A Novel System for Remote Control of Household Devices Using Digital IP Phones

Eric William Burger, *Senior Member, IEEE* and Ophir Frieder, *Fellow, IEEE*

Abstract — *The idea of using a phone as a remote control for household devices is not new. However, new digital technologies such as Voice-over-IP and signaling protocols such as SIP enable new methods of integrating the user interface of a phone with the digital IP infrastructure being deployed in the home. We developed and deployed the Key Press Markup Language and SIP Event Package (KPML) to address the need for a signaling-layer protocol for transmitting user stimulus from low-power, consumer devices such as IP phones to control household consumer devices.*

Index Terms — Home automation, remote control, tele-control, telephony.

I. Introduction

Many prior efforts describe the use of a phone as a remote control for household appliances. In such a system, the user presses buttons on the telephone keypad to send commands to a remotely controlled device. Most sSuch systems require establishing a voice path to carry Dual-Tone Multi-Frequency (DTMF) signals between a remote telephone and the device. These prior remote control efforts, however, require the consumer to introduce additional special-purpose, locally resident devices, which complicate and increase the cost of deployment. A primary novelty of our approach is the elimination of the need for additional, local devices via the development and deployment of a unified Key Press Markup Language and SIP Event Package (KPML) to address the need for a signaling-layer protocol for transmitting user stimulus from low-power, consumer devices such as IP phones to control household consumer devices.

Many prior systems enable users to remotely control devices remotely, such as household appliances, office equipment, or equipment at unmanned locations, via non-telephony signaling mechanisms, such as e-mail or short-message service (SMS). Others use a telephone, but require the consumer to install specialized hardware. Our approach enables any 12-key telephone, anywhere in the world, to control household devices.

Yamamoto, et al., [1] describe, amongst other things, a tele-control interface unit. This unit, which attaches to the described key system, translates touch-tone commands entered into a phone to the JEMA Home Appliance protocol standard. Wong [2] describes a purpose-built interface unit that appears to the phone network as a telephone, but presents a touch-tone-based user interface to allow a remote phone to control a set of switched power supply outlets. They introduce the concept of a local phone, which does not require user authentication by virtue of being locally connected. Koyuncu [3] describes effectively the same system, but one that uses a PC card as a telephone network interface.

Other work has leveraged the reality that the modern home network has TCP/IP and Bluetooth connectivity. For example, Kanma, et al. [4] describe a system where they introduce a Java application that uses the Bluetooth transceiver in a cellular phone to send and receive control commands from Bluetooth-linked home devices.

There are a few drawbacks with the current state-of-the-art. This is particularly true for home appliances that have digital, TCP/IP connectivity.

First, as previously stated, all of these schemes require the consumer to install some sort of local hardware. For example, the phone-based controllers require phone-line terminating customer premises equipment. Many of the network-based schemes require customer premises gateways.

Another problem is the treatment of local versus remote phones. For most of the home-based phone controllers, all phones look remote. That is, they use the same authentication scheme and in-band transmission of dual-tone multi-frequency (DTMF) tones. This has a number of implications. First of all, even if the phone and appliance are both TCP/IP-enabled, the phone must use its DTMF generators, and the appliance (or gateway/controller) must use its DTMF receivers to receive commands. This means that the methods do not leverage the inherent location-independence and authentication mechanisms present in the TCP/IP suite.

Schulzrinne, et al. [5] describe a Bluetooth, SIP, location-aware set of devices. Likewise, as described in Kanma, et al. [4], one could construct a rich client, using HTML or WML. However, that assumes a device with a non-trivial display and the processing power to do the rendering. Clearly, such an interface would not work over a plain-old telephone service device, which only has DTMF tone generators and a 12- or 16-key keypad for user input.

Even though a phone might be able to send digits as H.245 UserInputIndication (a signaling packet that indicates a DTMF key entered) [6], such messages are on a per-key-press basis, which wastes network resources and precious processing resources at the phone.

II. Home Real-Time Multimedia Network

In Fig. 1, we show a typical home real-time multimedia network. The various elements are a controller, a SIP Phone, gateways for traditional phones, and the appliances being under controlled.

E. W. Burger is with Brooktrout Technology, Inc., Salem, NH 03079 USA. (e-mail: eburger@brooktrout.com)

O. Frieder is with the Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: ophir@ir.iit.edu)

Manuscript received February 6, 2006

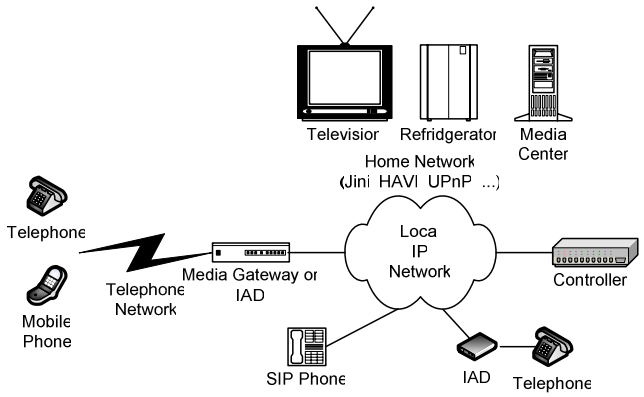

Figure 1 - Home Network

A SIP Phone [7] is a physical phone that includes the circuitry for digitizing, compressing, and packetizing voice or video. The SIP Phone also includes a SIP stack and a user interface. The SIP Phone includes at least the familiar 12-button keypad of the traditional phone. It may include a small display and additional keys. Throughout we do not consider intelligent devices, as they approach the display and user interaction qualities of a PC. For example, the cellular phone described by Kanma, et al. [4] has a sophisticated display and can run Java applications. Likewise, we do not expect the device to support UPnP, such as the sophisticated devices described by Verhoeven and Dees [8]. These devices are far beyond the capabilities of a traditional telephone.

One can connect a traditional phone to the IP network through a media gateway or integrated access device (IAD). The media gateway acts as a bridge from the traditional phone network to the IP network. IADs (and some media gateways) appear to the phone to be the phone network. For example, they provide dial tone and collect digits for formulating a connection request (dialed number).

The IAD is a consumer device. The media gateway can be either a consumer device in the home serving one or a small number of telephones or it can be carrier equipment serving hundreds or thousands of telephone lines. Either way, the gateway or IAD translates PSTN signaling to SIP and the bearer (voice) path to RTP [9].

An interesting feature of using SIP is that traditional phones connected locally to an IAD or remotely through the public telephone network appear the same to the network.

The function of the controller is to receive commands from the telephone and translate them into the appropriate appliance control commands.

In the prior art for using a telephone as a remote control, one establishes a full-duplex media connection between the telephone and the controller. This allows the controller to detect the user signaling, which in this case are in-band DTMF tones.

One can extend the model to the digital world. In VoIP, the bearer channel, where the actual media (e.g., voice) traverses the network, uses the real-time protocol, or RTP. RTP can transport the actual tones for DTMF. In addition, in VoIP, one can send named tones [10] over the bearer channel in RTP. Using named tones, the phone can send a packet indicating the user has pressed the "1" key, rather than, or in addition to, transmitting the DTMF for the "1" key, which happens to be 697 Hz + 1209 Hz [11]. For historical reasons, such named tone packets are called RFC 2833 packets.

One might consider RFC 2833 packets to be signaling, rather than media. However, RFC 2833 packets have a number of drawbacks when used for signaling. First, the packets physically travel in the bearer stream. This means that any application that has interest in the signaling must be in the bearer path. In the analog world, this was not a problem, as the only way to detect DTMF was to establish a bearer-channel connection to the phone, usually over the public telephone network. However, it is an excessive burden to place on an application to have to terminate and interpret the bearer channel if all it is looking for is key press user input.

A second drawback of using RFC 2833 is that RTP, the transport RFC 2833 uses, is not a reliable delivery mechanism. RTP trades off reliable delivery for real-time stream delivery, where it is more important that the packets come at regular (or predictable) intervals, even if that means dropping a packet.

Whereas this is an acceptable trade-off for the delivery of multimedia streams, it is not acceptable for the delivery of signaling.

In our system, we make only a signaling connection between the SIP Phone or gateway and the controller. This has a number of advantages. First, the controller does not need expensive tone detection circuitry. Second, the SIP Phone directly tells the controller the exact key pressed, rather than translating the key press into another format for the controller to decode. Third, the controller only needs a signaling stack (SIP in this case); it does not require a media stack (such as RTP). Fourth, if the user is dialing up remotely, and a service provider hosts the media gateway, only the signaling need traverse the access network. Given tThe typical rate of packets for signaling areis on the order of 1about one packet per /second. and tThe typical rate of ppackets for voice media areare on the order of 50about fifty packets per second, this is a significant savings. Fifth, there are well-known problems with establishing media connections through a network address translator[12], or NAT, which most residential network gateways are. For example, many signaling protocols, including SIP, insert the IP address of the media endpoint into some signaling messages. However, a NAT silently changes the endpoint's IP address, making the address in the signaling incorrect. Transmitting the user input in the signaling layer eliminates these problems.

III. KPML

SIP Phones and gateways use the KPML protocol [13] to transmit key presses in the signaling layer. KPML combines a markup, the Key Press Stimulus Markup Language, with the SIP SUBSCRIBE / NOTIFY protocol. The following is a brief overview of KPML.

There are three primary goals of KPML. The first is to present a compact, application-level representation to reduce the processing burden of application clients. This is particularly important for consumer electronic devices. The second is to reduce the number of messages required to transfer application-level state. The third is to reduce network traffic and reduce the application processing burden by

sending messages only to applications interested in a given user input pattern.

Applications send subscription messages to a device. The subscription message for each application identifies a pattern of user input for the device to notify the application is to be notified of.

KPML uses the SIP SUBSCRIBE/NOTIFY mechanism [14]. This mechanism provides a means for handling multiple, independent requests.

The device monitors input from the user to identify the occurrence of the patterns identified in the subscription messages. When the pattern occurs, the device notifies the corresponding application. The device only notifies the particular application that provided the subscription message, conserving processing and communications resources.

Subscription messages can also contain tags associated with patterns. When the device detects a match and reports it to the application, the device also returns the tag, enabling the application to easily determine easily exactly what response to the input is appropriate without needing to maintain a large amount of internal state information.

Some applications care to continuously monitor the stream continuously for a particular pattern. However, other applications look for only a single occurrence of a particular pattern, at which time the application is finishesd monitoring or may register a different set of patterns. The first type of request is referred to as a "persistent" request, whereas the second type is termed a "one-shot" request. KPML provides for the requesting application to specify the nature (persistent or one-shot) of the request.

Indicating the nature of the request in the subscription reduces the protocol overhead for terminating the subscription. For many interactions, the subscription termination messages may be a significant portion of the overall number of messages and bytes transferred.

In Fig. 2, we illustrate a sample KPML request. Here the key sequence *1 indicates a command for the PC and *2 indicates a command for the oven. The trailing * indicates to turn the appliance on whereas # indicates to turn the appliance off.

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml-request xmlns="urn:ietf:params:xml:ns:kpml-
request"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:kpml-
request kpml-request.xsd"
 version="1.0">
    <pattern>
        <regex tag="PCOn">*1*</regex>
        <regex tag="PCOff">*1#</regex>
        <regex tag="OvenOn">*2*</regex>
        <regex tag="OvenOff">*2#</regex>
    </pattern>
</kpml-request>
```

Figure 2 - KPML Request

By using a tag, one can abstract the control program from the actual interface. That is, rather than having code looking for the string "*1*", the code looks for the tag "PCOn". This makes it easier to introduce new modalities, such as HTML on a PDA, with only minor changes to the control program.

Note the SIP Phone will only notify the controller when the user enters a matching pattern. This is useful if multiple applications are looking for different patterns.

IV. Local and Remote Operation

The concept of having different user interfaces depending on the physical location of the phone is not new. For example, Wong's controller [2] had a switch that enabled a user with a locally-connected phone to bypass entering a pass code. The idea was that if your phone was connected "in the home", the fact of physical access was enough to authenticate the user.

Most controllers use a password challenge. That is, when the remote user calls the controller, the controller prompts the user to enter a series of digits to indicate the caller is really the homeowner.

We can improve upon this distinction of who is an authorized user. In the traditional phone system, all we can do is detect a directly connected phone (with a non-public interface, as in Yamamoto, et al. [1]) or require the pass code. In the digital SIP network, we can leverage modern, digital security methods[15]. One example of such methods is X.509 certificates for TLS [16].

With a pre-loaded X.509 certificate, the user's SIP phone can automatically authenticate itself with the network. This method does not necessarily require the user to enter a pass code (although one can require this for added security).

A novel feature is that now, rather than having authentication based on physical connectivity, we achieve authentication is achieved with distributed cryptographic algorithms. This means that the user can take their phone with them when they travel, and the phone itself is their authentication token, rather than a digit string.

Likewise, if the user uses, for example, a remote soft phone, they can enter a considerably richer and longer password for a digest authentication [17]. This retains the ability of a user to use a public phone, yet have the benefit of a streamlined (and much more secure) authentication mechanism.

V. Protocol Comparison

We implemented KPML in a network IP gateway [18], and others are implementing it in a SIP Phone and circuit switched network media gateway. Significant bandwidth and packet reduction occurred. In addition, much simpler controllers that do not need bearer channel circuitry were constructed. Moreover, KPML enables one to build low power consumer SIP Phones to be built that do not require elaborate browsers or displays. At the same time, the KPML application model makes it easier to build controllers that offer both DTMF interfaces to phones and rich, graphical user interfaces to PCs and PDAs.

As an example of the efficiencies of KPML, we look atconsider a typical remote home control scenario. In this scenario, we wish to control the power to a local PC through a switched outlet power controller. The purpose is to remotely power-on (and off) the PC before accessing it over the Internet.

In the call flow, the user connects to the controller, hears a voice prompt asking for a pass code (if the device is nothas no authenticationed), and then hears a prompt asking which outlet number to turn on or off. The pass code is eight digits and the user enters four commands before hanging up.

In the following analysis, we consider the approach of detecting tones in-band as compared to the KPML approach.

Note that the traditional approach of dialing in to an analog controller over the PSTN is functionally equivalent to using in-band tones. This is because the SIP gateway function is simply translating the analog phone signals into the digital domain. Likewise, if the call is already in the VoIP domain, but the tones are still in-band, a network IP gateway performs the tone detection and injection of the KPML messages into the SIP signaling path.

A. Detailed Call Flow

In Fig. 3, we illustrate shows the general call flow.

First there is a call from a gateway to the controller. In this case,Here the user calls their controller. This can occur by translations from a dialed number. In this case, the user dials a number, like 800 555 1212, which routes over the PSTN to the gateway, which then maps the number to the SIP address of the controller. Instead of calling over the PSTN, the user could connect directly to the controller from a SIP phone by entering the Request-URI (address) of the controller. Likewise, an IAD can do the mapping from a number to the controller Request-URI. In this example, the Request-URI is "sip:controller@user.example.com".

Since the call flows are identical for each of the access cases, namely via a gateway, an IAD, or a SIP phone, we use the term gateway to represent all of these scenarios, including other call establishment procedures and devices known in the art. Details of the session establishment protocol are in the SIP specification, RFC 3261 [19].

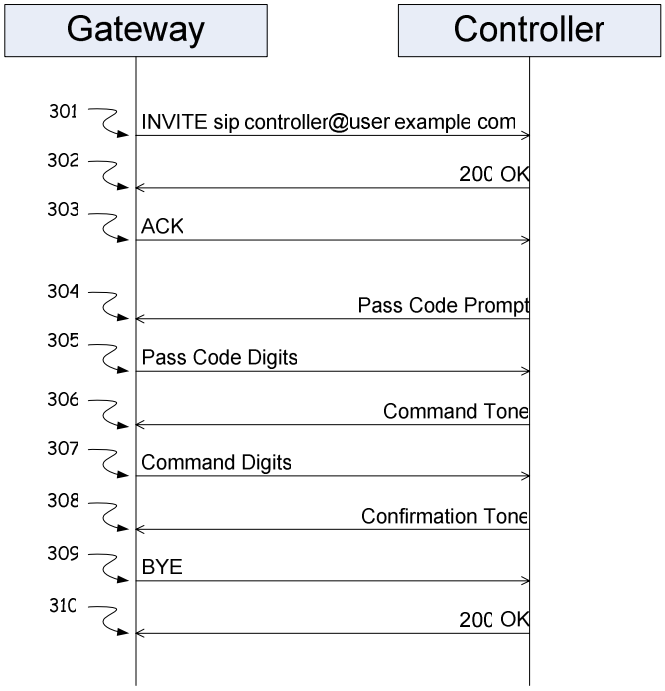

Figure 3 - Call Flow

In all scenarios, one establishes the session with the SIP handshake in messages **(301)**, **(302)**, and **(303)**. The handshake establishes the session by exchanging and negotiating capabilities.

The controller then prompts the user by streaming the prompt requesting the caller for a pass code **(304)**. The user enters the pass code digits **(305)**. Upon correct entry of the pass code, the controller streams a command tone **(306)**, after which the caller enters the command digits **(307)**. The controller emits a confirmation tone **(308)**. The cycle of command digits and confirmation tones can repeat, until the caller hangs up the phone. At this point the phone or gateway sends a disconnect message **(309)**, which the controller acknowledges **(310)**.

B. Named Tones

For the named tones case, the digit signaling **(305)** from Fig. 3 becomes expanded by the redundancy protocol of RFC 2833. The RFC 2833 redundancy protocol is to repeat a given digit, piggy-backed on a new digit, for some number of digits, usually 5. The gateway sends the redundant digits in the same packet with new digits, saving the packet overhead for the digits. If the caller does not enter a new digit within the repeat period, the gateway emits a packet with the remaining redundant digits.

Note that this is an example of "reliability by hope." RFC 2833 assumes there is a loss of connectivity for no more than the redundancy period. The redundancy period is the time period from when the gateway sends the first copy of the digit to the last redundant copy. RFC 2833 "hopes" that the network will deliver at least one copy of the digit during the redundancy period. Besides being inefficient, in that the gateway sends many more packets than necessary, the redundancy scheme does not provide a reliable digit delivery mechanism.

In Fig. 4, we shows the flow for the pass code digits. A similar expansion occurs for the command digits.

Steps **(401)**, **(402)**, and **(403)** in Fig. 4 are the SIP session establishment protocol, INVITE, 200 OK, and ACK, respectively.

In step **(404)**, the controller streams the pass code prompt to the gateway.

The user enters the first digit, which the gateway encodes as a RFC 2833 named tone packet **(405)**. The user enters the second digit, which the gateway encodes as a RFC 2833 named tone packet, with a redundant copy of the first digit **(406)**. The gateway encodes and sends the user's subsequent digits in steps **(407)** through **(417)**.

C. KPML

In Fig. 5, we shows the entire call flow for this scenario using KPML.

Steps **(501)**, **(502)**, and **(503)** in Fig. 5 are the SIP session establishment protocol, INVITE, 200 OK, and ACK, respectively.

The controller subscribes for the tone detection events **(504)**, which the controller gateway acknowledges **(505)** and sends an instant notification **(506)**, which the gateway controller acknowledges **(507)**.

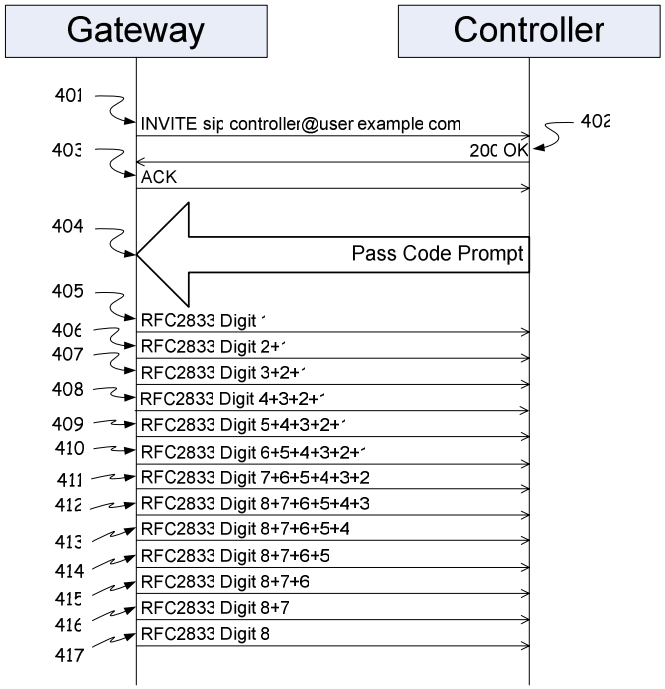

Figure 4 - Named Tones Call Flow Extract

The controller then streams the pass code prompt to the gateway **(508)**. The user enters the digits, and the gateway sends the collected digits to the controller **(509)** which the controller acknowledges **(510)**.

The controller streams the command tone to the gateway **(511)**. In response, the user enters the command, which the gateway sends to the controller **(512)** and the controller acknowledges **(513)**. This cycle completes three more times **(512514)** through **(523522)**. After the next confirmation tone **(523)**, , at which point the user hangs up **(524)** and the controller acknowledges the disconnect **(525)**.

The gateway then informs the controller of the subscription termination due to the user hanging up **(526)** which the controller acknowledges **(527)**.

VI. Network Utilization

We set out to compare the efficiencies of using various digit signaling schemes, including pure in-band transmission of DTMF tones (G.711), sending named tones (RFC 2833), and using KPML.

In-band transmission is simply where the gateway converts the analog signals into a continuous stream of G.711 RTP packets. RFC 2833 sends media packets during prompt playing, but sends only named tone packets (RFC 2833) and redundancy packets in the media stream when the caller enters digits. KPML sends media packets during prompt playing, but sends only KPML events (and handshakes) in the signaling stream when the caller enters digits.

To determine what the associated cost is, if any, to use a reliable protocol in the signaling channel (KPML), as opposed to sending signaling in the media channel (G.711 and RFC 2833), we conducted an experimental study.

A. G.711

The pass code prompt is 2140ms long.

G.711 places 20ms of audio into a 238 byte RTP packet. The 238 byte figure includes 160 bytes for the audio payload, as well as the RTP, UDP, IP, and Ethernet headers. Thus, the pass code prompt uses (2140ms) / (20ms/packet) = 107 packets. At 238 bytes per packet, the prompt uses 25,466 bytes.

The command **(306)** and confirmation **(308)** tones are 400ms each, or 20 G.711 20ms packets using 4,760 bytes.

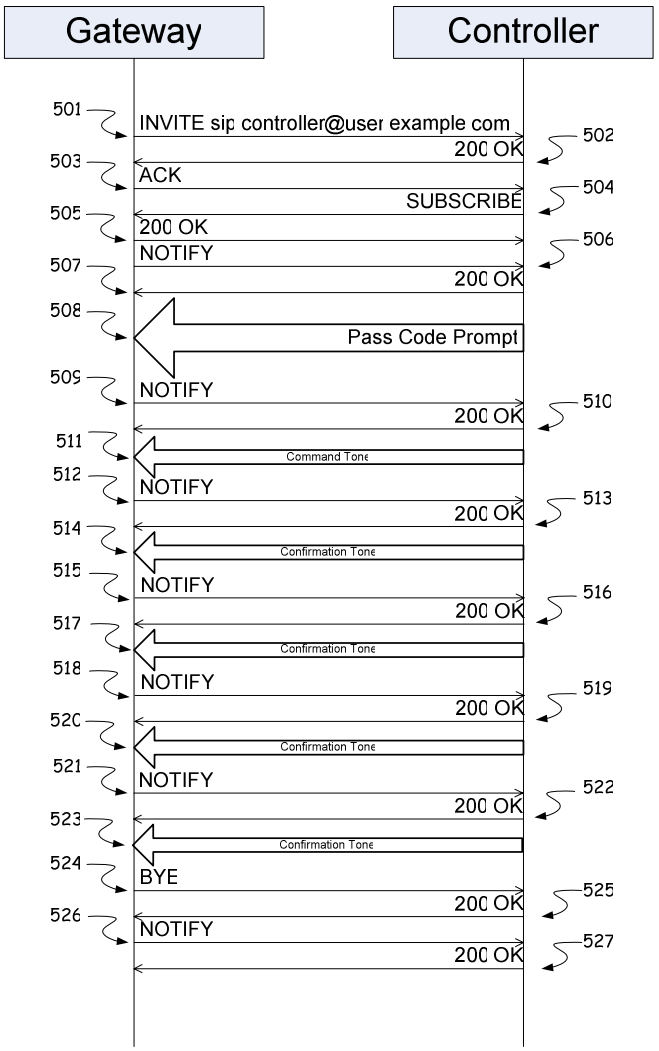

Figure 5 - Entire KPML Call Flow

There is a system processing delay of 500ms after the controller receives the command, before sending each confirmation tone.

VI. Network Utilization

We set out to compare the efficiencies of using various digit signaling schemes, including pure in-band transmission of DTMF tones (G.711), sending named tones (RFC 2833), and using KPML.

In-band transmission is simply where the gateway converts the analog signals into a continuous stream of G.711 RTP packets. RFC 2833 sends media packets during prompt playing, but sends only named tone packets (RFC 2833) and redundancy packets in the media stream when the caller enters digits. KPML sends media packets during prompt playing, but sends only KPML events (and handshakes) in the signaling stream when the caller enters digits.

We were looking to see what the cost, if any, there is to use a reliable protocol in the signaling channel (KPML) as opposed to sending signaling in the media channel (G.711 and RFC 2833).

A. G.711

The pass code prompt is 2140ms long.

G.711 places 20ms of audio into a 238 byte RTP packet. The 238 byte figure includes 160 bytes for the audio payload, as well as the RTP, UDP, IP, and Ethernet headers. Thus, the pass code prompt uses (2140ms) / (20ms/packet) = 107 packets. At 238 bytes per packet, the prompt uses 25,466 bytes.

The command **(306)** and confirmation **(308)** tones are 400ms each, or 20 G.711 20ms packets using 4,760 bytes.

There is a system processing delay of 500ms after the controller receives the command, before sending each confirmation tone.

Summing up these times, we find the total transaction time for the scenario examined in *Protocol Comparison* above is 2140ms for the prompt, 3380ms for the pass code entry, 400ms for the confirmation tone, 1350ms for the command entry, 500ms of system processing time, and 400ms for the command confirmation tone. This results in a grand total of 6480ms8170ms. For the scenarios where the gateway and controller continuously transmit G.711, the 8170ms transaction time represents the transmission of 409 packets and 97,342 bytes in each direction.

Note that TCP, IP, and Ethernet headers add 40 bytes to a TCP packet.

B. RFC 2833

In Table 1, we enumerate shows the packet and byte counts for the message and RTP exchanges for the long transaction described in *Protocol Comparison*, using RFC 2833 for signaling transport. The numbers in the "#" column refer to the message number. Numbers past 417 are message number from the exchange that occur after the dialog depicted in Fig. 4.

In Fig. 4, the messages labeled "RFC 2833 *n*" means a message with *n* digits in it. For example, message **(407)**, labeled "RFC 2833 3", has three digits in it, the third user-entered digit, the first redundant copy of the second digit, and the second redundant copy of the first digit.

TABLE 1 – LONG RFC2833 TRANSACTION

#	Msg.	Len.	w/TCP	Bytes	I/O	Pkts. In	Pkts. Out
401	INVITE	586	626	626	I	1	0
402	OK	608	648	648	O	0	1
403	ACK	194	234	234	I	1	0
404	PC Prompt	25466	0	25466	O	0	107
405	RFC 2833 1	86		86	I	1	0
406	RFC 2833 2	95		95	I	1	0
407	RFC 2833 3	103		103	I	1	0
408	RFC 2833 4	111		111	I	1	0
409	RFC 2833 5	119		119	I	1	0
410	RFC 2833 6	127		127	I	1	0
411	RFC 2833 6	127		127	I	1	0
412	RFC 2833 6	127		127	I	1	0
413	RFC 2833 5	119		119	I	1	0
414	RFC 2833 4	111		111	I	1	0
415	RFC 2833 3	103		103	I	1	0
416	RFC 2833 2	95		95	I	1	0
417	RFC 2833 1	86		86	I	1	0
418	Cmd Tone	4760		4760	O	0	20
419	RFC 2833 1	86		86	I	1	0
420	RFC 2833 2	95		95	I	1	0
421	RFC 2833 3	103		103	I	1	0
422	RFC 2833 3	103		103	I	1	0
423	RFC 2833 3	103		103	I	1	0
424	RFC 2833 3	103		103	I	1	0
425	RFC 2833 2	95		95	I	1	0
426	RFC 2833 1	86		86	I	1	0
427	Conf Tone	4760		4760	O	0	20
428	RFC 2833 1	86		86	I	1	0
429	RFC 2833 2	95		95	I	1	0
430	RFC 2833 3	103		103	I	1	0
431	RFC 2833 3	103		103	I	1	0
432	RFC 2833 3	103		103	I	1	0
433	RFC 2833 3	103		103	I	1	0
434	RFC 2833 2	95		95	I	1	0
435	RFC 2833 1	86		86	I	1	0
436	Conf Tone	4760		4760	O	0	20
437	RFC 2833 1	86		86	I	1	0
438	RFC 2833 2	95		95	I	1	0
439	RFC 2833 3	103		103	I	1	0
440	RFC 2833 3	103		103	I	1	0
441	RFC 2833 3	103		103	I	1	0
442	RFC 2833 3	103		103	I	1	0
443	RFC 2833 2	95		95	I	1	0
444	RFC 2833 1	86		86	I	1	0
445	Conf Tone	4760		4760	O	0	20
446	RFC 2833 1	86		86	I	1	0
447	RFC 2833 2	95		95	I	1	0
448	RFC 2833 3	103		103	I	1	0
449	RFC 2833 3	103		103	I	1	0
450	RFC 2833 3	103		103	I	1	0
451	RFC 2833 3	103		103	I	1	0
452	RFC 2833 2	95		95	I	1	0
453	RFC 2833 1	86		86	I	1	0
454	Conf Tone	4760		4760	O	0	20
455	BYE	192	232	232	I	1	0
456	OK	223	263	263	O	0	1
	Total In	5,597	Total Out	50,177	Total Pkts.	48	209

C. KPML

In Table 2, we enumerate the packet and byte counts for the message and RTP exchanges for the long transaction described in *Protocol Comparison*, using KPML for signaling transport. The numbers in the "#" column indicate the message number from Fig. 5.

C. KPML

Table 2 shows the packet and byte counts for the message and RTP exchanges for the long transaction described in *Protocol Comparison*, using KPML for signaling transport. The numbers in the "#" column indicate the message number from Fig. 5.

TABLE 2 – LONG KPML TRANSACTION

#	Msg.	Len.	w/TCP	Bytes	I/O	Pkts. In	Pkts. Out
501	INVITE	586	626	626	I	1	0
502	OK	488	528	528	O	0	1
503	ACK	192	232	232	I	1	0
504	SUBSCRIBE	851	891	891	O	0	1
505	OK	257	297	297	I	1	0
506	NOTIFY	285	325	325	I	1	0
507	OK	239	279	279	O	0	1
508	PC Prompt	25466	0	25466	O	0	107
509	NOTIFY	619	659	659	I	1	0
510	OK	239	279	279	O	0	1
511	Cmd Tone	4760		4760	O	0	20
512	NOTIFY	619	659	659	I	1	0
513	OK	239	279	279	O	0	1
514	Conf Tone	4760		4760	O	0	20
515	NOTIFY	619	659	659	I	1	0
516	OK	239	279	279	O	0	1
517	Conf Tone	4760		4760	O	0	20
518	NOTIFY	619	659	659	I	1	0
519	OK	239	279	279	O	0	1
520	Conf Tone	4760		4760	O	0	20
521	NOTIFY	619	659	659	I	1	0
522	OK	239	279	279	O	0	1
523	Conf Tone	4760		4760	O	0	20
524	BYE	192	232	232	I	1	0
525	OK	223	263	263	O	0	1
526	NOTIFY	276	316	316	I	1	0
527	OK	239	279	279	O	0	1
	Total In	5,323	Total Out	52,901	Total Pkts.	11	217

VII. ANALYSIS

We examined the message flows and bandwidth usage of various representative methods of transmitting signaling information from a user with a plain old telephone. Namely, we examined sending DTMF tones from the telephone, through a gateway, to a controller (the G.711 case); sending packets that are representative of the DTMF tones (the named tones or RFC 2833 case); and using a signaling-level protocol (the KPML case).

Using signaling instead of the actual digital waveforms for transporting the user input clearly is a benefit to network resources consumption. Both the named tones without G.711 and KPML have more than an order of magnitude fewer inbound bytes and inbound packets than the continuous RTP stream (DTMF). In Table 3, shows a comparison of we compare the results.

TABLE 3 - SUMMARY

Protocol	Inbound Bytes	Outbound Bytes	Inbound Packets	Outbound Packets
DTMF	97,342	97,342	409	409
RFC 2833	5,597	50,177	48	209
KPML	5,323	52,901	11	217

The prompts dominate the outbound byte count, is dominated by the prompts, but only having to send packets when actually playing a prompt clearly is advantageous.

Examining Table 3, we find that the byte counts for KPML and RFC 2833 are on the same order of magnitude. We see that RFC 2833 uses approximately 5% fewer outbound bytes and packets than KPML. However, it uses approximately 5% more inbound bytes and over four times the packets than KPML.

Most important, RFC 2833 does not have the protocol property of reliable delivery. Moreover, the KPML model enables stateless servers, which the named tone model does not. KPML provides the properties of an easier-to-program model and reliable delivery at only a small premium over RFC 2833.

In addition, the more digits captured and interpreted by KPML, the smaller the difference in network utilization becomes on the outbound side. KPML has a more pronounced benefit when considering, whereas the difference in network utilization on the inbound side becomes more pronounced in favor of KPML.

The programming flexibility of KPML over RFC 2833 is an important feature. Consider how the program logic at the controller changes when one goes from eight-digit pass codes to four-digit pass codes. In the RFC 2833 case, one must modify the program logic at the controller to expect a different number of digits. In the KPML case, one only needs to modify the KPML markup. For either four or eight digit pass codes, the controller receives a NOTIFY with the "pw" tag, informing the controller that the digits represent a valid pass code. This logic is much simpler than having to parse out each and every digit individually.

VIII. CONCLUSIONS

This paperWe described a new environment, KPML, . This environment makes it possible towhich easily and efficiently controls devices in the home environment remotely, without the need for specialized hardware in the home devices. This environment imposes no requirements for complex line sharing or specialized control hardware. The ubiquitous plain old telephone with a 12-digit keypad is all that one needs to control the home devices.

We have shown how KPML provides an efficient, reliable protocol for the remote control of consumer devices using plain old telephones with 12-digit keypads using Internet transport technologies. More importantly, KPML enables device developers to create dynamic user interfaces, which are much easier to create and maintain.

Further contributions not explored in this paper include the ability of the protocol KPML approach to allow multiple devices to simultaneously get input from a single controlling device, how KPML maps to the web model of application development, and enhancements that reducereduce the number of SIP messages for single, stand-alone user interface interactions.

IX. ACKNOWLEDGMENT

The authors would like to thank Martin Dolly for his editorial contribution to the IETF KPML standardspecification.

X. REFERENCES

[1] K. Yamamoto, S. Shinohara, and H. Yokota, "New home telephone system using Japanese Home Bus System standard," *IEEE Transactions on Consumer Electronics*, vol. 35, pp. 687-697, 1989.

[2] E. M. C. Wong, "A phone-based remote controller for home and office automation," *IEEE Transactions on Consumer Electronics*, vol. 40, pp. 28-34, 1994.

[3] B. Koyuncu, "PC Remote Control of Appliances by Using Telephone Lines," *IEEE Transactions on Consumer Electronics*, vol. 41, pp. 201-209, 1995.

[4] H. Kanma, N. Wakabayashi, R. Kanazawa, and H. Ito, "Home appliance control system over Bluetooth with a cellular phone," *IEEE Transactions on Consumer Electronics*, vol. 49, pp. 1049-1053, 2003.

[5] H. Schulzrinne, X. Wu, S. Sidiroglou, and S. Berger, "Ubiquitous computing in home networks," *IEEE Communications Magazine*, vol. 41, pp. 128-135, 2003.

[6] ITU-T, "Control protocol for multimedia communication," ITU, Geneva, Recommendation H.245, July 2003.
[7] H. Schulzrinne and J. Rosenberg, "The Session Initiation Protocol: Internet-centric signaling," *IEEE Comm. Mag.*, vol. 38, pp. 134-141, 2000.
[8] R. Verhoeven and W. Dees, "Defining services for mobile terminals using remote user interfaces," *IEEE Transactions on Consumer Electronics*, vol. 50, pp. 535-542, 2004.
[9] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF, RFC 3550, July 2003.
[10] H. Schulzrinne and S. Petrack, "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals," IETF, RFC 2833, May 2000.
[11] ITU-T, "Technical Features of Push-Button Telephone Sets," International Telecommunications Union, Geneva, ITU-T Recommendation Q.23, November 1998.
[12] T. Hain, "Architectural Implications of NAT," IETF, RFC 2993, November 2000.
[13] E. W. Burger and M. Dolly, "A Session Initiation Protocol (SIP) Event Package for Key Press Stimulus (KPML)," IETF, Internet Draft draft-ietf-sipping-kpml-07, December 29, 2004, work in progress.
[14] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification," IETF, RFC 3265, June 2002.
[15] S. Moyer, D. Marples, and S. Tsang, "A protocol for wide area secure networked appliance communication," *Communications Magazine, IEEE*, vol. 39, pp. 52-59, 2001.
[16] T. Dierks and P. L. Karlton, "The TLS Protocol Version 1.0," IETF, RFC 2246, January 1999.
[17] J. Franks, P. M. Hallam-Baker, J. L. Hostetler, S. D. Lawrence, P. J. Leach, A. Luotonen, and L. C. Stewart, "HTTP Authentication: Basic and Digest Access Authentication," IETF, RFC 2617, June 1999.
[18] E. Burger, "A New Inter-Provider Interconnect Technology for Multimedia Networks," *IEEE Communications Magazine*, vol. 43, pp. 147-151, 2005.
[19] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," IETF, RFC 3261, June 2002.

Eric W. Burger (M'84-SM'00) is the Chief Technology Officer of Brooktrout Technology, Inc. Prior to Brooktrout, he founded SnowShore Networks, Inc., where he invented the SIP Controlled Multifunction Media Server. He held various research and management positions with MCI, Cable & Wireless, ADC/Centigram, The Telephone Connection, Valid Logic Systems, and Texas Instruments. He holds degrees from the Massachusetts Institute of Technology and K.U. Leuven, and is currently affiliated with the Illinois Institute of Technology. He currently serves as the Chair of the SPEECHSC and LEMONADE Work Groups in the IETF and is active in signaling and applications protocol development in the IETF and W3C. He is a member of the ACM. His research interests focus on real-time multimedia protocols and architectures for large-scale multimedia systems.

Dr. Ophir Frieder (SM '93, F '02) is the IITRI Chair Professor of Computer Science and the Director of the Information Retrieval Laboratory at the Illinois Institute of Technology. His research interests focus on scalable information retrieval systems spanning search and retrieval and communications issues. He is an AAAS Fellow, ACM Fellow, and IEEE Fellow.