cron path="/_wave/robot/fu" timerinseconds="1" />
```

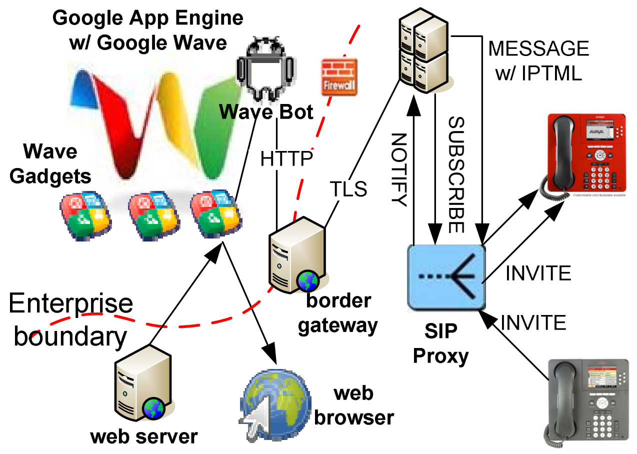

Figure 2. Our implementation to integrate voice communication functions into Google Wave

IV. Enhanced call handling in Google Wave

There are two benefits of introducing enterprise voice communication functions into Google Wave: on one hand, it adds another communication means into Google Wave and makes Google Wave a more complete unified communication platform. On the other hand, it also enhances voice communication experience by using Google Wave functions. For example, by bringing voice communication participants into Wave conversation, the participants can then use all the functions provided by Wave, such as instant message, wiki, and drag-and-drop file sharing for collaboration. Their voice communication can also be recorded and appended into the wave. In addition, we can also use Wave gadget to bring external web applications into the wave. For example, we can show remote party's colleagues upon receiving incoming call events, as shown in Figure 3.

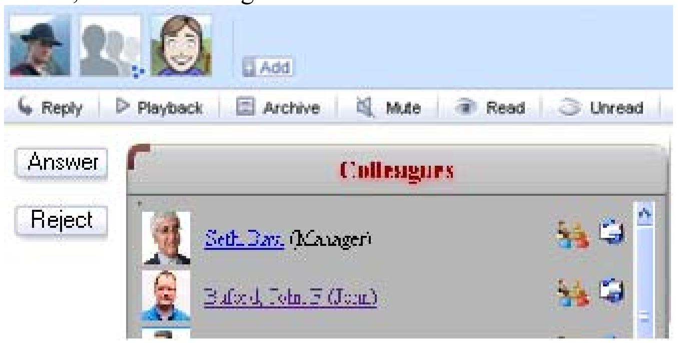

Figure 3. A wave showing an incoming call and the colleagues of the caller

V. Call control in Google Wave

In our implementation, we use first-party call control to control calls in Google Wave. In first-party call control, the controller can directly control the phone. The Wave bot can send call control commands through HTTP requests to the border gateway. The border gateway relays the commands to the application server. The application server then wraps the commands in an XML document and sends it using SIP MESSAGE requests to phones. Figure 4 shows the signaling flow of answering a call in Google Wave.

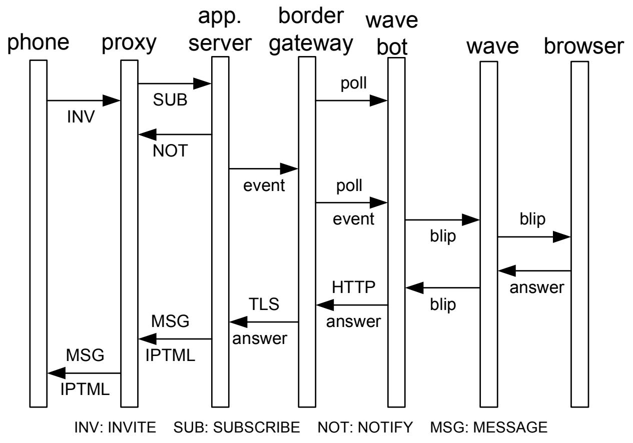

Figure 4. Answering a call in Google Wave

VI. Conclusion and Future Work

This paper introduces our approach on integrating enterprise communication functions into Google Wave. On one hand, the integration adds a new communication means in Google Wave. On the other hand, the integration also enhances enterprise communication by employing the features in Google Wave. Google Wave is still in its early stage. Some claimed functions are still not available and the whole platform is not stable. So, our current implementation is different from the ideal architecture of integrating enterprise communication functions into Google Wave. But our implementation proves the viability of the integration and shows the promising value of the integration. During our implementation, we also noticed several drawbacks of Google Wave, e.g., there should be Wave robots that are not bound to any specific waves and can initiate new waves. The feedback from us to the Google Wave community can help improving the product. In our future work, as Google Wave becomes more stable and can allow users to run a Wave server outside Google App Engine, we may experience the ideal architecture proposed in this paper and introduce more innovative features into Google Wave and enterprise communication networks.

References

[1] Google Wave: http://wave.google.com/

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002

[3] Twiliobot: click-to-call bot for Google Wave: http://code.google.com/p/twiliobot/

[4] A. Roach. Session initiation protocol (SIP)-specific event notification. RFC 3265, Internet Engineering Task Force, June 2002