Implementation and Evaluation of a Mobile Tetherless VoIP/PSTN Gateway

Jui-Hao Chiang Tzi-cker Chiueh
Computer Science Department
Stony Brook University
Email: {j-chiang, chiueh}@cs.sunysb.edu

***Abstract*—A voice-over-IP (VoIP) gateway bridges IP-based packet-switched networks (i.e. Internet) with public circuit-switched telephone networks (i.e. PSTN). The key building block of a VoIP gateway is a telephony card that interfaces with the PSTN and converts signals from the PSTN to bits that can be manipulated by a computer and vice versa. Because commercially available telephony cards only work with wired PSTN lines, almost all existing VoIP gateways are tethered and therefore do not support the kind of mobility enabled by modern wireless communications technology. This paper describes the implementation and evaluation of a mobile VoIP gateway called *WGate* that is designed specifically to bridge wireless VoIP clients and cellular phones, and can thus be easily deployed on demand in particular geographical locations. The key innovation of *WGate* is the ability to use a Bluetooth link as a wireless backplane by exploiting the Hands-Free profile of the Bluetooth protocol stack and eventually turning a set of commodity bluetooth-capable cell phones into a multi-port telephony card. Empirical measurements on a working prototype show that this approach can scale a VoIP gateway up to 8 cell phones because state-of-the-art Bluetooth adapters can only support up to 8 simultaneous Synchronous Connection-Oriented (SCO) connections when they operate in physically close proximity.**

I. Introduction

Voice over IP (VoIP) technology enables a packet-switched IP network to support voice connections with a similar quality of service (QoS) as in public switched telephone networks (PSTN). Rather than relying on specific QoS mechanisms on the network, most existing VoIP applications use sender-side transmission rate adaptation and receiver-side packet loss recovery, and exploit the increasing availability of broadband connectivity to deliver an acceptable level of QoS. VoIP technology is not only available on wired IP networks, but is also emerging as a killer application for wireless LANs (WLAN), especially with the advent of new WLAN standards such as IEEE 802.11n.

As WLAN-based VoIP technology matures, increasingly cell phones [1]–[3] are equipped with a WLAN interface, which is initially for Internet access but is designed to leverage WLAN-based VoIP whenever possible and resort to cellular-based voice communication only in the absence of WLAN connectivity. Such hybrid WLAN/cellular phones are quite compelling because previous studies show that most voice communications are between parties that are within a small geographical area and thus coverable by wireless LANs. At one extreme of the hybrid phone's design space, e.g., unlicensed mobile access (UMA) [4], a voice connection can be handed over from VoIP to cellular technologies and vice versa *during* its lifetime. Unfortunately, such seamless vertical handoff is still uncommon because coordination between these two technologies, e.g., translation between a PSTN number and a VoIP ID, remains largely unavailable. A more practical design for the hybrid phone is to support each voice connection either through VoIP or through a combination of VoIP and cellular technologies when the target party is not reachable via VoIP. However, this design requires a gateway that interfaces WLAN-based VoIP and cellular voice networking.

Because most of the world's telecommunication networks are still based on the largely closed PSTN technology, VoIP clients can only communicate with PSTN clients through a gateway that bridges IP telephony and PSTN telephony. The key building block of these VoIP/PSTN gateways is telephony cards that are plugged into a computer and interface with PSTN lines. Unfortunately, existing VoIP gateways have two limitations. First, they are typically quite expensive because the telephony cards are mostly proprietary and thus not commoditized. Second, most if not all of existing gateways are tethered and cannot support the type of mobility enabled by WLAN-based VoIP, because commercially available telephony cards only work with wired PSTN lines. This paper describes the design and implementation of the first known mobile tetherless VoIP/cellular gateway (called *WGate*) that eliminates both of these two limitations.

The most important innovation in *WGate* is its application of commodity cell phones as telephony cards. More specifically, *WGate* is equipped with WLAN interfaces and cell phones and is able to exercise control over these cell phones. As a result, *WGate* can seamlessly bridge WLAN-based VoIP connections with cellular phone connections, and is completely tetherless and fully mobile. The ability to support tetherless operation makes *WGate* an attractive building block for first response applications. In addition, applying cell phones as telephony cards entails several cost advantages. First, because of the enormous economies of scale (over 1 billion cell phones sold each year), the price of a commodity Bluetooth-capable cell phone is below $35 USD, which is one third to one half of the per-port price of a modern telephony card. Second, the service charge rate of cell phones is already pretty competitive when compared with land-line phones, and is expected to be even more competitive as more and more land-line phones are replaced by cell phones. Consequently using cell phones

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

in VoIP/PSTN gateways make perfect economic sense from the standpoints of both initial hardware acquisition cost and recurring service charge.

The main technical challenges of using cell phones as telephony cards are how to programmatically set up, tear down, and manage phone connections on proprietary and heterogeneous cell phones, and how to integrate multiple cell phones into a gateway without requiring any modification on them. We solve both problems by exploiting the Bluetooth interface embedded in commodity cell phones. In particular, we leverage the Bluetooth Hands-Free profile support in cell phones for call control, and apply the Bluetooth channel as a wireless backplane to integrate multiple Bluetooth cell phones into a single VoIP/PSTN gateway.

The rest of this paper is organized as follows. Section 2 reviews previous works on supporting multimedia connections over Bluetooth links, and building systems using cell phones as building blocks. Section 3 describes the system architecture of *WGate* and its various system components. Section 4 presents the results of a comprehensive study of the current *WGate* prototype. Section 5 concludes this paper with a summary of research contributions and a brief outline of future work.

II. Related Work

Bluetooth's Synchronous Connection-Oriented (SCO) link is aimed at supporting symmetric and real-time voice traffic between master and slave devices. Zurbes and Stahl [5] have done an extensive simulation study on the scalability of Bluetooth's SCO connections. From their simulation results, 30 concurrent SCO connections with packet format HV3 result in an average Frame Erasure Rate of 1 %, which is acceptable for voice data transmission. However, from our empirical result, the maximum number of SCO connections in a single collision domain is 8 due to commodity hardware constraints. Also, our testbed is setup such that all the Bluetooth adapters are placed very close to one another, which is the worst case for signal interference. On the other hand, Kapoor et al. [6] have suggested to use the Asynchronous Connection-Less (ACL) connection for voice data transmission to reduce the bandwidth consumption. However, the commodity Bluetooth cell phones only support the voice data transmission over SCO instead of ACL connection. Thus, our design still relies on the SCO connection.

Because Bluetooth operates in the same ISM band as IEEE 802.11b/g, the two can interfere with each other. Golmie et al. [7], [8] have run experiments on IEEE 802.11b connections co-existing with Bluetooth connections of different traffic types. The result shows that Packet Error Rate (PER) of Bluetooth voice links is low because its Frequency Hopping technique limits the impact from IEEE 802.11b interference. However, IEEE 802.11b links suffer severely from interference coming from Bluetooth connections. When one IEEE 802.11b link and 10 Bluetooth voice links co-exist in the same geographical area, the PER of IEEE 802.11b link is almost 100%. To address this interference problem, *WGate* uses IEEE 802.11a/n, which operates at the 5GHz band, as the wireless LAN link technology.

Cellular network infrastructure can also be used to expand the coverage of popular peer-to-peer applications, e.g. Skype [9]. Lie et al. [10] have proposed a new architecture for running peer-to-peer (P2P) applications on mobile cellular networks based on Session Initiation Protocol (SIP). Specifically, with the help of SIP, the network operators can manage and charge these P2P applications and overcome the user mobility and security problems. Kim et al. [11] have proposed a handoff mechanism for P2P VoIP applications that achieves seamless hand-over between IP-based networks and cellular networks. *WGate* provides a general infrastructure for the above applications to run on, and helps to enable commodity wireless VoIP phones such as iSkoot mobile phone [12] and Netgear Skype Wi-Fi phone [13].

Cellular voice service has seen a tremendous growth in recent years. For example, in United States, the number of cellular network sites has grown from 30,045 in December 1996 to 175,725 in 2004 [14]. Recent government reports [15] indicated that the customer expenditure on cell phone services has already exceeded the expenditure on residential phone services in 2007. Therefore, the economies of scale of cellular voice service is expected to further drive down its service charge in the next few years.

On the other hand, the availability of standardized mobile cell phone reference design [16] revolutionizes the cell phone manufacturing industry by driving down the time to market for low-end cell phones and thus their prices, which in turn stimulates the growth of the low-end cell phone market, especially in developing countries. A key feature of *WGate*'s design is its use of commodity cell phones. Although in theory, *WGate* could be designed to use a proprietary backplane that interfaces directly with cell phones' radio circuitry, such a design is considered undesirable in practice because it does not take advantage of the economies of scale associated with cell phone industries.

Through an extensive survey, we have determined that more than 50 commercial VoIP/PSTN gateways [17] leverage the cellular network technology to route voice calls. The use cases for them include

- Provide VoIP connectivity in areas where traditional telephone lines are unreachable but the cellular network is present.
- Reduce international/long-distance cellular-to-cellular call by first routing a call to a VoIP gateway via a domestic GSM call, then forwarding it through Internet to another VoIP gateway, which interacts with the target cell phone via another domestic GSM call.
- Avoid the high cost associated with peering between landline and cellular networks by routing a VoIP call destined to a local customer's cell phone through the cellular network instead of the telephone network.

In addition to the ability to bridge VoIP calls and cellular calls, *WGate* itself is also mobile because of its tetherless

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

capability. This mobility is particularly important for applications that require quick deployment of instant communications infrastructure, for example, emergency response, sports event (college football games), trade show, etc. Similar to our design, GP-712 GSM VoIP Gateway [18] is a commercial product that is meant to bridge VoIP calls and GSM calls via Bluetooth connectivity, but is less scalable than *WGate* since it only supports 2 concurrent VoIP calls.

III. Design

A. System Architecture

WGate is designed to interface WLAN-based VoIP applications and cellular telephony, and thus consists of a Linux-based server, a set of wireless LAN interfaces and a set of Bluetooth-equipped cell phones. The main software component is an open-source VoIP package called Asterisk [19], which supports such VoIP protocols as Session Initiation Protocol (SIP) and is able to interface with various telephony cards such as those from Dialogic [20]. The main implementation challenge of *WGate* is to modify Asterisk so that it can perform call control over cell phones through the Hands-Free profile of the basic Bluetooth connectivity.

Assume the VoIP protocol used is SIP, when a SIP client calls a target PSTN phone, it specifies the target's phone number to its associated SIP server, which, after recognizing that the call's target is specified in terms of a PSTN number, sets up a PSTN connection to reach the target, and bridges the VoIP connection and the PSTN connection until the call is terminated. To establish a PSTN connection, *WGate* interacts with a commodity cell phone over a Bluetooth link. In the following subsections, we will briefly describe each of *WGate*'s components in more detail.

The Session Initiation Protocol (SIP) is one of the most popular VoIP protocols that set up, tear down and manage voice connections between two or more end hosts across the IP network. SIP is an application layer signaling protocol for creating and terminating a phone call session defined by IETF [21]. SIP User Agents (UAs) run on the end-user devices, and create and manage a SIP session. A SIP UA has two main components, the User Agent Client (UAC) sends messages and answers with SIP responses and the User Agent Server (UAS) responds to SIP requests sent by the peer. SIP UAs may work in point to point mode. Typical implementations of a UA are SIP softphones or SIP hardphones. A SIP proxy server is an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other SIP clients.

B. Call Control Using Bluetooth's Hands-Free Profile

1) Bluetooth Protocol Stack: Bluetooth is a wireless protocol for exchanging data over short distances between fixed and mobile devices, and for creating personal area networks (PANs). In this project, we use Bluetooth 2.0, which operates in the license-free ISM band at 2.4-2.4835 GHz and provides a sustained data transfer rate of about 2.1 Mbits/sec. Communications between Bluetooth devices take place in an ad hoc

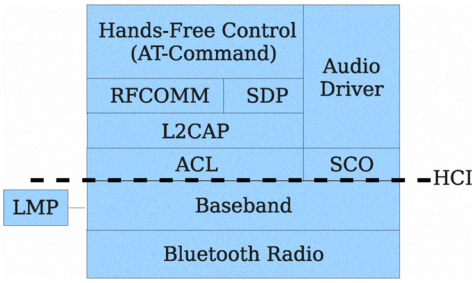

Fig. 1. *Bluetooth protocol stack.*

structure called a *piconet*, in which one device is designated as *master* and all other devices, up to 7, serve as *slaves*. Inside a piconet, all communications are synchronized with the master device, with time divided into slots of 625 usec. In addition, another structure called *scatternet* could be used to expand the physical range of a Bluetooth network by interconnecting two or more piconets. *WGate* uses the piconet instead of scatternet structure because the focus is to expand the capacity of a Bluetooth network rather than its coverage range.

To avoid interfering with other protocols that use the 2.45 GHz band, Bluetooth uses Frequency Hopping, in which the entire frequency band is divides into 79 channels (each 1 MHz wide) and time is divided into slots of size 625 μsec, and each piconet picks a unique hopping pattern. With their clocks synchronized with the master, all slaves in a piconet hop from one frequency carrier to another at the end of every time slot.

As Fig. 1 shows, Bluetooth assumes a layered protocol architecture consisting of core protocols, optional protocols and service profiles. The three mandatory protocols for the Bluetooth protocol stack are Link Management Protocol (LMP), Logical Link Control and Adaptation Protocol (L2CAP) and Service Discovery Protocol (SDP). Host/Controller Interface (HCI) and Radio frequency communications (RFCOMM) are two protocols that are almost universally supported. the full Bluetooth protocol stack. The Radio layer of the stack specifies the hardware parameters used in radio transmissions, and is followed by the the Baseband layer that specifies lower-level communication operations such as radio channel selection, data error correction and encryption. Two types of links are supported in the Baseband layer: asynchronous connectionless (ACL) and synchronous connection-oriented (SCO). The LMP layer manages the medium access control operations, and specifies how connections are established and released for ACL and SCO links.

The ACL link provides a packet-oriented data transmission interface to the upper L2CAP layer, which in turn provides a data transport interface to higher-layer protocols by taking care of issues such as data segmentation, reassembly, and multiplexing. On top of L2CAP, the RFCOMM layer emulates a serial port interface and supports reliable data transmission between two end devices. Service Discovery Protocol (SDP) is a control protocol that applications use to determine the profile and connection parameters supported by a Bluetooth device.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

2) Synchronous Connection-Oriented Link: Bluetooth supports SCO connections, which are designed to support real-time and symmetric constant-data-rate connections between two Bluetooth devices. More specifically, each SCO connection carries a 64-kbps voice connection using Continuous Variable Slope Delta Modulation (CVSD), which is designed to protect data transmitted over error-prone wireless media. The input and output of Bluetooth's CVSD module is 16-bit PCM voice with a sampling rate of 8 KHz.

A normal SCO packet contains the following fields:

- 72 bits of access code that represents the identity of the piconet master,
- 54 bits of packet header that contains such link control information as 3-bit slave address and forward error correction (FEC) code for packet header recovery, and
- 240 bits of voice data payload.

To meet the real-time communication requirement, the baseband controller schedules the transmission of a SCO packet controller every 2, 4, or 6 time slots for the HV1, HV2, and HV3 format, respectively. The HV1 packet has 10 information bytes and is protected by a rate 1/3 FEC while HV2 has 20 information bytes and is protected by a rate 2/3 FEC. As for HV3 packet, it has no FEC protection and contains 30 information bytes. In other words, a SCO connection using HV1 packet supports the same 64kbps data rate as other packet formates, but uses more FEC code and time slot to protect the data completeness. Packet transmissions from the master to a slave in a piconet can only start at the beginning of even slot numbers whereas packet transmissions from a slave to the master happen at odd slot numbers and right after the master transmits data to the slave. Thus, SCO packet transmission is effectively based on a polling mechanism from the master to slaves.

When a user application requests to initiate a SCO connection to a remote Bluetooth device, the SCO layer sends an HCI command to inform the baseband controller to start the connection establishment procedure. After the controller successfully sets up the requested SCO connection with the remote device, it sends up an HCI event packet to provide the SCO layer with the corresponding connection handle. Then the user application can start sending raw voice data of 16-bit 8KHz PCM format to the SCO layer, which encapsulates them into HCI SCO data packets and send them to baseband controller, which in turn schedules them for physical transmission. Inside the SCO layer, the voice data from a user application is partitioned into fixed-sized units whose size is equal to Bluetooth's Maximum Transmission Unit (MTU) size, which is usually set to 48 bytes for USB-based Bluetooth adapters.

A Bluetooth master device may support up to 3 SCO connections to the same slave or different slaves. However, commercial Bluetooth devices do not support more than one SCO connection. Thus, to support N voice connections, *WGate* sets up N master-slave pairs, which forms N independent Bluetooth piconets.

3) Hands-Free Profile: A Bluetooth profile describes an interface specification for a high-level communications service that is built on top of Bluebooth's core and optional protocols. At a minimum, a Bluetooth profile specification contains information on (a) dependencies on other profiles, (b) suggested user interface formats, and (c) specific parts of the Bluetooth protocol stack required.

In this project, we leverage the Hands-Free profile to enable a VoIP gateway to perform call control on a Bluetooth-equipped cell phone. The Hands-Free profile is originally designed to allow an end user to perform standard telephony functions through a separate Bluetooth device that serves as the voice input and output of a Bluetooth-equipped cellular phone without interacting with the cellular phone. The *control* plane of the Hands-Free profile is based on the Hands-Free Control layer, which utilizes the conventional AT command set for call monitor and control. The *data* plane of the Hands-Free profile involves audio input and output and is based on the Audio Driver layer of the Bluetooth stack, which acts as an interface to the SCO layer and adapts the audio encoding based on the application requirement.

Let's take an in-car Hands-Free unit as an example. Two types of devices are involved when a Hands-Free unit services a phone call: the Audio Gateway (AG), usually a cellular phone, communicates with the remote end via the cellular network, whereas the Hands-Free unit (HF) acts as the input and output device interacting with the user on behalf of AG, and is able to set up, tear down, and manage calls through the associated AG. Most commercial Bluetooth-equipped cell phones support the Hands-Free profile, because many states prohibit by law users from operating cell phones in moving vehicles without using the Hands-Free mode. Although the Hands-Free profile is originally meant to support hands-free operation of cell phones, it also provides the necessary primitives for a VoIP gateway to use a cell phone as a telephony card. The following describes how a VoIP gateway use these primitives to start, to receive, and to hang up a cellular phone call. In this case, the VoIP gateway is an HF and the cell phone is an AG.

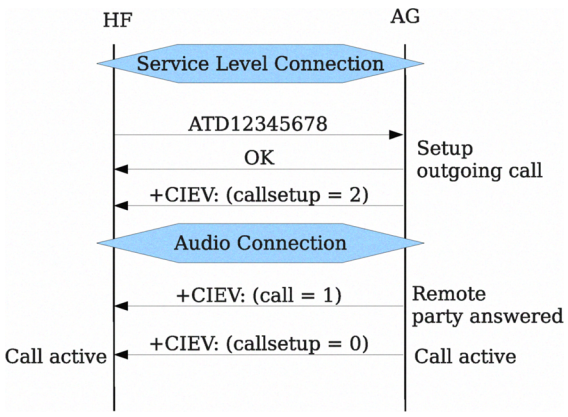

Fig. 2. *Making a phone call to a remote party through a cell phone from a VoIP gateway.*

As shown in Figure 2, the VoIP gateway first sets up a Service Level Connection with the cell phone, which is built

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

on top of the interface provided by the RFCOMM layer. To initiate a phone call, the VoIP gateway sends to the cell phone the AT command *ATD* with the target phone number as the operand (in this case 12345678). When the cell phone receives this command, it makes an outgoing call to the designated target and replies with an *OK* response followed by the result code *+CIEV (callsetup = 2)* to notify the VoIP gateway that the call has been successfully initialized. After that, an audio connection based on the Audio Driver layer is established between the two sides. As the called party answers the phone call, the cell phone sends two *+CIEV* commands to indicate to the VoIP gateway the phone connection is up and the user can start to talk with the remote party. From now on, the cell phone will route the voice input and output on the audio connection between the called party and the VoIP gateway.

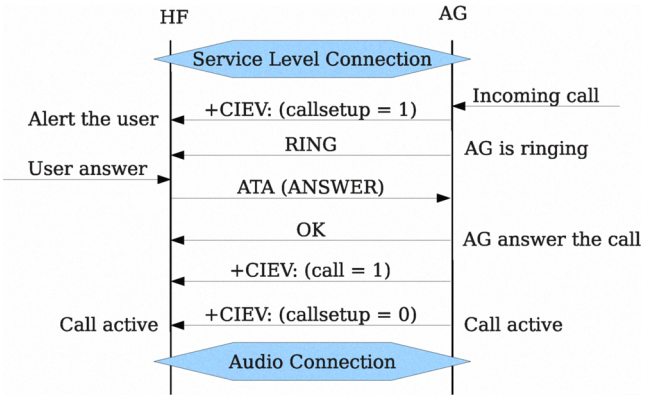

Fig. 3. *Receiving a phone call from a remote party to a VoIP gateway through a cell phone.*

As shown in Fig. 3, after the Service Level Connection is established between a VoIP gateway and a cell phone, the cell phone notifies the VoIP gateway with *+CIEV:(callsetup = 1)* and a *RING* message when there is an incoming call from a remote party. The VoIP gateway in this case sets up a VoIP connection to reach the target end user, and after the user answers, sends an *ATA* response message to the cell phone, which in turn returns a response to the remote calling party. After that, the cell phone sends the VoIP gateway two more *+CIEV* messages and creates an audio connection between them to transport subsequent voice inputs and outputs. When a user hangs up the call, a *AT+CHUP* message is used between the cell phone and VoIP gateway to terminate the voice connection.

IV. Tetherless VoIP Gateway

We chose Asterisk [19] as the base VoIP gateway implementation, because it is an open-source Private Branch Exchange (PBX) software that can be configured as a media gateway to bridge traditional PSTN telephony with IP telephony. In terms of IP telephony support, Asterisk supports both SIP and H.323 [22]. In addition, Asterisk can be configured as a SIP proxy server that propagates a SIP request to a SIP client or a PSTN end point. Asterisk mediates between these two telephony network through a *Dialplan*, which translates user identities from different networks. For example, a Dialplan can specify that a SIP ID *john@cs.sunysb.edu* should be translated into a PSTN or cellular telephone number *12345678*. This means when Asterisk receives a SIP connection request targeted at *john@cs.sunysb.edu*, it knows that it should accept the SIP connection request and initiate a PSTN call through one of its telephony cards so as to reach the actual target user.

As we were developing an interface software that allows Asterisk to interact with a cell phone through the Hands-Free profile, the Asterisk project independently develops an add-on module called *chan_mobile*, which supports a similar functionality of connecting Asterisk with the cellular network via Bluetooth-equipped cellular phones. Thus, we apply the module directly and mainly put focus on solving hardware compatibility problems, which are addressed later.

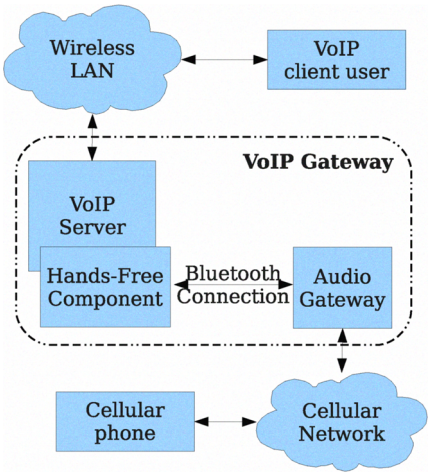

Fig. 4. *System architecture of the proposed tetherless VoIP gateway*

To be completely tetherless, *WGate* interfaces VoIP connections over WLAN interfaces with cellular telephony connections on cell phones. Instead of using standard telephony cards, *WGate* uses Bluetooth-equipped cell phones as telephony cards by treating them as an Audio Gateway (AG) in the Hands-Free profile. In the following, we describe how to build and combine each system component of *WGate* as Fig. 4.

- **VoIP Server:** We install Asterisk on a Linux-based industrial PC, set up its Dialplan, and configure the PC's wireless LAN interfaces and Bluetooth interfaces so that Asterisk can work with them smoothly.
- **Hands-Free Component:** The Bluetooth hardware adapter supports the Baseband layer of the Bluetooth protocol stack, whereas Linux's BlueZ supports the higher-layer protocols such as RFCOMM, L2CAP and SDP. The *chan_mobile* add-on module of Asterisk provides a basic implementation of the Hands-Free profile. Together, these Bluetooth components form a Hands-Free (HF) device that is able to communicate with an Audio Gateway, and can be used to perform basic call control over a cell phone.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

- **Audio Gateway:** An AG is a software module running on a Bluetooth cell phone that is responsible for mediating phone calls between the cellular network and a Hands-Free (HF) device, and mainly supports the following functionalities:
  - Set-up and release of service level connections over the RFCOMM interface for exchanging AT commands and responses,
  - Set-up, release and transfer of audio connections over the SCO interface for voice packet transport, where "transfer" means routing an incoming cellular call to an HF device and vice versa,
  - Answering, rejecting, terminating, holding an incoming cellular call,
  - Placing a cellular call using a PSTN phone number coming from an HF device,
  - Transmission of DTMF signals during an ongoing call, and
  - Activate/deactivate some other features or notification functions on AG, e.g. voice recognition.

Because modern Bluetooth cell phones support most of the above AG functionalities, using these cell phones to replace traditional wired telephony cards is not only cheaper, but also much more functional.

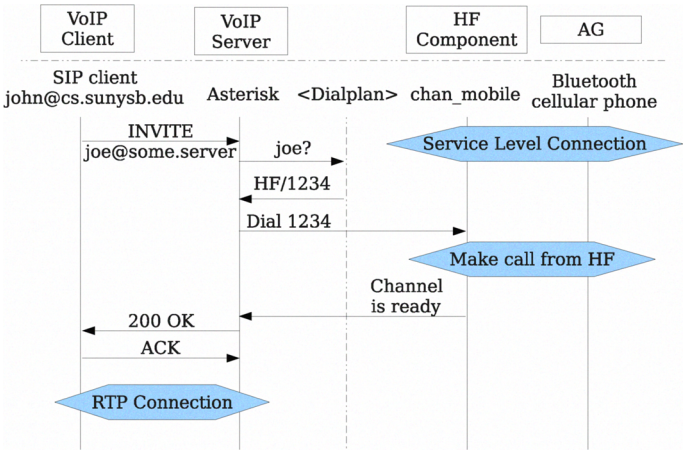

Fig. 5. *Making a SIP phone call through the gateway.*

As shown in Figure 5, when a SIP client user makes a phone call to *joe@some.server*, the request comes to *WGate*, whose Asterisk component checks its Dialplan and finds out the target user matches a cellular phone number *1234*. Then the Asterisk component asks its HF device, *chan_mobile*, to dial the identified phone number through one of the AGs, which initiates a cellular call using the procedure in Figure 2. After a service level connection between the HF device and the chosen AG is established, the HF device notifies the Asterisk component accordingly, which in turn responds to the original requesting SIP client user and starts an RTP connection for voice transport. Asterisk and chan_mobile use an internal buffer for voice traffic exchange because they reside in the same address space.

More specifically, the RTP streaming between the SIP client and Asterisk is segmented by time unit, e.g. 0.02 second, and the packet size depends on the encoding data rate of the SIP client. For the voice data between Asterisk and Bluetooth SCO layer, chan_mobile segments them to MTU size as the input to Bluetooth SCO layer, and assembles them to RTP packet as the input to the SIP client.

One of the most time-consuming aspects of this project is to resolve the hardware compatibility issue. Although many Bluetooth cell phones support the Hands-Free profile, their support is often incomplete and non-conforming because they are designed to work only with proprietary in-car Hands-Free devices. For example, we found their AT command messages for call control usually do not follow the standard format as defined in the Hands-Free profile. Eventually we chose to use Motorola Razr V3 in this project because we found it is often used in vendors' testing of Bluetooth adapters and service profile softwares.

As for Bluetooth adapters, we have tried adapters using chips from several vendors such as Broadcom, CSR, and ISSC, and most of them support the USB interface. The Broadcom adapter with chip identity BCM92035DG fails to establish an SCO connection with the V3 cellular phone while BCM92045B3_ROM works improperly when sending the AT commands. Finally, we decided to use the Bluetooth adapters from ISSC with chip identity ISSCEDRBTA, which works fine with the V3 cellular phone for most of the AT commands.

A Bluetooth 2.0 link is rated at 2.1 Mbits/sec, which is equivalent to 34 64-Kbps SCO connections. To scale up *WGate* to support these many SCO connections, we need to be able to integrate into it as many Bluetooth cell phones as possible. For each cell phone, we need to install a USB-based Bluetooth adapter. However, most industrial PCs do not support more than 8 USB ports. This puts a hardware limit on the number of phone calls that *WGate* can practically support. To circumvent this, we have experimented with multi-port USB hubs, which extends a single server-side USB port to multiple USB ports into which USB-based adapters can be plugged. Unfortunately, none of the USB hubs we have tested support multiple concurrent SCO connections. Luckily, we have found a PCI-based 4-port USB card from VIA that can support 4 concurrent SCO connections simultaneously. With three of such cards, the current *WGate* prototype can support up to 12 SCO connections.

V. Performance Evaluation

A. Experiment Methodology

The main technical challenge in the development of *WGate* is its scalability, specifically, the number of PSTN connections it can support concurrently. Because *WGate* uses Bluetooth as a wireless backplane to connect cell phones to the VoIP gateway server, its scalability is mainly limited by the number of SCO connections that a Bluetooth link can accommodate at a time. In theory, a Bluetooth 2.0 link provides a sustained throughput of 2.1 Mbits/sec, which is 33 to 34 times of a single SCO connection's bandwidth requirement, 64 Kbits/sec.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

However, the actual number of concurrent SCO connections that *WGate* can support is expected to be much lower because the backplane Bluetooth link suffers from heavy interferences. First, because the Bluetooth adapters and the cell phones are packaged in close proximity, they interfere with one another significantly. Second, because Wi-Fi links and Bluetooth links both operate in the ISM band, there exists some hidden interference in indoor environment, especially from IEEE 802.11b devices.

To empirically determine the scalability of the *WGate* prototype, we built a testbed to measure the scalability of its Bluetooth-based backplane. The testbed consists of three computers, one acting as VoIP clients, one as the VoIP gateway, and the third as an emulator for a set of Bluetooth cell phones. All the computers used are a Dell desktop machine with a Pentium-4 3-GHz CPU, 1 Gbytes of RAM and an 100Mbps Ethernet interface. All of them run the Linux 2.6.22 kernel with BlueZ core version 2.11. The WLAN interfaces are Wistron NeWeb 802.11 a/b/g mini-PCI card (Model No. CM9), which are installed on a computer via a 4-port miniPCI-to-PCI adaptor. This adapter uses the AR5004X chipset, which is composed of an AR5213 MAC controller chip and an AR5112 dual-band radio. In the experiment, it is configured to use one of the 802.11a channel. The Cyber-Bluetooth Slim USB adapter with ISSC chipset is used together with the DYNEX USB 2.0 4-port PCI Host Adapter with VIA chipset, which is used to connect multiple Bluetooth adapters to a computer.

To emulate a SIP client, we use the PJSIP tool [23], which is an open-source program providing SIP and media stacks for presence, messaging, and multimedia communication. To set up the VoIP gateway, we install Asterisk 1.6.0-beta4 with the add-on package containing the *chan_mobile* component. Because we don't have many Motorola Razr V3 cellular phones, we chose to implement a simple emulator that emulates the AG behavior in the Hands-Free profile and supports the following features:

- Set up and release a Service Level Connection.
- Transfer basic phone status by sending *+CIEV* messages to an HF device.
- Process a phone call initiation request from an HF device, which corresponds to the *ATD* message.
- Process a phone call hangup request from an HF device, which corresponds to the *AT+CHUP* message.

Together with Linux's BlueZ stack and a Bluetooth adapter, the above AG emulator can effectively act as a real Bluetooth cellular phone. Although this emulator does not support all the features specified in the Hands-Free Profile, it is sufficiently functional to be used in our testbed for the study of SCO connection scalability.

B. Performance Metrics

To determine the maximal number of SCO connections that *WGate*'s Bluetooth backplane can support, we need to first decide the criteria used to assess the quality degradation of a SCO link. One possibility is to use objective network performance metrics such as packet loss ratio, packet delay, or jitter. Unfortunately, these metrics are surprisingly difficult to measure because the lossy compression algorithm implemented inside Bluetooth adapters makes it difficult to correlate between sent packets and received packets. Also, because *WGate* is designed to support voice communication, it is not clear how to map these objective metrics to a subjective voice quality metric even if we can successfully compute them. To overcome this problem, we decided to adopt the voice quality metric produced by the Perceptual Evaluation of Speech Quality (PESQ) method [24], which is recommended by ITU, to measure the degree of performance degradation of SCO connections.

The PESQ method is designed to compare the quality of an original input voice signal with that of its corresponding degraded output of a communication system. The result of this method is a qualitative score that has the same interpretation as the subjective Mean Opinion Score (MOS), which is typically produced by a panel of human testers [25]. Aiming to capture the end-to-end voice quality, the PESQ method takes into account the following factors that contribute to the eventual voice quality perceived by the receiving user:

- Transmission channel errors,
- Environmental noise at the sending side,
- Impact of varying delay or jitter on listening only tests,
- Short-term and Long-term time warping of audio signal

To prevent error accumulation, the PESQ standard cuts each voice input into chunks, each 8 to 20 seconds in length, and analyzes each of them independently of one another. Since the PESQ method focuses on measuring one-way voice quality, it may result in different user experience of our system for two-way communication applications.

In addition to voice quality, we are also interested in the end-to-end connection set-up time, because this is another important metric that affects the user experience when making a phone call. Finally, we also measure the CPU and memory usage of the VoIP gateway itself to quantify their impact on the entire system's scalability. In this paper, we don't vary the WLAN conditions in the experiment since lots of research works have been done for the voice quality analysis in WLAN environment, e.g. [26], [27].

C. Scalability of Bluetooth Backplane

To measure the voice quality of a Bluetooth SCO connection, we set up a pair of Bluetooth adapters, each residing on a separate machine, and establish a SCO connection between these adapters. Then we feed a pre-recorded human voice stream into one adapter, record the voice stream output by the other adapter, and apply the PESQ method to compare the original voice input with the degraded voice output to assess the degree of distortion introduced by the SCO connection.

Originally we use two computers each equipped with 12 Bluetooth adapters such that we can run 12 concurrent SCO connections between them. However, in this set-up the distance between some Bluetooth adapters is approximately one quarter of an inch, which leads to strong interference

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

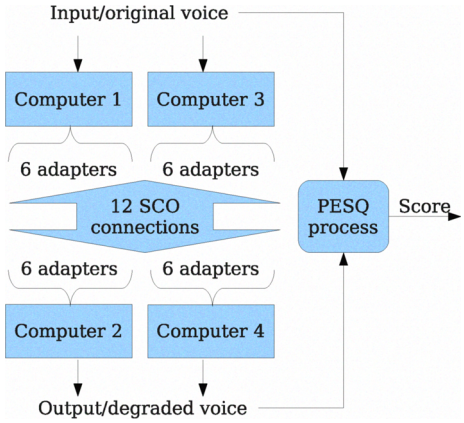

Fig. 6. *Testbed for measuring the voice quality of concurrent SCO connections. Each of these four computers is equipped with 6 USB-based Bluetooth adapters.*

among these Bluetooth adapters. To overcome this problem, we use a different set-up as shown in Figure 6, where the 24 Bluetooth adapters are distributed to 4 computers, and the closest distance between any two adapters is at least 1 inch. Then, we start the SCO connection between each of the 12 pairs of Bluetooth adapters, e.g. first adapter of computer 1 connects with first adapter of computer 2, etc. Finally, we feed into one end of each SCO connection a 207-second English female voice stream from the online Open Speech Repository [28], which provides several human voice data of different languages, and record the voice output on the other side.

When we set up 9 or 10 HV3-based SCO connections, we found that one or more SCO connections may get disconnected after a while. We take this to mean a Bluetooth link cannot support more than 8 HV3-based SCO connections because the degradation of some of the SCO connections is so bad that the associated Bluetooth adapters decide to terminate the connections altogether. Also, we found that if we tried to start all SCO connections at the same time, some of the SCO connections may not be established successfully. This result is very different from previously reported results [5], which were based only on simulations. We conjecture the main explanation for this discrepancy is that the interference among the Bluetooth piconets is much more serious than were assumed in these simulation models because they are so close physically in the *WGate* prototype.

From the above experience, we decide to reduce the total number of SCO connections to 8. In addition, we start each SCO connection one after another, and put a time gap, 2 seconds, between consecutive SCO connection set-up steps. Then we perform the following steps to measure the PESQ score for each SCO connection:

- Remove from consideration the first and last 15 seconds of both input and output voice streams because the last SCO connection is started 14 seconds later than the first one.
- Decompose each voice stream into 12-second chunks.
- Calculate the PESQ score for each pair of voice input/output chunks, e.g. chunk 1 of the input voice and chunk 1 of output voice, etc.

We repeat the same experiment 7 times, gather 100 PESQ scores for each SCO connection, and calculate their means. The SCO packet format is first set to HV3.

Figure 7 shows the arithmetic average of the mean PESQ scores of all SCO connections decreases with the number of concurrent SCO connections in a Bluetooth link when the HV1, HV2 and HV3 formats are used. As more SCO connections are present in a Bluetooth link, the amount of interference among them increases, and their avearge mean PESQ score decreases. When the HV1 format is used, the average mean PESQ score is 3.9, which represents good voice quality, if there is only one SCO connection, and still remains above 3 even when the number of SCO connections is increased to 8.

When there is only one SCO connection, there is no noticeable difference among the three packet formats. However, when the number of SCO connections increases, the quality of HV1 and HV2 start to degrade substantially more than HV3. HV1 takes more channel time to send a fixed number of bits than HV2 because HV1 uses stronger FEC code and spreads the channel time over more time slots than HV2. HV2 in turn takes more channel time to send a fixed number of bits than HV3 because HV2 uses stronger FEC code and spreads the channel time over more time slots than HV3. When more total channel time is required and more time slots are needed to send a fixed number of bits, the probability of inter-SCO connection collision increases, and the resulting voice quality degrades more. Empirically, the effect of higher collision probability overshadows the quality gain from stronger FEC code.

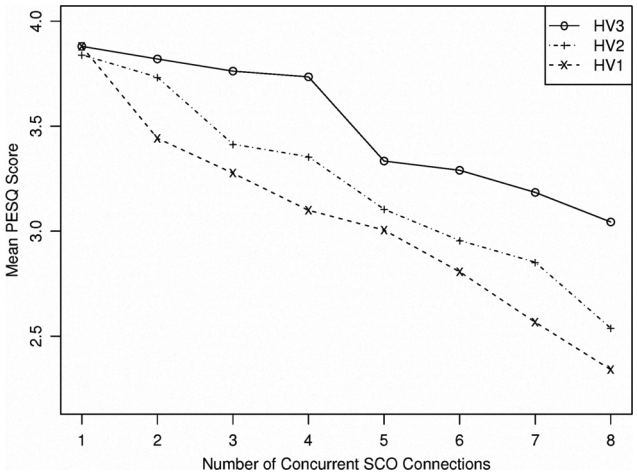

Fig. 7. *The average mean PESQ score of all SCO connections when the number of concurrent SCO connections sharing one Bluetooth link is increased from 1 to 8. These three lines corresponds to three different SCO packet formats HV1, HV2, and HV3.*

Figure 8 shows that when the number of concurrent SCO connections is 8, the mean values of most SCO connections

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

are greater than 2.5, which means the voice quality is a little annoying but still acceptable. The quality of the third SCO connection is noticeably worse others, even though the same hardware and software are used in all these SCO connections. We conjecture this anomaly is due to the defects in the adapter hardware. Figure 8 also shows the 95% confidence interval of the PESQ scores of each SCO connection, which corresponds to the interval in which 95% of a SCO connection's PESQ scores fall. In all cases, the 95% confidence interval of a SCO connection's PESQ scores is concentrated around its mean value.

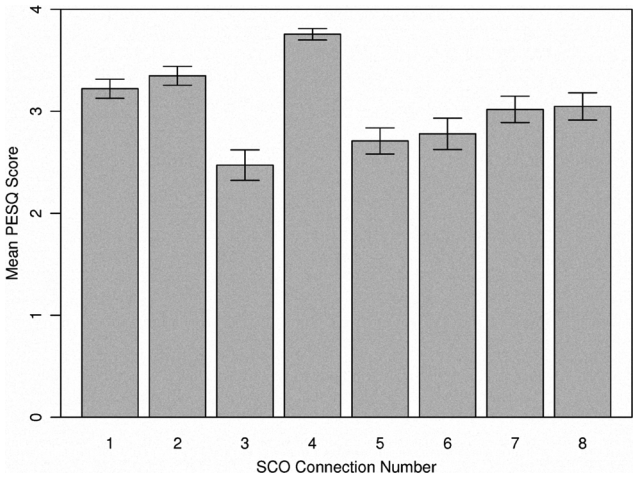

Fig. 8. *Mean value and 95% confidence interval of the 100 PESQ scores associated with each SCO connection when 8 concurrent SCO connections are sharing one Bluetooth link.*

Figure 9 shows the temporal evolution of each SCO connection's PESQ score in one run when 8 concurrent SCO connections are sharing one Bluetooth link. Although at some point in time, the PESQ scores of some SCO connections drop below 2, but they climb back up later so that the mean value is above 2.5. Consequently, the overall voice quality of each SCO connection in this run remains acceptable for human consumption.

D. End-to-End Performance Testing

To test the end-to-end performance of the *WGate* prototype, we set up another testbed shown in Figure 10. On Computer 1, we run PJSIP to initiate 8 concurrent SIP phone calls. An IEEE 802.11a WLAN interface is installed on both Computer 1 and Computer 2 so that they can communicate with each other over a WLAN link. Asterisk and its *chan_mobile* module on are installed on Computer 2. We set up one cell phone, Cell 1, and Computer 3 with 7 Bluetooth adapters each of which has established a service level connection with one of the eight adapters installed on Computer 2. Another cell phone, Cell 2, serves as the remote party for Cell 1, so that these two cell phones can serve as targets of VoIP calls accessible through T-mobile's cellular network. For timing measurements, all computers in the testbed run NTP to synchronize their clocks.

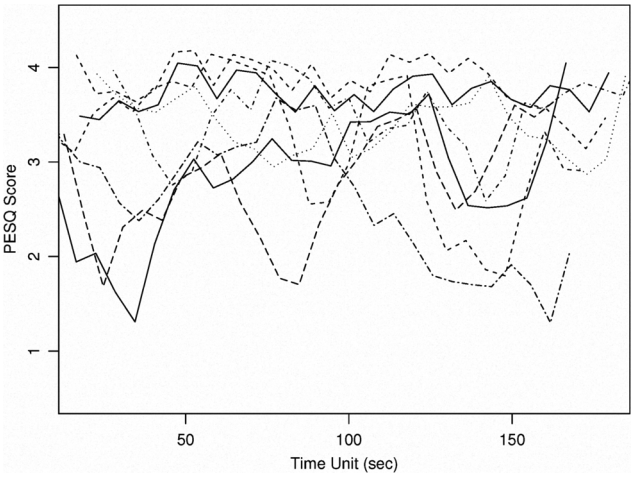

Fig. 9. *The temporal evolution of each SCO connection's PESQ score when 8 concurrent SCO connections are sharing one Bluetooth link.*

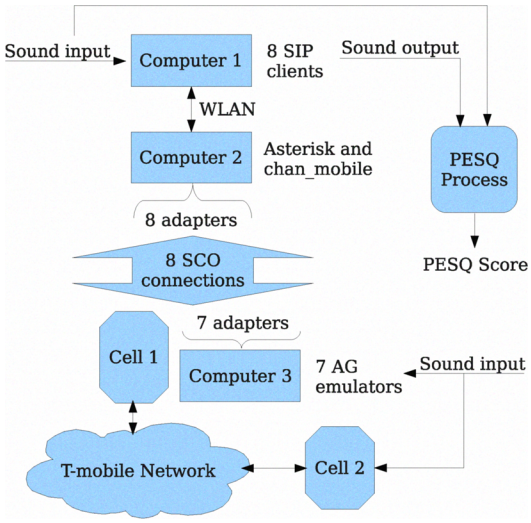

Fig. 10. *The end-to-end performance testing testbed consists of three computers, one running SIP clients, another acting as the VoIP gateway and the third containing 1 real and 7 emulated Audio Gateways. Another real cell phone serves as the remote party through T-mobile cellular network.*

A phone call starts with PJSIP sending out a SIP INVITE message over the WLAN to the VoIP gateway, which, after consulting with Asterisk's Dialplan, maps the target user of the INVITE message to a cellular phone number, picks a local HF device, and asks the HF device's associated AG to initiate a cellular call to the target user. At Cell 2, we pick up the phone call as soon as the first hint of the ringing tone shows up. We measured the *connection establishment time*, which is the interval between when the PJSIP program sends the initial INVITE message and when the VoIP gateway returns a final *OK* to the PJSIP after the associated cellular call is established. The minimum and maximum of the connection establishment times are 245 msec and 3487 msec respectively.

To measure the end-to-end quality of a VoIP/PSTN call going through *WGate*, after such a call is set up, we feed

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026

the PJSIP, 7 AG emulators, and Cell 2 with a 16-bit 8-KHz linear PCM human voice data stream from the Open Sound Repository, record the voice output from the PJSIP program, and calculate the output voice stream's PESQ score with respect to the original voice input. Note that this voice input experiences several voice codec transformations, such as conversion to 8-bit 8-KHz PCM μlaw format between PJSIP and Asterisk and 64kbps CVSD encoding before being sent on SCO connections. Each call lasts for 15 minutes and we repeat each call 5 times.

The mean PESQ score of a VoIP/PSTN call set up this way is 3.52, when there is only one SCO connection running on *WGate*'s Bluetooth backplane, and drops to 2.62 when there are totally 8 SCO connections running on the Bluetooth backplane. The difference between the two scores results from the interference among concurrent SCO connections when the 7 AG emulators on Computer 3 are sending voice streams to Computer 1. We also measured the CPU and memory usage of the VoIP gateway machine when it bridges one VoIP/cellular call and 7 emulated calls, and both CPU and memory usage turn out to be below 1%, which suggests that the scalability of the current *WGate* prototype mainly is limited by its Bluetooth backplane rather than its CPU or memory.

VI. Conclusion

Although advances in wireless communications technologies such as wireless LAN and cellular telephony have enabled unprecedented mobility for end users, VoIP/PSTN gateways, which bridge between VoIP calls with PSTN calls, remain largely immobile, because their key building blocks, computer telephony cards, only work with wired PSTN lines. This paper describes the design, implementation and evaluation of the first known tetherless and thus mobile VoIP/PSTN gateway called *WGate*. *WGate* is equipped with wireless LAN interfaces to support VoIP connections and with cell phones to support PSTN connections. A key innovation of *WGate* is its novel use of the Hands-Free profile in the Bluetooth protocol stack to connect multiple Bluetooth-equipped cell phones into a VoIP gateway without requiring any modifications to these phones. The ability to integrate commodity cell phones into a server computer not only renders a mobile VoIP/PSTN gateway possible, but also greatly reduces the acquisition and service cost of the telephony hardware, thanks to the enormous economies of scale of cellular phones and services. More specifically, this project makes the following contributions to the research area of mobile computing/networking systems:

- A novel application of Bluetooth's Hands-Free Profile to convert a Bluetooth link as a backplane of integrating cell phones into a computer telephony server.
- The first working prototype of a tetherless VoIP/PSTN gateway that can be quickly deployed to set up an instant communication infrastructure for situations such as disaster response or crisis management.
- The first empirical study on the research question of how many SCO connections a Bluetooth link can support simultaneously. The result of this study is very different from previous studies, which are all based on simulations.

References

[1] Apple Inc., "iphone, an internet-connected multimedia smartphone," http://www.apple.com/iphone/.

[2] T-mobile Inc., "G1, smartphone with google android software platform," http://www.htc.com/www/product/g1/overview.html.

[3] AT&T Inc., "Tilt, windows mobile pocket pc phone," http://www.wireless.att.com.

[4] UMAToday Inc., "Unlicensed mobile access (uma) technology," http://www.umatechnology.org/overview/.

[5] S. Zubres, W. Stahl, K. Matheus, and J. Haartsen, "Radio network performance of bluetooth," *IEEE International Conference on Communications*, Jun. 2000.

[6] R. Kapoor, L.-J. Chen, Y.-Z. Lee, and M. Gerla, "Bluetooth: carrying voice over acl links," *4th International Workshop on Mobile and Wireless Communications Network*, 2002.

[7] N. Golmie, R. V. Dyck, and A. Soltanian, "Interference of bluetooth and ieee 802.11: Simulation modeling and performance evaluation," *4th ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, July 2001.

[8] N. Golmie, R. E. V. Dyck, A. Soltanian, A. Tonnerre, and O. Rebala, "Interference evaluation of bluetooth and ieee 802.11b systems," *ACM Wireless Networks*, May 2003.

[9] Skype Inc., "Skype, voip software," http://www.skype.com/.

[10] S. Liu, J. Chen, S. Zhao, and F. Na, "Peer-to-peer application in mobile cellular systems," *Proceedings of the Fifth International Conference on Information Technology: New Generations*, 2008.

[11] S. C. Kim, M. G. Kim, and B. H. Rhee, "Seamless connection for mobile p2p and conventional wireless network," *The 9th International Conference on Advanced Communication Technology*, Feb. 2007.

[12] iSkoot Inc., "iskoot, free mobile software for skype," http://www.iskoot.com/.

[13] Netgear Inc., "Netgear skype wi-fi phone," http://www.netgear.com/Products/CommunicationsVoIP.aspx.

[14] C. C. Carbone, "Cutting the cord: telecommunications employment shifts toward wireless," *Monthly Labor Review, Bureau of Labor Statistics, United States Department of Labor*, 2006.

[15] United States Department of Labor, "Consumer expenditure survey," http://www.bls.gov/cex/cellphones2007.htm.

[16] Mediatek Inc., "Mobile reference design," http://www.mediatek.com.

[17] Voip-info Org., "Voip gsm gateways," http://www.voip-info.org/wiki/view/VOIP+GSM+Gateways.

[18] Gempro Technology Inc., "Gp-712 gsm/3g/cdma sip voip gateway," http://www.gempro.com.tw/.

[19] Asterisk Org., "Open source private branch exchange (pbx), telephony engine, and telephony application toolkit," http://www.asterisk.org/.

[20] Dialogic Inc., "Multimedia and signaling technologies and platforms provider," http://www.dialogic.com/.

[21] J. H. Schulzrinne, "Internet telephony: Architecture and protocols - an ietf perspective," *Computer Networks and ISDN Systems*, Feb. 1999.

[22] *Recommendation H.323: Packet-based multimedia communications systems*, International Telecommunication Union, Jun. 2006.

[23] PJSIP Org., "Open source sip stack and media stack for presence, im/instant messaging, and multimedia communication," http://www.pjsip.org/.

[24] *Perceptual Evaluation of Speech Quality (PESQ), an objective method of end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. ITU-T Recommendation P.862*, International Telecommunication Union, 2001.

[25] *Methods for subjective determiniation of transmission quality. ITU-T Recommendation P.800*, International Telecommunication Union, 1996.

[26] M. Narbutt and M. Davis, "An assessment of the audio codec performance in voice over wlan (vowlan) systems," *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems*, 2005.

[27] F. Anjum, M. Elaoud, D. Famolari, A. Ghosh, and R. Vaidyanathan, "Voice performance in wlan networks - an experimental study," *Global Telecommunications Conference*, 2003.

[28] VoIP Troubleshooter Inc., "Open sound repository," http://www.voiptroubleshooter.com/open_speech/index.html.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.7026
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.7026