Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 493261, 10 pages
doi:10.1155/2012/493261

Research Article

Enhancing Existing Communication Services with Context Awareness

Bachir Chihani,[1,2] Emmanuel Bertin,[1,2] Irsalina Salsabila Suprapto,[3] Julien Zimmermann,[1] and Noël Crespi[2]

[1] *Orange Labs, 42 rue des Coutures, 14066 Caen, France*
[2] *Service Architecture Lab, Telecom SudParis, Institute Telecom, CNRS 5157-9 rue Charles Fourier, 91011 Evry, France*
[3] *Telecom Bretagne, Institute Telecom, Campus de Rennes, 2 rue de la Chtaigneraie, 35576 Cesson Sévigné Cedex, France*

Correspondence should be addressed to Bachir Chihani, bachir.chihani@orange.com

Received 30 March 2012; Accepted 30 May 2012

Academic Editor: MoonBae Song

Context aware communication services rely on information sources and sensors, to derive users' current situation and potential needs, and to adapt their communication services accordingly. If extensive studies have been driven on context awareness by industrials and researchers from academia, the design of such systems without modifying uses and manners of underlying communication services—while keeping them simple, intuitive, and reactive—remains a challenge. In this work, we introduce a context aware communication system that takes into account user's preferences, workload, and situation to customize telephony services. In this implementation, we use IMS for communication management. The benefits of this implementation are the enhancement of IMS with context awareness features, and the coupling of user preferences with contextual information to provide improved service customization, without modifying the user experience.

1. Introduction

In terms of usage, people still mostly use telephony like in the early 1970s; Alice dials Bob's number and hears a ring back tone, Bob picks up if he is available, and both communicate. Also, communication systems typically ignore the callee's situation when establishing a communication channel between him/her and the caller. For example the time might not be appropriate (e.g., the callee is in meeting), or the communication channel is not suitable (e.g., the callee is far from his fixed phone or is handling a mobile phone in a public space, such as a cinema).

Recently, we are witnessing big advances in devices capabilities and technology as they become smarter with the introduction of sensors (e.g., accelerometer, GPS). These advances should help to improve the legacy telephony services and to design services in adequate way with users' increasing requirements [1]. Now, effort of developing innovative communication services has to consider providing flexible mechanisms to support service adaptation based on observed changes in user situation.

Context-aware communication (CAC) [6–8] services attempt to enhance communication systems with the ability to deduce callee and caller situations in order to decrease the probability of communication failures. The situation of a user can be defined as a snapshot of the user's context at a certain instant or period of time. For a communication service, most relevant contextual information is user related (e.g., location, presence, and activity), device related (e.g., access network, battery level), and environment related (e.g., noise level).

In order to enhance existing communication services, three main challenges should be addressed.

(i) First, CAC services should extend current communication services, but without modifying the way they are used by end users, as telephony usage is very well established.

(ii) Second, CAC services should reduce the amount of interruption that current system causes and should not increase it (e.g., with inopportune notifications).

(iii) Third, CAC services should support multichannel communication (e.g., voice and text-based communications) and reactivity (e.g., not extending the call establishment delay).

Most existing work in CAC, even if some were innovative, did not address all these challenges. We intend here to tackle them by proposing a framework able to manage multiple contexts, and able to facilitate the integration of CAC services with communication services.

The rest of the paper is organized as follows. In Section 2, we present two different visions for context-awareness and we introduce the benefits and the requirements for CAC systems. In Section 3, we introduce a generic framework for supporting the development of CAC system, which is used then in Section 4 to implement CoAR (context-aware reachability), a prototype for handling incoming calls. In Section 5, we present the existing works on CAC systems, and then we discuss these works and our work in Section 6 based on the previously introduced requirements. We conclude the paper in Section 7.

2. Context-Awareness and Its Requirements

Context is all information intrinsic to an entity (e.g., user, device) that can be acquired, made explicit, and published to applications, in order to enable them to adapt their behavior to the entity's state. In this section, we present two viewpoints on context and context-awareness, typical scenarios of CAC systems, and then we present the challenges and requirements for building such innovative systems.

2.1. Theorist versus Practitioner. From the academic viewpoint [9, 10] context is about facts, rules, and axioms that can be used to describe a state of an entity (e.g., user, device) at a given time. Context-awareness is considered as a special kind of formal logic systems on which well-established artificial intelligence theories and algorithms (e.g., rule-based inference) can be applied for automating the processing of inferring new knowledge and reasoning on facts representing user situation.

From the industry viewpoint (e.g., travel or retail companies [11]) context is about preferences, activities, and geospatial information related to a given user [12, 13]. This information is acquired to answer questions like the following.

(i) Who is using the service? What are his/her preferences or habits?

(ii) What is the user doing? What are his current activities?

(iii) Where is the user? What is the logical or physical place where the service is invoked?

(iv) When? At what time are his/her actions occurring?

(v) How? Which device is used to access the service?

With this in mind, context-awareness for industries is defined as the ability of a service to adapt its response to user's requests depending on whether the service is accessed from a mobile phone or accessed from a laptop, on his/her identity and his/her role (e.g., particular or professional client), from where the user is looking for the service, and so forth.

In this paper, while keeping in mind the academic viewpoint, we intend to take into account the industry viewpoint that is usually less investigated.

2.2. Benefits of CAC Systems. In order to offer context-awareness, communication systems should be able to detect user-relevant situations and to adapt their behavior accordingly, as well as their user interface. For example, the fact that a user is working or on a holiday could impact the handling of incoming calls (e.g., to reject a call from a best friend or a call from a business partner), and even the user's contact list (how the contact list is ordered).

Also, the way received messages and event notifications are displayed should be adapted to the user situation; an urgent voicemail can be played by the TV if the user is watching TV alone, but when someone is with them then the message will be displayed on his/her smartphone. Such systems will be smart enough to take into account rules specifically related to the user environment (e.g., do not answer calls during a movie), while considering exceptions (e.g., receiving a call considered urgent by a well-known caller).

2.3. Requirements for CAC Systems. The scenarios presented above involve different equipments with different capabilities and technologies, for example, hardphone, softphone, smartphone, TV, and PC screens. They involve different sources of information (e.g., agenda, contact list, preferences) concerning the users and their surrounding environment (e.g., nearby communication devices, transportation means) to help in deducing a users' situation. To realize such scenarios, big challenges have to be faced like how to enhance existent services without having to modify their implementation and usage, how to make system as simple as possible even when deployed in complex environments, how to consider multiple communication channels, and how to support real-time interaction with the user.

To summarize, CAC systems should address the six following requirements, which can be grouped as follows.

User-related requirements:

(1) Heterogeneity of components (sensors, mobile devices, servers), communication technologies and protocols (e.g., SIP, XMPP);

(2) Ease of use because end users will not be IT experts;

(3) Privacy of users, which has to be controlled (e.g., how user's information will be used by applications) and preserved (e.g., who can see what); and

System-related requirements:

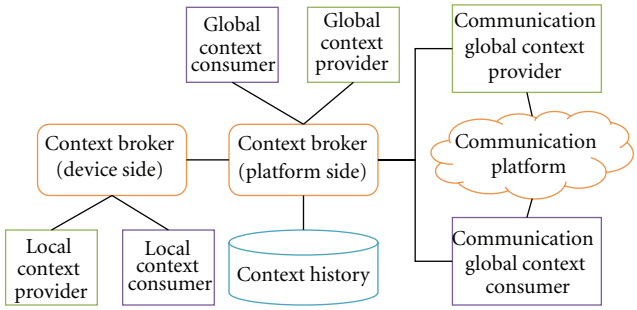

Figure 1: Context management framework.

(4) Flexibility of information modeling and management (acquisition, distribution, processing and storage);

(5) Scalability from few to many users and devices, often belonging to different administrative domains;

(6) Reactivity in order to guarantee a real time adaptation and a reasonable response time.

These requirements have to be addressed over the whole life cycle of contextual information (from the extraction to its use) and at the different parts of the CAC system.

3. Framework for Developing CAC Systems

3.1. Context Management. To develop prototypes of context-aware systems, we propose a generic framework for integrating context and communication management.

Figure 1 illustrates the architecture of our framework, which is based on the consumer/provider design pattern with the introduction of brokers to decrease the amount of point-to-point communication between the different components. In this figure, the green color is used to represent providers, the orange for brokers, the blue for databases, and the purple for consumers. The device side broker (implemented for a PC and an Android client) aims to decouple local context providers from local context customers and remote components, while the platform side broker (implemented as a Restlet application running on a Jetty web server) aims to decouple the different components and third-party applications. Local context providers/consumers are components present on a device, while global context providers/consumers are components (e.g., third-party applications) that are connected directly to the platform-side context broker. In this architecture, local brokers are able to aggregate contextual information produced by local context providers before publishing it into the global context broker to be available for all the different components in order to reduce the amount of exchanged messages between both sides. We used the CometD (http://cometd.org/) server as the notification server to implement the publish/subscribe mechanism for context distribution.

For gathering communication-related context, a specific global context provider is used, the communication global context provider. We use the communication global context consumer for performing adaptation as a way to improve communication management.

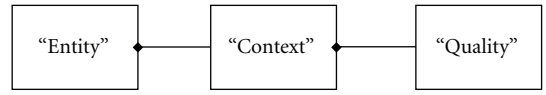

Figure 2: Context modeling approach.

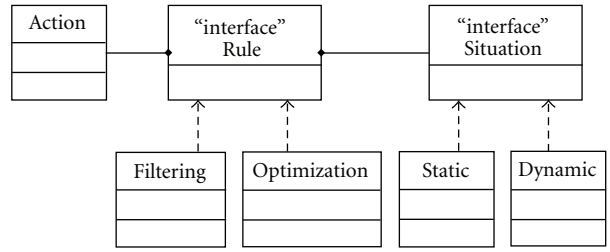

Figure 3: Rule diagram.

3.2. Context Modeling. Figure 2 illustrates the generic context model used by the framework. It is composed of three interfaces: Entity (e.g., person, place, device, network), Context (e.g., location of a person, connectivity of a device), and Quality (e.g., error rate of a sensor). The interfaces have to be implemented in order to create a complete context model. The context is transported in XML between the architecture components.

3.3. Context Reasoning. We propose a generic diagram (Figure 3) for modeling context reasoning rules. A reasoning *Rule* is an IF-THEN rule; the condition part is a set of *Situation* instances, and the action part is a set of *Action* instances. A rule can be for filtering (e.g., blocking calls when a user is in a meeting) or for optimization (e.g., forward a call to a hardphone instead of the mobile if the user is in his/her office). A situation is a set of contextual information; it can be static and set by a user before runtime (e.g., the user is in a meeting) or dynamically known at runtime (e.g., a user's current situation).

The Algorithm 1 for handling context reasoning rules takes the list of registered rules (e.g., user preferences) and the current user situation as parameters. First, filtering rules are evaluated; the situation part of each rule is evaluated against the user's current situation to retrieve the corresponding actions. Then, the resulting actions are evaluated against the user's devices' current situation in order to eliminate the nonapplicable actions (e.g., forward call to a not registered phone). Finally, the optimization rules are used to evaluate the applicable actions and to output the actions that most fit the user's current situation. They allow the framework to deal with rule collision cases, where more than an action can be performed by ranking applicable actions and eliminating those not fitting user current situation. A threshold α is used to measure the fitness (weight) of an action given user current situation; actions

Input:
Current user situation or context data $C = \{c_1, \ldots, c_n\}$,
Fitness Threshold α,
Filering rules FR $= \{\text{fr}_1, \ldots, \text{fr}_p\}$,
Optimization rules OR $= \{\text{or}_1, \ldots, \text{or}_q\}$
Output:
Applicable actions $A = \{a_1, \ldots, a_m\}$
1: A$\leftarrow \emptyset$
2: *Applicable* $\leftarrow \emptyset$
3: **for each** rule $\text{fr}_i \in$ FR **do**
4: **if** $\text{fr}_i \cdot condition \in C$ **and** d $\text{fr}_i \cdot action \cdot is\ Applicable\,()$:
5: *Applicable* $\leftarrow$*Applicable* $\cup\ \text{fr}_i \cdot action$
6: **for each** rule $\text{or}_i \in$ OR **do**
7: **for each** action $a_j \in$ *Applicable* **do**
8: $a_j \cdot weight = a_j \cdot weight + \text{or}_i \cdot evaluate\,(a_j)$
9: **for** action in $a_i \in$ *Applicable* **do**
10: **if** $a_i \cdot weight > \alpha$:
11: $A \leftarrow A \cup a_i$
12: **return** A

Algorithm 1: Reasoning on contextual information.

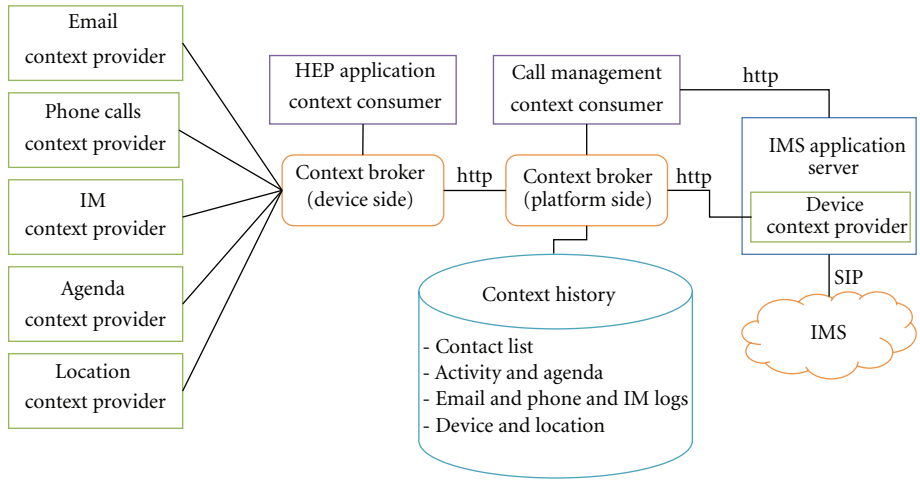

Figure 4: Architecture of the call management solution.

with low weight will be eliminated and not executed. An example of optimization rules would be a rule that prioritize the desk phone over cell phone in case both can receive the incoming call.

In our implementation, the threshold is set to zero so that any rule positively evaluated will be executed.

4. CoAR: Implementation with an IMS Platform

4.1. Architecture Overview. IMS [14] is a standardized architecture for converged communication services. It is structured into three planes: the application plane for offering end users' services through IMS application servers, the control plane for implementing access control and call/data routing, and the transport plane, which consists of physical resources for transporting voice and data.

Figure 4 illustrates the architecture of our prototype for call management. The components are as follows.

(i) A set of local context providers that periodically retrieve contextual information (e.g., the number of received/rejected/answered phone calls) from various communication tools (e.g., E-mail, IM, phone, and agenda).

(ii) A local context consumer in charge of generating the user workload level from the gathered context, in order to publish this workload so that it is accessible to users' contacts.

(iii) A device-side broker that decouples local context providers, consumers, and remote components.

(iv) A platform-side broker to facilitate the distribution of users' workload levels among the different consumers.

(v) A database for storing all the gathered context information for a future use.

(vi) A location context provider to track user's location (e.g., at office, home) and his/her transportation means (e.g., highway, train).

(vii) A device context provider to provide device-related contextual information (e.g., type, battery level, nearby devices) and IMS-related information (e.g., registration status, timestamp and expiration time, presence status). It plays the role of the communication global context provider, and it is implemented with the service triggering module provided by Mobicents.

(viii) A call management context consumer in charge of making decisions related to call routing in IMS architectures (e.g., call transfer/forwarding/completion). In the request body sent to this component, devices' IMPUs are included to identify corresponding users. This device IMPU represents an ID in REST requests.

The call management context consumer plays the role of the communication global context consumer. It is implemented as a RESTful web application. It has two interfaces: one with the context management framework for retrieving contextual information and another with the IMS AS for handling routing decision requests. The use of RESTful web services with HTTP as underlying transport protocol instead of SIP aims to make the framework generic enough to be used in other environments than SIP/IMS.

4.2. Use Case: Handling of Incoming Calls. When the IMS receives a call request (i.e., a SIP Invite) for a given IMPU (IMS public user identity, that is, the identifier that is used to reach a user on corresponding device), the Mobicents AS is triggered. The AS sends a request to the call management context consumer with the IMPUs of both caller and callee. It receives in response the action to perform on this call request. The different actions are as follows:

(i) nothing: to let the call reach the callee;

(ii) busy: to interrupt the incoming call;

(iii) redirect: to redirect the call to a voice server; or

(iv) transfer: to transfer the call to another destination.

In the redirect and transfer cases, an optional parameter specifies the target IMPU (e.g., IMPU of the voice server). If the call management context consumer module does not respond in due time, the "nothing" action is performed.

When reasoning based on contextual information and deciding how to handle incoming calls based on the user situation, it is very important to include user preferences. For example, users may not want to be interrupted by a call from a certain class of contacts (e.g., colleague) if they are very focused on their current work (high workload level). In this implementation we used the previously described rule diagram (Figure 3) to represent user preference rules.

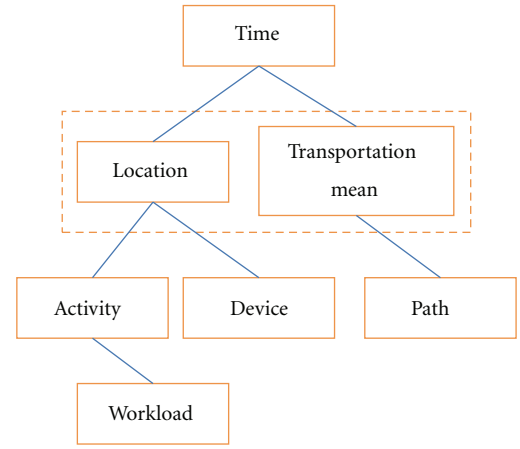

Figure 5: Hierarchical view of contextual information.

Concretely, these rules are represented in XML so that they can be exchanged easily between framework components; they are instantiated as java objects to be processed by the internal reasoning engine. We are using a custom rule engine that consumes java objects representing filtering or optimization rules to generate the action to be performed.

To help users defining their preference rules and to guide them through this process, we propose a hierarchical view (Figure 5) for representing contextual information to users. For a preference rule, users may first indicate the type of time (e.g., workday, weekend, vacation, lunch time). Then, they have to choose a location (e.g., office, home) or the transportation means and type of path (e.g., home-work path, occasional path) if the rule will be applied when a user will be on the move. Next, they can specify the type of activity (e.g., reading mails) and device (e.g., hardphone) affected by this rule. For the activity, they can specify the workload level (e.g., busy). Following this approach, an example of a generated rule will be as Algorithm 2.

CoAR here allows users to have control over their communication. Users specify their preferences regarding how they can be reached under which situation. It provides an extension to existent communication platform without modifying underlying communication service and reduces amount of interruptions caused by existing communication service as it decides on how calls are managed on behalf of user. Also, it considers context information from multiple sources.

4.3. Infrastructure and Evaluation. Our testbed IMS platform is built on the Open IMS Core [15] which implements the control plane of IMS architecture (P-CSCF, I-CSCF, S-CSCF, HSS). At the application plane, we used Mobicents [16] application server (AS) for communication service creation, through its JAIN SLEE API. We use also different types of IMS clients (e.g., IMSDroid for Android-based smartphones, Uct IMS client for softphones).

Filtering rule:

If isTime (Workday) == true
  and isLocation (AtWork) == true
  and isActivity (EditDocument) == true
  and isWorkload (Do_Not_Disturb) == true
  and isCallFrom (Boss) == false
Then
  RedirectCallToVoiceServer (VoiceXML_File)

Algorithm 2

Figure 6: Test environment.

Figure 6 depicts our test environment configuration. The Open IMS Core is installed on Ubuntu/Linux 10.0 operating system running on Oracle VirtualBox (https://www.virtualbox.org/) virtual machine. The latter, as well as Mobicents AS, is running on Microsoft Windows XP SP3 installed on Dell D630 laptop with the following characteristics vender: Intel, model: Core 2 Duo, CPU: 2,2 GHZ, memory: 2 GB.

For persistence, contextual information are stored on an object database DB4O (http://www.db4o.com/). Context Broker and call management context consumer are both implemented as web applications with the Restlet (http://www.restlet.org/) framework running on a Jetty web server. Both are installed on Microsoft Windows 2003 Server SP2 with the following hardware charcteristics: vender: Intel, model: Xeon, CPU: 2,8 GHZ, memory: 2 GB.

Main contribution of this paper is the design and implementation of a context-aware communication management framework able to provide user with support and assistance service by handling his/her preferences and context parameters. Therefore, it is important to evaluate the cost to achieve context-awareness in terms of response time as it is an additional amount of time to the response time of the communication service. The summation should remain acceptable for users.

The response time of CoAR depends on type of operation (context publication, context query/consumption, or context subscription), and needed time to upload contextual information from database to memory in case it is stored. The response time for the call management context consumer depends on response time of the broker plus the service reasoning time. User response time depends on response time of Mobicents AS that depends on call management context consumer response time.

The following graphs illustrate the evolution of response time for broker according to the number of parallel context publications (Figure 7) and number of parallel context consumption requests (Figure 8).

The aim is to figure out the response time as function of these parameters in order to measure the reactivity of the overall system and to be able to estimate in future the expected response time.

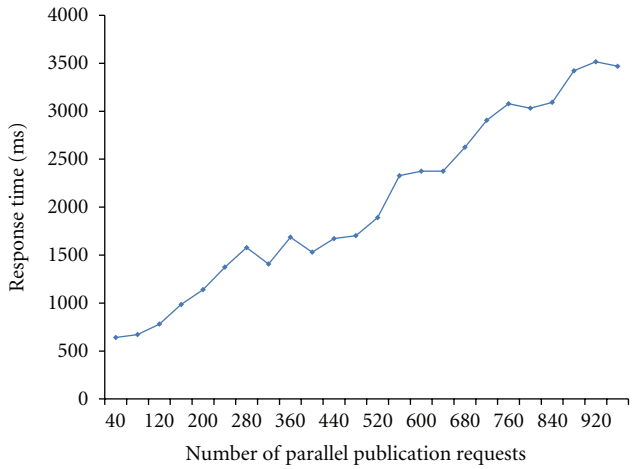

Figure 7: Response time variation according to number of publication requests.

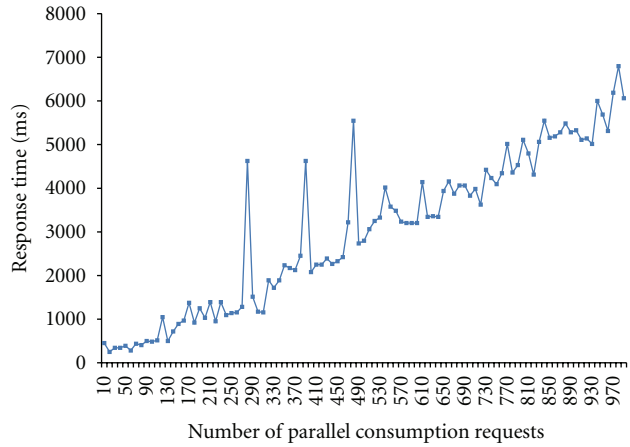

Figure 8: Response time variation according to number of consumption requests .

The two above graphs present the variation of response time of parallel context publication/consumption requests. Both match the following linear function:

$$\text{Response Time (Request Number)} = \alpha + \beta * \text{Request Number} \quad (1)$$

The function parameters α, β can be approximately computed as follows:

$$\beta = \text{average}\left(\frac{\Delta\text{Response Time}_{i+1,i}}{\Delta\text{Request Number}_{i+1,i}}\right), \quad (2)$$

$$\alpha = \text{average}\left(\text{Response Time}_i - \beta * \text{Request Number}_i\right). \quad (3)$$

For the first graph data we find $\beta \approx 3{,}07$ and $\alpha \approx 516{,}46$, for second graph we find $\beta \approx 5{,}72$ and $\alpha \approx 143{,}47$.

We believe that the current CoAR response time should be acceptable even under extreme conditions, that is, when hundred of requests are received in the same time. To enhance it further, we should investigate IO operations (input/output) response time derived by storage and retrieval of contextual information from the database, for example, by using cache systems.

The delay time engendered by our framework before call establishment between two users depends on needed time for sending an HTTP request to the call management service and the reasoning time needed to decide how an incoming call should be handled. Figure 9 depicts a comparative summary of delay distribution in case user preferences are used to decide how to handle incoming calls (i.e., with), or not used (i.e., without). The distributions are generated by sending about one hundred sample requests to the call management context consumer component. In the figure, the median (50th percentile) of each distribution is represented with a darkened line; bottom side of boxes represents the 25th percentile of corresponding distribution; top side represents 75th percentile; bottom line represent 10th percentile; top line represents 90th percentile.

From the figure, the two distributions are very close in terms of shape, similar dispersion (spread) of delay samples. Distribution median is about 300 ms when reasoning on user preferences and about 240 ms when no reasoning is performed, this difference is due to reasoning overhead. All values of both distributions are between 50 ms to 500 ms, with an interquartile (samples in the box between its bottom and top) ranging from 200 ms to 350 ms. As a result, the delay is more influenced by time needed to send HTTP request to the call management service and reasoning overhead is negligible against it. This delay range is fully compatible with the duration of an end-to-end call setup.

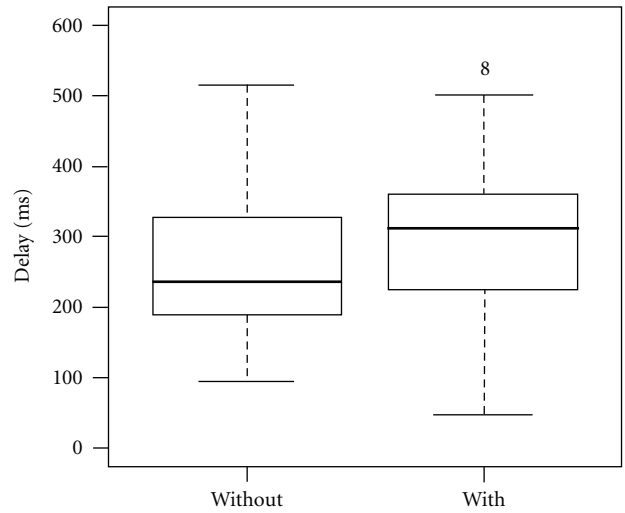

Figure 9: Distribution of call establishment delay.

5. Related Work

A first experiment with communications management has already been successfully conducted in [17, 18] where we defined HEP (enHanced unintErruPtibility) which provides users with the workload level of their contacts on different communication channels to help them to evaluate the impact of the interruption they will cause by initiating a communication with a given contact. The system was efficient in representing the workload of contacts, but in some cases users ignored the provided information and still initiated communications with the callee. We believe that a communication management system only based on publicizing information about the availability of callees is not sufficient. However, communication management systems based only on routing policies entered by the callee, such as call transfer, suffer from a lack of flexibility. Hence, we propose CoAR a communication management system that merges a context management framework with a communication platform in order to best manage users' communications.

Cals.Calm [19] provides information about callees by publishing them on a web page. Thus any device with Internet connectivity can access them, enabling the support for heterogeneity but also presenting a serious threat to callees' privacy. Considered context is limited to location and activities which are manually edited by users, and the system does not provide any intelligence to make use of this information. The system is based on a web server that may be configured to support scalability and real time responsiveness.

MOCSP [20] defines communication ext-links, URI, to a web-based communication service (e.g., IM, E-mail, etc.), as the address to be used by a caller to reach users. In [21], MOCSP is used for managing communication sessions across callee's multiple devices based on user preferences. Users have to specify which communication service to be used for a group of contacts. The system is open to the Internet and users may access it with any device having Internet connection. The system is based on a P2P model in which each user has its own communication server.

In [2] a CPL- (Call-Processing-Language-) based CAC for customizing call control services (forward, transfer, and completion) is presented. Decisions rely on user-entered

preferences which may lead to incorrect call filtering decisions and/or threaten user privacy. The system considers users as having the capability to build CPL scripts and provides a CPL scripting editor and a command line for configuration. Contextual information is limited to location and calendar entries. System architecture is client-server, providing a means for support scalability.

In [3] a centralized system based on otology approach for context modeling and reasoning to adapt the way incoming calls (e.g., not answered) are handled (e.g., forwarding) is presented. Considered context is historical information of communication sessions already established between the callee and the caller without exposure to third parties, that is, no support for heterogeneity. Routing decisions rely on a common threshold to all users, making it inflexible and not able to adapt to quick variations in a user's situation.

AmbiTalk [22] is an SIP-based CAC system to enable smart places to enforce their communication policies on mobile devices, which limits supported devices. It allows mobile devices to adapt their configuration (e.g., turn phone to vibrating mode when it enters a meeting room) and to negotiate communication sessions (e.g., IM instead of phone call if two devices are located within places enforcing different policies). Thus, it may restrict device usage and raise usability and privacy concerns. Considered context is limited to device-related (e.g., location, alert mode, or owner).

In [4] a centralized policy-based context-aware message delivery system is presented. The system enhances OMA (Open Mobile Alliance) Converged IP Messaging (CPM) with context-awareness for selecting appropriate messaging services (e.g., SMS, E-mail). User preferences are used in the condition portion of IF-THEN routing policies which may lead to wrong decision, that is, routing a message to a bad destination. It requires some user configuration efforts to specify for each contact the preferred communication channel.

INCA [23] is a layered CAC assistant with a P2P-based architecture enabling scalability. Two consecutive layers interact with event notifications and requests for action execution, which may lead to conflicts between actions made by different layers and/or to wrong decisions. Contextual information are limited to user preferences, used devices, type of communication technology (e.g., VOIP), and subject of the communication.

SECE [24] (Sense Everything, Control Everything) is a context management framework supporting the creation of mashable communication services that combine multiple context dimensions (location, presence, agenda). The framework enables end users to write context-aware Tcl-based scripts composed of conditions on user context and corresponding actions which are described in a simple and natural language terminology. Even if writing context-aware rules (e.g., send a happy birthday message on behalf of user when it is the birthday of a friend) is intuitive, it is hard for users to maintain and keep such rules up to date, especially for actions that are not frequent in a uniform way or may happen under ambiguous situations.

In [5], the authors present an approach for embedding device context into mobile search queries to enhance the relevance of the result to device characteristics and user situation. They propose a rule-based approach for abstracting raw physical measurements (e.g., GPS location) acquired from the device sensor into user high-level contextual information (e.g., user is at home). These derived information are then transferred to search engines (e.g., Bing) as additional parameter using the corresponding API which may limit the potential of the solution as it depends on a limited set of acceptable parameters for the search engine.

6. Discussion

Most of the works presented in the previous section allow users to define rules that specify how calls have to be routed, or provide caller with information related to callee's situation to make decision on whether or not making a call. The first option lacks flexibility, and is hardly maintainable as users may not update their rules every time their preferences change. The second option assumes willingness of caller to use the provided information to make better decisions and avoid interrupting callee. Another option would be to automatically detect user context and adapt service behavior correspondingly as proposed by CoAR.

We summarize the discussion of the different works in Table 1 where evaluation criteria are based on the requirements already introduced in Section 2.2. The symbol "+" means the existence of a support for the corresponding requirement, whereas the symbol "−" indicates its absence; "±" is used to describe cases where support exists but needs to be enhanced.

Based on this analysis, we can conclude that most of the previous work in developing CAC systems has been focusing more on enhancing the system-related requirements rather than user-related requirements. In our work, we try to provide a better means to meet both system- and user-related requirements.

Our framework (Figure 1) supports the enhancement of communication platform generally and IMS in particular with context-awareness features for enabling service customization. Yet, it does not require changes in user habits regarding the use of their favorite communication services, and it reduces interruptions caused by underlying services. Also, it is generic enough to be used in other environments than SIP/IMS as it is based on RESTful web service with HTTP as underlying transport protocol and not SIP.

The main difference between our model (Figure 2) and the different context models that have been proposed is its simplicity that enables developers to use the framework without having to understand complex modeling schema, thereby making it possible to rapidly prototype context-aware systems.

The distribution of brokers makes the system scalable and provides a better support for heterogeneity because a device-side broker can be implemented on different devices with different technologies. Implementing lightweight providers/consumers on the device side provides an enhanced privacy protection because contextual information can be used locally; they can also be carefully designed to provide an enhanced usability. Implementing

Table 1: Comparison between CAC systems.

	(1) Heterogeneity	(2) Usability	(3) Privacy	(4) Flexibility	(5) Scalability	(6) Reactivity
Cals.Calm	+	−	−	−	+	+
MOCSP	++	±	−	−	++	+
Framework [2]	+	−	−	+	+	+
Framework [3]	−	−	−	−	−	−
AmbiTalk	−	−	±	−	+	+
Framework [4]	+	−	−	−	−	−
INCA	±	−	−	+	++	+
SECE	++	+	−	±	+	+
Framework [5]	±	+	+	±	+	+
CoAR	+	±	±	+	±	±

providers/consumers on the platform side and the simplicity of the context modeling approach enable the use of any contextual information sensed or derived from running data mining algorithms.

Our framework provides a support for reasoning on context and system adaptation via the Rule class diagram (Figure 3) which is generic enough to represent any kind of adaptation rules. In addition, the reasoning engine is able to interpret two types of rules (i.e., filtering and optimization) to produce a more flexible adaptation. This distinction between rules is a logical distinction as filtering rules implement hard constraints on actions (if condition is not verified then eliminate corresponding action) while optimization rules implement soft constraints for evaluating and ranking actions. This distinction motivated the choice of a custom rule engine instead of using a generic rule engine.

For the moment, the framework does not support cases where multiple devices use the same SIP ID/IMPU. To address this limitation, contact header in SIP messages can be used to store more information (e.g., device location, IP address) that help distinguish between multiple devices with same IMPU. For improving performance of reasoning, we should use more sophisticated engines able to process adaptation policies described in a domain-specific language.

7. Lessons Learnt and Perspectives

In this work, we present a framework that aims to manage contextual information on behalf of applications and provide them with mechanisms for context-awareness that does not imply any changes to the original application logic and the interaction between user and the application. Three main lessons were learned from our work.

(i) First, the IMS is sufficiently flexible to support an external call routing decision based on the context of the callee.

(ii) Second, http quasi-synchronous mechanisms like CometD enable to develop nearly real-time publish/subscribe RESTful services to distribute the context among the providers and consumers.

(iii) Third, a cache system seems more adequate than a database for context storage, as context information has a limited validity time. This leads also to better performances.

More generally, this work reveals the limitation of rule-based contextualization approaches as they are not flexible enough due to the need for maintaining continuously the rule base. It also shows the importance of user feedback in the process of customizing adaptation. This feedback can be acquired directly from user (explicit feedback) or by observing user behavior and reactions (implicit feedback).

An interesting perspective would be to use contextualization approaches that take as input user feedbacks (implicit and explicit) in addition to context data in order to generate and update autonomously inference rules. Machine learning approaches are promising for this purpose and should be deeply investigated. These approaches will enable automated reasoning on contextual information without need for predefining and maintaining inference rules.

In the future, we plan to make the decision process more flexible by supporting the automated learning of routing rules. We intend also to interface our framework with an IPTV system so that we can implement the second part of the presented use cases.

8. Conclusion

Communication services extended with end users context awareness features will help reduce and in some case eliminate barriers that complicate communication between individuals and groups.

This paper introduces a framework that integrates a communication platform with a comprehensive context management system. It describes how this framework was used to prototype CoAR, a CAC system that combines contextual information with user preferences to decide how to handle incoming calls. We believe that our work illustrates the possibility of building innovative CAC systems that rely on existing infrastructure without disturbing their usage or their performances.

What still need to be investigated by researchers from industrial companies and academic universities is how

to gain user's trust and acceptance toward context-aware systems, as the context-awareness benefits do not come without cost, especially the threats on user privacy implied by the system autonomous monitoring of context and activities in a daily manner.

References

[1] D. Tacconi, D. Miorandi, I. Carreras, F. De Pellegrini, and I. Chlamtac, "Cooperative evolution of services in ubiquitous computing environments," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 6, no. 3, article 20, 2011.

[2] M. Görtz, R. Ackermann, J. Schmitt, and R. Steinmetz, "Context-aware communication services: a framework for building enhanced IP telephony services," in *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN'04)*, pp. 535–540, Chicago, Ill, USA, October 2004.

[3] K. Hamadache, E. Bertin, A. Bouchacourt, and I. Benyahia, "Context-aware communication services: an ontology based approach," in *Proceedings of the 2nd International Conference on Digital Information Management (ICDIM'07)*, Lyon, France, October 2007.

[4] N. Blum, S. Lampe, and T. Magedanz, "Enabling context-sensitive communication experiences," in *Proceedings of the 14th International Conference on Intelligence in Next Generation Networks (ICIN'10)*, Berlin, Germany, October 2010.

[5] E. Yndurain, D. Bernhardt, and C. Campo, "Augmenting mobile search engines to leverage context awareness," *IEEE Internet Computing*, vol. 16, no. 2, pp. 17–25, 2012.

[6] B. N. Schilit, D. M. Hilbert, and J. Trevor, "Context-aware communication," *IEEE Wireless Communications*, vol. 9, no. 5, pp. 46–54, 2002.

[7] A. Ranganathan and H. Lei, "Context-aware communication," *Computer*, vol. 36, no. 4, pp. 90–92, 2003.

[8] F. Toutain, A. Bouabdallah, R. Zemek, and C. Daloz, "Interpersonal context-aware communication services," *IEEE Communications Magazine*, vol. 49, no. 1, pp. 68–74, 2011.

[9] P. Mehra, "Context-aware computing: beyond search and location-based services," *IEEE Internet Computing*, vol. 16, no. 2, 2012.

[10] J.-D. Kim, J. Son, and D.-K. Baik, "CA_{5W1H} onto: ontological context-aware model based on 5W1H," *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 247346, 11 pages, 2012.

[11] W. Clark, *How Context Can Improve Your Customer Relationships*, Gartner Webinars, 2010.

[12] Z. Zhao, N. Lage, and N. Crespi, "User-centric service selection, integration and management through daily events," in *Proceedings of the 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM'11)*, pp. 94–99, Seattle, Wash, USA, March 2011.

[13] R. Tusch, M. Jakab, J. Köpke et al., "Context-aware UPnP-AV services for adaptive home multimedia systems," *International Journal of Digital Multimedia Broadcasting*, vol. 2008, Article ID 835438, 12 pages, 2008.

[14] R. Copeland, *Converging NGN wireline and mobile 3G networks with IMS: converging NGN and 3G mobile*, CRC Press, 2008.

[15] D. Vingarzan, P. Weik, and T. Magedanz, "Development of an open source IMS core for emerging IMS test-beds, academia and beyond," *Journal for Mobile Multimedia*, vol. 3, no. 2, pp. 131–149, 2006.

[16] A. Bhayani, "Developing converged application using open source software," in *Proceedings of the IEEE Symposium on Industrial Electronics and Applications, ISIEA 2009*, pp. 452–456, October 2009.

[17] B. Chihani, E. Bertin, and N. Crespi, "A comprehensive framework for context-aware communication services," in *Proceedings of the 15th International Conference on Intelligence in Next Generation Networks (ICIN'11)*, Berlin, Germany, October 2011.

[18] B. Chihani, E. Bertin, F. Jeanne, and N. Crespi, "Context-aware systems: a case study," in *Proceedings of International Conference on Digital Information and Communication Technology and its Applications (DICTAP'11)*, Dijon, France, 2011.

[19] E. R. Pedersen, "Calls.calm: enabling caller and callee to collaborate," in *Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI'01)*, pp. 235–236, New York, NY, USA, 2001.

[20] S. Shanmugalingam, N. Crespi, and P. Labrogere, "My own communication service provider," in *Proceedings of the International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT'10)*, pp. 260–266, Moscow, Russia, October 2010.

[21] S. Shanmugalingam, N. Crespi, and P. Labrogere, "User mobility in a web-based communication system," in *Proceedings of the IEEE 4th International Conference on Internet Multimedia Services Architecture and Application (IMSAA'10)*, Bangalore, India, December 2010.

[22] E. Karmouch, A. Nayak, P. Martin, and H. Hassanein, "Experimenting with mobile context-aware sip-based multimedia communications," in *Proceedings of the 1st International Global Information Infrastructure Symposium (GIIS'07)*, pp. 6–13, Marrakech, Morocco, July 2007.

[23] B. E. Saghir and N. Crespi, "A generic layer model for context-aware communication adaptation," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'08)*, pp. 3027–3032, Las Vegas, Nev, USA, April 2008.

[24] O. Boyaci, V. Beltran, and H. Schulzrinne, "Bridging communications and the physical world," *IEEE Internet Computing*, vol. 16, no. 2, pp. 35–43, 2012.