A Context Aware Gateway for SIP-based Services in Ubiquitous Smart Homes

Bo-Chao Cheng[1], Huan Chen[2] and Ryh-Yuh Tseng[3]
[1,3]Department of Communications Engineering, National Chung-Cheng University
[2]Department of Electrical Engineering, National Chung-Cheng University
e-mail : [1]bcheng@ccu.edu.tw, [2]huan@ee.ccu.edu.tw,
[3]rytseng@insa.comm.ccu.edu.tw

Abstract

The smart home concept brings innovation and convenience to our every day's life style. Without context awareness, smart home applications can neither offer adaptive services based on users' dynamic situations nor provide preferential treatments according to users' various preferences. Such a design paradigm forces human to live in the world of machines rather than a human-centric world. The use of SIP-protocol allows smart home devices and services to be connected compatibly and effectively. In this paper, we propose an adaptive SIP Context Aware Gateway (SIP-CAG) for the ubiquitous SIP-based services. With the SIP-CAG solution, the home owner is able to use preferred devices for communication and to post his/her context information to home appliances or to people and civil society communities on an "as needed" basis.

1. Introduction

The general goal of the smart-home development is to integrate home devices, services and resources by the use of network technologies so that home appliances would become more "intelligent" to adapt to users' needs. With the advances of the sensor technology and wireless communication, more and more companies such as IBM, Microsoft, Simens and Cisco are conducting projects to work on this field. In particular, the Korea government has committed to develop on-going programs for promoting the applications of smart home environment [5]. In August 2003, the Ministry of Commerce, Industry and Energy (MOCIE) of Korea government announced a ten year investment plan which includes the Smart Home to be one of the national development industries in Korea. In the same year, Ministry of Information and Communication (MIC) proposed the "Smart Home Vision 2007" to promote Korea to be the leading smart home country in the ubiquitous era.

Mark Weiser pointed out that the ubiquitous computing era is coming soon to a life near you [16]. Context awareness is the key technology concept in ubiquitous computing, where context is defined as the information characterizing the situations of an entity [4][16]. From the perspective of human, the semantic meaning of "smartness" for home appliances shall mean that they are adaptive to inhabitants according to their current environment situations (i.e., context). In the near future, smart home applications are expected to take more responsibility to cover home-care, security and personal assistance tasks.

Most smart home applications focus on enhancing the functionality and capability, but they can neither adapt to the users' dynamic situations nor provide preferential treatments to their preferences. Such a design paradigm forces human to live in the world of machines rather than a human-centric world.

Session Initiation Protocol (SIP) [10] is among the most important Internet protocols used today. It is designed to establish, modify, and terminate a session of the application services. The use of SIP-protocol allows smart home devices and services to be connected compatibly and effectively. SIP has been used in many popular services such as Voice over IP (VoIP), Instant Message (IM), Presence Service and File Transfer. In the near future, it is expected that SIP will play an important role in the smart home environment. In this paper, we design and implement an adaptive SIP Context Aware Gateway (SIP-CAG) based on the IXP425 network processor technology. SIP-CAG is an intelligent gateway to be deployed between home networks and the Internet. In literature, Hoecke et al. [6] proposed a web-services based middleware for e-Homecare services. Both Hoecke's work and our work offer dynamic selection and composition of services. However, the major differences between these two works are two folds. First, Hoecke's work focuses on the web services while our work emphasizes on the SIP-based services. Second, Hoecke's work is on the development of a middleware while ours is on a gateway.

With the SIP-CAG solution, the home owner is able to use SIP User Agent (SIP UA) or Presence User Agent (PUA) to communicate with home appliances, people and civil society communities smoothly. As such, every participant can interact with each other adaptively and collaboratively.

The remainder of this paper is organized as follows: Section 2 describes the system architecture and components of the SIP-CAG. Section 3 provides an example to illustrate the information flow for the presence information posts and service requests. Section 4 presents the performance evaluation of the SIP-CAG prototype implemented on an IXP425 network processor. Finally, Section 5 makes a brief conclusion.

2. System architecture and components

In practice, a home inhabitant could answer the incoming call by choosing his/her preferred SIP UA devices and post the outgoing presence information on an "as needed" basis (based on pre-defined context aware service criteria such as entity-to-entity relationships and current situations). An effective way to do this is to deploy an intelligent gateway to select/control communication sessions between home networks and the Internet. SIP Context Aware Gateway (SIP-CAG), serving as a SIP-based home application secretary, provides tremendous flexibility and capability in adapting to diverse environment situations. SIP-CAG supports the SIP-based context aware applications based on the inhabitant's motivations, behaviors and needs. Figure 1 shows the proposed SIP-CAG architecture that comprises the following major components: SIP Server and Presence Server, Stateful Inspection Layer (SIL), Content Filter (CF), SIP Application NAT (SAN), Feature Selector (FS), Policy of Context (PoC) and SIP-CAG Management Agent (SMA). Components are collaborated together to perform the context management. In the following paragraphs, we describe the function of each component in more detail.

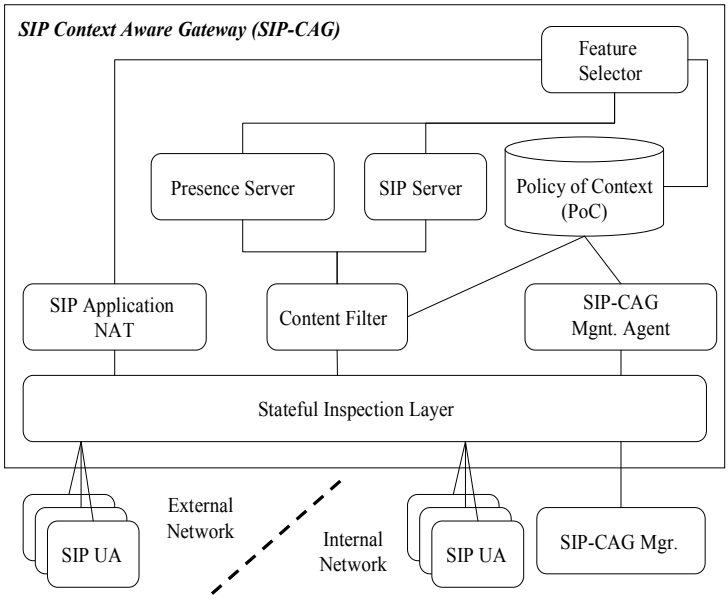

Figure 1 SIP-CAG architecture

2.1. Stateful Inspection Layer (SIL)

Providing the firewall functionality (such as IP address and TCP port filtering), SIL is designed to defense the low-level protocol exposure attacks (e.g., IP spoofing and vulnerability associated with the TCP/UDP ports). By default, firewalls would block all traffic originated from outside networks, but accept those from inside. After a connection is established, the firewall would accept all data streams on it. For the SIP-based communication model, the media connections are setup dynamically. Media Transport Address (MTA), declared in the "c=" and the "m=" lines in the Session Description Protocol (SDP), specifies the IP address and port number at which RTP/RTCP packets can be received. SIP-CAG is a SIP awareness device that dynamically opens MTA ports to accept the corresponding media traffic.

2.2 Content Filter (CF)

Like many services posing serious security threats, SIP applications (e.g., file sharing) increase the potential security risks with viruses/worm infection and copyrights/intellectual property infringements [14]. Unfortunately, current commercial firewall and Intrusion Detection System (IDS) products could not block or detect this kind of vulnerability. Content Filter (CF) provides an efficient solution that scrutinizes the incoming data traffic with deep content inspection to the SDP payload based on predefined security policies. Figure 2 shows an example where the contact point (a specific URL http://insa.comm.ccu.edu.tw/process) in the SDP payload is illegal and it shall be blocked. CF module will tear down the connection between the SIP UA and this contact point. The CF module uses signature/pattern to recognize illegal contents in the SDP payload. As such, CF module can reject malicious or illegal requests.

```
v=0
o= 289024 289024 IN IP4 comm.ccu.edu.tw
s=
t=0 0
c=IN IP4 0.0.0.0
m=data 8080 HTTP application/soap+xml
a=contact: http://insa.comm.ccu.edu.tw/process
a=directionassive
a=wsdl:http://ant.comm.ccu.edu.tw/Guest.wsdl
```

Figure 2 SDP payload for contact negotiation

2.3 SIP Application NAT (SAN)

The traditional network address translation (NAT) only operates at layer 3 and layer 4 [12] and does not modify the IP addresses/port information embedded within the SDP payload. To be interoperable with the traditional NAT/PAT deployed at the front defense line, SAN module replaces the following parameters in the SIP header and SDP payload (as shown in Table 1) [2]. In addition to revising the above fields, SIP-CAG also keeps the status of every session in order to map the internal IP addresses to public IP addresses until the session terminates. Finally, SAN recomputed the content length.

Table 1 SIP-CAG NAT actions

Fields	Description	Modification
Via header	A *via header* field value is added only after the transport that will be used to reach the next hop has been selected	Yes
contact header	The *contact header* provides a SIP or SIPS URI where the client would to receive subsequent requests	Yes
Call-ID	*Call-ID* contains a globally unique identifier for this call, generated by the combination of a random string and the softphone's host name or IP address	No, in order to reflect the mapping
o=	The originator of the session	No, in order to reflect the mapping
c=	The address is provided where media should be sent	Yes
m=	This indicates the transport port where media should be sent	Yes

2.4 Feature Selector (FS)

In the ubiquitous smart home environment, proven context awareness is a must requirement. Constructing a SIP gateway with context awareness architecture involves more than just placing a few databases. FS module in SIP-CAG is designed to direct incoming

service requests to the right receiver and post outgoing presence information traffic with the best-appropriate context.

2.5 Policy of Context (PoC)

A flexible rules language, Behavior Access Manuscript (BAM), is designed to describe the context policy. The following is the format of BAM language:

Action	Response	Protocol	Filter	Direction	Filter	Policy_List

- Action: describes how the CF handles applications traffic, e.g., accept, reject, redirect, encrypt and drop.
- Response: dictates how SIP-CAG responses for a specified session, e.g., none, alert and log.
- Protocol: expresses traffic type, e.g., SIP, TCP and UDP.
- Filter: states what specific content should be examined, e.g., SIP URL.
- Direction: specifies the traffic direction, e.g., <, >, and <>.
- Policy List: depicts the specification of communication preference such as content, presence settings and equipment.

Context policy described by BAM language enables the CF module to check specific SIP events (e.g., Invite, ACK, Cancel and Bye) and explicit SIP message patterns (e.g., "m= data" or "a=*example.com"). The detailed BAM syntax is depicted in Appendix. We show a simple example to demonstrate that Policy of Context (PoC) can be described by BAM language. In this example, we illustrate how a professor interacts with his student depends on his current location. Figure 3 shows the PoC used in this scenario.

```
Accept  Log    SIP  Student@insa.ccu.edu.tw     6000:  <>
        Professor@ccu.edu.tw  5060  Available  Office  Office Phone  IM

Accept  Alert  SIP  Student @insa.ccu.edu.tw     6000:  <>
        Professor @ccu.edu.tw  5060  Away  Home  Mobile  PDA
```

Figure 3 Examples of the PoC

Policy 1 describes that the professor allows his student to get his presence status (Available) and location (Office). Policy 1 also indicates that the student could use office phone or instant message (via port 5060) to communicate with him. When this session terminates, SIP-CAG will log the conversation record into the data base. Policy 2 describes that the mobile phone or the PDA (soft phone) is a permissible device to communicate with the professor when his presence status becomes "Away" and location is "Home". An alert message will be generated when this session is finished. All the connections are limited to port 6000.

2.6 SIP-CAG Management Agent (SMA)

SIP-CAG management consists of the SIP-CAG agent and the SIP-CAG manager. Similar to the way that most network management agent software does, SIP-CAG agent is responsible for monitoring all traffic through the gateway, for communicating management information to the SIP-CAG manager, for configuring the SIP-CAG gateway itself and for maintaining the conversation records. Powered by the web, the SIP-CAG manager can provide a user-friendly management interface where the home owner can define all the required context policy and configure the necessary SIP-CAG service parameters. SIP-CAG management is able to analyze the transaction logs and pinpoint the security risks that are most threatening to the smart home applications.

3. Presence information and service requests

In this section, we show a case study to illustrate how the SIP-CAG works. As shown in Figure 4, SIP-CAG comprises three major components: SIP Server, Presence Server and Feature Selector. The SIP Server is a general SIP module with integrated functions of SIP Proxy Server, Redirect Server and Registrar Server and Location Server. These servers can be either co-located in the same place or deployed in different places. Presence Server complies with the SIP/SIMPLE protocol to perform presence information management tasks. Feature Selector (FS) selects the best communication device and determines appropriate presence information to post for Alice based on her current context and preferences. The decision can be made based on either different profiles of the Watcher (e.g., Bob) or the policies of context (PoC) defined by the Presenity (e.g., Alice).

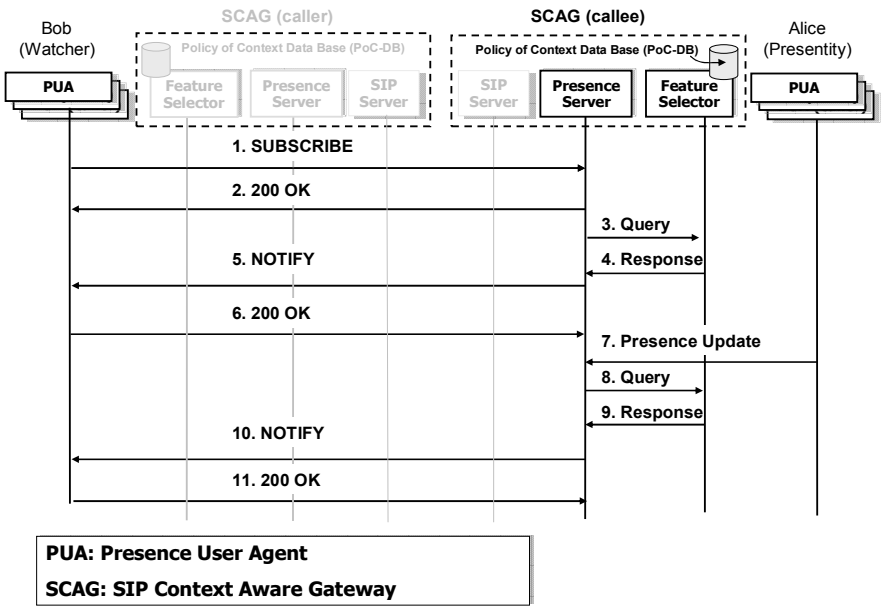

Figure 4 Presence information flow

3.1 Customized Presence Information

The presence information flow presented below shows a simple scenario for a presence user agent (PUA) of "Bob" to subscribe the presence information. For the sake of clarity, the complete verbose texts of the exchanged messages are not shown here. The steps are discussed as below.

Step 1:	Bob sends a SUBSCRIBE request to obtain the presence information of Alice. This request is captured and processed by the Presence Server module in the Alice's SIP-CAG.
Step 2:	The Presence Server acknowledges the request by sending the SIP/2.0 200 OK message to Bob.
Step 3:	Presence Server sends out a Query message to the Feature Selector to learn the appropriate way for posting presence information.
Step 4:	Feature Selector responds the Query with appropriate presence document (containing the presence information of Alice). The appropriate presence document is chosen based on Bob-Alice trust relationships and desired presence information.
Step 5:	The Presence Server sends the NOTIFY message (containing the context presence information of Alice) to Bob. Unlike the conventional design, Alice's presence information is allowed to be different according to the policy of context (PoC). Suppose that Bob is in Alice's close buddy list, and Charley is in her general friend list. In such a scenario, the Bob's PUA will show more detailed location information about Alice (such as "library"), while Charley's PUA, only shows rough location information about Alice (such as "school").
Step 6:	The Bob's PUA acknowledges the message by sending a 200 OK response.
Step 7:	When Alice changes her status, the PUA of Alice sends to the Presence Server to update the presence information of Alice.
Step 8:	Same as Step 3.
Step 9:	Same as Step 4.
Step 10:	The Presence Server sends Bob the NOTIFY message to refresh presence information of Alice.
Step 11:	The Bob's PUA sends a 200 OK response to acknowledges the NOTIFY message.

Please note that the Presence Server can send the "Watcher" a NOTIFY message posting the context information of the Presentity at Step 5 and Step 10 by using PIDF (Presence Information Data Format) [13] to encapsulate Presentity context information. PIDF consists of "Presentity URL", "Presence Tuples" and "Presentity human readable comment" in the message format. After defining the customized namespaces, the context of the Presentity can be expressed in the format <prefix:element-name>value</prefix:element-name>. For example, the element-name, lcs (location context symbol), is defined to describe Alice's current location as shown in the following Figure 5.

```
NOTIFY sip:allen@watcherhost.ccu.edu SIP/2.0
Via: SIP/2.0/TCP server.ccu.edu;branch=z9hG4b423
From: <sip:ryan@ccu.edu>;tag=kiky45
To: <sip:allen@ccu.edu>;tag=frgt23
Call-ID: 6367@watcherhost.ccu.edu
Event: presence
Subscription-State: active;expires=599
Max-Forwards: 70
CSeq: 815 NOTIFY
Contact: sip:server.example.com
Content-Type: application/pidf+xml
Content-Length: 302
```

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
      xmlns:lcs="urn:example-com:pidf-locationstatus "
       entity="pres:ryan@example.com">
   <tuple id="ub93s3">
      <status>
         <basic>open</basic>
         <lcs:location>home</lcs:location>
      </status>
      <contact>im:ryan@example.com</contact>
   </tuple>
</presence>
```

Figure 5 Embedded context information in NOTIFY message

3.2 Customized SIP Service Request

Figure 6 depicts how the SIP-CAG deals with the incoming SIP service requests. Since some components do not participate in the interaction of the incoming SIP connection signaling handshake process, we mark them as dim components (e.g., Presence Server). Upon receiving an INVITE request originated from Bob, Alice's SIP-CAG will determine for her how Alice is connected and at which device. Following the same design concept of the Third Party Call Control (3PCC) defined in RFC 3725 [7], SIP server leaves the decision of how Alice answers the request up to the Feature Selector. The detailed processes are illustrated as follows.

Step 1:	To contact Alice, Bob's SIP UA sends an INVITE request to his local SIP Server.
Step 2:	The SIP INVITE request is routed in the SIP domain and arrives at the SIP Server on the callee side (Alice).
Step 3:	Since the SIP Server is not aware of which device Alice answers, SIP Server sends a query message to Feature Selector by sending an INVITE message without the SDP payload.
Step 4:	Feature Selector acknowledges the INVITE message by replying SIP Server with a 200 OK but with the Alice's SIP UA (callee) information

	contained in the SDP message body.
Step 5:	The SIP Server obtains the appropriate communication method such as at which device and how Alice shall be connected. Then, it acknowledges FS by sending an ACK response.
Step 6:	The SIP Server next initiates a new INVITE request to the appropriate SIP UA of Alice.
Step 7:	The appropriate SIP UA of Alice acknowledges the INVITE message by sending a 200 OK to SIP Server.
Step 8:	The 200 OK message is routed back in the SIP domain and arrives at the SIP Server on the caller side (Bob).
Step 9:	The 200 OK message is forwarded to the SIP UA of caller (Bob).
Step 10:	The Caller (Bob) acknowledges the SIP communication negotiation process by sending an ACK response.

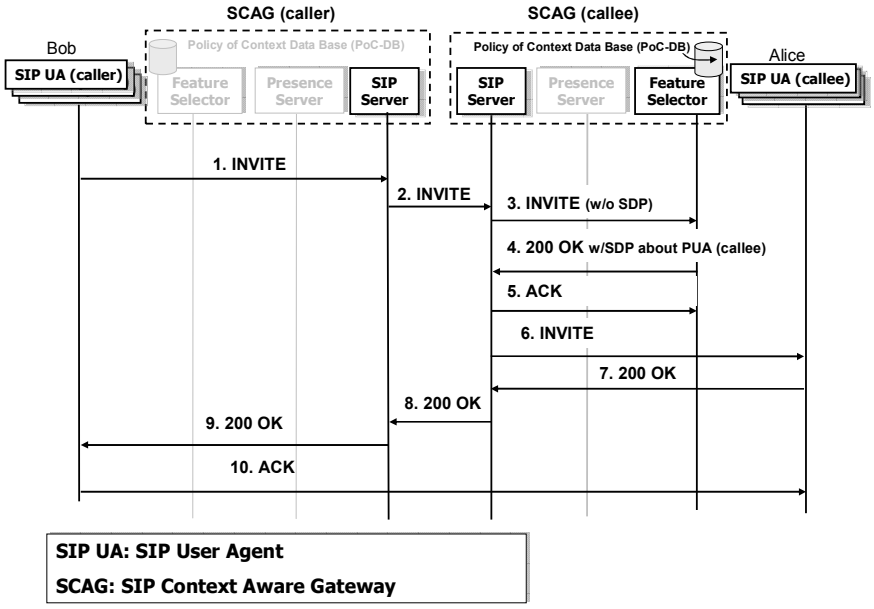

Figure 6 Incoming service request flow

4. Implementation and experimental results

Every network system is unique and has its own special capabilities. From the economic perspective, it is impossible to manufacture customized ASICs for each of them. Therefore, network processor (NP) becomes a cost-effective core technology to implement specialized network systems. NP technology leverages the advantages from both hardware and software solutions. NP is advantageous over ASIC-based solutions due to its programmable capability, which provides great design flexibility. On the other hand, NP is considered advantageous over the software-based solutions because it provides better performance in terms of mW/MIPS and packet processing speed (sometimes wired speed under specific conditions). NP delivers all the flavors with programmability, flexibility and performance as well as with the added benefits of low cost such that it helps network engineers intelligently and efficiently designing and implementing their network systems.

According to the Tolly Group report on the performance analysis for home gateways [15], Intel IXP425 based solutions outperform other processors consistently in terms of the throughput and latency. The emphasis of the evaluation conducted here is on the feasibility assessment of the practical IXP425 based implementation for the proposed SIP-CAG. The ADI's Coyote Gateway Reference Design Board (an Intel IXP425 based platform) is chosen as the implementation platform. The following paragraph describes the implementation of SIP-CAG on the Coyote reference platform, which is followed by the experimental results. The testing environment is set up to examine SIP-CAG performance. SIP traffic is generated by the Smart Bits 2000 (traffic coming out from the Tx port) and is fed to the WAN port of the SIP-CAG/Coyote target board. The network connections between the SmartBits and SIP-CAG are described as follows.

- The Transmitting port (Tx) on SmartBits is connected to the SIP-CAG/Coyote target board's WAN port.
- The Receiving port (Rx) on SmartBits is connected to the SIP-CAG/Coyote target board's LAN port.

The following two benchmarks are used for evaluating the performance of the SIP-CAG implemented on a Coyote target board.

- Benchmark for data traffic: To evaluate the performance for data traffic, the SIP-CAG functionality is disabled (denoted as "Disable SIP") to test the bridge function of the SIP-CAG/Coyote target board (data traffic conveys from WAN port to LAN port).
- Benchmark for signaling traffic: For signaling traffic, the SIP-CAG functionality on the Coyote target board is enabled (denoted as "Enable SIP").

For both benchmarks, packets of different sizes ranging from 64 to 1518 bytes are generated to test the performance of the SIP-CAG. The SIP-CAG performs all of necessary functions including (1) capturing the incoming INVITE requests, (2) extracting SIP fields (such as From, To, and Call-ID), (3) comparing extracted fields with the rules of the PoC and (4) forwarding the INVITE requests. Please note the performance of SIP-CAG may greatly depend on how fast the comparing tasks can be performed, which in turn depends on how many context policy entries are used in the PoC. In this research, three context policy entries are used for all tests.

Performance evaluation on throughput

Figure 7 presents the experimental result on throughput evaluation. The vertical axis represents the values of throughput (in Kbps) and these values are expressed in the logarithm scale. The horizontal axis indicates the packet size (in bytes). We observed that the maximum throughput of SIP-CAG is 4.99 (about 97 Mbps) and the minimum is 4.07 (11 Mbps) when SIP-CAG is disabled (Benchmark for data traffic). The maximum throughput of SIP-CAG is 3.2 (about 1640 Kbps) and the minimum is 2.5 (about 280 Kbps) when SIP-CAG is enabled (Benchmark for signaling traffic). Since SIP-CAG processes the application layer information, the string matching operation needs a lot of computation power and thus degrades the throughput. For a general traffic distribution pattern (90% data traffic and 10% signaling traffic), experimental throughput results of our implementation range from 10 Mbps to 96 Mbps that satisfies the throughput requirement for the current Internet access speed on a Home Network (such as ADSL and T1). Figure 7 also shows that the throughput increases proportionally to the packet size and approaches to the wire-speed (100Mbps) when packet size reaches 700 bytes.

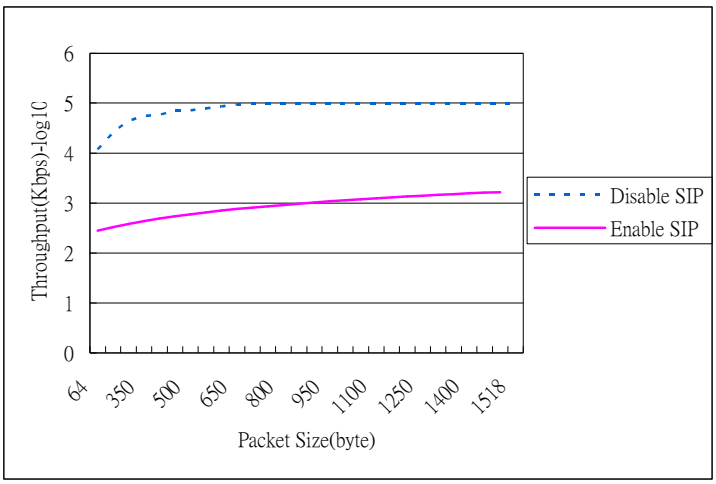

Figure 7 Throughput on different packet sizes

Performance evaluation on latency

Figure 8 presents the latency evaluation of the SIP-CAG. The vertical axis shows the latency value in the logarithm scale (in μsec) and the horizontal axis indicates the packet sizes (in bytes). We compare the packet delay of two systems. One system enables the SIP-CAG function to test the speed of signaling traffic (solid line), and the other system disables the SIP-CAG function to test the speed of data traffic (dotted line). When SIP-CAG functionality is disabled (for data traffic), the maximum and minimum latency tested are 355μs and 55μs, respectively. When the SIP-CAG functionality is enabled (for signaling traffic), the latency jumps up to about 45ms due to the increase of the computation time spent on pattern string matching. The time scale for setting up conversation is in seconds which is far larger than the observed experiment value resolved in millisecond scale. As such, the latency of SIP-CAG for signaling is acceptable in the practical usage. Experimental results for both benchmarks show that our implementation meets the latency requirement.

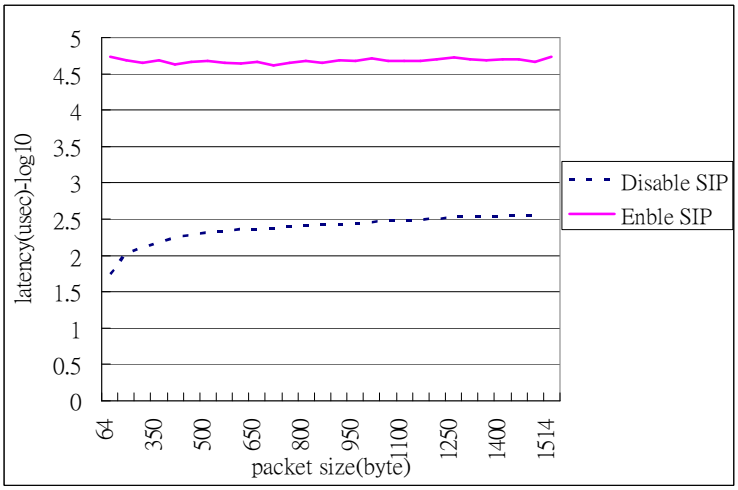

Figure 8 Latency on different packet sizes

5. Conclusions

In the ubiquitous computing era, smart home networks are expected to integrate home devices, services and resources compatibly and effectively. Context awareness is the key technology for adaptations on an "as needed" basis. Smart home applications running on such networks need to recognize and accommodate users' current contexts. With the power of the flexibility and the extensibility, SIP contains endless possibility for the smart home environment. In this paper, we proposed a SIP Context Aware Gateway (SIP-CAG) to provide an adaptive solution for the ubiquitous sip-based services in smart homes. SIP-CAG, deployed between home networks and the Internet, serves as a SIP-based home application secretary providing tremendous flexibility and capability in adapting to diverse environment situations according to the context of the inhabitants. A flexible rules language, Behavior Access Manuscript (BAM), was designed to describe the context policy. The prototype of SIP-CAG was implemented on an IXP425 network processor and the experimental results assess its implementation feasibility is very promising. This study has confirmed that SIP-CAG meets the context aware requirements in the smart home environment and the network processor technology provides a practical implementation solution for the proposed SIP-CAG.

6. Acknowledgements

This research was supported in part by the National Science Council (NSC) in Taiwan under the grant numbers TWISC@NCKU NSC 94-3114-P-006-001-Y and NSC-96-2221-E-194-034.

7. References

[1] J. E. Bardram, "Applications of Context Aware Computing in Hospital Work -- Examples and Design principles," ACM, SAC '04, March 14-17, 2004, Nicosia, Cyprus.

[2] B. Biggs, "A SIP Application Level Gateway for Network Address Translation," IETF Internet Draft <draft-biggs-sip-nat-00>

[3] B. Caswell and J. Beale, et al., "Snort 2.1 Intrusion Detection, 2nd Edition," Published by Syngress, ISBN 1931836043

[4] A. K. Dey, and G. D. Abowd, "Toward a better understanding of context and context-awareness," GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology. ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

[5] H. Cho, "Next Generation Growth Engines. A Korean Perspective," IMS NoE Conference on Advanced Manufacturing, Leuven, November 2005.

[6] S. Van Hoecke, K. Vlaeminck, F. De Turck, B. Dhoedt, "Open web services-based middleware for brokering of composed eHomeCare Services," IEEE MWS2005, Middleware for Web Services workshop, Enschede, Netherlands, September 2005, pp. 46-53.

[7] J. Rosenberg et al, "Best Current Practices for. Third Party Call Control (3pcc) in the Session. Initiation Protocol (SIP)," IETF RFC 3725, April 2004.

[8] J. Rosenberg, "A Presence Event Package for the Session Initiation Protocol (SIP)," IETF RFC 3856, August 2004.

[9] A. B. Roach, B. Campbell, and J. Rosenberg, "A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists," IETF RFC 4662, August 2006.

[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Hand-ley, E. Schooler, "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.

[11] IETF working group, "SIP for Instant Messaging and Presence Leveraging Extensions (simple)," http://www.ietf.org/html.charters/simple-charter.html.

[12] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, January 2001.

[13] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, J. Peterson, “Presence Information Data Format (PIDF),” IETF RFC 3863, 08/2004.

[14] Symantec, “Symantec Internet Security Threat Report,” http://www.symantec.com/ threatreport/

[15] Tolly Group, “Intel IXP425/IXP422/ IXP420 Network Processors Performance Analysis of 802.11 Broadband Routers and Access Points,” Tolly Group, June, 2004

[16] M. Weiser, “A Computer for the 21st Century,” Scientific American (265): pp. 94-104, 1991.

8. Appendix. BAM syntax

BAM=	*[action response 1*[protocol filter direction filter| SSID | IP_range] policy_list]
filter=	content ":" port | ":" port
action=	"accept" | "reject" | "drop" | "encrypt" | "update" | redirect | "update"
response=	"none" | "alert" | "log"
protocol=	"sip" | "tcp" | "udp" | "ip" | "icmp"
content=	enum | sip_url | string | hostnum
enum=	digits
sip_url=	user "@" host
string =	*[uchar]
SSID=	string
IP_range=	hostnum"/" digits
user=	*[uchar | ";" | "?" | "&" | "="]
uchar=	alpha | digit | safe | extra
safe=	"$" | "-" | "_" | "." | "+"
extra=	"!" | "*" | "'" | "(" | ")" | ","
host=	hostname | hostnum
hostname=	*[domainlabel "."] toplabel
toplabel=	alpha | alpha *[alphadigit | "-"]alphadigit
alphadigit=	alpha | digit
alpha=	lowalpha | hialpha
lowalpha=	"a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" |"k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
hialpha=	"A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
hostnum=	digits "." digits "." digits "." digits
port=	digits " : " digits | digits " : " | " : " digits
digit=	"0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
digits=	1*digit
direction=	< | " > " | " < > "
policy_list=	*[cpl_tuple | "next"]

cpl_tuple= status | equipment | location | manuscript
status= "online" | "offline" | userdefine
userdefine= 1*[1*[alphadigit] " "] 1*[alphadigit]
equipment= "handset" | "phone" | userdefine
location= "home" | "office" | userdefine
manuscript= timestamp | 1*[alphadigit]
domainlabel= alphadigit | alphadigit *[alphadigit | "-"] alphadigit
timestamp= times digits

Authors

Bo-Chao Cheng

Is an Assistant Professor of Department of Communication Engineering at National Chung-Cheng University. Cheng received a PhD degree in CIS from New Jersey Institute of Technology in 1996. After graduations, he also worked for Transtech Network (2000-2002), Bellcore (1998-2000) and Racal DataCom (1996-1998) respectively. His broad interests include network security, network management and real-time embedded system design.

Huan Chen

Received his B.S. and M.S. (Electrical Engineering) degrees from the National Tsing Hua University (NTHU), Hsing Chu, Taiwan, R.O.C. in 1993 and 1995, respectively. He received his Ph.D. (Electrical Engineering) degree from the University of Southern California (USC) in 2002. He is currently with the Electrical Engineering Department at the National Chung Cheng University, Chiayi (Taiwan). His principle research interests include QoS support, Ad Hoc routing and network optimization. Other research interests include ubiquitous computing and network security.

Ryh-Yuh Tseng

Received a B.S. degree in mathematics from Fu-Jen Catholic University, Taiwan, in 1987, and a M.S. degree in computer science at Utah State University, USA, in 1992. He is a lecturer of Department of Information Management at Wu-Feng Institute of Technology in Taiwan. He is also a PhD candidate of Department of Communication Engineering at National Chung-Cheng University. His research interests include Network Security, Computer Networking, Artificial Intelligence, Multimedia Communication systems.