A VoIP Emergency Services Architecture and Prototype

Matthew Mintz-Habib, Anshuman Rawat, Henning Schulzrinne, and Xiaotao Wu
Department of Computer Science
Columbia University
{mm2571,asr,hgs,xiaotaow}@cs.columbia.edu

***Abstract*— Providing emergency services in VoIP networks is vital to the success of VoIP. It not only presents design and implementation challenges, but also gives an opportunity to enhance the existing emergency call handling infrastructure. We propose an architecture to deliver emergency services in SIP-based VoIP networks, which can accommodate PSTN calls through PSTN to SIP gateways. Our architecture addresses the issues of identifying emergency calls, determining callers' locations, routing emergency calls to appropriate public safety access points (PSAP), and presenting required information to emergency call takers. We have developed a prototype implementation to prove our architecture's feasibility and scalability. We expect to undertake a pilot project at a working PSAP with our implementation once it is thoroughly tested.**

I. Introduction

VoIP telephony services are increasing in residential and enterprise communication market penetration due to their attractive service enhancements and cost savings. One feature from the traditional public switched telephone network (PSTN) that is essential for VoIP telephony is the ability to summon emergency services, such as by dialing "911" in the United States and "112" in parts of Europe. Transitioning to VoIP networks offers the opportunity to add significant enhancements to emergency call handling services, rather than simply duplicating the existing feature set. The enhancements include higher resilience, faster call setup, better information presentation, multimedia support, and lower costs. To achieve the enhancements, we designed an architecture and developed a prototype of our architecture that can provide emergency services in VoIP networks based on the Session Initiation Protocol (SIP) [1]. Our architecture can also accommodate PSTN calls bridged into VoIP networks through gateways. Even though our architecture is based on SIP, the same concepts and design principles can also be applied to other VoIP networks, such as H.323-based VoIP networks.

SIP is an application layer signaling protocol for initiating sessions between hosts to exchange media content. In SIP, sessions can be negotiated by SIP user agents (UAs) communicating directly with each other, or through a series of SIP servers, using SIP methods like INVITE, REGISTER, or BYE. Typically, SIP UAs are configured with an outbound proxy that forwards SIP messages on their behalf. SIP uses REGISTER requests to bind users' logical addresses to their physical addresses. This way, SIP can easily handle routing services, like the follow-me service, on an inbound SIP proxy server. Our architecture involves efforts on different SIP entities, including both caller and emergency call taker's user agents, and inbound and outbound SIP proxy servers.

SIP does not transport media content itself, but facilitates communicating parties to agree on what media to exchange and how to exchange it. Specifically, this is accomplished by using an offer/answer model with the Session Description Protocol (SDP) [2] as SIP message content. Note that SIP uses MIME [3] to format its content so we can put other information, such as location information, in addition to media description to form a multipart entity in SIP message content. Location information is essential for emergency call handling.

Proxy servers require emergency callers' location information to route calls to proper public safety answering points (PSAP), which are responsible for coordinating local or regional emergency services, because each PSAP is dedicated to a specific geographic area. Location information is also necessary for dispatching help to emergency callers. Since VoIP callers are nomadic, their location may not be readily apparent. Considering a user located in New York communicating through a SIP proxy in Hong Kong over a VPN tunnel. If the user were to request emergency services, the call should be routed to a call center in New York, not Hong Kong! Our architecture defines several methods to determine the location of VoIP callers.

Location information can be geographic coordinates, such as latitude, longitude, or altitude values, or civic location information, such as country, city, and street names. Civic location information needs to be general enough in an international context, since the Internet knows no national boundaries.

While there are currently no accepted standards on VoIP emergency services, related work can be found in several Internet Drafts addressing the subject [4], [5], [6]. The National Emergency Number Association's (NENA) [7], the organization promoting a universal emergency service number in the United States, recently published a list of requirements for IP enabled PSAPs [8]. Our prototype fulfills most of the requirements listed. Similarly, Arai and Kawanishi are pursuing VoIP emergency services requirements in Japan [9]. Within Columbia University, we have spent some time working on the VoIP emergency services problem [11], [12], [13], [14].

Our work brings together design features from various sources into one cohesive architecture, contributes novel design elements at the PSAP, and implements the system as a

prototype. We have demonstrated our prototype implementation at working PSAPs, for local and state authorities, as well as for the members of the NENA's Next Generation E9-1-1 committee. Our demo works very well, and we expect to undertake a pilot project at a working PSAP with our implementation once it is thoroughly tested.

The remainder of our paper is organized as follows. Section II describes our emergency call handling architecture. Section III discusses challenges implementing our prototype. Section IV provides system performance and security analysis. Section V concludes the paper and discusses future work.

II. Architecture

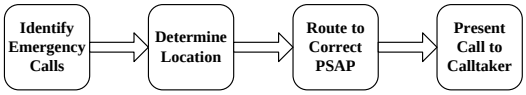

Fig. 1. Control flow for emergency call handling

Emergency call handling can be divided into four steps that are executed in sequence for each emergency call (Fig. 1). Each step involves one or more entities in the system architecture as shown in Fig. 2. The first step identifies emergency calls. For outgoing calls, the caller's user agent and outbound proxy server are responsible to check whether the call is an emergency call or not. Once an emergency call is identified, the second step determines the caller's location, and integrates the location information into call setup messages. The third step finds an appropriate PSAP based on the location information. A proxy server can then route the emergency call to the PSAP. The fourth step presents the emergency call to the emergency call taker at the PSAP. The emergency call taker utilizes the information in the call setup messages to handle the emergency call, such as pinpoint the caller on a map and bring police, fire, and medical supports into a conference call. At any point, a SIP entity may query third party services for information, such as caller location or medical records. We discuss each component in detail below.

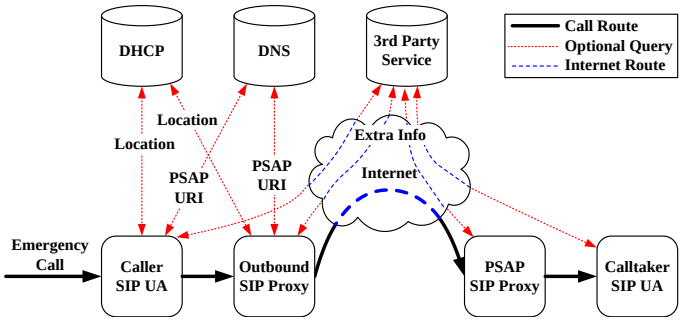

Fig. 2. Emergency services system architecture

A. Identifying emergency calls

Emergency calls are identified by their destination URIs and the location of the caller. Work is in progress to standardize the use of "`sos`" as the username part of a SIP URI [5] to represent an emergency call. Telephone URIs [15] for conventional emergency numbers, such as "`tel:911`", can be aliased to the emergency URI "`sos`", either by a SIP proxy or a SIP UA, based on the location of the caller.

B. Determining location

There are several ways to determine a calling party's location, either by the calling UA's outbound proxy or by the calling UA itself. The outbound proxy can determine the caller's location based on the calling UA's MAC address. In enterprise networks, the location of ethernet jacks and desktop machines, as well as the MAC addresses of the desktop machines are usually stored by system administrators. The outbound proxy can determine the location of the emergency caller simply by sending a `DHCPINFOM` query with the MAC address retrieved from the packets it received.

The calling UA can determine its own location directly, such as from a GPS receiver, a bluetooth beacon, DHCP options, or manually entered by a user. It can also get location information from a location server. For example, through triangulation calculation, multiple wireless access points can pinpoint a mobile station's location and store the location on a location server. The mobile station can subscribe to its own location from the location server by using SIP event notification architecture [16].

C. Routing emergency calls

Different location information require different techniques to determine appropriate PSAPs to route emergency calls to. Location information in an emergency call can be civic location, geographic coordinates, or no location. We use DNS Naming Authority Pointer Resource Records (NAPTR) [17] to find appropriate PSAPs.

To determine the correct PSAP for calls with civic location, the caller's location elements can be transformed into a period-separated form hierarchically from most granular to least granular location element. The corresponding NAPTR record has the service type "`SOS+ECC`", indicating that it represents an emergency call center's URI. DNS is first queried for the most granular location, and if no match is found, successive layers of granularity are stripped and queried until a match is found. Each location entry is suffixed with `sos-arpa.net` as the top level of the hierarchy, thus ensuring a default match if no better match is found. Upon success, a NAPTR record is returned with the emergency URI [6]. The example below shows the DNS record for the location "`Houston, TX`".

```
houston.tx.us.sos-arpa.net IN NAPTR 50 50 "u" "SOS+ECC"
"/.*/sip:houston_tx@emergency.info/i" .
```

We can also use DNS to determine the proper PSAP URI based on geographic coordinates, but with a different service type: "`SOS+POLYGON`" [6]. The result of the DNS query will be a pointer to an XML document defining a specific geo-political boundary, such as a state, county, or PSAP coverage area. These boundaries are unlikely to change often, so the

DNS record can be set with a large TTL value and the returned boundary information can be cached.

```
tx.us.sos-arpa.net IN NAPTR 50 50 "u" "SOS+POLYGON"
"/.*/http:\/\/www.emergency.info\/polygons\/texas.xml/i" .
```

The example above shows the record pointing to the XML document defining the polygon boundary for the state of Texas (special characters escaped). A proxy server can search the "SOS+POLYGON" records from least granular to most granular, linearly, to check whether a polygon contains the caller's geographic coordinates or not. Once the most granular match is found, the corresponding URI found in the "SOS+ECC" record is returned for emergency call routing.

Emergency calls that contain no location can be routed to a default PSAP URI. This URI can be determined by the outbound SIP proxy server of the calls. The proxy server queries DNS for the PSAP URI based on its own location. The default PSAP URI can also be stored as a configuration parameter in the SIP proxy.

D. Call presentation at the PSAP

A general feature list for presenting emergency calls to emergency call takers has been defined by NENA [18], which also documents features specifically for IP-enabled PSAPs [8]. PSAPs need to display caller locations on a map, automatically distribute incoming calls to available call takers, log emergency call details to database, archive call media content, view call logs and generate statistics, and monitor currently active calls. We have achieved these requirements in our prototype implementation, which we will discuss in detail below.

III. Implementation

We implemented a prototype based on the architecture defined above. To place VoIP calls, we use the Columbia SIP User Agent (SIPC) [19], as well as hardware UAs, like the Cisco 7960 [20] SIP phone. Location can be entered manually into SIPC, automatically looked up using host-specific DHCP options, received from GPS receivers, acquired from a location server through SIP event notifications, or queried through MapInfo's Envinsa [21] platform.

For call routing, we use the Columbia InterNet Extensible Multimedia Architecture (CINEMA) [10] architecture for SIP services, and use SIP-CGI [22] Perl scripts to make routing decisions. PSAP identification is accomplished by using MapInfo's Envinsa service or DNS-based lookups.

To present caller information to call takers, we use GeoComm's GeoLynx Dispatch Mapping System [23] to display caller location on a map and have SIPC interface with GeoLynx through TCP connections. Also at the PSAP level, we created a system to distribute calls among multiple call takers, enabled conferencing of multiple parties using CINEMA or the Brooktrout Technology's Snowshore Media Server [24], enhanced SIPC to log call details, and created a web-based system to manage PSAP end-systems.

A. Identifying emergency calls and determining locations

Identifying emergency calls was straightforward to implement simply by identifying calls addressed to "SOS" as emergency calls. We also aliased the URIs "911" and "112" to the emergency URI for ease of use. To speed up the dialing process in an emergency, SIPC has an SOS button to quickly make emergency calls.

The more challenging aspect was determining caller location. Our prototype uses manual location entry in SIPC, though it is also capable of utilizing GPS measurement and acquiring location information from a location server. Many SIP UAs may not support manual location entry, or cannot measure or lookup their location. To accommodate such UAs, we implemented an automatic lookup feature at the outbound SIP proxy as described in Section II-B based on DHCPINFOM queries for the caller's MAC address. This ensures that every SIP UA can participate in emergency services without modification to the UA. We leave the issue of MAC address availability for calls passing through layer 3 devices as future work, though it may simply be included as a SIP header.

Complementing the DHCP lookup, the proxy can find locations for calls originating from a PSTN-to-VoIP gateway by querying the source telephone number in MapInfo's Envinsa server over an HTTP SOAP interface. This allows us to find the GPS location of cellular phones from a set of demonstration units.

Once the proxy gets the location for an incoming SIP INVITE request, it can encode the location in presence-based GEOPRIV location object format [25], and incorporate the encoded document into the message body of the INVITE request to form a multipart message body in MIME format.

These features are diagrammed in Fig. 3, which also shows the logical information flow. A user optionally enters location information into SIPC manually (1a), then makes an emergency call (1b). The call is sent to the outbound proxy (2), and may or may not include location information, depending if the location was entered manually. The outbound proxy receives the emergency call, and launches a SIP-CGI script. If necessary, the script looks up the location, either using DHCP for local callers (3a), or using MapInfo's Envinsa service for calls brought in over an IP gateway (3b). In either case, the script returns the location information (4a,4b), and further processing for routing decisions ensues.

B. Routing emergency calls

We have described the DNS-based routing strategy in Section II-C. Complimentary to the DNS lookups, we also utilize MapInfo's Envinsa platform to look up PSAP information for geographic locations. We use SIP-CGI scripts running on users' outbound proxy servers to handle emergency call routing. We allow proxy server administrators to choose Envinsa or DNS for geographic lookups by configuring the SIP-CGI scripts. If SIPC knows both civic and geographic location information, it will send both in its outgoing INVITE requests. In that case, the outbound proxy will check civic location first.

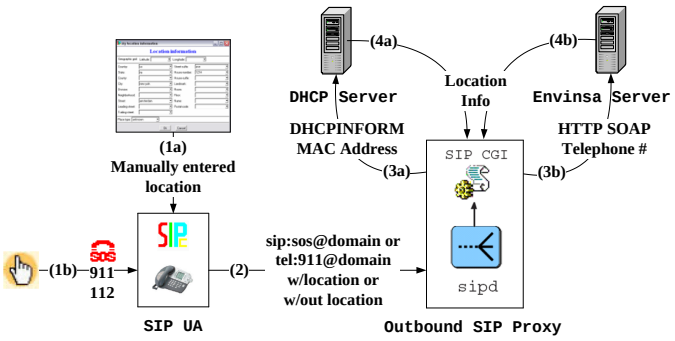

Fig. 3. Identifying emergency calls and determining location

Fig. 4 shows the overview of these features. The proxy receives an emergency call (1). If no location is available, the proxy attempts to determine the location as described in Section III-A. If location is still not available, the proxy simply routes the call to a default PSAP URI (4). If the proxy receives geographic coordinates, it will either query MapInfo's Envinsa server (2a) or DNS for PSAP boundary information (2b), depending on how the administrator of the proxy server configured the SIP-CGI script. If a civic address is received, the system queries DNS (2c) for the PSAP URI (3c). Once the SIP-CGI script get the PSAP URI (3a,3b), it will proxy the the call to the PSAP URI (4) along with the location information.

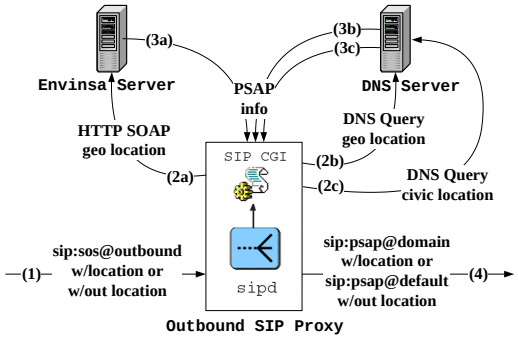

Fig. 4. Routing emergency calls

C. Call presentation at the PSAP

Once an emergency call reaches an appropriate PSAP, the PSAP UA will display the location details graphically using GeoComm's GeoLynx dispatch mapping system. When the call taker ends the session, locations are cleared from the GeoLynx display. We use SIPC as the PSAP UA, which uses TCP sockets to communicate with GeoComm's GeoLynx system. SIPC also has a button allowing emergency call takers to manually refresh location information for mobile stations using MapInfo's Envinsa platform.

SIPC has an interface to classify calls, log additional details and notes, and speed dial buttons to request police, fire, or medical support. SIPC can also transfer calls to another PSAP. For PSAPs using SIP hardphones, we implemented a mechanism for the Cisco 7960 series SIP phones to display location information, which is encoded in XML format, and retrieved via HTTP.

We use an automated controller system at the PSAP to handle all calls. The controller is responsible for distributing incoming calls to available call takers and logging the details of each call. In our prototype system, we treat every call as a conference call to allow multiple parties, including emergency call takers, police, fire, and medical support, to participate in the conference call. We have integrated two conference modules, one is CINEMA's conference server, SIPCONF, and the other is Brooktrout's Snowshore media server, both of which can be used interchangeably. The controller is responsible for managing and logging these conferences as well. Logging is performed at the earliest opportunity to provide accountability for incomplete calls.

To assist in the management of the PSAP components, we created a web interface to browse and search call logs, view call statistics, view and join active calls, update incident types, and manage associated DNS records.

Fig. 5 shows the general PSAP architecture and logical information flow. The controller receives an incoming call (1), starts logging the details (2), then creates a conference for the call (3). The controller then selects among the available call takers (4), who joins the conference in turn (5). At this point, the caller is connected to a call taker. The call taker may choose to update the caller's location information from the Envinsa Server (6a), which is then displayed in GeoLynx (7a). If necessary, the call taker may conference in additional parties such as police (6b,7b). These actions are logged (6c), and the call taker is able to log additional notes and classify the call (6c). The web management system uses the information in the datastore to generate its pages.

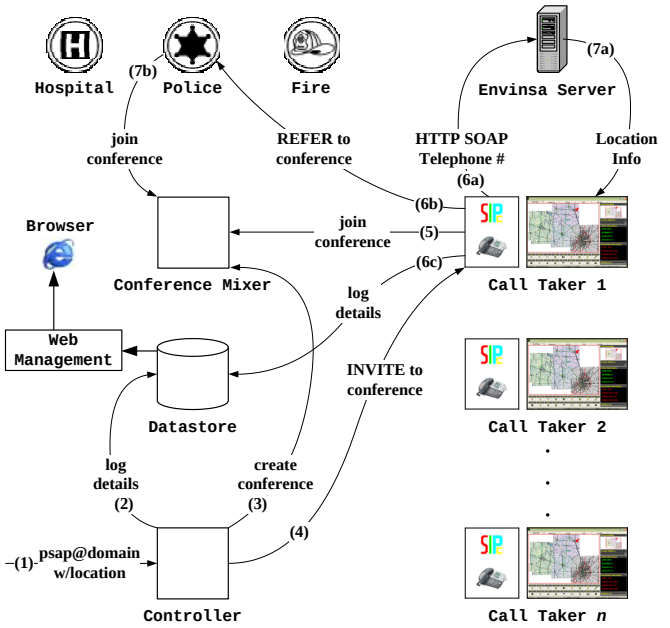

Fig. 5. PSAP architecture and logical information flow

As shown in Fig. 6, the controller uses the SIP third party

call control architecture [26] to bring call takers and additional third parties in to a conference call. The process is completely transparent to participating parties. To begin, the caller initiates a SIP call, which includes location, if available. When the controller at the PSAP receives the call, it sends an INVITE to the conference server along with the caller's SDP, SDP1 (2), but without location information because the conference server is only responsible for media mixing. The conference accepts the INVITE, and sends a 200 OK with SDP1' as its message content. The controller forwards SDP1' on to the caller (4), then the controller and the caller both ACK the 200 OKs each received, respectively (5)(6). At this point, the caller is able to talk to the conference server (7).

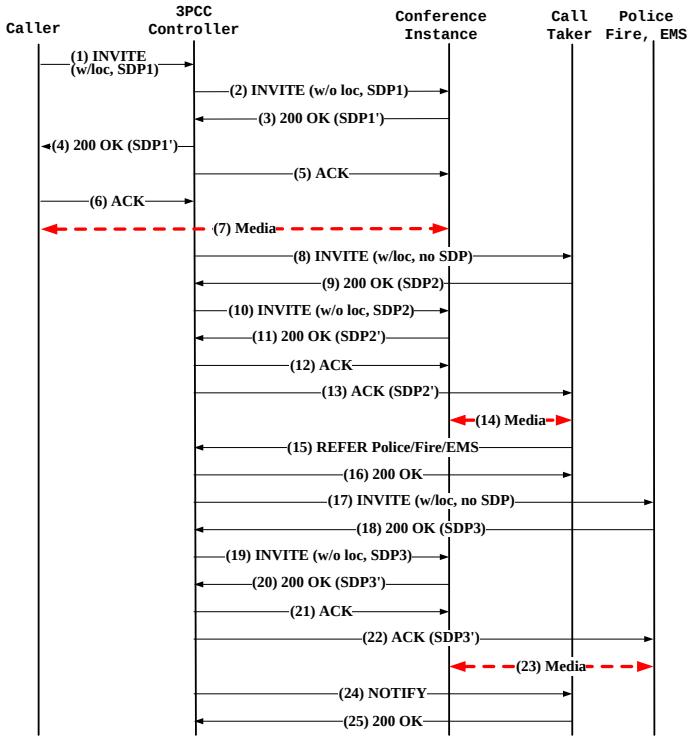

Fig. 6. Third party call control message flow

The next step is to bring a call taker into the conference to handle the emergency call. In this case, the controller sends an INVITE without an SDP body (8) so that the call taker's UA can negotiate its own media with the conference. Note that the emergency caller's location is included in the INVITE so that the call taker can immediately display the caller's location. The call taker's UA replies with a 200 OK to the INVITE and offers SDP2 (9). This offer is forwarded to the conference server in an INVITE to bring the call taker into the conference (10). The conference server accepts the INVITE and sends a 200 OK with SDP2' (11). The controller sends an ACK to the conference server (12), then puts SDP2' in its ACK to the call taker (13). Now the call taker can also talk to the conference server (14). With both the caller and call taker in the same conference, they can communicate with each other.

As is commonly the case, the call taker may want to bring in additional third parties' assistance, such as police or fire departments. We use the SIP REFER method [27] to handle this on our PSAP UA, SIPc. SIPc has speed dial buttons to bring additional parties in. Instead of sending REFER requests directly to a third party, SIPc sends the REFER requests to the controller, and has the controller to bring the third party into the conference. This way, the third party user agent does not have to support the SIP REFER method. As shown in the diagram, the call taker initiates the request to bring in a third party by sending a REFER message to the controller (15), who responds with a 200 OK (16), indicating that it is ready and able to process the request. From here, steps (17)-(23) are identical to steps (8)-(14) to bring the third party into the conference. The controller then sends a NOTIFY to the call taker (24) to update the status of the REFER request, to which the call taker responds 200 OK (25). All three parties can now communicate with each other.

IV. Performance and Security

Our prototype system has not yet undergone a comprehensive performance evaluation. The main concerns are system throughput and the latency. We define throughput as the number of emergency calls that can be handled per second, and latency as the time elapsed between emergency call initiation and the time the emergency call taker joins the call.

The throughput can be considered at the proxy level and the PSAP level. At the PSAP level, the number of simultaneous calls is most likely bounded by the number of call takers. At the proxy level, the throughput is determined by the number of requests a SIP proxy can handle. Our empirical tests have shown the CINEMA SIPd proxy running on very moderate hardware (500 MHz CPU, 128 MB RAM) capable of supporting 86 proxy requests per second [28]. More recent work shows a stateful CINEMA load-sharing architecture with failover support running on contemporary hardware (3 GHz CPU, 1 GB RAM) capable of supporting 800 calls per second. NENA estimates about 200 million 911 calls in the United States per year, or roughly 6.3 calls per second nationwide on average [7], though some emergencies may elicit a burst of calls. While the CINEMA performance evaluations did not consider SIP-CGI execution and traffic may be bursty, it is unlikely that the SIP proxy will be a bottleneck.

Latency will be a bigger concern. Many factors contribute to call setup latency, such as UA processing, network conditions, SIP proxy processing, and call distribution at the PSAP. In our prototype, much of the delay is incurred by SIP-CGI scripts waiting for queries executed on remote machines. For instance, tests on our local network show that emergency calls sent without location and routed to the default PSAP take an average of 0.57 seconds. This can be seen as a lower limit for emergency call latency in our prototype. However, calls sent with geographic location that is queried in MapInfo's Envinsa service take 1.70 seconds on average. The exact modules invoked at script run dictate the latency characteristics incurred at the SIP proxy during call setup. We will study both latency and throughput in our system as a future work.

Security considerations for our prototype implementation are less imperative than in a live, public system. Accordingly, we did not build explicit security features into our prototype. In a public system, there are some enhancements that could be added. To prevent PSAP impersonation by manipulating DNS entries, secure DNS could be used. To protect signaling integrity and media integrity and confidentiality, calls could be routed using TLS and exchange media using SRTP. Other considerations include the security involved in querying external services, such as MapInfo's Envinsa platform.

V. Conclusion

We have presented an architecture for providing emergency services in VoIP networks. Our design is based on end-to-end IP connectivity and facilitates PSTN calls bridged into the network over IP gateways. The system addresses the issues of identifying emergency calls, determining location, routing to the appropriate PSAP, and presenting the emergency call to the call taker.

The architecture was implemented into a prototype system based on Columbia University's SIP infrastructure consisting of the CINEMA platform and SIPc, as well as with components provided by MapInfo and GeoComm. We developed several software solutions to provide enhanced functionality to call takers at PSAPs, as well as provided a web-based system to manage aspects of the system.

There are many areas we are looking to explore in our prototype system. These can be grouped into the addition of new features at the PSAP, enhancements to the call delivery architecture, and performance evaluation.

At the PSAP level, we are interested in adding advanced multimedia functionality, such as playing back instructional video to callers, e.g., a CPR how-to. Another item is to implement media archiving and incorporate retrieval via the web management system. The call conference mixers we use have the ability to record audio, but no additional media types. One solution is to have an automated robot that retrieves and archives media streams join each conference call. One more useful feature we will implement is the ability to call back abandoned or disconnected emergency calls. While SIPc is capable of calling a disconnected call back directly, we currently do not support conferencing and logging of these actions.

In the call delivery architecture, we intend to add redundancy and failover features to enhance the system's robustness as described by Singh [10]. Another item is to add backup PSAP support so that if a particular PSAP's resources are occupied, incoming calls are redirected to a backup call center.

Also, we intend to conduct a comprehensive performance evaluation of the prototype system. This would empirically study both throughput and latency metrics at the system and component level.

Acknowledgments

The work described in this paper was supported in part under a grant by SIPquest and a Technology Opportunities Program (TOP) grant by the National Telecommunications and Information Administration (NTIA). The authors would also like to acknowledge Amrita Rajagopal for her contribution implementing the DNS-based XML boundary graph lookup.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," RFC 3261, June 2002.

[2] J. Rosenberg and H. Schulzrinne, "An offer/answer model with the session description protocol (SDP)," RFC 3264, June 2002.

[3] N. Freed and N. Borenstein, "Multipurpose Internet mail extensions (MIME) part one: Format of Internet message bodies," RFC 2045, Nov. 1996.

[4] H. Schulzrinne and B. Rosen, "Emergency services for Internet telephony systems," draft-schulzrinne-sipping-emergency-arch-01, Internet Draft, July 2004, work in progress.

[5] H. Schulzrinne, "Emergency services URI for the session initiation protocol," draft-ietf-sipping-sos-01, Internet Draft, Feb. 2004, work in progress.

[6] B. Rosen, "Emergency call information in the domain name system," draft-rosen-dns-sos-01, Internet Draft, July 2004, work in progress.

[7] NENA, National Emergency Numbers Association. [Online]. Available: http://www.nena.org

[8] *NENA IP Capable PSAP Features And Capabilities Standard*, NENA Std. 58-001, Feb. 2005.

[9] H. Arai and M. Kawanishi, "Emergency call requirements for IP telephony services in japan," draft-arai-ecrit-japan-req-00, Internet Draft, Feb. 2005, work in progress.

[10] CINEMA, Columbia InterNet Extensible Multimedia Architecture. [Online]. Available: http://www.cs.columbia.edu/IRT/cinema/

[11] J. Lee, K. Singh, and H. Schulzrinne. SIP 911 implementation. [Online]. Available: http://www.cs.columbia.edu/~kns10/projects/spring2002/911/

[12] H. Schulzrinne and K. Arabshian, "Providing emergency services in Internet telephony," *IEEE Internet Computing*, vol. 6, no. 3, pp. 39–47, May/June 2002.

[13] X. Wu and H. Schulzrinne, "SIPc, a multi-function SIP user agent," in *IFIP/IEEE International Conference, Management of Multimedia Networks and Services (MMNS'04)*, San Diego, CA, Oct. 2004, pp. 269–281.

[14] ——, "Location-based services in Internet telephony," in *IEEE Consumer Communications & Networking Conference (CCNC'05)*, Las Vegas, NV, Jan. 2005.

[15] H. Schulzrinne, "The tel URI for telephone numbers," RFC 3966, Dec. 2004.

[16] A. B. Roach, "Session initiation protocol (SIP)-specific event notification," RFC 3265, June 2002.

[17] M. Mealling and R. W. Daniel, "The naming authority pointer (NAPTR) DNS resource record," RFC 2915, Sept. 2000.

[18] *NENA Generic E9-1-1 Requirements Technical Information Document*, NENA TID 08-502, July 2004.

[19] sipc, Columbia SIP User Agent. [Online]. Available: http://www1.cs.columbia.edu/~xiaotaow/sipc/

[20] Cisco Systems. VoIP phones. [Online]. Available: http://www.cisco.com

[21] MapInfo Corporation. Envinsa Location Platform. [Online]. Available: http://www.mapinfo.com

[22] J. Lennox, H. Schulzrinne, and J. Rosenberg, "Common gateway interface for SIP," RFC 3050, Jan. 2001.

[23] GeoComm Corporation. GeoLynx Dispatch Mapping System. [Online]. Available: http://www.geo-comm.com

[24] Brooktrout Technology. Snowshore Media Server. [Online]. Available: http://www.brooktrout.com

[25] J. Peterson, "A presence-based GEOPRIV location object format," draft-ietf-geopriv-pidf-lo-03, Internet Draft, Sept. 2004, work in progress.

[26] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo, "Best current practices for third party call control (3pcc) in the session initiation protocol (SIP)," RFC 3725, Apr. 2004.

[27] R. Sparks, "The session initiation protocol (SIP) refer method," RFC 3515, Apr. 2003.

[28] J. Lennox, "Services for Internet telephony," Ph.D. dissertation, Department of Computer Science, Columbia University, New York, New York, 2004, pp.113-117.