TOWARDS PROGRAMMABLE CONTEXT-AWARE VOICE SERVICES

Kerry Jean, Nikolaos Vardalachos, Alex Galis
Department of Electronic Engineering, University College London,
Torrington Place, London, WC1E 7JE, U.K.; Tel: +44-20-7679-5752
{kjean, nvardala, agalis@ee.ucl.ac.uk

Abstract. Programmable context-ware services use context information and programmable networks technology in the provision of easily customised and personalised services, which can respond appropriately to changes in their environment. This paper presents one such service, which is used to enable the provision of VoIP services in crisis situations. This service, the context-aware VoIP (CaVoIP) service is built upon the CONTEXT platform, an innovative middleware designed for the creation, deployment and management of context-aware services. The platform consists of a programmable layer, a context-aware service engine and a policy-based service layer. The voice services in the CaVoIP service are provided by a session initiation protocol (SIP) platform called Siptrex. The result is an easily customised, flexible and scalable context-ware service, which suppresses, non-essential traffic during crisis situations allowing greater bandwidth for essential traffic.

Keywords: programmable services, programmable networks, context, and context-aware services

1 Introduction

Context consists of the implicit and explicit information of an entity, be it an application, network or service, which can be used to characterise it. This context information can be used to enhance a service or application, personalising it, enriching it or making it more responsive to changes in its environment or situation [1]. When a service makes use of its context, it becomes context-aware. Context-awareness is characterised by the ability of a service or system to react to its changing environment. Context-aware services can be developed through the creation of a context infrastructure on top of a programmable network [3][6]. This paper presents a programmable context-aware service designed to efficiently use network resources to cope with the huge increase in voice traffic during crisis situations.

This research was carried out as part of the an EU funded project called CONTEXT [3][4], which created the CONTEXT platform. This platform made use of programmable networks and policy-based service management (PBSM) for the efficient creation, deployment and management of context-aware services. The CONTEXT project created a context-aware VoIP (CaVoIP) service where context and programmable networks are used to enhance a VoIP service enabling it to react appropriately to crisis situations such as terrorist attacks or major disasters. During a crisis there is a huge strain on a network due to the great increase in voice traffic, both essential from the emergency services, doctors, hospitals) and non-essential (from general public). Without proper management, there is less bandwidth available for essential traffic (resulting in a degradation of the quality of the service. The aim of the CaVoIP service is to suppress non-essential traffic and only allow traffic from priority users to be carried on the network. This service would monitor for different crisis conditions and enable the network to respond appropriately. In the CaVoIP service a session initiation protocol (SIP) [2] VoIP platform called Siptrex [5] is used to provide voice services.

The next section, the background, provides an overview of the technologies used to create the service namely programmable networks, context and context-awareness. Then the service specifications, realisation, components and interactions of the CaVoIP service are detailed. A service scenario and evaluation is then presented. This paper ends with the conclusions.

2 Background

2.1 Context and Context-awareness

Context is defined as any information that can be used to characterise the situation of an entity, where an entity can be a person, place, physical or computational object. Typical examples of context information are: (1) user location information (e.g. outdoors/indoors, street, city, etc.); (2) social context (e.g. role –student/staff/faculty; wife; boss; colleague, etc.); (3) personal preferences (e.g. food preferences, favourite sports, etc.); (4) user's behaviour (e.g. task, habits); (5) device and network characteristics (e.g. network elements, bandwidth).

A system is context-aware if it uses context to provide relevant information and/or services to the user. The user here can be a human end user or an-

other application, service or system. Context-awareness enables a new class of computing and network services [3]. These services can be easily personalised to help users find nearby services or devices, decide the best devices to use, receive messages in the most useful and least intrusive manner, and can enable systems to react appropriately to certain situations etc. As the CaVoIP service is context-aware, it uses network information to respond appropriately to a crisis and can be easily personalised.

2.2 Programmable/Active Networks

Traditional networks passively transport data packets from one host to another via routers. Each node performs only the processing necessary to forward packets towards their destination. In programmable or active networks, the packets contain code, which can be executed by active routers. Active or programmable routers are network nodes that execute the code contained in an active packet [10]. This code can be used to make the network nodes more intelligent and programmable. Active network architectures enable a massive increase in the complexity and customisation of the computation that is performed within the network [14].

There exist two main approaches to realise active networks: *programmable nodes* and *encapsulation.* In the first approach, the user injects programs into the programmable node separately from the actual packet using existing network packet formats and providing a discrete mechanism for downloading programs to the active nodes. The program is executed when data packets associated with it arrive at the node [10] [13].

In contrast, in the *capsule* approach, a program is integrated into every packet. Encapsulation replaces existing packet structures with programs that are encapsulated within the transmission frames. In this approach, the active node has a built-in mechanism to load the encapsulated code, an execution environment to execute the code and semi-permanent storage where capsules can retrieve or store information [13].

3 The Context-aware VoIP Service

3.1 CaVoIP Service Specification

The CaVoIP service is a programmable network service which uses context-awareness and PBSM to deal with crisis situations in VoIP networks. These

situations cause an upsurge in network traffic. This upsurge has to be managed carefully to ensure that the quality of the essential traffic (from emergency services, police, hospitals, government etc) is not compromised. The solution is to terminate all non-essential traffic when a crisis occurs and from then on only allow essential traffic. The CaVoIP service is built on top of the CONTEXT platform, a programmable, policy-based network platform. The CONTEXT service creation subsystem creates the service specific code and policies for the CaVoIP service. The network is monitored for a crisis using context-aware service code. The PBSM along with the programmable network is used to deploy and manage the service. To fulfil these objectives, the service needs:

- A means of obtaining and analysing context information
- A means of interacting with the underlying programmable network
- A means of interacting with the PBSM used to manage the service.
- A means of accessing the Siptrex platform.

3.2 CaVoIP Service Realisation

The CaVoIP service works as follows. A context computational object (CCO) monitors the network, computes context from local information and publishes it to the Context broker. The Context broker is a programmable network interface which allows the CaVoIP service to interact with the programmable network and obtain context information. A service execution condition evaluator (SICE) subscribes to this information and determines when a crisis has occurred. A SICE monitors a particular condition to determine when a service should start. A crisis is defined as the moment when calls made to the emergency number (911) have passed a predefined threshold in a fixed period of time. When a crisis occurs the SICE informs the PBSM and certain execution policies are invoked resulting in the deployment of a service level object (SLO) to the nodes in the crisis area. A SLO is a programmable application that provides a context-aware service. This SLO begins executing by terminating all non-essential calls to and from the crisis area. It then takes over call admission control from the SIP servers and only allows privileged callers access to the network. When the crisis is over the SLO stops executing, relinquishes call admission control and the CaVoIP service reverts to its default state.

The CaVoIP service is realised through the service code (the SICE, SLO and CCO), the Siptrex and CONTEXT platforms and the SIP and Context brokers. The CONTEXT platform is used to create, deploy and manage context-aware services. The brokers serve as interfaces between the programmable network and the service code. They allow programmable entities access to the network and context information and enable them to perform network reconfigurations needed to implement services. The user agents and SIP (session initiation protocol) servers of the Siptrex platform were modified to interface with the SIP broker and the service code. The Siptrex and CONTEXT platforms, brokers and service code are all described below.

3.2.1 Context Platform

The CONTEXT project [4] designed and developed an innovative platform and middleware solution to efficiently provide context-aware services making use of programmable networks technology and PBSM.

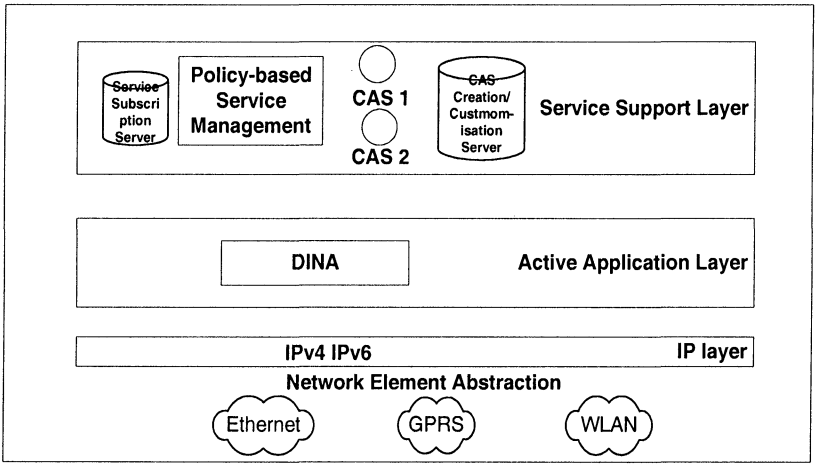

Figure 1: CONTEXT framework architecture

Figure 1 illustrates the high level architecture of the CONTEXT platform [3]. It consists of a distributed service execution environment (EE) on top of a transport layer. The EE is composed of two layers, the service support layer (SSL) and the programmable application layer AAL. The features provided by these two layers are applied throughout the service creation, deployment, and operation phases of the service lifecycle.

The AAL provides the programmable network functionality to the higher layers. A programmable platform, DINA, was developed based on concepts used in ABLE [7][8][9]. DINA is a programmable middleware, which can be attached to different types of routers to make them programmable routers [10]. It enables the deployment of programmable services on network nodes. The APIs of the DINA platform allow the service code access to local, network, and context information, allowing it to perform actions (such as network level configurations) as needed. The ABLE platform was chosen, as it was lightweight, scalable and easily extensible. ABLE's extensibility and scalability results from the use of brokers, software components which enable access to all sorts of services and technologies e.g. SIP, context, QoS, GPRS etc. The creation of DINA was achieved by rewriting much of the ABLE C code in Java (to enable interoperability). During this process security was improved and support for context-aware services and IPv6 were provided.

The SSL consists of two main subsystems, the context-aware service (CAS) creation and customisation subsystem and the PBSM subsystem. The former enables context-aware service creation and customisation and also contains a service subscription server. The PBSM contains the policy management infrastructure. This consists of the policy manager, the code and policy repository as well as the code distributor and the code execution controller. Policies are used for service creation, deployment and operation.

The CONTEXT solution is characterised by three phases:

- *Service Creation and Customisation Phase:* In this phase, the behaviour of a context-aware service is defined based on the capabilities of the AAL. This behaviour is modelled as a set of policy rules in XML. Based on that model and the capabilities of the AAL, the service code and policies are then generated using a code generator. The generated service code and policies are then customised according to the customer's specifications.
- *Service Deployment Phase*: The code and policies are stored in code and policy repositories respectfully. According to the code distribution policies for the CaVoIP service, the customised code and policies are distributed throughout the network to the code storage and execution points (DINA nodes).

- *Service Operation Phase:* After the code has been distributed to the execution points, the service awaits triggers, as defined by the respective code execution policies, for code execution to begin. In the Ca-VoIP service, the trigger to start executing the service is when a crisis has occurred. The trigger or event is raised by a service invocation condition evaluator (SICE) when it detects the context conditions identifying a crisis.

3.2.2 DINA Components

DINA is a modular and scalable programmable network platform that enables the deployment, control, and management of programmable services over networks entities such as routers, WLAN access points, media gateways, and servers in IP-based networks. In addition, DINA provides interfaces (brokers) that can be used by the programmable services to retrieve information and perform configuration operations on local nodes. Figure 2 below presents the main DINA platform components.

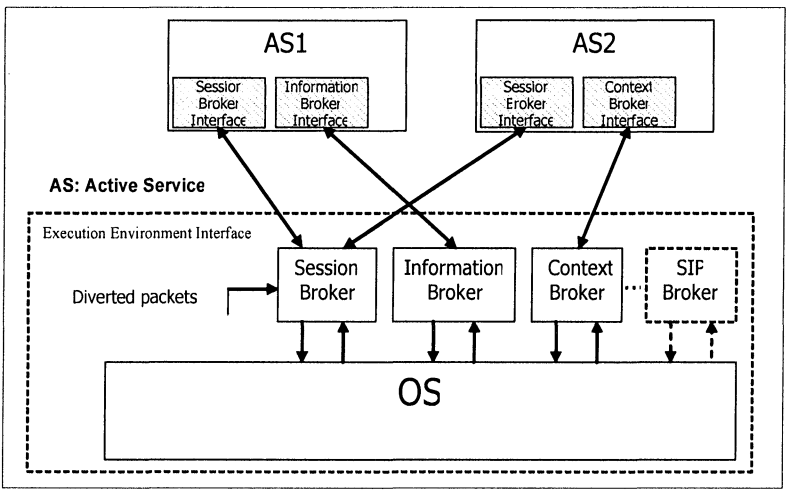

Figure 2: The DINA platform components

The DINA platform consists of two main components, the Session broker and the Diverter that run in parallel on each programmable node. The Session Broker is the core of the DINA platform. It receives and parses programmable packets, handles and manages existing services, and distributes

programmable packets according to service requests. The Diverter receives the programmable packets captured by the programmable node and forwards them to the Session broker. In addition to these two components, other components, called brokers, run in parallel and provide enhanced services to the programmable sessions. These services extend the DINA platform allowing it to integrate with other technologies and services e.g. SIP, QoS, context etc. The following are two DINA components created for use in the CaVoIP service.

3.2.2.1 SIP Broker

The SIP broker is a programmable application, running on DINA, which interfaces between the AAL and the Siptrex platform used to implement the CaVoIP service. The broker provides to the programmable entities the ability to control and manage the SIP components such as proxy servers, and user agents. The SIP broker also provides the service code the ability to obtain information from the SIP server and to control aspects of call control during a crisis situation. The SIP broker performs three primary functions; provide information about the SIP servers, sessions and users; terminate voice sessions; and delegate call admission control to service level objects.

3.2.2.2 Context Broker

The Context broker implements the mechanism for enabling service code to access the necessary context information from the various context sources. It provides the methods that enable context producers to publish their context information and context consumers (service code) to retrieve it. In the CaVoIP service, it is used to store SIP, network and user context.

3.2.3 Siptrex System

The Siptrex system [5] is a versatile and extensible platform that provides VoIP services based on SIP. The Siptrex system software was developed in Java using the Java API for Integrated Networks (JAIN), the Java Media Framework (JMF) and a SIP parser from the National Institute of Science and Technology (NIST). The Siptrex platform was chosen to use in the CaVoIP service due to its simplicity, easy extensibility and interoperability.

The Siptrex system is of a client server design. The Siptrex clients (user agents) are SIP software phones (softphones) through which Siptrex users

access the Siptrex services (through the Siptrex server) and are used to initiate and terminate SIP sessions. Siptrex service deployment is supported through the provision of a Siptrex API. The Siptrex platform components were modified to allow the SIP broker access to information from the Siptrex system enabling call admission control delegation and session termination.

3.2.4 CaVoIP Service Components

3.2.4.1 Service Invocation Condition Evaluator (SICE)

The SICE for the CaVoIP service is the SIP_SICE. It is configured to detect a crisis. It queries the Context broker to determine whether the number of SIP calls from a SIP domain has exceeded a certain threshold. When this threshold has been passed, a crisis is deemed to have occurred. The SIP_SICE then sends an event to the PBSM notifying it of an observed crisis.

3.2.4.2 Service Level Object (SLO)

The SLO for this service is the CH_Main. It is the key program in the CaVoIP service. It is deployed through the programmable network to the DINA nodes that are in the crisis area, as soon as the PBSM is informed of the crisis. Under a crisis situation the SLO takes over call admission control from the SIP server, allowing only privileged users access to the VoIP service. Non-privileged callers are blocked. Furthermore, the SLO terminates all non-essential calls when a crisis begins. It also fetches the user context (privileges) from the Context broker to determine whether they are privileged users.

3.2.4.3 Context Computational Object (CCO)

The CCO for this service is called the CH_Publisher. It collects SIP call information, relating to 911 calls, from the SIP broker. CH_Publisher compiles and publishes this information as context items to the Context broker.

3.3 CaVoIP Service Interactions

3.3.1 Service Creation and Deployment

The CaVoIP service is created using the CAS creation and customisation subsystem of the CONTEXT platform. The Siptrex and CONTEXT plat-

forms are first installed. Then the Context broker, SIP broker and CH_Publisher are installed on the DINA nodes The CAS code generator generates the service code and policies for the CaVoIP service. The service code and policies are then customised to fit the requirements of the customer (e.g. the 112 versus the 911 emergency number, definition of a crisis situation etc.). The result is customised CH_Main and SIP_SICE, customised policies and customisation parameters for the CH_Publisher. The SIP_SICE along with the generated policies and customisation parameters is distributed to the code execution points (DINA nodes). This process is carried out by the code distributor and is controlled by a Distribute_Service_Code policy. Other policies control service code removal and revision.

3.3.2 Crisis Detection

The CH_Publisher registers to publish the 911 call context to the Context broker. The SIP_SICE is configured by the PBSM with specific policies, and subscribes to the context items associated with the statistics of the emergency number, and the crisis condition. At every reporting period the CH_Publisher asks the SIP broker running in the same domain for session statistics detailing the number of 911 calls. This context is then exported to the local Context broker. If a crisis condition is detected, the SIP_SICE is notified by the CH_Publisher through the Context Broker. Then the SIP_SICE creates a Start_CH_Main event and sends it to the PBSM. This event contains the address of the DINA node that hosts the SIP broker responsible for the domain in which the crisis has occurred. Upon receiving this event the PBSM distributes CH_Main to the specified DINA nodes. Figure 3, illustrates the interactions among the components of the CaVoIP service involved in crisis detection.

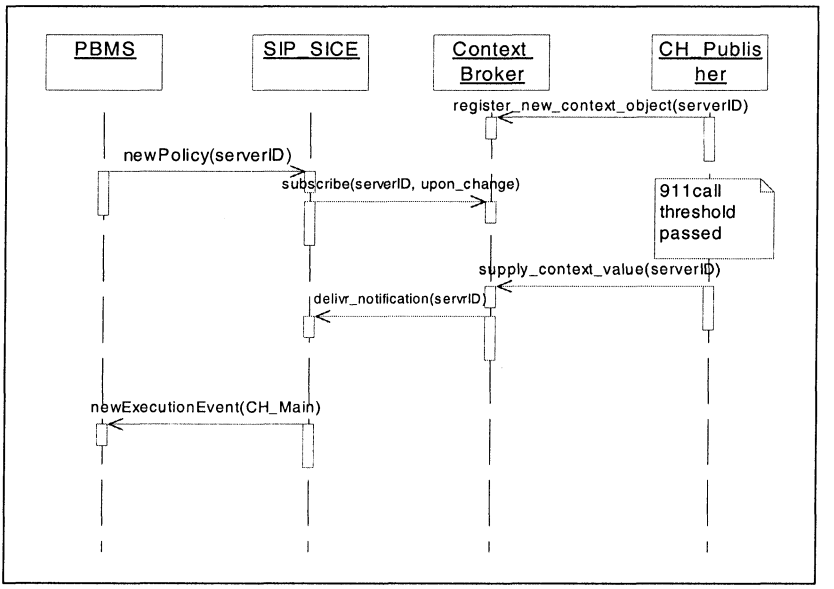

Figure 3: Sequence of interactions for the crisis detection stage

3.3.3 Service Execution

On arrival at the DINA node, CH_Main begins its execution by instructing the SIP broker to terminate all ongoing non-essential calls to and from the crisis domain. Then CH_Main is delegated call admission control. From this point on, all new sessions to be established must be authorised by the CH_Main. CH_Main checks the privileges of the callers and callees of all new call requests, by contacting the Context broker, allowing only privileged callers to make calls. When the crisis is over, CH_Main terminates and relinquishes call admission control. Figure 4 illustrates the interactions among components involved in the service execution stage.

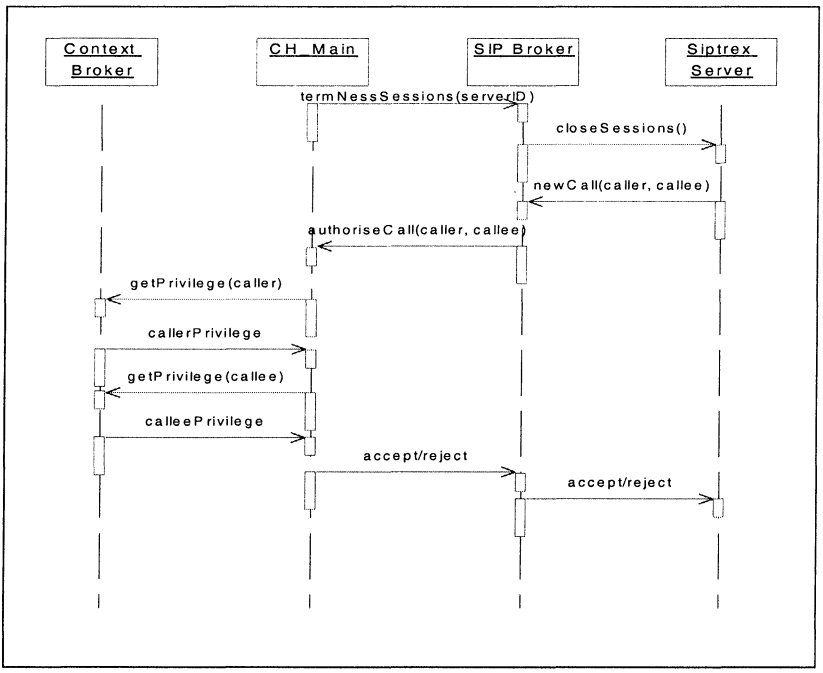

Figure 4: Sequence of interactions for the service execution stage

4 CaVoIP Service Evaluation

4.1 Service Scenario

A scenario can be envisaged where a terrorist bomb has exploded at a train station in a city. The scene is horrific and there are casualties. An eyewitness calls the emergency services. Many people gather at the site and there is a flurry of calls from them as well as other neighbourhood residents. The emergency services and police arrive and try to get the situation under control. An eyewitness calls the local newspaper to report the breaking news. When it has been established that a crisis is underway this call is terminated as the system terminates all the non-essential calls. Further call attempts fail and other low priority users cannot make calls. If a paramedic needs to call a doctor at the hospital for advice, the network accepts the call, due to his privileged status.

4.2 Service Testing

The aim of this test was to prove that the CaVoIP service works as envisaged during the design process. The test would be considered successful if as soon as a crisis occurs all the non-essential calls are dropped and from then on only essential calls are allowed. Figure 5 illustrates the testbed used for the CaVoIP service.

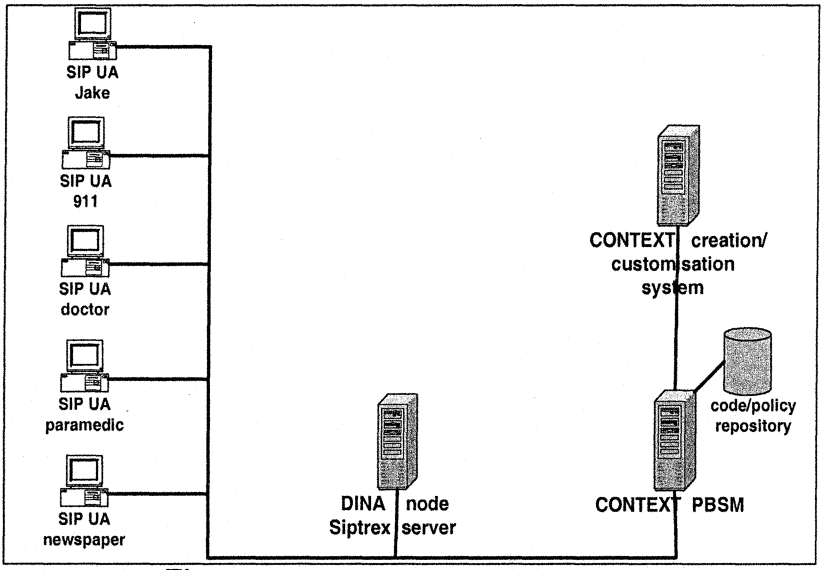

Figure 5: Testbed for CaVoIP service

The components of the CaVoIP service testbed are described below:

- One Siptrex server hosted on a Linux machine.
- One DINA node on the same Linux host as the Siptrex server. It contains the DINA programmable platform with the SIP and Context brokers as well as the CaVoIP service components: CH_Main, SIP_SICE and CH_Publisher.
- The CONTEXT PBSM, which consists of the code execution controller, the code distributor, the policy engine and the code and policy repositories.
- The CONTEXT service creation and customisation subsystem.

- Several user agents acting as callers and callees, one for the eyewitness Jake, one for the 911 contact centre, one for the newspaper and one for the doctor.

4.2.1 Test Procedure

The tests were conducted with the SIP_SICE, CH_Publisher, SIP broker, Context broker, Siptrex server, DINA components and CONTEXT platform.

1. The user privileges for each of the user agents were introduced in the Context broker.
2. Seven completed calls were made to 911 in less than 5 seconds (this was the definition of a crisis) and then a call was made between Jake's user agent and the newspaper. The crisis was detected and CH_Main deployed and started to execute in the DINA node.
3. A call was then made between the paramedic's user agent and the doctor's user agent.
4. Then another call was made between Jake and the newspaper.
5. After a period of time, an event was raised indicating the end of the crisis. Then another call was made between Jake and the newspaper.

4.2.2 Test Results

As soon as the crisis was detected and the CH_Main SLO began to execute, the call between Jake and the newspaper was terminated (as Jake is not a privileged user). The next call between Jake and the newspaper was not authorised and did not go through. The call between the paramedic and the doctor was authorised and went through. This was because the paramedic and doctor are privileged users. After the crisis was over the call between Jake and the newspaper went through. In fact all calls in the system were then authorised. The service tests were all carried out successfully and hence the CaVoIP service works exactly as was designed within the limits evaluated in the above test.

4.3 Service Extensibility and Flexibility

The CaVoIP service is very flexible and extensible due to its context-awareness and use of policies. More complex privilege allocation and logic to deal with it can be introduced. This can be done by altering the service logic found in the SLO to allow several permutations of privilege and access

control to be used. The user privilege could be allocated through a number scale. For instance, a paramedic could have a privilege of two, a doctor three while the chief of police has five. Ordinary users will have a privilege of one. The decision to authorise the call could then be made through a combination of the privileges of the caller and callee. This can be achieved just by changing the user context published to the Context broker and the call admission logic in CH_Main.

Changes to the service can be introduced during the service customisation phase. The definition of a crisis situation could be changed (e.g. 10 calls to 911 in 5 seconds or 20 calls to 911 in one minute) or a totally new crisis situation can be defined (e.g. 20 dropped calls in one minute). The context-awareness and policy-based infrastructure allows such easy customisation and flexibility. The policy-based nature allows easy extensibility of the service. Some users could be allowed video calling, others voice only calls, some short message service (SMS) calls while blocking others, all according to their privileges.

4.4 Service Scalability

As both the underlying platforms used to create the CaVoIP service, the Siptrex and CONTEXT platforms, are easily scalable then the CaVoIP service can also easily scalable. To allow scalability, much use is made of the modular design of the CaVoIP service and the policy-based infrastructure. Additional Siptrex servers can be accommodated by using a policy defined naming scheme both for the individual services and the context associated with the service. For instance if a Siptrex server is called SS001, all the context associated with it would be preceded with this tag. Different crisis conditions could be defined for each Siptrex domain. These parameters can all be defined during the service customisation phase.

5 Conclusions and Future Work

The CaVoIP service presented is a programmable context-aware service. It uses context, programmable network technology and PBSM to enable a network to deal with the huge increase in voice traffic during a crisis situation. It is easily customisable, personalisable and extensible due to the use of context and a policy-based infrastructure. The CaVoIP service could easily be developed further through more complex privilege allocations and

more complex logic to deal with them.. Also the services provided by the CaVoIP service can be extended to provide video calls and instant messaging in which case it could be possible to define privileges that allow users to either have video calling, voice only calls or short message service (SMS) calls according to their user privileges

References

[1] Dey, A. and Abowd, G., "Towards a better understanding of context and context-awareness". Proceedings of Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), The Hague, Netherlands. April, 2000.

[2] Sinnreich, H., Johnston, A., "Internet Communications Using SIP," John Wiley & Sons, New York, 2001.

[3] CONTEXT Consortium, "Context project deliverable D2.2: CONTEXT Architecture: Solution for provisioning and delivery of context aware services" ed. UCL, 2004.

[4] CONTEXT project website, http://context.upc.es/index.htm.

[5] Siptrex website, www.siptrex.net.

[6] Sygkouna, I. et al., "Context-Aware Services Provisioning on Top of Active Technologies," IFIP 5th International Conference on Mobile Agents for Telecommunication Applications (MATA 2003), Marrakech, Morocco 8-10.10, 2003.

[7] Kornblum, Jessica. Raz, Danny, Shavitt, Yuval. "The Active Process Interaction with its Environment," Computer Networks, 36(1):21--34, June 2001.

[8] Raz, D. and Shavitt, Y., "Towards Efficient Distributed Network Management," Journal of Network and Systems Management, September 2001.

[9] ABLE: The Active Bell Labs Engine http://www.cs.bell-labs.com/who/ABLE/

[10] Denazis, S. G., Galis, A., "Open Programmable & Active Networks: A Synthesis Study" IEEE IN 2001 Conference, Boston, USA, 6- 9 May 2001, ISBN 0-7803-7047-3.

[11] Galis, A., Denazis, S., Brou, C., Klein, C. (eds), "Programmable Networks and Programmable Network Management," ISBN 1-58053-745-6, Artech House, London April 2004.

[12] Ebling, M., Hunt, G. and Lei, H., "Issues of Context Services for Pervasive Computing," Proceedings of Workshop on Middleware for Mobile Computing, Heidelberg, Germany, 2001.

[13] Tennenhouse, D. L. and Wetherall, D. J., "Towards an Active Network Architecture" Computer Communication Review, Vol. 26, No. 2, April 1996.

[14] Tennenhouse D., Smith J., "A survey of Active Network Research," IEEE Communications Magazine, January 1997.