Università degli Studi di Napoli Federico II

Facoltà di Ingegneria

Dottorato di Ricerca in Ingegneria Informatica ed Automatica
XXV Ciclo
Dipartimento di Informatica e Sistemistica

Real-time and content-aware applications and infrastructure: a holistic approach dealing with architectural, performance and security issues

Roberta Presta

Ph.D. Thesis

Tutors

Prof. Simon Pietro Romano
Prof. Giorgio Ventre

Coordinator

Prof. Francesco Garofalo

March 2013

Abstract

This thesis has been carried out in the field of Computer Networks.

We have looked after two challenging and popular multimedia applications over the Internet, namely Real-time Multimedia Applications, considering Multimedia Conferencing and Voice over IP (VoIP) services in general, and Content-Oriented Applications, that are multimedia web services such as User Generated Contents (UGC) platforms and Online Social Networks. We have conducted our studies starting from a thorough review of both functional and non-functional issues related to the above mentioned application families.

We first consider a number of different facets of Real-time Multimedia Applications, dealing with architectural, performance and security issues. We embrace an engineering approach, by involving modeling, implementation and simulation. We try and adhere to standard architectures by also actively contributing to them, arriving at the definition and completion of a fully-fledged standards-compliant web conferencing platform. We propose original contributions to face performance and security issues. With respect to the former aspect, we address scalability and performance analysis by leveraging formal methods. As to the latter point, we design an effective profiling system for VoIP users which is able to detect an interesting class of VoIP threats represented by "social" attacks.

In the second part of the thesis, we focus on Content-Oriented Applications, once again basing our work on a study of current architectural and performance challenges. We propose a framework for the performance optimization of content-delivery based on the awareness of the contents to be delivered, as well as of the users' behavior in terms of mobility, request patterns and content preferences.

Web conferencing systems have gained a growing attention during the last years. Besides real-time multi-point video communications, they allow real-time collaboration by remote, making companies able to dramatically reduce travel expenses and improve their pollution impact in terms of CO2 emission. Based on VoIP technologies, web conferencing architectures have

been the subject of standardization efforts within the Internet Engineering Task Force (IETF). Such efforts have mainly focused on centralized frameworks as defined by the Centralized Conferencing (XCON) Working Group (WG). A state of the art analysis of the standard multimedia conferencing architectures is provided in Chapter 1. We have actively contributed to such efforts by designing and implementing one of the most important protocols involved in the standard framework devoted to the conference control [18], the Centralized Conferencing Manipulation Protocol (CCMP), and by writing Internet drafts that have now become *Request For Comments* (RFC) [17]. We have contributed both to the definition of the protocol and to the documentation of *best current practices* in real-world conferencing scenarios by exploiting Meetecho, a standards-compliant multimedia conferencing and collaboration platform developed at the Computer Engineering Department of the University of Napoli Federico II [8]. Meetecho has been conceived at the outset to be extensible towards a distributed architecture, yet being fully compliant with the XCON specification, in order to better fulfill scalability requirements resulting from its large-scale deployment. Details on the design and implementation activity in this field are provided in Chapter 2.

Experimental campaigns have demonstrated the performance improvements attainable with the distribution of the centralized conferencing framework, but also show the strong dependency of the performance results on the conference features that have been configured by means of the conference control protocol. From the perspective of a service provider having the possibility to turn on multiple conferencing server nodes, the ability of assessing the performance behavior of the system under different configurations looks definitely appealing. To this aim, we model the conferencing system by means of formal methods [57] and validate the model against the results obtained from real-world measurements in Chapter 3. We demonstrate the flexibility of our model-based performance evaluation approach by simulating different scenarios with varying distribution and conferencing settings.

Thanks to the experience gained while working on standard multimedia conferencing architectures, we are further contributing to current standardization activities (Chapter 4), by focusing in particular on telepresence systems. Telepresence systems provide high definition, high quality audio/video, enabling a “being-there” experience. We document current work we are carrying out within the *ControLling mUltiple streams for tElepresence* (CLUE) IETF WG on the design of both the framework and the data model [67], as well as on the definition of the orchestration of both existing and new standard protocols to the purpose.

Standard VoIP technologies like the Session Initiation Protocol (SIP) and the Real-time Transport Protocol (RTP) are the foundation of advanced real-time multimedia services such as conferencing and telepresence. Basic VoIP services are nowadays offered by most telecom providers and, by looking at their growing diffusion, they will probably completely replace the circuit-based telephony in the future. Compared to old telephony systems, securing a VoIP platform is a much more challenging task, since such a platform is exposed to both IP network attacks and new application-specific threats. In Chapter 5, we deal with issues related to security of real-time communications over the Internet. In particular, we focus on the detection of social attacks, i.e., attacks that are carried out by directly contacting VoIP users with malicious purposes (such as telemarketing and phishing). We propose a behavioral model for VoIP users that allows for the detection of anomalous social behaviors in VoIP networks by using unsupervised clustering techniques [30]. We test our approach on a very large dataset of VoIP calls provided by an important Italian Telecom Operator and illustrate the results of our analysis.

Finally, we consider the challenges raised by Content-Oriented Applications, that are the leading actors of the dominant multimedia entertainment Internet traffic. With such an explosion of mobile access to multimedia web contents, there is an unmet demand for a higher quality and speed in multimedia content and service delivery to the connected devices. The network has become a dynamic space of contents, populated by users, as well as explored by them. The use of the network has then moved from a host-centric to a content-centric paradigm. Given the above considerations, Internet researchers are starting to re-consider and re-design the network architecture according to the Content-Centric approach. While there are disruptive proposals focusing on the definition from scratch of the network core, a different kind of solution envisions an overlay infrastructure on top of the existing host-centric platform. Content Distribution Networks are one of the most representative architectural examples of such an alternative. We propose to leverage knowledge about users' behavior in order to optimize the performance of content distribution networks. In particular, we identify three main research areas dealing with, respectively, user mobility patterns, user request patterns, and user content preferences.

We first investigate how to exploit users' preferences and community analysis to optimize content delivery within the context of the Behavior-Aware Content-Oriented Networking (BACON) project, that has been financed by Telecom Italia within the context of a national initiative named Working Capital and aimed to support innovation in our country. We design a behavior-aware content-oriented framework exploiting users' preferences for contents, expressed as ratings. The context of the aforementioned work, together with information about the current achievements in the field of mobility and request patterns are provided in Chapter 6. In Chapter 7 we report some preliminary experimental results on the interest-based community detection needed to apply the behavior-aware content distribution strategy we designed. We therein further introduce an ad-hoc Content Delivery Network (CDN) simulation framework we conceived for assessing the effectiveness of different content distribution strategies in the presence of realistic request workloads. The simulator is indeed able to reproduce web users' behaviors in terms of mobility and request patterns according to the most realistic models we found in the literature. Validation results about the accuracy of the models' implementation are also provided.

Contents

Chapter 1

Real-time applications and infrastructure part I - current achievements

1.1 Introduction

Multimedia conferencing is one of most challenging real-time multimedia application over the Internet. It imposes a number of stringent requirements to the underlying network infrastructure. First, the intrinsic multimedia nature of a conference (which typically involves a combination of audio, video, instant messaging, desktop sharing, etc.) requires coping with complex management issues. Second, the real-time features of conference-based communication demands a high level of Quality of Service (QoS).

In this chapter we provide the reader with the background needed to place our contribution in the field of standard multimedia conferencing over the Internet. We indeed both participate in the IETF standards design and definition and produce prototype implementation, as documented in the next chapter.

The Internet Engineering Task Force (IETF) is the most outstanding open international community concerned with the evolution of the Internet architecture and protocols. In particular, standards related to multimedia conferencing have been developed witihin the IETF RAI (Real-time Application and Infrastructure) area.

One of the first RAI Working Group (WG) attempting to standardize the multimedia conferencing service was the Session Initiation Proposal Investigation (SIPPING) WG. It developed the very first framework for multi-party conferencing based on the SIP protocol [72]. The Session Initiation Protocol (SIP) [73] is the most widespread signaling protocol for IP networks, providing users with the capability to initiate, manage, and terminate communication sessions and natively supporting many models of multi-party communications. Conferencing models considered by the SIPPING WG are documented in Section 1.2.

In particular, the centralized model has been object of further standardization efforts. The Centralized Conferencing (XCON) WG was born to focus on centralized architectures "signalling-agnostic", i.e., transparent to the employed signaling protocol. The XCON architecture is described in Section 1.3. Our standardization work has been carried on that architecture, in particular on the control plane of centralized conferencing.

A further IETF contribution which is worth to be mentioned is the one of the Media Control (MEDIACTRL) WG, devoted to dealing with the separation of concerns among the entities involved in multimedia sessions (Section 1.4).

To complete the panorama on current achievements on centralized conferencing standards, we finally refers to the 3GPP IMS conferencing support in Section 1.5.

While centralized conferencing has been paid strong standardization efforts in last years, distributed architecture are still an open field for research. Pioneering examples of proposals of such kind are documented in Section 1.6.

1.2 Conferencing with the Session Initiation Protocol (SIP)

The SIPPING WG reports in [72] three major conferencing models:

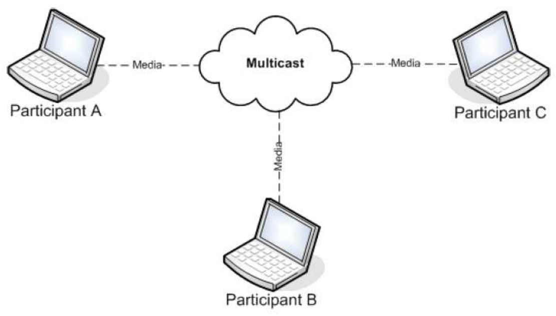


Figure 1.1: Loosely coupled conference

- *loosely coupled conferencing*

  This model makes use of multicast media groups (see Fig. 1.1). There is no SIP signaling relationship between participants in the conference, nor a central point of control. Participation is gradually learned through control information that is passed as part of the conference (using the Real-Time Control Protocol (RTCP), for example). Loosely coupled conferences are easily supported in SIP by using multicast addresses within its session descriptions.

- *fully distributed conferencing*

  Here, each participant maintains a SIP signaling relationship with the other participants (see Fig. 1.2). Again, there is no central point of control. The control function is completely distributed among participants .

- *tightly coupled conferencing*

  This third model, instead, envisages the presence of a central point of control to which each participant connects to. Such central point provides a variety of conference functions, and may possibly perform media mixing functions as well.

The last centralized scenario was the one fully defined in [72]. The central component is called *focus*, and maintains a SIP signaling relationship with

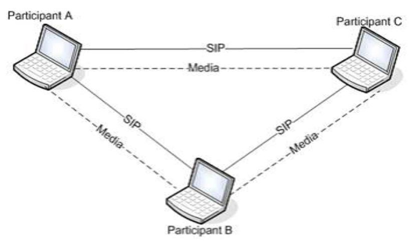


Figure 1.2: Fully distributed multiparty conference

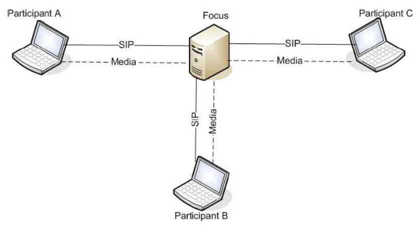


Figure 1.3: Tightly coupled conference

each participant in the conference. The result is a star topology, as depicted in Fig. 1.3. The focus is a SIP user agent that is addressed by a conference URI (a SIP URI) and identifies a conference.

The centralized architecture designed by the SIPPING WG constitutes a reference model for the following architecture proposals. It identifies the main actors of the framework (participants, focus, conference policy server) and defines the fundamental interactions among them. The conference policy server is a logical entity managing the conference policies, i.e., the complete set of rules governing a particular conference. The application of the conference policies on the managed conferences is against the focus. The focus is also responsible for ensuring, in some way, that each participant receives the conference media. It does that through the use of one or more mixers, each of which combines a number of input media streams to produce one or more output media streams. The focus uses the media policy to determine the proper configuration of the mixers. Finally, the focus is also responsible of the delivery of conference events (such as the joining of new participant in the conference, the egress of a participant from the conference, and the like) to participants, by acting as a notifier. Conference events are expressed as changes to the conference data model representing each instance of a conference managed by the system. That model is defined using the XML Schema language in RFC4575 [74] and contains information about participants, involved media streams, conference host information, as well as generical metadata about the conference.

1.3 Centralized Conferencing Framework (XCON)

The XCON framework (RFC5239, [15]) extends the SIPPING model by making it independent from the employed signaling protocol, while following the same principles and adopting the same terminology. Besides basic conferencing features, the XCON model offers richer functionality, by supporting different conference templates and providing standard protocols for managing and controlling conferences' attributes. Such protocols are depicted in Figure 1.4.

Namely, they are:

- A signaling protocol

  Signaling is necessary for the (multi-party) call initiation, management, and termination. Such task can be performed by any signalling protocol, being XCON signalling-agnostic. The server side of the protocol is played by the focus.

- A floor control protocol

  A "floor" is a token controlling the access to a pool of resources by participants. Participants need to access floor associated with the right to talk to transmit their audio in the conference. A floor control protocol enables the moderated access to such resources to participants. In moderated conferencing scenarios, a moderator

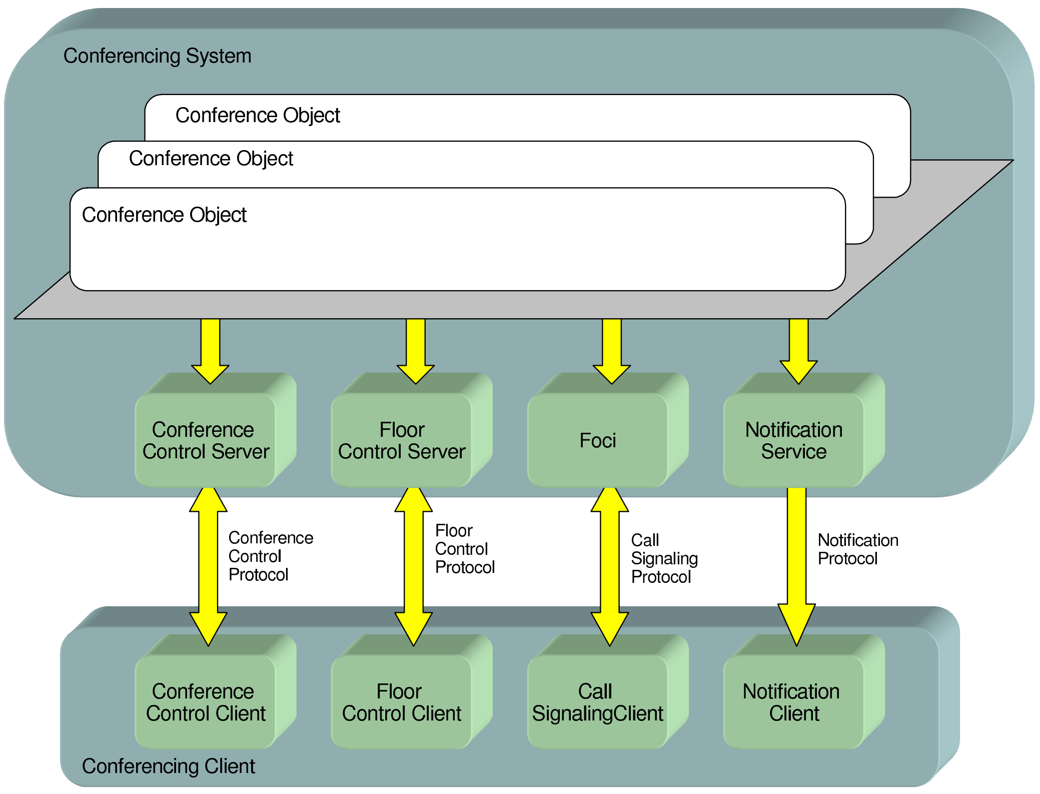


Figure 1.4: The XCON framework: protocols

("chair" of the conference) decides who can join or must release a certain floor by means of the floor control protocol. The Binary Floor Control Protocol (BFCP) defined in RFC4582 [27] is the standard candidate to do the task.

- A notification protocol

  The notification protocol is the mechanism needed to provide participants with information related to conference events, as already envisioned in the SIP conferencing framework. XCON conferencing events are described according to the formalism defined by RFC6502 [28], which extends conference information defined by the SIPPING WG in RFC4575.

- A conference control protocol

  A protocol devoted to the creation, update and canceling of conferences. The Centralized Conferencing Manipulation Protocol [16] is the latest output of the XCON WG for the management and manipulation of conferences. It has been one of the objects of our efforts. It represents a powerful means to control basic and advanced conference features (conference state and capabilities, participants, relative roles and details).

The XCON Framework introduces the concept of "conference object" as a logical representation of a conference instance, representing the current state and capabilities of a conference. To each XCON conference is uniquely associated an identifier named "XCON-URI".

Figure 1.5 illustrates the typical life cycle of a conference object in the XCON framework. At each instant in time, a conference object is associated with an XML representation compliant with the XCON data model specification provided in in RFC6501 [64]. The XCON data model describes all of the features of a conference, starting from its general description (purpose, hosting entity, status, etc.) and arriving at much more detailed information

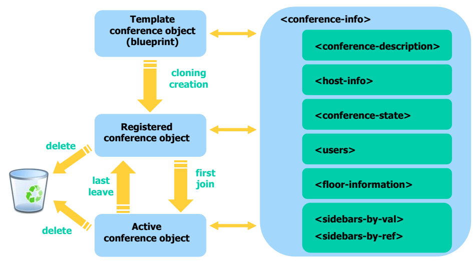


Figure 1.5: Conference object's life cycle

related to participants, available media and their features, floor informations, as well as sidebars associated with the conference (i.e. subconferences involving part of the users participating in the main conference).

Creation of conference objects is usually performed through a cloning operation, i.e. by replicating the structure of one of conference object templates (also known as "blueprints") available at the server. A newly created conference object is typically marked as "registered" until the first user joins the conference and it will stay "active" until either the last user leaves the conference (in which case it comes back to the "registered" state) or a user (holding the right to do so) deletes it.

The conference control protocol is the protocol used to manipulate conference objects during its lifetime. While other XCON protocols implicitly interact with conference objects somehow, the conference control protocol directly affects conferences by manipulating their logical representation. We directly intervene on definition, design and implementation of the control protocol, as it will be further documented in the following.

1.4 A standard framework for Media Server Control (MEDIACTRL)

The MEDIACTRL Working Group aims at specifying an architectural framework to properly cope with the explicit separation of responsibilities between an Application Server (AS), which is in charge of handling the services application logic, and a Media Server (MS), whose task is instead the low-level manipulation and delivery of media streams. The framework defined within such WG is described in RFC5567 [61].

The current specification of the framework envisages a modular approach when looking at the functionality to provide. This means that, inside the same MS, different inner components take care of different processing that may be required. Examples of operations a MS is supposed to be able to achieve include: mixing, transcoding and adaptation of media streams; manipulation of media streams (e.g. gain levels, video layouts, and so on); playing and recording of media streams from and to both local and network environments; tone and DTMF detection; Text-To-Speech and Speech Recognition; and so on.

To achieve this, the framework currently specifies a general invocation mechanism with opaque payloads, whereas such payloads can be directed to the proper component according to a header in the request itself. This way, new components providing additional media capability can be added at any time without affecting the specification of the core framework. Such components are called “control packages”. So far two different packages have been standardized: a package providing basic Interactive Voice Response (IVR) functionality (RFC6231 [59]), and a package for managing mixers for media conferences and connections (RFC6505 [60]).

When looking at the protocol itself, the interaction between an AS and a MS relies on a so-called “control channel”. This channel is where MEDIACTRL-related messages flow between the AS (the client side) and the MS (the server side). An AS would instruct a MS into a specific media operation by placing a request on this channel, and the MS would reply to such request, as well as notify events, through the same channel. Of course, such a channel can only be set up as the result of a transport-level connection. This implies that either the AS or the MS must previously know the transport address of the other party in order to set up the connection. To allow this, a COMEDIA-based approach has been standardized (RFC4145, [90]): AS and MS make use of a SIP dialog to negotiate and set up the control channel.

1.5 3GPP IP Multimedia Subsystem (IMS)

The 3GPP has worked on the specification of a tightly coupled conferencing service within the IP Multimedia Subsystem.

The IP Multimedia Subsystem (IMS) is a standardized Next Generation Networking (NGN) architecture that has been conceived for telecom operators willing to provide advanced services on top of both mobile and fixed networks [26]. In the IMS envisioned scenario, heterogeneous devices are supported and users have to be able to ubiquitously exploit the entire portfolio of available services, which entails support for roaming as well as for flexible and transparent adaptation to context changes. To achieve the aforementioned goals, IMS makes extended use of Voice over IP (VoIP) technologies and open IP protocols and adopts the principle of separation of concerns, which reflects in the definition of a number of core IMS components (such as Call/Session Control Function - CSCF, Home Subscriber Server - HSS, Media Resource Function - MRF and Application Server - AS), each in charge of looking after a specific function of the overall system. The identified components must also be scalable and able to provide advanced features, like five nine reliability.

Due to its recent birth, the IMS architecture is currently far from reaching its steady-state with respect to the complete definition of the overall infrastructure and related standards. Furthermore, to date only a few early trials and deployments of the architecture are underway. This leaves space to a number of open issues that still have to be faced, both at the infrastructure and at the service level, as well as, crosswise, for a systematic approach to the performance evaluation.

Both the requirements and the architecture for tightly coupled congerencing have been defined in [1]. The cited document indeed represents a sort of integrated specification within the IMS, aimed at harmonizing the combined use of existing standard protocols, like SIP, SIP Events, the Session Description Protocol (SDP) and the Binary Floor Control Protocol (BFCP). Based on the mentioned 3GPP specification, two logical planes can be identified:

the "control plane" and the "media plane". The control plane deals with issues related to the set-up of the multi-media multi-user communication, as well as those related to overcoming the heterogeneity of both the access networks and the end-user devices. The media plane faces all the matters related to the media flows transmitted and received by users, such as, for example, the transcoding across different media formats and the switching of different media streams among conferencing participants.

1.6 A distributed proposal for improvement

Based on the concept of achieving scalability through the cooperation of more centralized conferencing islands, the Distributed Conferencing (DCON) model is the first proposal of a fully-fledged distributed conferencing architecture. It has been presented in its different facets in several IETF drafts ([6, 5, 7]).

DCON is based on the idea that a distributed conference can be setup by appropriately orchestrating a set of XCON focus elements, each in charge of managing a certain number of participants distributed across a geographical network. In such a distributed scenario, each focus manages the associated local users who subscribed to the conference.

The focus in the conference initiator's home network must provide all other foci with up-to-date conference information. Foreign foci play a two-fold role. On the one hand, they act as a regular conference focus for the participants belonging to their managed network; on the other, they appear as normal participants to the focus in the conference initiator's home network.

Each single realm keeps on being based on a star-topology graph for all what concerns the call signaling part. The different stars are then connected through an upper-layer mesh-based topology providing inter-focus communication. The overall topology of the distributed conferencing scenario thus looks like an overlay network of focus entities.

This architecture requires a dedicated communication channel among the foci that are participating in the conference. This channel is used to propagate information about conferences, as well as to manage synchronization issues.

A coordination channel is needed in order to manage a distributed conference along its life-cycle. For instance, once a user decides to create a new conference, the corresponding conference focus has to distribute conference information to all other foci, so to enable other potential participants to retrieve the needed data and possibly subscribe to the event. Indeed, whenever a new conference is created (or an active conference changes its state) such an event has to be communicated to all interested (or active) participants. Given the intrinsic nature of the distributed framework, the actual flow of information will always foresee the interaction among conference focus entities for both conference information exchanging and state changes notifications.

While interaction between each participant and the corresponding conference focus is based on the standard XCON framework, the inter-focus interaction has been completely defined and specified from scratch.

Besides architectural details of the distributed solution, the work in [5] discuss the possibility of exploiting for inter-focus communications an Instant Messaging (IM) channel. The authors propose for such a role a Server-to-Server XMPP (Extensible Messaging and Presence Protocol (XMPP)) channel, being XMPP a standard IM protocol defined in RFC 6120.

The work in [7] on the other hand defines a protocol designed to get centralized protocols to work in a distributed environment. Two logical entities that are strictly coupled can be identified: (i) the XCON focus, i.e., the entity acting as a conference focus within the scope of its local conferencing cloud (ii) the DCON focus, i.e., the entity acting as the corresponding conference focus in the context of the distributed conferencing framework, in charge of communicate with the other clouds' foci. Between the XCON focus and its DCON counter part, it is needed a protocol that properly translates centralized conferencing protocols effects into the distributed environment. Such a protocol has been called XSDP (XCON-DCON Synchronization Protocol) since its task is to synchronize the XCON focus with its DCON peer.

Chapter 2

From theory to practice

2.1 Introduction

We described current achievements in centralized conferencing standardization in the previous chapter. Here we deal with our contribution, constituted by the design and implementation of the standard conference management protocol [18], that three years ago was still the missing piece of the standard picture. The standard protocol for conference control within the XCON architecture has been named Centralized Conferencing Manipulation Protocol (CCMP) and the draft describing it has recently reached the RFC stadium (RFC6503, [16]) after a long-lived refinement process. We first focus on the design of the protocol and then discuss how we implemented and integrated it in a real-world standard conferencing system, Meetecho [8]. Meetecho is a collaborative framework developed at the University of Napoli as an active playground for IETF standardization activities in the field of real-time applications and infrastructure. We then present, in section 2.3, a bird's eye view of the Centralized Conferencing Manipulation Protocol. The same section also provides some insights on the history of the overall specification process. Section 2.4 drills down on the specific messages that can be carried inside the body of the CCMP protocol, while section 2.5 concludes the part associated with our standardization work by depicting a typical call flow associated with a CCMP-based interaction between a conferencing client and an XCON Conferencing server. The second part of the chapter is entirely devoted to the implementation of CCMP within the Meetecho system and how such a system has been made fully compliant with the XCON standards. The implementation of the centralized conferencing control protocol has been heavily used to both test the protocol's behavior and provide very useful feedback to the authors of the CCMP document. Furthermore, to aid implementors, a specific draft focusing on call flows has been written in order to provide the Internet community with guidelines in the form of Best Common Practices. The outcome of our work has recently become an RFC (RFC6504, [17]).

2.2 Centralized Conferencing Manipulation Protocol

The Centralized Conferencing Manipulation Protocol (CCMP) is the latest output produced by the IETF XCON working group. As we have already mentioned, that protocol has been conceived at the outset as a light-weight protocol allowing conferencing clients to access and manipulate objects describing a centralized conference. The CCMP is a state-less, XML-based, client-server protocol carrying in its request and response messages conference information in the form of XML documents and fragments conforming to the centralized conferencing data model schema. It represents a powerful means to control basic and advanced conference features, such as conference state and capabilities, participants and relative roles and details. CCMP allows authenticated and authorized users to create, manipulate and delete conference objects. Operations on conferences include adding and removing participants, changing their roles, as well as adding and removing media streams and associated end points. CCMP is based on a client-server paradigm and is specifically suited to serve as a conference manipulation protocol within the XCON framework, with the Conference Control Client and Conference Control Server acting as client and server, respectively. The

CCMP uses HTTP as the protocol to transfer requests and responses, which contain the domain-specific XML-encoded data objects defined in the Conference Information Data Model for Centralized Conferencing (XCON Data Model, [64]).

2.3 Protocol overview

CCMP is a client-server, XML-based protocol, which has been specifically conceived to provide users with the necessary means for the creation, retrieval, modification and deletion of conference objects. CCMP is also stateless, which means implementations can safely handle transactions independently from each other. Conference-related information is encapsulated into CCMP messages in the form of XML documents or XML document fragments compliant with the XCON data model representation.

The core set of objects manipulated in the CCMP protocol includes conference blueprints, conference objects, users, and sidebars. CCMP is completely independent from underlying protocols, which means that there can be different ways to carry CCMP messages across the network, from a conferencing client to a conferencing server. Indeed, there have been a number of different proposals as to the most suitable transport solution for the CCMP. It was soon recognized that operations on conference objects can be implemented in many different ways, including remote procedure calls based on SOAP [43] and by defining resources following a RESTful [38] architecture. In both approaches, servers will have to recreate their internal state representation of the object with each update request, checking parameters and triggering function invocations. In the SOAP approach, it would be possible to describe a separate operation for each atomic element, but that would greatly increase the complexity of the protocol. A coarser-grained approach to the CCMP does require that the server process XML elements in updates that have not changed and that there can be multiple changes in one update. For CCMP, the resource (REST) model might appear more attractive, since the conference operations nicely fit the so-called *CRUD (Create-*

Retrieve-Update-Delete) approach. Neither of these approaches was finally selected. SOAP was not considered to be general purpose enough for use in a broad range of operational environments. Similarly, it was deemed quite awkward to apply a RESTful approach since CCMP requires a more complex request/response protocol in order to maintain the data both in the server and at the client. This doesn't map very elegantly to the basic request/response model, whereby a response typically indicates whether the request was successful or not, rather than providing additional data to maintain the synchronization between the client and server views.

The solution for the CCMP at which we arrived can be viewed as a good compromise amongst the above mentioned candidates and is referred to as “HTTP single-verb transport plus CCMP body”. With this approach, CCMP is able to take advantage of existing HTTP functionality. As with SOAP, it uses a “single HTTP verb” for transport (i.e. a single transaction type for each request/response pair); this allows decoupling CCMP messages from HTTP messages. Similarly, as with any RESTful approach, CCMP messages are inserted directly in the body of HTTP messages, thus avoiding any unnecessary processing and communication burden associated with further intermediaries. With this approach, no modification to the CCMP messages/operations is required to use a different transport protocol. That has been the approach that finally has been selected for CCMP.

We will show how the CCMP protocol inserts XML-based CCMP requests into the body of HTTP POST operations and retrieves responses from the body of HTTP 200 OK messages. CCMP messages will have a MIME-type of “application/ccmp+xml”, which appears inside both the “Content-Type” and “Accept” fields of HTTP requests and responses.

2.3.1 Protocol Operations

The main operations provided by CCMP belong in four general categories:

- create: for the creation of a conference, a conference user, a sidebar, or a blueprint;

- retrieve: to get information about the current state of either a conference object (be it an actual conference or a blueprint, or a sidebar) or a conference user. A retrieve operation can also be used to obtain the XCON-URIs of the current conferences (active or registered) handled by the conferencing server and/or the available blueprints;
- update: to modify the current features of a specified conference or conference user;
- delete: to remove from the system a conference object or a conference user.

Thus, the main targets of CCMP operations are: (i) conference objects associated with either active or registered conferences; (ii) conference objects associated with blueprints; (iii) conference objects associated with sidebars, both embedded in the main conference (i.e. `<entry>` elements in `<sidebars-by-value>`) and external to it (i.e. whose XCON-URIs are included in the `<entry>` elements of `<sidebars-by-ref>`); (iv) `<user>` elements associated with conference users; (v) the list of XCON-URIs related to conferences and blueprints available at the server, for which only retrieval operations are allowed.

Each operation in the protocol model is atomic and either succeeds or fails as a whole. The conference server must ensure that the operations are atomic in that the operation invoked by a specific conference client completes prior to another client's operation on the same conference object. The details for this data locking functionality are out of scope for the CCMP protocol specification and are implementation specific for a conference server. Thus, the conference server first checks all the parameters, before making any changes to the internal representation of the conference object.

For example, it would be undesirable to change the `<subject>` of the conference, but then detect an invalid URI in one of the `<service-uris>` and abort the remaining updates.

Also, since multiple clients can modify the same conference objects, conference clients should first obtain the current object from the conference server and then update the relevant data elements in the conference object prior to invoking a specific operation on the conference server. In order to effectively manage modifications to conference data, a versioning approach is exploited in the CCMP. More precisely, each conference object is associated with a version number indicating the most up to date view of the conference at the server's side. Such version number is reported to the clients when answering their requests. A client willing to make modifications to a conference object has to send an update message to the server. In case the modifications are all successfully applied, the server sends back to the client a "success" response which also carries information about the current server-side version of the modified object. With such approach, a client which is working on version "X" of a conference object and finds inside a "success" response a version number which is "X+1" can be sure that the version it was aware of was the most up to date. On the other hand, if the "success" response carries back a version which is at least "X+2", the client can detect that the object that has been modified at the server's side was more up to date than the one it was working upon. This is clearly due to the effect of concurrent modification requests issued by independent clients. Hence, for the sake of having available the latest version of the modified object, the client can send to the conference server a further "retrieve" request. In no case a copy of the conference object available at the server is returned to the client as part of the update response message. Such a copy can always be obtained through an ad-hoc "retrieve" message. Based on the above considerations, all CCMP response messages carrying in their body a conference document (or a fragment of it) must contain a "version" parameter. This does not hold for request messages, for which the version parameter is not at all required, since it represents useless information for the server: as long as the required modifications can be applied to the target conference object with no conflicts, the server does not care whether or not the client had an up to date view of the information stored at its side. This said, it stands clear that a client which has subscribed at the server, through the XCON event package [28], to notifications about conference object modifications, will always have the most up to date version of that object available at his side.

A final consideration concerns the relation between the CCMP and the main entities it manages, i.e. conference objects. Such objects have to be compliant with the XCON data-model, which identifies some elements and attributes as mandatory. From the CCMP standpoint this can become a problem in cases of client-initiated operations, like either the creation or the update of conference objects. In such cases, not all of the mandatory data can be known in advance to the client issuing a CCMP request. As an example, a client has no means to know, at the time it issues a conference creation request, the XCON-URI that the server will assign to the yet-to-be-created conference and hence it is not able to appropriately fill with that value the mandatory 'entity' attribute of the conference document contained in the request. To solve this kind of issues, the CCMP will fill all mandatory data model fields, for which no value is available at the client at the time the request is constructed, with fake values in the form of wildcard strings (e.g. AUTO_GENERATE_X, with X being an incremental index initialized to a value of 1). Upon reception of the mentioned kinds of requests, the server will: (i) generate the proper identifier(s); (ii) produce a response in which the received fake identifier(s) carried in the request has (have) been replaced by the newly created one(s). With this approach we maintain compatibility with the data model requirements, at the same time allowing for client-initiated manipulation of conference objects at the server's side (which is, by the way, one of the main goals for which the CCMP protocol has been conceived at the outset).

2.4 CCMP messages

As anticipated, CCMP is a request/response protocol. Besides, it is completely stateless, which explains why HTTP has been chosen as the perfect

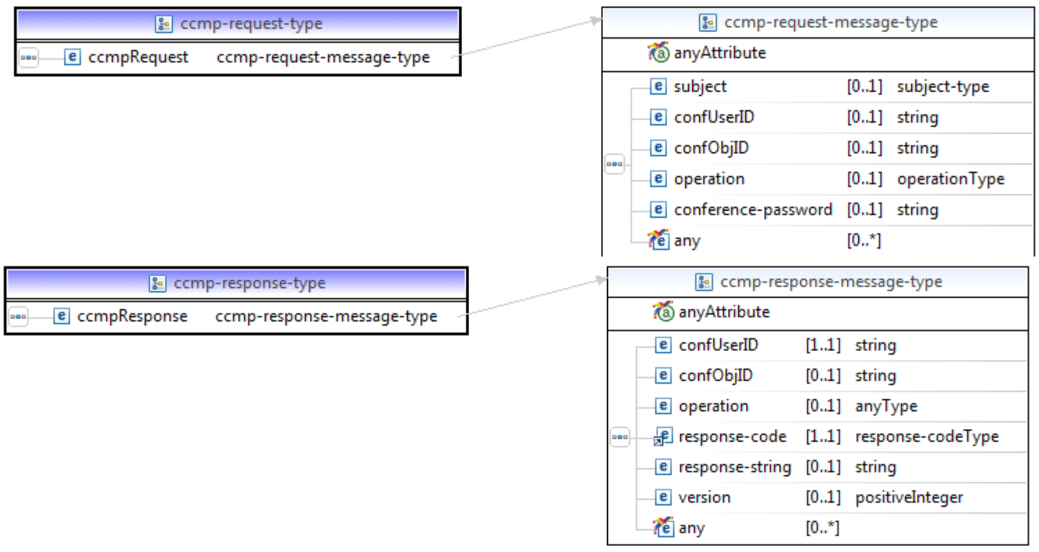

ccmp-request-type
ccmpRequest ccmp-request-message-type
ccmp-request-message-type
anyAttribute
subject [0..1] subject-type
confUserID [0..1] string
confObjID [0..1] string
operation [0..1] operationType
conference-password [0..1] string
any [0..*]
ccmp-response-type
ccmpResponse ccmp-response-message-type
ccmp-response-message-type
anyAttribute
confUserID [1..1] string
confObjID [0..1] string
operation [0..1] anyType
response-code [1..1] response-codeType
response-string [0..1] string
version [0..1] positiveInteger
any [0..*]

Figure 2.1: CCMP Request and Response messages

transport candidate for it.

For what concerns the protocol by itself, both requests and responses are formatted basically in the same way, as depicted in Figure 2.1. In fact, they both have a series of heading parameters, followed by a specialized message indicating the particular request/response (e.g., a request for a specific blueprint). This makes it quite easy to handle a transaction in the proper way and map requests and related responses accordingly.

For what concerns the shared parameters:

- `confUserID` indicates the participant making the request;
- `confObjID` indicates the conference the request is associated with;
- operation specifies what has to be done, according to the specialized message that follows.

Other parameters are defined which are more strictly related to either requests or responses. There is, for instance, a "password" parameter participants may need to provide in CCMP requests for password-protected conferences, as well as a "response-code" parameter (which is carried just by responses) providing information about the result of a requested operation.

That said, the core of a CCMP message is actually the specialized part. In fact, as stated in the previous section, the CCMP specification describes several different operations that can be made on a conference object, namely: (i) blueprints retrieval, (ii) conference creation and manipulation, (iii) users management, (iv) sidebar-related operations. All these operations have one or more specialized message formats, instead of a generalized syntax, in order to best suit the specific needs each operation may have.

Indeed, requesting a blueprint and adding a new user to a conference have very different requirements for what concerns the associated semantics level, and as such they need different modes of operation. This is reflected in what is carried in the specialized message body, which will always contain information (compliant with the XCON common data model specification) strictly related to the operation it is associated with. The specialization of the message then allows for an easier and faster management at the implementation level.

To better highlight the considerations above, we show in Figure 2.2 the structure of a CCMP `confRequest` message, which is used in all operations concerning the manipulation and control of an entire conference object. As described in the picture, each such message is a specialization of the general CCMP request message, specifically conceived to transport, through the `confInfo` element, an XCON-compliant conference object (i. e. an object whose representation conforms to the common data model specification) towards the CCMP server.

2.5 CCMP sample call flow

To better clarify how a CCMP transaction can occur, this section presents a sample call flow. This example comes from a real implementation deployment, as it will be explained in 2.8, providing some details on our implementation experience with the protocol. Specifically, we will address the way we designed the process according to the specification (from both a client and server perspective), and the related implementation choices.

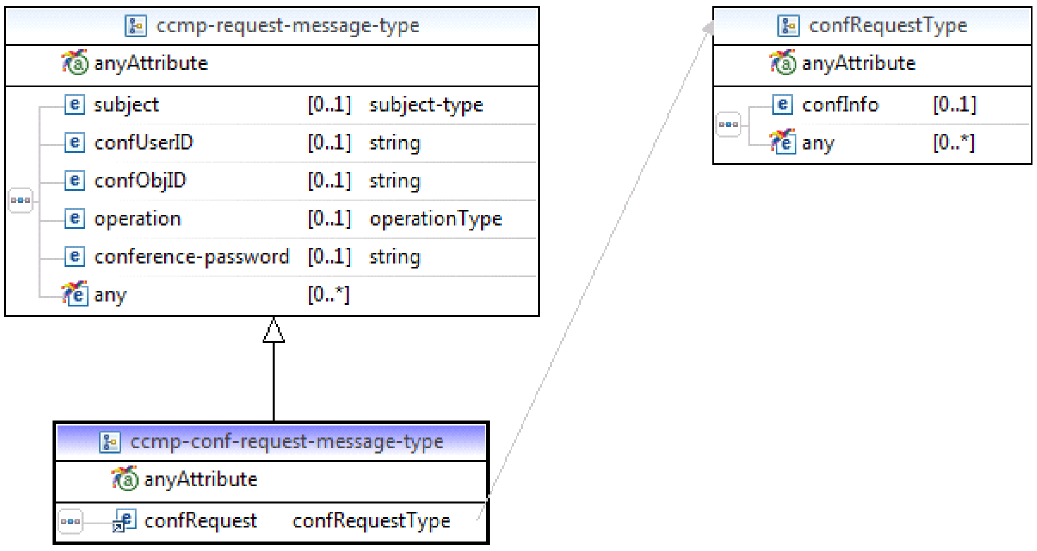

ccmp-request-message-type
anyAttribute
subject [0..1] subject-type
confUserID [0..1] string
confObjID [0..1] string
operation [0..1] operationType
conference-password [0..1] string
any [0..*]
confRequestType
anyAttribute
confInfo [0..1]
any [0..*]
ccmp-conf-request-message-type
anyAttribute
confRequest confRequestType

Figure 2.2: CCMP confRequest message

For the sake of conciseness, we chose a very simple example of call flow, which nevertheless provides the reader with a general overview of both CCMP requests and responses. As mentioned previously, HTTP is suggested by the CCMP specification as a transport for the protocol messages, and Figure 2.3 shows the typical request/response paradigm involved in that case.

As it can be seen, the CCMP request (in this case, a <`blueprintRequest`>) is sent by an interested participant to the conference server. This request is carried as payload of an HTTP POST message:

```
POST /Xcon/Ccmp HTTP/1.1
Content-Length: 657
Content-Type: application/ccmp+xml
Host: example.com:8080
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.0.1 (java 1.5)

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpRequest
      xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
  <ccmpRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="ccmp:ccmp-blueprint-request-message-type">
        <confUserID>xcon-userid:Alice@example.com</confUserID>
        <confObjID>xcon:AudioRoom@example.com</confObjID>
        <operation>retrieve</operation>
        <ccmp:blueprintRequest/>
  </ccmpRequest>
</ccmp:ccmpRequest>
```

The Content-Type header instructs the received that the content of the message is a CCMP message (application/ccmp+xml). For what concerns

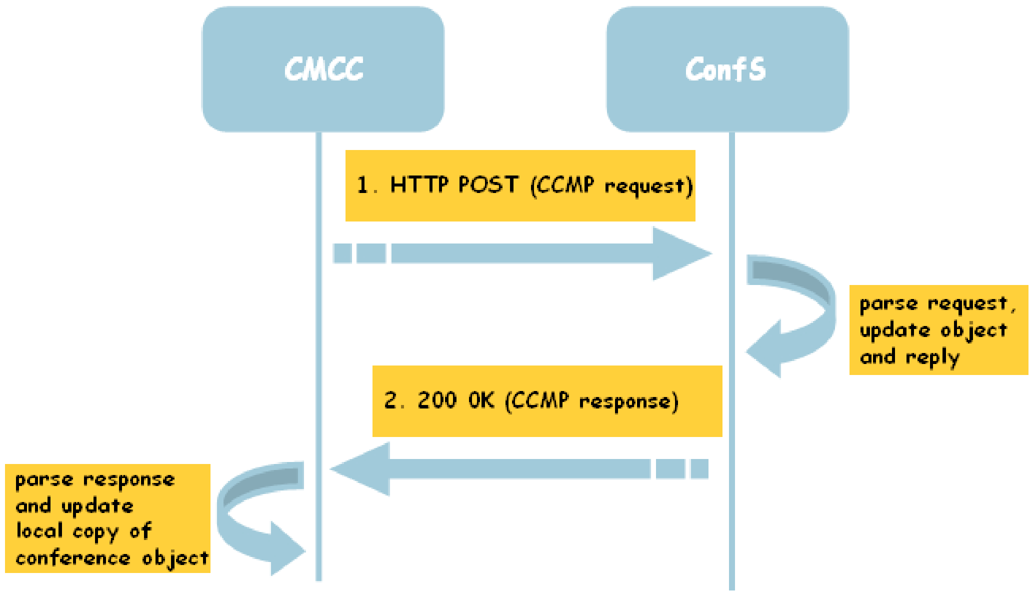

CMCC
ConfS
1. HTTP POST (CCMP request)
parse request, update object and reply
2. 200 OK (CCMP response)
parse response and update local copy of conference object

Figure 2.3: CCMP transported in HTTP

the request itself, as mentioned, it is a “blueprintRequest”: this means that the participant is interested to the details of a specific blueprint availabe on the server. This is reflected by the specialized part of the message, i.e., the <`ccmp:blueprintRequest`> element. The element is empty since no further details need to be provided in this case: this is not always true, since a participant may often be required to provide additional details in a request in order to see it fulfilled (e.g., when adding a new user to a conference). The generic parameters introduced in the previous section are also provided as part of the request: “confUserID” addresses the requester (Alice’s XCON URI), “confObjID” in this case addresses the blueprint to retrieve (as an XCON conference URI), while “operation” clarifies what needs to be done according to the request (retrieve the blueprint).

The CCMP response, inturn, is carried as payload of an HTTP 200 OK reply to the previous POST:

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/2.5
Server: Sun GlassFish Communications Server 1.5
Content-Type: application/ccmp+xml;charset=ISO-8859-1
Content-Length: 1652
Date: Thu, 04 Feb 2010 14:47:56 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ccmp:ccmpResponse
      xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
      xmlns:info="urn:ietf:params:xml:ns:conference-info"
      xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
  <ccmpResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="ccmp:ccmp-blueprint-response-message-type">
    <confUserID>xcon-userid:Alice@example.com</confUserID>
    <confObjID>xcon:AudioRoom@example.com</confObjID>
    <operation>retrieve</operation>
    <response-code>success</response-code>
    <ccmp:blueprintResponse>
      <blueprintInfo entity="xcon:AudioRoom@example.com">
        <info:conference-description>
           <info:display-text>AudioRoom</info:display-text>
           <info:maximum-user-count>2</info:maximum-user-count>
           <info:available-media>
             <info:entry label="audioLabel">
                 <info:type>audio</info:type>
             </info:entry>
             </info:available-media>
        </info:conference-description>
        <info:users>
           <xcon:join-handling>allow</xcon:join-handling>
        </info:users>
        <xcon:floor-information>
          <xcon:floor-request-handling>confirm
          </xcon:floor-request-handling>
          <xcon:conference-floor-policy>
                <xcon:floor id="audioLabel"></xcon:floor>
          </xcon:conference-floor-policy>
        </xcon:floor-information>
      </blueprintInfo>
    </ccmp:blueprintResponse>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

As a reply to a "blueprintRequest" message, the CCMP response includes a "blueprintResponse" specialized message in its body: this element includes the whole conference object (compliant with the XCON common data model specification) associated with the requested blueprint URI, as part of a <blueprintInfo> container. The same parameters the participant provided in the request can be found in the response as well (confUserID, confObjID, operation), together with an additional piece of information related to the result of the request: specifically, a "response-code" parameter tells the participant the request was successfully taken care of (success).

2.6 Call flows advanced scenarios

RFC6504 [17] provides guide lines about how to manage conference control by means of CCMP in several conferencing scenarios. We have therein reported CCMP dialogs for each use case.

The document [17] provides a sampling of detailed call flows that can be implemented based on a system realization of the XCON framework and implementation of CCMP. It is intended to be a simple guide for the use of the conference control protocol between the conference server and the conference control client. The objective is to provide an informational base reference for protocol developers, software developers, and researchers. The scenarios are based on those described in the XCON framework, many of which are based on the advanced conferencing capabilities described in the XCON scenarios. Additional scenarios have been added to provide examples of other real-life scenarios that are anticipated to be supported by the framework.

Basic conferencing scenarios are those related to conference creation and deletion. Besides the creation from a system blueprint, we have shown how it is possible to clone an existing conference, as well as how to create a new conference on the basis of user preferences expressed by means of CCMP.

We have illustrated also users manipulations examples. They show how an authorized CCMP-enabled user can add or remove a participant from the conference, or mute one of the participants, or join a password-protected conference.

While creating conferences and manipulating users and their media are sufficient for many scenarios, there may be cases when more complex management is needed. In fact, a feature typically required in conferencing systems is the ability to create sidebars. A sidebar is basically a child conference that usually includes a subset of the participants of the parent conference and a subset of its media as well. Sidebars are typically required whenever some of the participants in a conference want a private discussion, without interfering with the main conference. We have dealt with some typical scenarios using a sidebar, like whispering, private messaging, and coaching scenarios.

Whispering is a typical sidebar use case. In the reported example, there is a participant, say Alice, that is involved in a conference and wants to create a sidebar to have a side discussion with one of the other participants involved, say Bob. Alice may want still viewing the video associated with the main conference and mantain the main conference audio at a reduced volume, while keeping other participants unaware of her sidebar discussion with Bob. By means of sidebar-specific CCMP request, Alice can interact with the conferencing server to create the sidebar according to the aforementioned constraints.

Observing and coaching is one of the most interesting sidebar-related scenarios. In fact, it highlights two different interactions that have to be properly coordinated. A call center scenario is used to provide an example for such a use case. In this example, call center agent Bob is involved in a conference with customer Carol. Since Bob is a new agent and coach Alice sees that he has been on the call with Carol for longer than normal, she decides to observe the call and coach Bob as necessary. Of course, the conferencing system must make sure that the customer Carol is not aware of the presence of the coach Alice. This makes the use of a sidebar necessary for the success of the scenario. We have demonstrated how all the control operations needed to do the job can be performed by means of CCMP.

2.7 CCMP history and related work

Every time a framework for conferencing has been proposed, the need for a proper Conference Control mechanism has arisen as a consequence. For this reason, such mechanism has been the subject of a lot of efforts. Nevertheless, the proprietary nature of most of the conferencing solutions currently available paved the way for numerous heterogeneous and incompatible solutions for such a functionality. For the sake of conciseness, we don't provide in this section a list of such solutions, considering it would be quite incomplete. We instead focus on the related work carried within the standardization bodies. In fact, since the XCON architecture has been introduced within the IETF, several different candidates have been proposed to carry the role of the Conference Control Protocol. Such candidates differed in many aspects, which reflected the discussion within the standardization fora with respect to the approach that should be taken in that sense. An interesting debate took place, for instance, about whether a semantic or a syntactic approach would be better as a basis for a Conference Control Protocol. Besides, the best transport means to be adopted has also been the subject of investigation.

In this spirit, at least three candidates were proposed in the XCON WG before CCMP was chosen as the official protocol.

The first proposal, at the end of 2004, was the "Centralized Conference Control Protocol" (`draft-levin-xcon-cccp`) by O. Levin and G. Kimchi. This protocol, as CCMP, was XML-based and had a client-server organization, but unlike CCMP it was designed using SOAP as a reference model. It likely reflected the implementation work carried out within Microsoft at the time. Despite being in a quite advanced state (four updates were submitted), the proposal was eventually put aside.

Shortly thereafter the first individual submission of the CCCP, another candidate came to the light, "COMP: Conference Object Manipulation Protocol" (`draft-schulzrinne-xcon-comp-00`) by H. Schulzrinne. Like its predecessors, it heavily relied on Web Services as a reference, while stressing the use of SIP for notification purposes. Unlike CCCP, COMP had a strong semantic approach for what concerned the protocol specification. No updates published to the draft, which nevertheless paved the way for stimulating discussion that eventually led to CCMP.

One month later, another candidate was proposed, the "Conference State Change Protocol (CSCP)" (`draft-jennings-xcon-cscp`) by C. Jennings and A. Roach. Unlike both its predecessors, CSCP took a completely different approach towards the protocol. In fact, CSCP was basically a proposal to extend the already defined Binary Floor Control Protocol in order to allow it to also deal with conference manipulation functionality. CSCP motivated

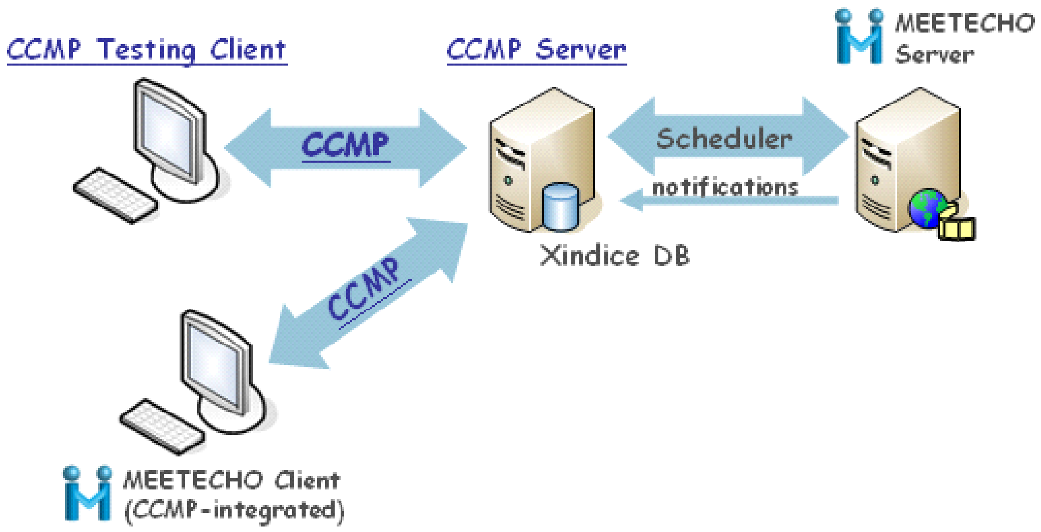


Figure 2.4: Reference scenario @ unina

such an approach stressing the fact that binary messages would be smaller and easier to handle, especially for mobile devices. Besides, it was the authors' opinion that every XCON-compliant entity would likely support BFCP already, and as such CSCP would prove a trivial addition. Nevertheless, the proposal was eventually abandoned, and a text-, possibly XML-based solution was decided to be a preferred approach.

Finally, a last proposal saw the light at the end of 2005, the individual submission that would subsequently become the official CCMP draft. Such a draft has seen many revisions and efforts since then, which have resulted in the RFC 6503 we strongly contributed in.

2.8 A real-world implementation

This section deals with our prototype implementation of the CCMP protocol. The reference scenario is the one depicted in Figure 2.4.

As the figure shows, in order to have a working instance of the CCMP protocol which could be used as a playground for testing and validation of the specification in progress, as a first step we have realized a stand-alone Java-based CCMP client and a Java-based CCMP server.

The CCMP testing client presents a very simple graphical user interface through which it is possible to create and send to the CCMP Server the desired CCMP request. All CCMP messages sent and received by the client are logged onto a debugging window which allows to easily visualize the entire call flow associated with client-server interactions.

As to the server, it has been integrated, since the beginning, with the Meetecho conferencing platform. Since Meetecho already makes use of a "proprietary" protocol[1] for conference creation, manipulation and scheduling (which is herein called "scheduler"), we had to implement the CCMP server as a proxy towards it. The CCMP server receives CCMP requests from the testing client, converts them into scheduler requests and forwards them to the Meetecho server by using the Meetecho scheduler protocol, which is a simple, text-based protocol based on TCP. When the Meetecho server is done with the forwarded request, it sends back to the CCMP server a scheduler-compliant answer, which is then converted into a CCMP-compliant response and forwarded to the CCMP testing client. The CCMP server takes care of the correct mapping between CCMP- and scheduler-compliant messages. We also remark that synchronization between the Meetecho server and the CCMP proxy server can be achieved through asynchronous notifications. As soon as something worth communicating happens at the Meetecho server, a notification can be sent to the CCMP proxy (which subscribes to the events associated with conference management and manipulation) in order to let it always have an up-to-date view of the actual situation inside the conferencing server.

Indeed, the notification mechanism described above, allows us to improve the overall performance of the integrated server made of the CCMP proxy combined with the Meetecho server. In fact, provided that the CCMP proxy is always kept aligned with the Meetecho server for all what concerns conference-related information, we can let it respond to CCMP client requests directly, thus skipping the complex operations associated with the needed "CCMP-Scheduler" mapping procedures, along both directions.

Upon activation, the CCMP Server retrieves, through specific scheduler requests, all the Meetecho blueprints and conferences and loads them into a native XML database. Conference objects hence take the form of XML conference documents compliant with the XCON data model. As stated above, the CCMP Server is also a subscriber to the Meetecho Notification Service, and is thus aware of all modifications taking place on the conferences managed by the Meetecho server (modifications which might also be due to actions undertaken by non-CCMP aware Meetecho conferencing clients). Accordingly to the received notifications, the database is updated. In such a way, the CCMP Server has always available an aligned image of the confer-

[1]This is due to the fact that, when the Meetecho XCON-compliant conferencing platform has been conceived, inside the XCON Working Group there was no consensus yet as to the standard conference control protocol to be adopted.

ence information set managed by the Meetecho platform. This allows the CCMP Server to immediately answer to CCMP retrieve requests, without forwarding the corresponding Scheduler request to the Meetecho server each time this kind of message arrives. Unlike the retrieve case, the CCMP requests associated with an operation of either create, or update, or delete must be translated into the equivalent Scheduler messages to be sent to Meetecho, in order to have an actual effect on the Meetecho Server side. The Scheduler responses are then interpreted and converted into the appropriate database updates, as well as translated into the equivalent CCMP responses to be returned to the CCMP Client.

Having transposed the Meetecho Conference Control plane to the CCMP world, we have integrated a library of CCMP APIs into our Meetecho client, thus allowing it to make use of CCMP (instead of the legacy Meetecho scheduler protocol) as the Conference Control Protocol, in such way completing the scenario we presented in 2.4.

In the following subsections, we delve somehow into the details of the main actors involved in the Conferencing Control environment we have introduced, namely the CCMP client, the CCMP proxy server and the native XML database. We will discuss both the design and the implementation choices associated with the above mentioned components. Some notes and considerations about the way CCMP has been integrated into the Meetecho client are also reported.

2.8.1 Managing XML CCMP messages and conference information

The CCMP server and client components have both been implemented in Java. Since CCMP messages, as well as the conference-related information they carry, are formatted as XML documents, we faced the need of generating, parsing and handling such items in Java. Besides, as it will be explained in the following section, we also needed a proper way to handle an XML-aware database, which could manage the manipulation of conference objects.

In order to facilitate these operations, we chose to exploit the JAXB API (Java Architecture for XML Binding API) 2.1, which is the last API version at the time of this writing. This API allows to represent XML documents (that also have to be validated against a given XML Schema) in a Java format, i.e. through Java objects representing their different composing parts. The binding indeed represents the correspondence between XML document elements and the Java objects created with JAXB. Accessing XML contents by means of JAXB presents several advantages in terms of both efficiency and easiness with respect to SAX and DOM parsing. In fact, just like DOM, the output analysis can be saved at once and then consulted at any time without having to re-parse the whole document again, while the concerning memory occupation turns out to be lower than the one of the DOM tree; like SAX, on the other hand, it is possible to access specific document parts without performing a further complete document parsing and without traversing the XML tree until the leaf to be examined is reached.

JAXB not only allows for easy access to XML documents, but also for a seamless creation of XML document from the representative Java counterparts. This operation is called "marshalling". The inverse operation, from XML to Java objects, is instead called "unmarshalling".

Each JAXB generated class, corresponding to a specific type of XML element or attribute described in the schema file, is equipped with get and set methods that make it very easy to both extract information values and set them.

The JAXB architecture is composed of a set of APIs (contained in the `javax.xml.bind` extension package) and of a binding compiler, called XJC, which generates, starting from an XML Schema, the set of Java classes representing the element types embedded in XML documents compliant with it. In this context we have used the XJC Eclipse plug-in and produced the package of Java classes related to the XML Schema files collected from the data model documents [74, 64], as well as from the CCMP RFC [16].

2.8.2 Managing HTTP

Considering the suggested transport for CCMP messages is HTTP (precisely, POST and 200 messages for requests and responses, respectively), we also had to cope with the issue of handling HTTP messages both at the client and at the server sides.

For the CCMP server implementation, we made use of the Apache open-source servlet engine Tomcat. The CCMP server business logic is realized through a servlet which, in the `doPost()` method, extracts the CCMP body from the HTTP POST request and, once the proper CCMP request type has been detected, starts the specific management thread accordingly.

On the client side, instead, we made use of the HTTP open source package provided by Apache, Apache Commons HTTP Client 3.1. This package is widely deployed in several projects, and allowed us to easily create and send to the CCMP server HTTP POST requests containing the CCMP message inside their payload, as well as handle the associated HTTP response accordingly.

2.8.3 Xindice database

We previously mentioned the need for an XML-aware database. In fact, CCMP handles the manipulation of conference objects compliant with the XCON common data model specification. Such conference objects are XML documents, and so, having an XML-aware database to store and manipulate them instead of relying on a relational databases relieved us form the burden of taking care of the transformation from tables to XML documents and viceversa whenever needed.

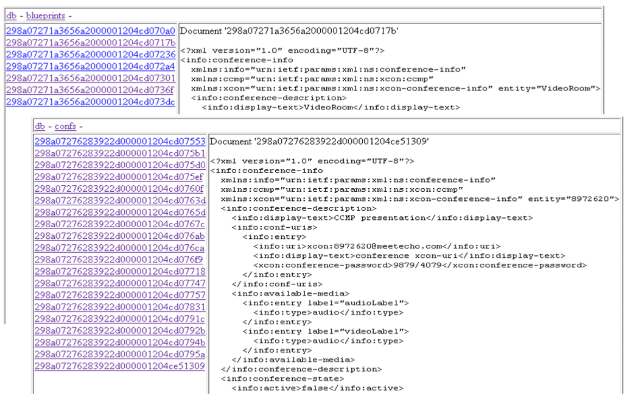
To cope with this requirement we chose Xindice, an open source Apache server handling an XML-native database specifically conceived for storing XML documents. Just as we needed, it allows to simply insert the XML data as it is when writing to the database, as well as to return the data in the same format when accessing the database. This feature is very useful when having to deal with complex XML documents like XCON conference objects, which might become very difficult or even impossible to be effectively stored in structured databases.

Xindice is installed as a Tomcat web application, and as such it was seamlessly integrated into our CCMP server prototype implementation. The `XML:DB` Java API is used to access the XML database. Such API are vendor-neutral, meaning that they are independent of the specific native XML database implementation, and operate on XML document collections, allowing the user to perform, on the collected XML documents, XPath queries as well as XUpdate modifications. Document collections are created and accessed through Xindice-specific Java APIs (Xindice Collection Manager Service).

In this context, we generated two main collections: (i) *confs* – the set of active and registered conferences hosted on the Meetecho server, reported in the form of conference documents compliant with the XCON data model; (ii) *blueprints* - the set of the Meetecho conference templates, in the XML XCON data model compliant format as well. A snapshot of the database content is showed in Figure 2.5.

XPath queries are then executed by the CCMP server whenever needed, for instance to select the conference object referred to by the confObjID in CCMP requests, or to retrieve specific conference information from the XML conference documents grouped in the database collections.

XUpdate queries are instead performed to update the conference documents according to received Meetecho notifications (generated, for example, as a consequence of a new user join or leave event) and CCMP client requests (e.g. when a client sets via CCMP a participant as chair of a certain floor).



db - blueprints -

298a07271a3656a2000001204cd070a0
298a07271a3656a2000001204cd0717b
298a07271a3656a2000001204cd07236
298a07271a3656a2000001204cd072a4
298a07271a3656a2000001204cd07301
298a07271a3656a2000001204cd0736f
298a07271a3656a2000001204cd073dc

Document '298a07271a3656a2000001204cd0717b'

```
<?xml version="1.0" encoding="UTF-8"?>
<info:conference-info
  xmlns:info="urn:ietf:params:xml:ns:conference-info"
  xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
  xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info" entity="VideoRoom">
  <info:conference-description>
    <info:display-text>VideoRoom</info:display-text>
```

db - confs -

298a07276283922d000001204cd07553
298a07276283922d000001204cd075b1
298a07276283922d000001204cd075d0
298a07276283922d000001204cd075ef
298a07276283922d000001204cd0760f
298a07276283922d000001204cd0763d
298a07276283922d000001204cd0765d
298a07276283922d000001204cd0767c
298a07276283922d000001204cd076ab
298a07276283922d000001204cd076ca
298a07276283922d000001204cd076f9
298a07276283922d000001204cd07718
298a07276283922d000001204cd07747
298a07276283922d000001204cd07757
298a07276283922d000001204cd07831
298a07276283922d000001204cd0791c
298a07276283922d000001204cd0792b
298a07276283922d000001204cd0794b
298a07276283922d000001204cd0795a
298a07276283922d000001204ce51309

Document '298a07276283922d000001204ce51309'

```
<?xml version="1.0" encoding="UTF-8"?>
<info:conference-info
  xmlns:info="urn:ietf:params:xml:ns:conference-info"
  xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
  xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info" entity="8972620">
  <info:conference-description>
    <info:display-text>CCMP presentation</info:display-text>
    <info:conf-uris>
      <info:entry>
        <info:uri>xcon:8972620@meetecho.com</info:uri>
        <info:display-text>conference xcon-uri</info:display-text>
        <xcon:conference-password>9879/4079</xcon:conference-password>
      </info:entry>
    </info:conf-uris>
    <info:available-media>
      <info:entry label="audioLabel">
        <info:type>audio</info:type>
      </info:entry>
      <info:entry label="videoLabel">
        <info:type>audio</info:type>
      </info:entry>
    </info:available-media>
  </info:conference-description>
  <info:conference-state>
    <info:active>false</info:active>
```

Figure 2.5: An image of the Xindice native XML database used in the prototype

2.8.4 CCMP-Meetecho integration

As anticipated, our reference conferencing platform, Meetecho, does not support CCMP natively. It instead currently relies on a proprietary protocol, called Scheduler, to handle conference objects and their manipulation. This protocol has a limited set of functionality available, which nevertheless can be logically mapped in a quite straightforward way to a subset of CCMP operations. This motivated us into integrating CCMP in our platform by handling at first CCMP as a simple wrapper to the operations made available by the Scheduler.

Specifically, the Scheduler protocol allows a participant to:

- create a new conference;
- delete existing conferences;
- retrieve the list of available blueprints;
- retrieve a specific blueprint;
- setting a participant as floor chair of a media;
- control the volume of the received audio;
- enable or disable mirroring, i.e., receive back or not receive back her/his video;
- select the audio and video contribution of the other participants she/he wish to receive;
- retrieving the list of users in a conference.

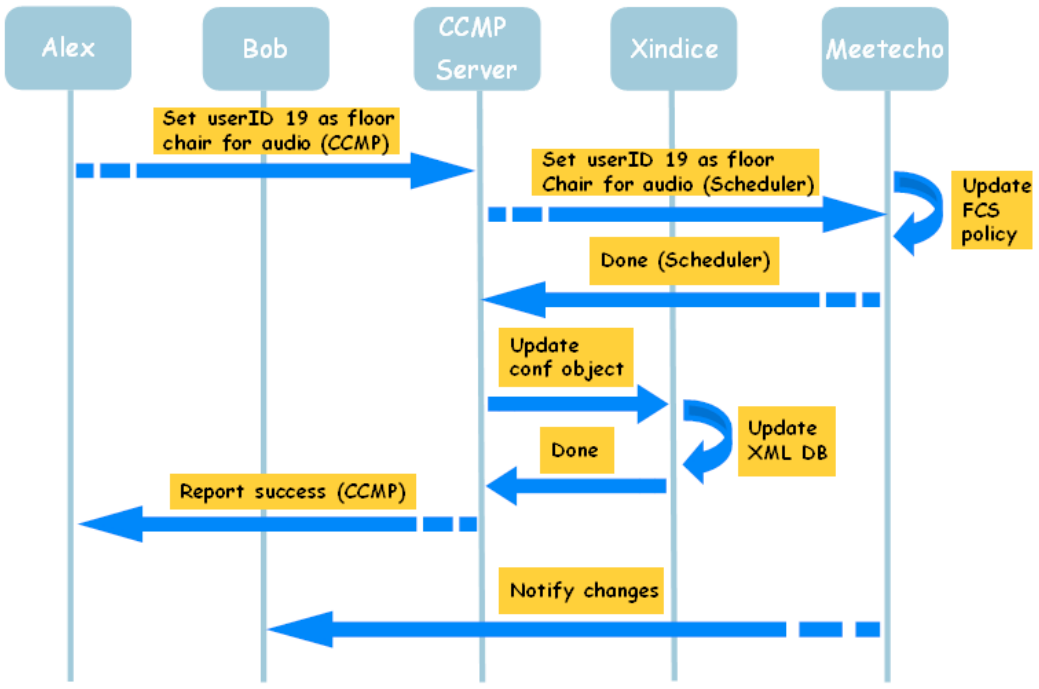


Figure 2.6: A sample CCMP-based interaction involving protocol mapping

All these operations are made available by CCMP as well, and so this allowed us to test our prototype CCMP implementation in realistic scenarios.

The integration was realized by implementing a wrapper on the server side. We deployed our CCMP server (Tomcat, JAXB and Xindice) by putting it side by side with the existing Meetecho server. We then added to the already implemented CCMP server logic a wrapping functionality, in order to handle incoming CCMP requests and translate them into Scheduler directives accordingly, where applicable, and viceversa. On the client side, we replaced the Scheduler client module with our CCMP client implementation and logic.

The mode of operation is quite straightforward. Any time a participant issues a CCMP request, it is handled by the CCMP server. The CCMP server maps the request to the Scheduler counterpart, translating the message. Such a message is then forwarded to the legacy Meetecho Scheduler server, where it is handled and enforced. According to the Scheduler reply that is received as a consequence, the CCMP server takes the related action, e.g., updating the XML conference object on the Xindice database if needed, and providing the participant with a coherent CCMP response.

An example is provided in Figure 2.6.

A dump of the CCMP messages exchanged follows:

```
ccmpRequest message sent:
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpRequest
        xmlns:info="urn:ietf:params:xml:ns:conference-info"
        xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp"
        xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info">
     <ccmpRequest
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="ccmp:ccmp-conf-request-message-type">
        <confUserID>xcon-userid:alex@meetecho.com</confUserID>
<confObjID>xcon:8977777@meetecho.com</confObjID>
        <operation>update</operation>
<conference-password>1377</conference-password>
        <ccmp:confRequest>
           <confInfo entity="xcon:8977777@meetecho.com">
<xcon:floor-information>
                   <xcon:conference-floor-policy>
                   <xcon:floor id="11">
<xcon:moderator-id>19</xcon:moderator-id>
                   </xcon:floor>
                   </xcon:conference-floor-policy>
        </xcon:floor-information>
           </confInfo>
        </ccmp:confRequest>
     </ccmpRequest>
  </ccmp:ccmpRequest>

  ccmpResponse message received:

  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <ccmp:ccmpResponse
       xmlns:xcon="urn:ietf:params:xml:ns:xcon-conference-info"
       xmlns:info="urn:ietf:params:xml:ns:conference-info"
       xmlns:ccmp="urn:ietf:params:xml:ns:xcon:ccmp">
    <ccmpResponse
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:type="ccmp:ccmp-conf-response-message-type">
       <confUserID>xcon-userid:alex@meetecho.com</confUserID>
       <confObjID>xcon:8977777@meetecho.com</confObjID>
       <operation>update</operation>
       <response-code>success</response-code>
       <version>10</version>
        <ccmp:confResponse/>
  </ccmpResponse>
</ccmp:ccmpResponse>
```

This simple example shows the process for a typical scenario. In an active conference (identified by its ID 8977777), a participant, Alex (who happens to be administrator of the conference), decides to assign a floor chair for the audio resource, in order to have it properly moderated by means of BFCP. In CCMP, this is achieved by issuing a “confRequest” with an “update” operation: the body of the specialized “confRequest” element contains the part of the conference object that needs manipulation, in this case the floor information associated with the existing audio resource. This audio resource is identified by means of a floor id (11 in the example), which in the conference object itself is explicitly mapped to the label assigned to the audio medium. Since Alex is interested in assigning a floor chair to take care of this medium, he specifies a “moderator-id” (19) which refers to a specific userID in the Floor Control Server. A password is also provided (1377) since this operation requires special permissions.

This request is sent to the CCMP server which, since a mapping with the Scheduler functionality exists, translates the message accordingly to the Scheduler format, and sends the newly created message to the legacy Meetecho Server. The server handles the request and enforces it, updating the

Floor Control Server policy accordingly. The successful result of the operation is reported by means of a Scheduler reply to the CCMP server, which in turn updates the Xindice database coherently with the request. This means that the XML conference object associated with conference 8977777 is updated. A success is finally returned to the participant by means of a CCMP response.

2.9 Standard Multimedia Conferencing in the wild: the Meetecho Architecture

In this section we describe the Meetecho conferencing system we contribute to make fully-compliant with the XCON standard [8]. We indeed have worked on the integration of the CCMP server within the Meetecho platform as a "na tive" component, in such a way as to avoid the unavoidable burden associated with proxying CCMP requests and mapping them onto the legacy scheduler protocol.

Meetecho is an open source centralized video-conferencing system capable to offer advanced communication experience to end-users through the effective exploitation of mechanisms like session management and floor control. It indeed has been designed to be fully compliant with the latest standard proposals coming from both the IETF and the 3GPP and can be considered as an outstanding example of a real-time application built on top of the grounds paved by the SIP protocol.

The Meetecho conferencing platform offers advanced conferencing features: system users, named "participants" or "conferencing clients", are enabled to create and join conferences involving any kind of media stream, including audio and video, as well as instant messaging or even gaming. Different conference kinds ("blueprints") are supported and can be highly customized by users aiming to create conferences best fitting their needs. The system is conceived to be scalable, i.e., able to support an increasing number of conferencing clients. This property is obtained by a proper co-operation mechanism among different "centralized conferencing islands", according to the proposals illustrated in [5].

In the following, we will stress how the interactions among XCON protocols and actors can realize advanced conferencing scenario in practice by means of Meetecho.

2.9.1 Designing a standard multimedia conferencing architecture

The first step was obviously identifying and locating all the logical elements which would be involved in a conferencing scenario. The next one, the one of investigating the possibility of replicating, or at least replacing, such elements with existing real-world components.

First of all, the scenario clearly addresses two distinct roles, a server side (the elements providing the service) and a client side (all users accessing the service).

At the client side, the very first mandatory element that comes into play is the *User Equipment (UE)*, which has to be both SIP-compliant and XCON-enabled in order to correctly support conferencing. On the server side, instead, different cooperating components have been identified: the *Application Server (AS)*, the *Media Server (MS)* and one or more *Gateways*.

The former can be further split into subcomponents, as it has to provide several functionality like dealing with the signaling plane of the conferencing scenario, handling the management of conferences (e.g., creating, modifying, deleting them), and in general taking care of all the business logic, which includes policies, related to the scenario. These policies include Floor Control, which implies that the AS will also have to manage access rights to shared resources in our conferencing framework. The media streams are manipulated and provided by the Media Server. It has to provide the resources, by offering functionality like mixing of incoming media streams (in our case, audio and video streams) and media stream processing (e.g., audio transcoding, media analysis). Finally, considering the XCON framework is conceived to be agnostic with respect to the signaling protocol used to access the service, specific elements are needed as gateways towards other technologies. For instance, a Gateway is needed in order to guarantee the interworking with the Public Switched Telephone Network (PSTN).

Furhtermore, the XCON framework defines a suite of conferencing protocols, which are meant to be complementary to the call signaling protocols, for building advanced conferencing applications and supporting complex scenarios. These protocols aim at providing means to manage conferences under all their aspects.

The realization of an XCON-compliant architecture then led to work both on the client and on the server side, with special focus on the communication protocols between them, as well as their scenarios of interaction. The client side work included the implementation of both roles envisaged in the architecture, namely the simple participant and the chair. On the server side, the main actor to implement is the *Focus*As such, the focus acts as

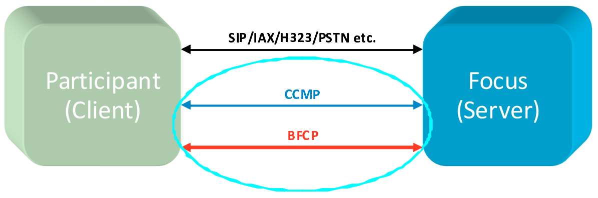


Figure 2.7: New protocols implemented

an endpoint for each of the supported signaling protocols and is responsible for all primary conference membership operations (e.g., join, leave, update the conference instance) and for media negotiation/maintenance between a conference participant and the focus),of the Floor Control Server, of the Conference Control Server, and of the Media Server. To make the client and server sides properly interact, all the envisaged protocols have been implemented(see Figure 2.7), specifically the already mentioned BFCP [27] and the Centralized Conferencing Manipulation Protocol (CCMP) [16].

As to BFCP, it has been implemented as a dynamic library, which has then been integrated into both client and server entities of the architecture. All the media management, manipulation and delivery have been bound to an event-driven mechanism, according to the directives coming from the Floor Control Server.

The CCMP protocol has been implemented and integrated as well , in order to allow clients to dynamically manage conference creation as well as conference manipulation.

2.9.2 Server side components

On the server side, we adopted Asterisk[2], a popular open source PBX which is constantly growing in popularity. The modular architecture behind Asterisk design allows it to be quite easily modified and enhanced, upon necessity. Specifically, we added to Asterisk the following new functionality:

- XCON-related identifiers, needed to manage conferences;
- Floor Control Server (FCS), by means of a dynamic library implementing the server-side behavior and policies of the BFCP;
- CCMP Server, the server side component implementing the conference scheduling and management protocol;

[2]See `http://www.asterisk.org`

- Video Mixer Client, the client side of the protocol implementing the interaction with the remote Video Mixer;
- Notification Service, to enable asynchronous events interception and triggering.

Most of these components have been realized as extensions to a conferencing facility already available as a module in Asterisk, called *MeetMe*. This facility acts as a set of configurable virtual "rooms" for channels that are attached to it, thus allowing users to access conferences by simply calling a predefined phone number, associated with a standard extension of Asterisk's dial-plan, independently from the client's signaling protocol. The addition of the above mentioned functionality allowed us to realize an XCON-compliant focus.

Conference management

The addition of the CCMP component to the "vanilla" MeetMe module allows for dynamic conference management in a user-friendly fashion: in fact, through this component clients become able to dynamically (i.e., both in an active way, as in scheduling, and in a passive way, as in retrieving information) manipulate conference objects. Considering the dynamic nature of the framework with respect to policies, settings and scheduled conferences, all the required changes in the dial-plan, as well as dynamic reloading upon necessity, have been accomplished by adding the related functionality to the extended MeetMe module.

Moderation

For what concerns BFCP, we had to implement the entire protocol, as well as its behavior (which includes queues and state machines) from scratch. In order to achieve this, BFCP has been realized as a dynamic library, which is loaded at run time by the Asterisk server and comes into play whenever a resource is to be moderated. In fact, Asterisk, as the entity in charge of the business logic, also acts as the Floor Control Server of the architecture (see Figure 2.8). The FCS functionality is involved every time a request is generated from a participant, asking for the right to access a specific resource (e.g., audio or video). As suggested by the picture, the FCS itself may or may not take any decision about incoming requests. In fact, automated policies may be involved to take care of floor control with a more straightforward approach (e.g., to always accept or reject incoming requests according to predefined policies). If they are not specified, the FCS rather

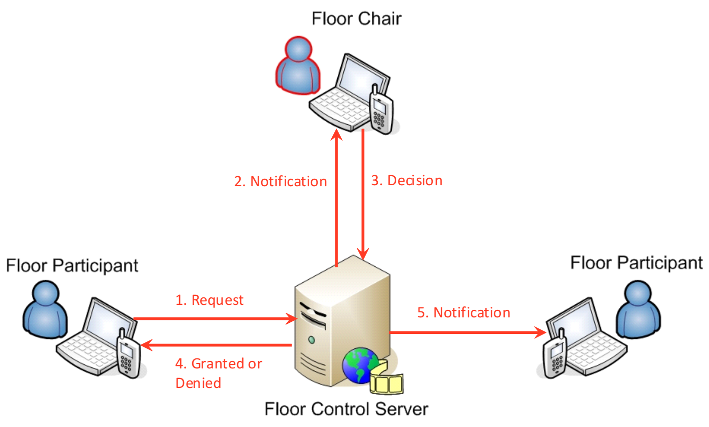


Figure 2.8: The BFCP protocol in action

forwards floor requests to the designated floor chair who is in charge of taking a decision that is eventually notified to all the interested parties.

2.9.3 Client side components

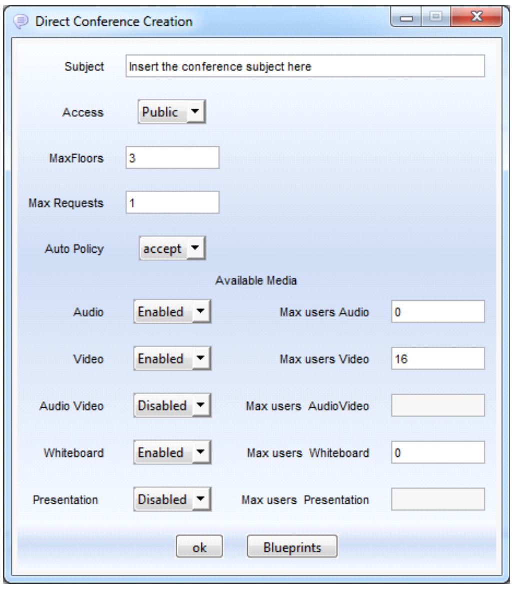
On the client side, we adopted an existing open source instant messaging client, called *Spark*[3], as the basis for our work. Spark is a piece of software written in Java, which implements the XMPP protocol to realize instant messaging scenarios. It is natively conceived to interact with the aforementioned Openfire server, and is easily extensible by developing custom plugins which implement new functionality. In our case, we introduced the support for the SIP/SDP, RTP, BFCP and CCMP protocols, besides some additional functionality

.

New graphical widgets have been realized as well, in order to enable user-friendly support for SIP- and BFCP-related settings and to allow users to take advantage of the functionality related to both conference scheduling and BFCP. As to conference scheduling, Figure 2.9 shows the widget by means of which it is possible to dynamically create new conferences by exploiting the CCMP protocol. Figure 2.10, instead, shows the list of scheduled conferences, retrieved by means of CCMP as well.

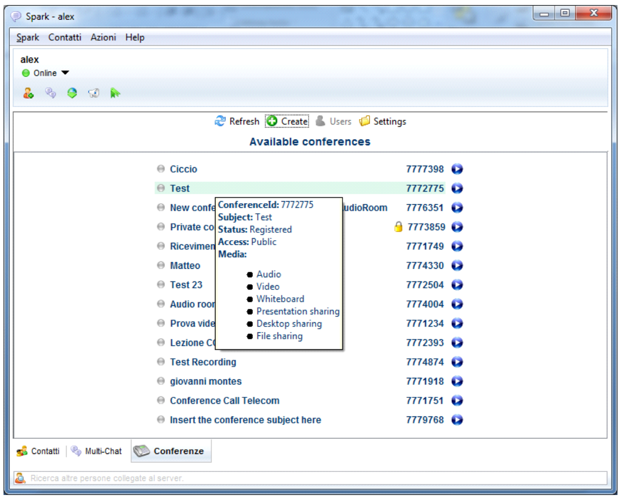
The client side BFCP behavior has been implemented as a Java library. An ad-hoc panel, associated with such library, has been introduced in order to enable users to:

[3]See `http://www.igniterealtime.org/projects/spark/`



Subject	Insert the conference subject here		
Access	Public		
MaxFloors	3		
Max Requests	1		
Auto Policy	accept		
Audio	Enabled	Max users Audio	0
Video	Enabled	Max users Video	16
Audio Video	Disabled	Max users AudioVideo	
Whiteboard	Enabled	Max users Whiteboard	0
Presentation	Disabled	Max users Presentation	

Figure 2.9: Conference creation



Available conferences

Ciccio	7777398
Test	7772775
New confe … udioRoom	7776351
Private co	7773859
Ricevimen	7771749
Matteo	7774330
Test 23	7772504
Audio roor	7774004
Prova vide	7771234
Lezione C	7772393
Test Recording	7774874
giovanni montes	7771918
Conference Call Telecom	7771751
Insert the conference subject here	7779768

ConferenceId: 7772775
Subject: Test
Status: Registered
Access: Public
Media:

- Audio
- Video
- Whiteboard
- Presentation sharing
- Desktop sharing
- File sharing

Figure 2.10: List of scheduled conferences

- Send BFCP messages to the BFCP server;
- Interactively build BFCP floor requests in a user-friendly fashion, either in *participant* or in *chair* (i.e., with enhanced floor management functionality) mode;
- Keep an up-to-date log of all the BFCP messages exchanged with the server (and optionally show each such message in further detail by simply clicking on the related entry in the history widget).

Finally, as to the VoIP functionality of the client, the open source SIP stack called *MjSip*[4] has been adopted and extended.

2.9.4 An example of client-server interaction

To provide the reader with a more detailed overview of the way the client-server interaction involves the introduced protocols, this subsection is devoted to presenting an example regarding a typical use case scenario. To ease the understanding of the sequence diagram depicted in Figure 2.11, each protocol is represented with a different line style:

1. A participant (client in the scenario) contacts the Focus (the server), through the CCMP protocol (dashed line), to ask for the list of currently active conferences, thus sending a *ConfsRequest* message request with a proper filter parameter;
2. The Focus processes the request and sends back to the participant (still through the CCMP protocol) a *ConfsResponse* message, containing the list of all active conferences;
3. The participant reads the list and decides to join the active conference identified by the number 8671000: to join the conference, she/he calls the conference number – as if it were a standard phone number – using SIP (solid line) as the call signaling protocol, thus placing a call to the SIP URI 8671000@*Focus* (where Focus is the SIP domain, in this case the IP address of the Asterisk server);
4. The Focus receives the call and, according to the specified dialplan rules, routes it to the XCON-enabled MeetMe instance managing the conference with the same call number;

[4]See `http://www.mjsip.org`

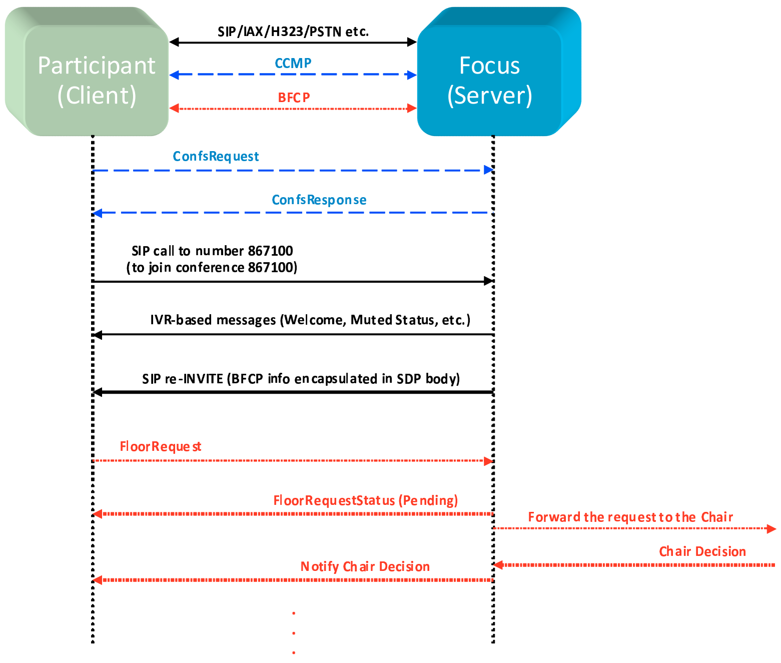

Participant (Client)
Focus (Server)
SIP/IAX/H323/PSTN etc.
CCMP
BFCP
ConfsRequest
ConfsResponse
SIP call to number 867100 (to join conference 867100)
IVR-based messages (Welcome, Muted Status, etc.)
SIP re-INVITE (BFCP info encapsulated in SDP body)
FloorRequest
FloorRequestStatus (Pending)
Forward the request to the Chair
Chair Decision
Notify Chair Decision

Figure 2.11: An example of signaling between client and server

5. The XCON-enabled MeetMe instance managing the conference, through IVR (*Interactive Voice Response*), plays back a series of pre-recorded voice messages to welcome the new user. It also warns the client about the fact that she/he is initially muted in the conference;

6. All the relevant BFCP information is encapsulated in an SDP body, and then sent back to the new user by means of a SIP re-INVITE;

7. Once the client receives the re-INVITE and becomes aware of the needed BFCP set of data, she/he, using the BFCP (dotted line), decides to make a *FloorRequest* to ask the Focus for the permission to talk;

8. The Focus, as Floor Control Server, answers the client by sending back a *FloorRequestStatus* BFCP message notifying that the request is currently pending. At the same time, the Floor Control Server forwards the message to the chair of the requested floor as well, to ask him to take a decision about the request.

From this point on, the BFCP transaction proceeds exactly as described before (see Figure 2.8). Once the chair grants the floor, the client is un-muted and thus given the permission to talk until the floor is not willingly released by the client herself/himself or revoked by the chair. Since a floor is a logical object, all BFCP transactions will proceed in the same way, independently from the set of resources (be it audio or video, in the case of our platform) the related floor(s) could be associated with. In case the floor request involved a manipulation of a video request, a subsequent interaction between the conferencing module and the remote videomixer would take place through the dedicated channel.

Chapter 3

A formal approach to performance assessment

3.1 Introduction

This chapter proposes a Stochastic Activity Networks (SANs) based approach to performance analysis of a conferencing framework compliant with the IP multimedia core network subsystem specification. Such conferencing framework is represented by the Meetecho conferencing platform, introduced in the previous chapter.

The proposed approach relies on the OsMoSys modeling methodology (Object-baSed multi-formalism MOdeling of SYStems, [86]) applying some concepts of component software engineering to the development of formal models. We extends such model by coping with reuse and complexity of models. As such, we introduce the concept of *model templates* enabling the definition of a family of models from one model description, as well as the usage of *model stubs* which can be used in order to reduce both simulation time and memory consumption during the analysis phase.

We describe the implementation of a library of reusable SANs modeling the conferencing framework components. Starting from the model library, we analyze the scalability performance of Meetecho by instantiating and composing SAN models. We validate the resulting model of the conferencing system and discuss the advantages of the proposed approach through a comparative analysis with the results of an experimental campaign conducted over a real-world testbed implementation involving the Meetecho platform.

Section3.2 provides the reader with the minimal background information about the SAN formalism and the OSmoSys modeling methodology. We first present the facets of the conferencing framework we take into account during the modeling phase (Section 3.3). We motivate and position our work in Section 3.4. Section 3.5 presents the methodological contribution and introduces the concept of *model templates*. The SANs models obtained by applying the modeling process are presented in Section 3.6. In Section 3.7 the role played by model stubs and reduction techniques is discussed. The models are validated in Section 3.8 through a comparative analysis with the results of an experimental campaign conducted over a real-world testbed to assess Meetecho performance. We show the advantages of the proposed modeling approach by exploiting both complete and reduced models.

3.2 Background

3.2.1 Stochastic Activity Networks

Stochastic activity networks, or SANs, are a convenient, graphical, high-level language for describing system behavior. SANs are useful in capturing the stochastic behavior of a system.

SANs [1] represent a variation of the Stochastic Petri Networks (SPN) aimed to overcome their limitations in terms of expressive power. Indeed, SPN may only perform very simple operations and that makes difficult to model complex interactions. Because there are a number of subtle distinctions relative to SPNs, stochastic activity networks use different words to describe ideas similar to those of SPNs. For example, the role played by transitions in SPNs is played by activites in SANs. Activities are similar to transitions in SPN, and the time needed to complete a transition can be expressed in several kinds of distributions.

The SANs have four new symbols in addition to those of SPNs (i.e., besides places, tokens, timed transitions, input and output arcs):

- Input gate: used to define complex enabling predicates and completion functions
- Output gate: used to define complex completion functions
- Cases: (small circles on activities) used to specify probabilistic choices
- Instantaneous activities: used to specify zero-timed events

Activities can presents more than one output gatew. The use of doors allows introducing more complex functions for enabling activities or at the completion of a given activity rather than the simple conditions expressable in SPN.

Moreover, SANs natively support model composition. Model composition has two operations:

- Replicate: combine two or more identical SANs together, holding certain places common among the replicas;
- Join: Combine two or more different SANs, combining certain places to permit communication.

That allows alleviating the cost of developing models of large scale and complex systems.

3.2.2 The OsMoSys modeling methodology

The OsMoSys modeling methodology apply the concepts of component software engineering to the development of formal models in order to provide a practical support to model engineering. According to OsMoSys a model is an object of a Model Class which encapsulates the details of its implementation (the model structure is expressed by a formal language). Hence, models are components which communicate via interfaces. An interface is a subset of elements of the model structure which may be used to exchange information among models (for example, the value of parameters, indices or the overall state of the model). The role of the interfaces has been formally defined in [39, 63].

3.3 Modeling the conferencing system

In this section we highlight the main aspects related to the conferencing system which has been the subject of our modeling efforts. In particular, we refer to the logical IMS entities it implements.

The most important entities for our work are:

- *User Equipment (UE)*: the device used by the conferencing user to participate in the conference;
- *Application Server (AS)*: the server-side entity responsible for the implementation of the conferencing application logic;
- *Media Resource Function Controller (MRFC)*: the logical entity devoted to control operations on media streams according to information provided by the AS;
- *Media Resource Function Processor (MRFP)*: the entity in charge to perform media processing by acting as a media mixer.

In Meetecho, the AS provides all the XCON functionality as described in [15]: the signaling management for the call set-up, the delivery of conference notifications to participants, the creation and modification of conference instances according to user preferences, the handling of moderation and so on. Moreover, it acts as a MRFC by managing the MRFP, which is implemented as a different component that we will call from now on “media mixer”.

In what follows, two different conferencing scenarios are considered: a centralized conferencing scenario and a distributed conferencing scenario. The second one has been introduced in order to improve the scalability of the conferencing framework. A recent implementation of such scenario has been proposed in [25].

3.3.1 Centralized Conferencing Scenario

Figure 3.1 presents a simplified view of the system, showing the main IMS components we mentioned before. We refer to such IMS cloud as to a "centralized island" of the system, since all signaling messages from conferencing clients are directed to the same centralized AS ("focus" of the conference) and hence realize, at the level of the control plane, a typical star topology.

On the basis of the adopted conference blueprint, the AS issues commands to the media mixer, which is the system entity in charge of performing the audio and video mixing functions needed to deliver to each conference user the proper mixed stream, according to her/his preferences as well as to the supported capabilities of the device she/he uses. The multimedia streams generated and/or consumed by each participant are in fact always exchanged with the media mixer, hence realizing a star topology also on the media plane.

The AS and the media mixer can be logically grouped into an integrated server-side logical entity, called "Conferencing Server", making available all the functions needed to provide the conferencing service, both on the control plane and on the media plane.

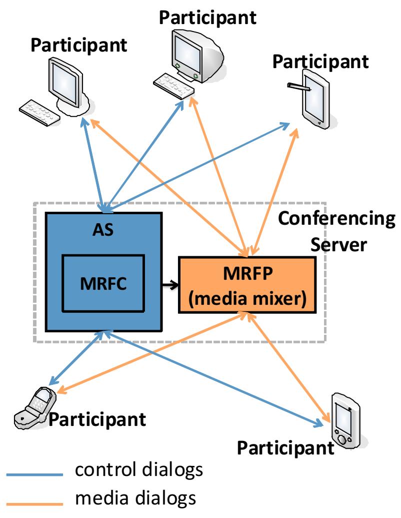


Figure 3.1: A view of the conferencing system

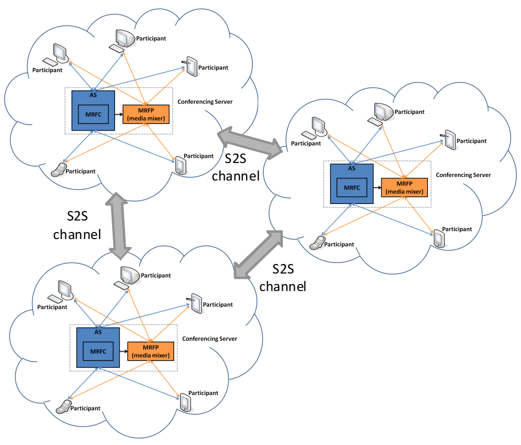


Figure 3.2: Distributed conferencing scenario

3.3.2 Distributed Conferencing Scenario

We herein move the attention to the distributed scenario, in which several centralized conferencing clouds are interconnected and cooperate in order to provide the conferencing service in a distributed manner (Figure 3.2).

Under the assumption that all conference participants refer to the focus in the home network of the conference initiator, which we call the "main" focus, the IMS centralized conferencing solution keeps on working also in the scenario where the conference participants belong to networks owned by different telecom operators. Though, in a fully distributed scenario, the above case should be dealt with by providing each involved network with a specific focus managing the associated local users who subscribed to the conference. It is up to the main focus in the conference initiator's home network to provide all other focus entities with up-to-date conference information. Such foreign focus entities thus play a twofold role. On one hand, they act as a regular conference focus for the participants belonging to their underlying managed network; on the other hand, they appear as normal participants to the focus in the conference initiator's home network.

Similarly, on the media plane, multiplexing/demultiplexing of multimedia streams generated by local users is up to each local media mixer. The resulting stream coming from foreign media mixers to the main media mixer (i.e., the mixer located in the island where the distributed conference has been created) is handled in the same way as the media flow of an ordinary conferencing client. In order to achieve consistency in a conference involving more islands, a dedicated communication channel exists between each pair of focus entities participating in the conference ("Server to Server (S2S) Channel" in Figure 3.2). Such channel is used to exchange conference information, as well as to manage synchronization issues.

3.4 Motivation and Related Work

This work is willing to provide a modeling approach to accomplish performance evaluation of actual IMS-compliant conferencing platforms in order to support their design and deployment phases. With this goal in mind, we start considering the IMS-compliant conferencing framework described in Section 3.3.

The considered system has been already the subject of a thorough experimental benchmarking campaign carried out on a real testbed [9, 10] aimed at conducting a performance and scalability analysis of both the centralized and the distributed architectures. The experimental results showed that, in a controlled network scenario, as the number of users increases, CPU utilization of the centralized server becomes the most critical performance index, since such server is responsible for the management of both the signaling dialogs with each conferencing client and the media streams mixing function. In particular, managing the media plane turned out to be much more onerous than controlling the signalling plane. This led to focus on the performance of the media plane, which definitely represents the limiting factor as far as scalability is concerned.

The mentioned experimentations required a lot of efforts. For each trial, a real laboratory testbed has been properly set up and a realistic set of conferencing sessions has been reproduced over it. Each such session was actually instantiated by replaying a proper set of recorded traces associated with actual conferences and hence mimicking in a reliable fashion the actual behavior of end-users and servers. This approach does not clearly solve the issue of assessing as timely as possible (i.e., during the design and capacity planning phases, before the actual deployment of the numerous infrastructure components) the main non-functional features of a conference, like performance and dependability. This clearly calls for a need to make available a set of flexible tools for the assessment of the aforementioned requirements, which naturally lend themselves to a multidimensional, multi-faceted characterization.

A common approach to performance evaluation of networked systems which does not require real testbed implementations is of course based on the usage of network simulators. Nevertheless the current leading network simulators (such as ns-2, ns-3, OPNET Modeler, etc.) do not offer complete support for IMS [83]; in particular, they do not support the modeling of the media plane through embedded components, but usually rely on third-party implementations of some of the required IMS media plane functions. Compared with the experimental and the simulation approaches, formal modeling presents several advantages. It allows for performance evaluation and performance prediction thus supporting the early phases of the development of the system and providing the possibility to easily evaluate several conference blueprints. This also allows for sensitivity and stability analysis on a number of parameters, including those associated with Quality of Service, which gives the service providers an effective means to plan and schedule conferences according with the expected QoS levels. A further advantage in using formal modeling is the possibility to model and analyze anomalous behaviors due to attacks, faults, overloads or degraded operating modes. Here we mainly address the aspects related to the scalability and performance of the system. Several works can be found in the literature aiming at mainly analyzing access control and protocols by developing formal models, but focusing on security [45], protection [78, 55], dependability [44] and protocol validation [48]. To the best of our knowledge there are not many attempts at modeling the IMS-compliant conferencing frameworks specifically oriented towards the study of the media plane.

Multimedia conferencing systems have already been a playground for the application of formal modeling. Several works focused on the orchestration of the different building-blocks a multimedia conferencing platform relies on, as it is the case in [21]. In the cited work, the authors mainly use Petri Nets and related modeling languages to perform composition correctness validation. Other researchers leverage Petri nets to model conferencing media flows characteristics, as well as time synchronization mechanisms. These approaches are definitely more relevant to our work, since we are concerned with media plane modeling rather than compositional aspects. In [65] authors use Timed Petri Nets (TPNs) to properly describe the temporal scheduling of the multimedia presentation process, within the inter- and intra-streams synchronization constraints. TPN models of such mechanism are used to support the implementation of a system capable to control streams synchronization in a video conferencing application. As opposed to the mentioned work, we do not analyze media streams management to drive the construction of a system from scratch, but rather to model the workload of an existing conferencing media mixer and study its performance.

Since our goal is to provide a flexible tool for the evaluation of performance and scalability, we use SAN models to describe the conferencing server behavior, while dealing with users mixing preferences and coping with the diversity of the involved media and related codecs. SANs are more suitable than TPNs to our aims because of their modeling power and efficiency. Specifically, they provide the basic modeling mechanisms to easily integrate, in the models, data structures representing messages, as well as to replicate and compose submodels.

The focus of this work is on both scalability and performance aspects of the system. The analysis is carried out by alternately varying: a) the scenario (centralized vs distributed); b) the number of participants; c) the kind of deployed conference (e.g., presence/absence of the moderator, presence/absence of transcoding mechanisms). From now on, for the sake of simplicity, we consider conferences with only audio streams (audio conferences): this simplification allows us to better explain our approach without losing in generality.

3.5 Model Development Process

The modeling approach is founded on well known principles: it is compositional and hierarchical, and promotes model reuse. In particular we refer to the OsMoSys modeling methodology [86] which was born to support compositional and multiformalism modeling. Here we do not exploit multiformalism but we rather take advantage from the separation between interface and implementation of submodels. The IMS-conferencing case study requires that a clear separation of concerns is adopted in modeling the behavior of the different entities of the framework. Moreover, the heterogeneity of devices, the variability of the number of participants and the need for scalability suggest that proper solutions must be found to provide the modeling approach with the necessary flexibility. Possible answers to these issues are proposed:

1. separation of concerns: the first step of the model development process is to define the behavioral levels of the system. They will be modeled in isolation and then composed through model interfaces.

2. enabling model reuse: we propose to extend the OsMoSys methodology by introducing the concept of *Model Template* in order to easily define a family of models sharing part of their definition and allowing for parametric specification of some elements of the models themselves.

3. dealing with model scalability: the simulation time and space occupation needed to solve the models are very high as the number of participants increases: we propose two solutions to this issue, described in Section 3.7.

3.5.1 Separation of Concerns: Modeling Levels

In Figure 3.3 the structure of both the Participant (Client Side) and the Conferencing Server (Server Side) entities is shown. Since we deal with performance and scalability analysis of the system, we concentrate the modeling efforts on the media plane: in fact, as remarked in Section 3.4, experimental campaigns showed that operations like transcoding and mixing of the participants' media streams are the most demanding ones in terms of required resources and hence have the strongest impact on the overall system performance.

Client Side is composed by the following levels.

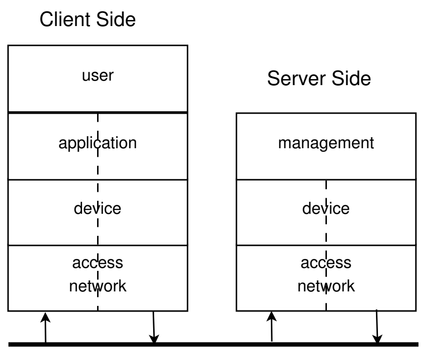


Figure 3.3: Reference modeling schema

- **User**: at this level the media streams are produced (output streams) or consumed (input streams). The user is characterized by the role she/he plays in the conference (moderator, speaker, observer, etc.) and by the load that she/he produces.

- **Application**: at this level the streams are encoded/decoded. For example, the application samples and encodes the audio signal coming from the user's microphone. The production rate of the messages depends on the type of encoding (i.e., codecs adopted), as well as on both the power and the number of CPUs.

- **Device**: this level is in charge of bundling output streams into packets or viceversa (i.e., assembling input streams). At this level the features of the participant's communication device (the User Equipment) are taken into account. The main parameters are the power and the number of CPUs.

- **Access network**: it represents the access network of the device and is characterized by its connection speed.

Server Side levels are the following ones.

- **Management**: it includes the server-side functionality for the provisioning of the conferencing service on the media plane in each centralized conferencing cloud. More precisely, as already mentioned, we skip the modeling of the Application Server component, while stressing the details of the media mixer component in charge to perform media streams management on the basis of the particular kind of conference.

- **Device**: the same as the client-side's one.
- **Access network**: the same as the client-side's one.

3.5.2 Enabling Model Reuse: Components and Templates

The OsMoSys methodology may be very effective in modeling the conferencing system since a component-based approach emphasizes the separation of concerns. Nevertheless, the problem we deal with is characterized by a variable number of peers (Conferencing Servers and/or Participants), as well as by the heterogeneity of the involved applications and devices. Hence, we need mechanisms to easily provide a specification for generating models based on parameters. With this aim in mind, we explore the possibility of extending OsMoSys by defining *model templates*.

Model Templates may introduce a powerful feature in formal modeling, since they allow to specify with a single model description an entire family of related models, called template models. Similarly to Class Templates introduced by several programming languages, *Model Templates* require one or more type parameters to specify how to generate a specific instance. The type parameters refer to the type of one or more elements of the model structure, including the type (i.e., the Model Class) of submodels. It is also possible to use *non-type* parameters, e.g., to specify the number of replicas of a subset of elements (including submodels). The complete definition of *Model Templates* in OsMoSyS is part of an ongoing work. We define and exploit a specific case of *non-type* parameters. The formal definition and application of *type* parameters are out of the scope of this work.

Some research papers have been published proposing template based approaches, all limited to some extent: in [92] typed parameters template are introduced for Petri Nets models. In [22] the focus shifts onto Stochastic Activity Networks and on the possibility to change the behavior of the model according to *non-type* parameter values. Other works focus on the possibility to both replicate and join submodels inside a compositional approach [82, 31]. To the best of our knowledge, there are no works aimed at defining a generic modeling approach in which formal models are templates developed in terms of *type* and/or *non-type* parameters that can be specified at instantiation time.

In order to define the *Model Templates* according to the OsMoSys notation, some preliminary definitions are due. Formally, a *Model Class* $MC_{\mathcal{F}}$ is a triplet (T, S, SM) where: T is the class name, S is the graph structure of the class, SM is a set of submodels. $MC_{\mathcal{F}}$ is compliant with a formalism $\mathcal{F}$ and its structure S is a graph of elements of $\mathcal{F}$. Specifically, from now on we will call N the set of nodes and E the set of edges of S. For example, if a *Model Class* is compliant with the Petri Nets formalism, its structure will comprise $Place$, $Transition$ (the nodes of the graph) and Arc (its edges). It holds: $S = External_S \cup Internal_S$; $External_S \cap Internal_S = \oslash$ where $External_S$ is the subset of the *interface* elements of the *Model Class* and $Internal_S$ is the subset of elements that are encapsulated by the class.

Let us define a *Model Template* $MT_{\mathcal{F}}$ as a pair $(MC_{\mathcal{F}}, PAR)$ where $MC_{\mathcal{F}}$ is a *Model Class* and PAR is a set of *non-type* parameters:

$$PAR = \ \{p_1, p_2, \dots p_n\}, n \geqslant 1.$$

A non-type parameter is a triplet $p_i = (l_i, SS_i, f_i)$, where l_i is the name of the parameter; SS_i is a subgraph of S, $SS_i = (NN_i, EE_i)$ with $NN_i \subseteq N$ and $EE_i = \{e = (m, m_1) \in E | m, m_1 \in NN_i\}$.

A synthetic notation for the Model Template is $MT_{\mathcal{F}} < l_1, l_2, \dots l_n >$.

An *instancing function* is $f_i : \mathbb{N} \longrightarrow \mathcal{M}^1_{\mathcal{F}}$ where $\mathcal{M}^1_{\mathcal{F}}$ is the set of the *Model Classes* compliant with the formalism $\mathcal{F}$.

An *instancing function* f_i must specify how the Model Template should be (automatically) instantiated by using the *non-type* parameters. A synthetic

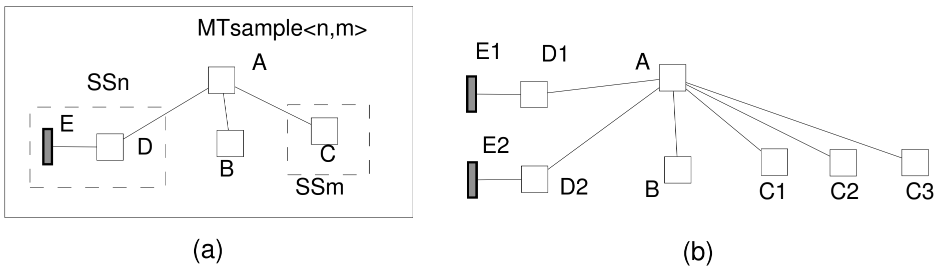


Figure 3.4: A Model Template example

notation for an instance is $MT_{\mathcal{F}} < v_1, v_2, \dots v_n >$ where v_i is the value on which f_i is computed.

An example is depicted in Figure 3.4, in which a simple case of replication is shown. A Model Template is on the left (Figure 3.4(a)), let it be:

$$MT_{sample} = (MC_{sample}, \{p_n, p_m\})$$

where: $p_n = (n, SS_n, f_n)$ and $p_m = (m, SS_m, f_m)$ are the *non-type* parameters. The subsets of the model structure in the dashed boxes (SS_n and SS_m) may be replicated and a template model may be generated by providing the value of the two non-type parameters. Both subnets include a submodel (the white square). The Model Template is denoted by $MT_{sample} < n, m >$ while the template model is denoted by $MT_{sample} < 2, 3 >$ and is shown in Figure 3.4(b). The model is obtained by specifying the values 2 and 3 for the number of replicas of SS_n and SS_m, respectively[1]. The *instancing functions* must be provided in order to state how the template models have to be generated. In this example we use a simple instancing function that copies a sub-graph as many times as indicated in the parameter value: all the replicas are then connected to the rest of the model by replicating arcs connecting the sub-graph with the rest of the structure, too. In the example, the subgraph SS_n and the arc from A to D are replicated twice. Of course, a renaming function is also needed to avoid conflicting names.

[1]Note that E is an interface element: this approach can be used to replicate model interfaces, too.

In the next section, the modeling approach herein described is exploited to develop a library of the models associated with the IMS-compliant conferencing framework. Such library contains both Model Classes and Model Templates, compliant with the SAN formalism, that must be customized and instantiated according to the different available conference blueprints.

3.6 Developed models

This section details how the modeling approach described in Section 3.5 is applied to generate the SAN models of the conferencing system. Models are described according to a bottom-up approach. First, we introduce the models realizing each level of the conferencing system in isolation (Figure 3.3). Then, we show how they are composed in order to obtain the models of both the Client Side and the Server Side. Some of the models described in this Section are template models, since they are obtained by instantiating specific Model Templates (specifically, User, Management, Client Side and Server Side). The subnets we will introduce to perform the composition in Par. 3.6.4 are template models, too. The remaining models (Application, Device and Access Network) are not templates. For the sake of conciseness, only the formalization of the User Model Template, according to the notation presented in Par. 3.5.2, is reported in the following. We describe only the upper layers. Other layers description can be found in [57].

3.6.1 User Level

In the conferencing context a user has the same behavior independently from the specific medium she/he uses (audio, video, etc). This behavior may be modeled by a state machine: a User Model Template that produces and consumes a generic multimedia stream $User_{SAN} = (UserMC_{SAN}, \{(K, S, f_K)\})$ is shown in Figure 3.5, where $UserMC_{SAN} = (userMedia, S, \varnothing)$: the SM set of $UserMC_{SAN}$ is empty and f_K is the instancing function described in the previous Section. User has one parameter K that represents the number of the media streams related to the user. Note that SS_K=S, i.e., this is the trivial case in which the entire structure of $UserMC_{SAN}$ must be replicated. Hence, $User_{SAN} < 1 >$ coincides with the Model Class $UserMC_{SAN}$. In order to show the flexibility of this mechanism, Figure 3.6 shows the template model $User_{SAN} < 2 >$ modeling a user who makes use of both video and audio.

The interface of this model is represented by the two places *MediaInbound* and *MediaOutbound*, respectively for inbound and outbound traffic. The user may be in a *Producing* or in a *Waiting* state (i.e., "talking" and "not talking" for the audio media) represented by ordinary places. The timed transitions (*P2W* and *W2P*) model the switch between the two states. When the transition *P2W* fires, a token is removed from the *MediaOutbound* place by the output gate *OG0*. On the contrary, when the transition *W2P* fires, a token is added to the place *MediaOutbound* by the gate *OG1*.

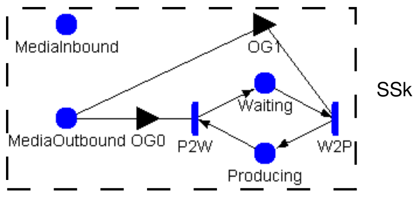


Figure 3.5: The User Model Template

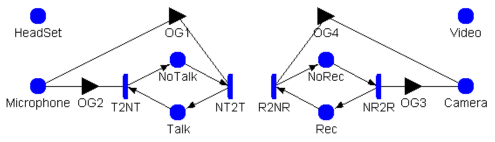


Figure 3.6: The audio conference user model

3.6.2 Application Level

At this level, we model the behavior of the encoding and decoding processes, on both Client and Server Sides. For each medium associated with the Participant, two different models are created, one for each stream direction: inbound-decode and outbound-encode. Therefore, we have a number of instances of the inbound and outbound Application model depending on the media enjoyed by the participant The two patterns of inbound and outbound application models are depicted in Figure 3.7 and Figure 3.8, respectively. The former model is able to maintain a token in the *output* place starting from a set of structured tokens (messages) arriving at the *app_in* place. The

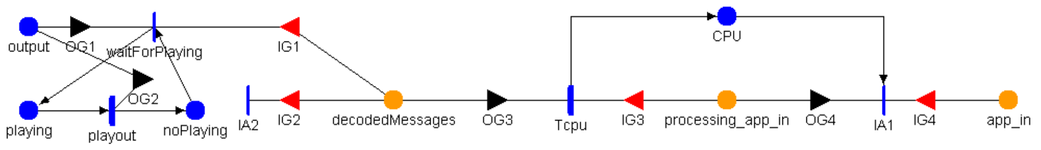


Figure 3.7: The inbound Application model

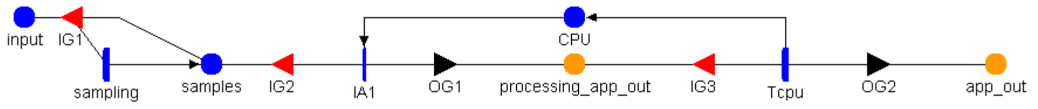


Figure 3.8: The outbound Application model

latter is instead devoted to the production of structured tokens to be put into the *app_out* place, with the presence of an incoming media flow being represented by a token in the *input* place. In detail, the interfaces of the inbound application model are represented by the places *output* and *app_in*: a structured token in *app_in* indicates the presence of a message ready to be decoded; after the decoding process (that keeps the CPU busy), the decoded data are ready to be played out. On the left of the image there is a watchdog that is responsible for maintaining the token in the *output* place: two places *playing* and *noPlaying* with the connected transitions and gates implement the state machine related to the watchdog.

A specular behavior is manifested by the outbound application model. A token in the *input* place indicates the presence of an external flow that needs to be encoded. In this case, $IG1$ enables the *sampling* transition that fires after a sample length and produces a new sample in the *samples* place. Upon arrival of a specific number of tokens in the *samples* place, depending on the encoding standard used, $IA1$ is enabled and, if the CPU is not busy, the generation of a message is activated. After a predefined processing time, *Tcpu* fires and the message is made available at the output interface *app_out*.

3.6.3 Management Level

The Management level is specific to the Conferencing Server and it is not present in the model on the Client Side. This level is responsible for transcod-

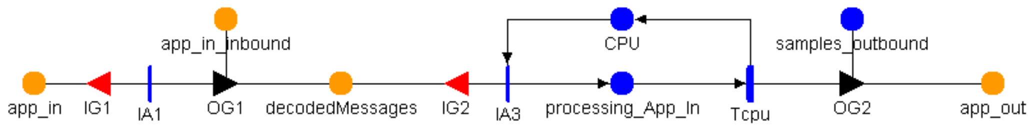


Figure 3.9: The Server Side Management additional model

ing between the different media codecs adopted by participants, as well as for forwarding messages to them, by relying on stored information about the conference (that the Conferencing Server needs to keep in memory). The overall model of this level instantiates the aforementioned application model templates (inbound and outbound) to transcode messages, in conjunction with another model, described in this paragraph, that deals with the generation and transmission of messages. The overall Management model, like the application models, is specific to the particular media and can hence be instantiated ST times (ST being the number of media streams involved in the conference). Figure 3.9 depicts the SAN representation of this additional model.

Given its function, the additional model offers two interfaces to the device level (*app_in* and *app_out*), as well as four connection points with the inbound application model (*app_in_inbound* and *decodedMessages*) and with the outbound application model (*samples_outbound* and *app_out*), both used for potential messages transcoding. When a new message arrives at the server application level on the interface *app_in*, $IG1$, $IA1$, *OG1* check whether the codec used in the message is the standard codec chosen for the conference. In this case the message is put into the *decodedMessages* place; otherwise, it may be transferred to the inbound application model through the *app_in_inbound* place. The transcoded messages produced by the inbound application model are put into the *decodedMessages* place, where the CPU finds only messages encoded with the standard codec, which are elaborated and sent to all participants through $OG2$. Before putting messages into the *app_out* place, the server checks the codec of choice for each participant and, if not consistent with the standard one, it puts the samples into the *samples_outbound* place (connected to the outbound application model) for the correct message encoding.

3.6.4 Composing Model Stacks

The aim of this Subsection is twofold: it first shows how composition has been applied and then how it has been implemented in the Möbius framework [33]. A two-level composition strategy has been implemented: an "intra-stack" composition to create Participant and Conferencing Server composed models, and an "inter-stack" composition at a higher level to create the overall conferencing system model. The "intra-stack" compositions representing the Client Side and the Server Side model stacks are shown in Figure 3.10 and Figure 3.11, respectively.

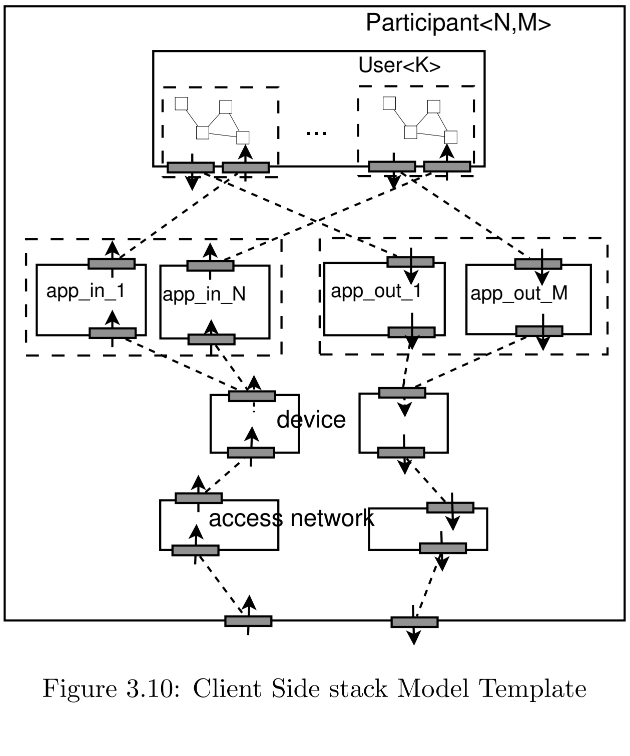


Figure 3.10: Client Side stack Model Template

Starting from the top of Figure 3.10, the Model Template described in Subsection 3.6.1 is used inside the User level according to the parameter K (the number of involved media). At the Application level, several models described in Par. 3.6.2 are used. The number of inbound and outbound

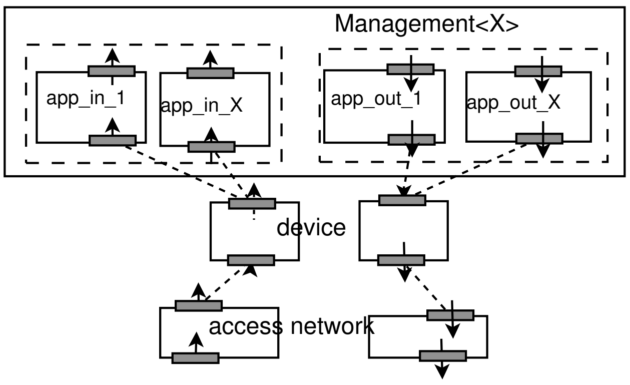


Figure 3.11: Server Side stack Model Template

application model replicas is determined from N and M, representing, respectively, the number of received and transmitted media streams. The two values of N and M can be different (and not equal to K) in those cases when a client is producing but not receiving a specific medium (like, e.g, for a mobile phone, not equipped with a camera, which is not capable to generate its own video stream, but can nonetheless receive video sent by other participants). In the simplest case of an audio conference where all participants can both listen and speak we have $K = M = N = 1$. Both at device and at access network levels, two models, according to the direction of the media flows, are instantiated. It is clear that the whole Participant model is itself a Model Template since entire submodels can be replicated. Figure 3.11 depicts the "intra-stack" composition at Server Side where the different part is represented by the Management model (since Server Side and Client Side use the same Device and Access Network models). The model is in charge of representing the Conferencing Server's audio and video mixing functions: in order to capture such concepts, the model is built by using hierarchical composition. In fact, transcoding media streams may be necessary depending on the adopted encoding-decoding formats, as well as on the capabilities of the involved devices. Encoding and decoding are modeled by using the application model templates (respectively *app_out* and *app_in*) described in the previous paragraph. These are model templates in X parameter, that is the number of the different encoders and decoders used in the conference.

The “intra-stack” composition is performed by place superpositions, where it is naturally possible, or by using some interconnection SANs: Figure 3.12 shows the Model Template of the interconnection network between device and application models on the inbound branch of both Participant Side and Server Side. Figure 3.13 shows the real instantiation obtained by replicating, for two media (audio and video), the SS_n subgraph. The latter model allows to connect the device level of a Participant/Conferencing Server to the two inbound application models. It basically represents a simple messages separator, working in accordance with the media to which they refer.

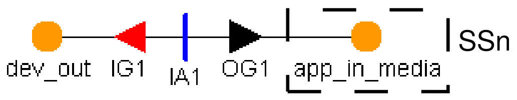


Figure 3.12: The Device-to-Application composition network Model Template

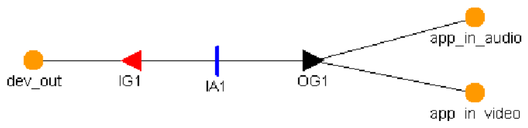


Figure 3.13: A Device-to-Application composition network model

The “inter-stack” composition, starting from the definition of conference deployment (in terms of the number of conference participants, as well as overall Conferencing Server configuration), joins all the instances of the peers stack model (Participants and Conferencing Server) through a proper network, on the basis of a provided topology.

The two composition steps described above are implemented into the Möbius framework by means of the Rep and Join operators [33]. The set of participants is created by replication and instantiation of the entire Client Side model stack. Figure 3.14 shows an example of an overall composed model of a centralized audio conferencing system. The “intra-stack” composition has been implemented through the *participant* and *server* Join models. Similarly, the“inter-stack” composition is given by the combined action of both the *Rep* (to replicate participants) and the *system* Join model (acting as a top level joining all the involved entities). To identify single participants, by assigning a specific id to each of them, a simple *idAssignment* model has been implemented. This model is similar to the one proposed in [31] and allows to associate an index to each participant instance. Indeed, having

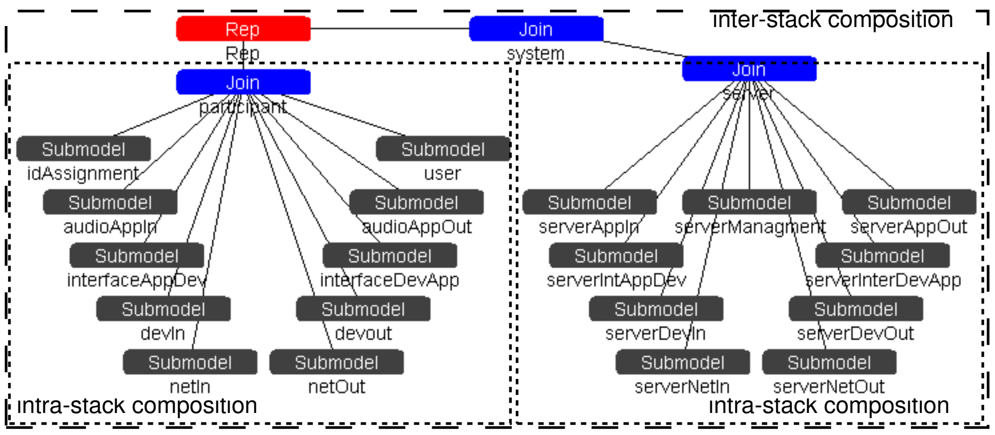


Figure 3.14: The overall composed system model

non-anonymous participants is critical when, for example, a measure on a specific instance is requested or a different behavior must be modeled for it. The composition SAN is in this case trivial, since different models have to share some communication places.

3.7 Dealing with Model Scalability: Stub and Reduced Models

The composed model of a complete conferencing system, shown at the end of the previous section, requires increasing simulation time and memory space during the solving phase and thus call for the introduction of scalable solutions. These problems are due to the huge use of extended places containing data structures arrays that need a high start time to be solved. A possible solution relies on storing data in a database, while keeping in memory just the references to the tuples. However, this strategy is not effective enough for our scope because it just helps overcome the memory consumption issue, while leaving unaffected the computation time. Therefore, we herein propose two different approaches to face scalability issues: (i) stub models, that replace some levels of peer stack with simplified models, and (ii) reduced models, that take the place of the entire peer stack. These two techniques are not mutually exclusive and can be jointly used to model the system. They introduce different approximation levels that we have analyzed by means of simulations. We show the obtained results in the following section.

3.7.1 Stub Model

A stub model, as in software development techniques, may simulate the behavior of an existing, more detailed, model or be a temporary substitute for a yet to be developed model, hence reducing the complexity of a complete one. The stub models can be used to perform analysis at a precise level among those previously identified, by simulating the behavior of the remaining ones. The use of stubs is simplified by the layered nature of system and models, as well as by the compositional nature of the modeling methodology: clearly, a stub for a generic model layer must exhibit the same interface of the models it replaces. Stub models, like the complete one, do not prevent from identifying the single participant and assigning it a different behavior. They also reduce memory occupation for the complete stack; though, as it will be shown in the following, they still entail a significant computational complexity. The approximation introduced by a stub model is proportional to its ability to represent the missing levels: the greater the accuracy of the model, the lower the error on results. The adoption of stubs can be useful for two different purposes: i) to focus on the model of a specific level, in which case a stub acts as a substitute for the models of the lower levels; ii) to build a smaller scale model of the overall conferencing system, by considering just one or few levels both for participants and for the server.

3.7.2 Reduced Model

In many cases the analysis of a conferencing system is conducted to calibrate or to estimate the performance features of the conferencing server: in these cases we can ignore the identification of each single participant, hence allowing for more participant stacks to be collapsed into a *reduced* model. The *reduced* model is here presented: it can be connected directly to the Conferencing Server model stack and it generates the load related to n participants

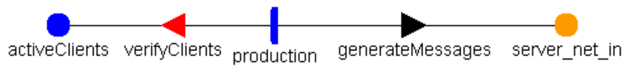


Figure 3.15: The client reduced model

without instantiating all of them. This model is depicted in Figure 3.15: (i) the distribution function of the *production* transition models the average rate with which messages are produced by a single participant; (ii) the *activeClients* place is used to store, at each instant of time, the number of transmitting participants; (iii) the *verifyClients* input gate checks the presence of at least one token in the associated place and is used to either increase or decrease the number of active participants, for example in case of configurations involving moderation; (iv) the *generateMessages* output gate injects packets into the network. This simple model introduces an acceptable approximation in the evaluation of server-side performance, with reasonable complexity, as well as reduced memory occupation, as demonstrated in the next Section. Though, it does not allow to conduct performance evaluations at the client side.

3.8 Evaluation of the Proposed Approach

This Section assesses the proposed approach by means of experimental campaigns. Our objectives are: i) validation of the proposed models through a comparison with real data; ii) assessment of the scalability of the modeling methodology; iii) analysis of its power and flexibility when addressing different conference configurations (e.g., with/without transcoding, with/without moderation, etc.).

To this purpose and according to related works [9, 10] we choose CPU usage of the Conferencing Server (for an audio conference) as a key performance indicator. Such measure is usually computed with respect to the number of participants connected to the conferencing system. Before showing the results of our analysis, a description of the system configuration parameters used in the trials must be given (Table 3.1). For the sake of coherence, such

Table 3.1: System parameters

Parameter	**Description**	**Value**
pkts_per_msg	Packets generated at device level for each application level message.	1
sample_len	Duration of a single audio sample (according to G.711 specification).	0.125 ms
samples_per_msg	Samples contained in an application level message (according to G.711 specification).	160
sample_dim	Dimension of an audio sample	8 bit
app_in_Tcpu_mean app_out_Tcpu_mean	Mean CPU time needed to decode/encode a message at the application level.	1.3E-4 ms
app_in_Tcpu_var app_out_Tcpu_var	Variance of CPU time needed to decode/encode a message at application level.	1.3E-5 ms
mngmt_Tcpu_mean	Mean of CPU time spent by the server to generate a single application message	1.3E-4 ms/msg
mngmt_Tcpu_var	Variance of CPU time spent by the server to generate a single application message	1.3E-5 ms/msg
dev_Tcpu_mean	Mean CPU time to decompose/recompose a message at device level.	0.6E-6 ms
dev_Tcpu_var	Variance of CPU time to decompose/recompose a message at device level.	0.6E-7 ms
net_band	Upstream/downstream bandwidth needed for a message	67 Kbit/s

parameters have been chosen according to the above cited works. The first two objectives are fulfilled in Subsection 3.8.1 and the third one in Subsection 3.8.2

3.8.1 Validation with Real Data

This group of simulations aims at demonstrating the equivalence between our models and real experimental data. In order to achieve this goal, we compared the estimated CPU usage obtained by solving models with the values reported in [9, 10]. This validation requires a high level of scalability in terms of number of participants: for this reason, we introduce at first a comparative analysis of the complete model with *stub* and *reduction* approaches to evaluate their scalability features.

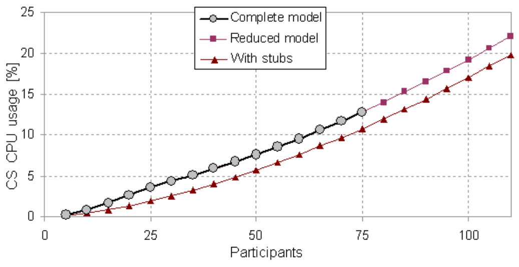


Figure 3.16: Scalability analysis: CPU usage

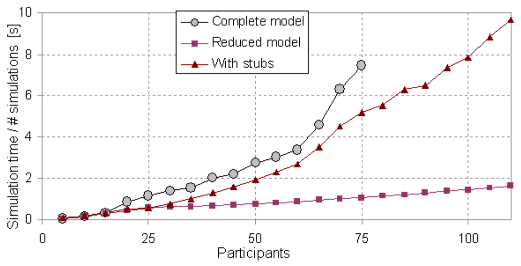


Figure 3.17: Scalability analysis: simulation time

Complete vs stub vs reduced model. Based on the specific well-defined reference scenario, the three models are compared: a complete model obtained by composing the server stack with the replicas of the participant stack; a model obtained by substitution of the *device* and *access network* levels with stub SAN models; a model built using reduction techniques as defined in Section 3.7. The accuracy of the *stub* and *reduced* models compared to the complete one, that is the estimated CPU usage of the Conferencing Server against the number of connected participants, is reported in Figure 3.16. The simulation times associated with these studies are instead reported in Figure 3.17. The two analyses have been conducted for a number of participants between 5 and 110: a quick look at the graphs tells us that the reduced model is able to analyze up to 110 participants within reasonable simulation time. The complete model is instead capable to arrive at a maximum of 75 participants on the computer used for the simulation (Intel(R) Core(TM) 2 Duo @ 2.53GHz, 2 GB RAM) due to RAM saturation during the solving phase. The reduced model scales very well in simulation time and, according to the first diagram, it is also as accurate as the complete model for the considered conference scenario. The drawback in its use resides in the potentially unsatisfactory level of detail: some analyses that require non-anonymous participants (e.g., the QoS of the conference as perceived by end-user), cannot be conducted because of its limited description of the participants' model stacks. On the contrary, a stub solution seems to best strike the balance between accuracy, simulation time and level of detail. In particular, the considered stub models superpose communication places without introducing CPU effort at device and network levels: the accuracy can be further improved by constructing more realistic (yet less performing) stub networks.

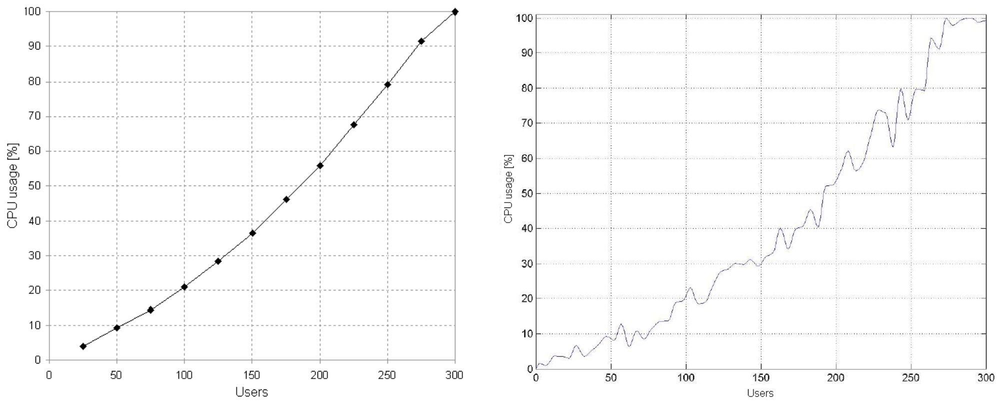


Figure 3.18: Reduced model vs real data: centralized scenario

Reduced model vs real data. We compare the solutions obtained with the reduced model with the real experimental data for the two different configurations described in Section 3.3: a) centralized scenario, where a single focus acts as the Conferencing Server for a number of participants varying from 25 and 300; b) distributed scenario, where a "two islands" configuration is used for load balancing purposes (150 participants per focus). The results are reported in Figure 3.18 and Figure 3.19, respectively. In both scenarios the reduced model fits well the real measured data.

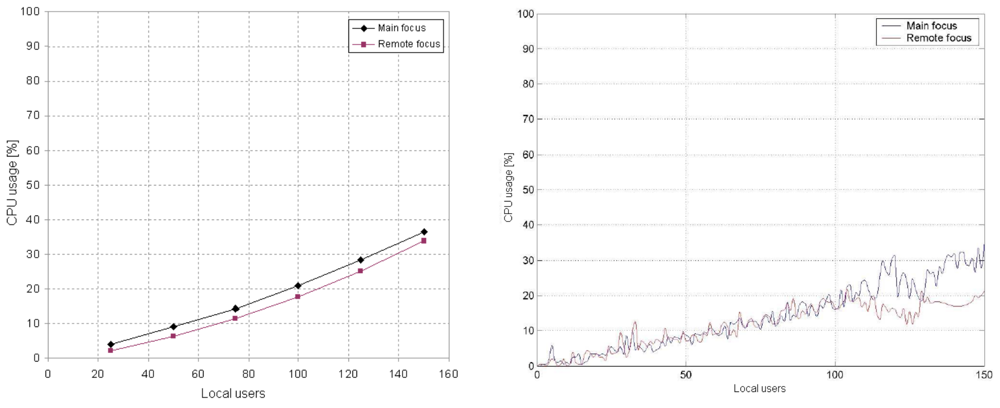


Figure 3.19: Reduced model vs real data: distributed scenario

3.8.2 Evaluating Power and Flexibility

So far we have validated our approach both in terms of accuracy of the results and scalability. The next two experiments point out the power and flexibility of the proposed modeling approach by evaluating the system's performance under different conference configurations. More precisely, we evaluate the effects of moderation and transcoding on the CPU usage of the central Conferencing Server for an audio conference. The aforementioned experiments are conducted by varying two key simulation parameters: the probability of being muted by the moderator and the probability of a flow to be transcoded. For both of them, we employed complete models, limiting to a value of 50 the number of participants. In the performed simulations, all participants use identical devices and present the same probability to talk and probability to have their flows be transcoded. However, the adoption of a complete stack model allows us to assign different behaviors and different devices to each user. This paves the way for further analysis that can be realized by introducing heterogeneity in the composition of the participants set. Moreover, this enables future works on performance studies conducted from a user's perspective, by focusing on parameters like, e.g., the performance of user devices, endpoint CPU utilization, perceived delay, etc..

Complete model: moderation-based conference. In this study, starting from a sample conference, we evaluate the difference between the CPU usage of the Conferencing Server either in the presence or in the absence of moderation. According to the User model described in Section 3.6, the evaluation can be characterized by the *Prel* parameter, that is the probability to release a conversation when talking (or, similarly, to be muted by the moderator in a moderation-based conference). Several values of the *Prel* parameter have been used for the analysis. Figure 3.20 and Figure 3.21 show the results of these simulations, by highlighting a reduction on the CPU usage as long as the *Prel* parameter increases. This is clearly due to the reduced number of incoming audio flows to be processed.

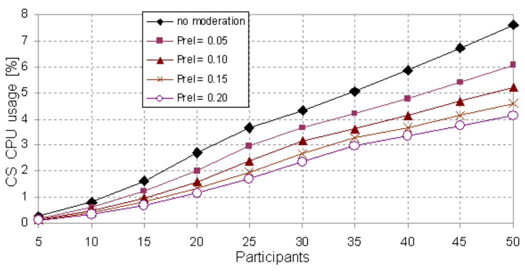


Figure 3.20: CPU usage vs number of users for different Prel values

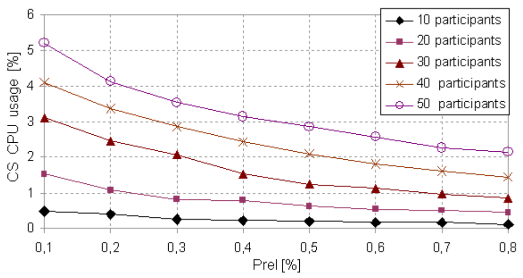


Figure 3.21: CPU usage vs Prel for different conference sizes

Complete model: conference with transcoding In the last experiment we evaluate the potential impact due to the presence of transcoding. We suppose that a portion *Cdtrcd* of participants does not need any transcoding, while the remaining part requires that the Conferencing Server transcodes messages (which increases server CPU usage). Several values of *Cdtrcd* have been used. The results are shown in Figure 3.22 and Figure 3.23. We can notice how the CPU workload is progressively lighter as long as the number of participants needing server-side transcoding decreases (i.e., by increasing the value of the *Cdtrcd* parameter).

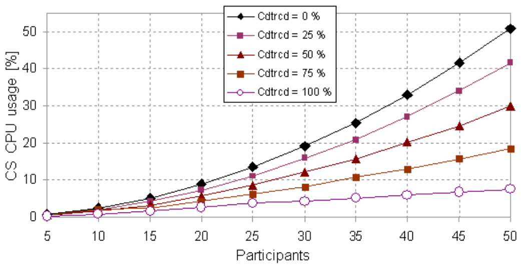


Figure 3.22: CPU usage vs number of users for different Cdtrcd

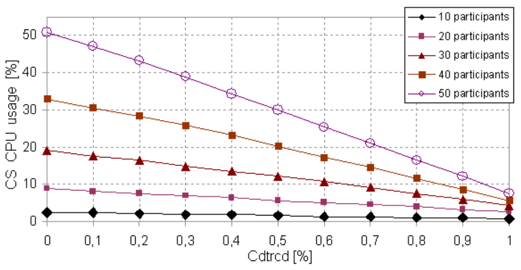


Figure 3.23: CPU usage vs Cdtrcd for different conference sizes

Chapter 4

Real-time applications and infrastructure part II: ongoing standardization work

4.1 Introduction

Real-time multimedia communications over the Internet will assist to a major breakthrough in the near feature. Both IETF and W3C (World Wide Web Consortium) are jointly working on the definition of a new architecture enabling browser-to-browser interactive communications. Two cooperating working groups are committed on such a task: the IETF RTCWeb (Real-Time Communication in Web-browsers) WG and the W3C WebRTC (Web Real-Time Communications) one.

On the other hand, multimedia web conferencing standardization efforts are focusing on standard architectures enabling interoperability among telepresence systems, where new needs arise for the comprehensive metadata description of media streams and for the management of an increased number of contemporary media flows. The IETF CLUE (ControLling mUltiple streams for tElepresence) WG is dealing with these issues.

Both the ongoing works are strictly interconnected since major decisions on common issues, such as the standardization of media stream multiplexing [47], will affect both the activities' outputs.

In this chapter we provide an overview of the current efforts, by focusing in particular on telepresence activities we are involved in.

4.2 Real-time communications in the Web

Integrating interactive multimedia features in web browser could let real-time communications over the Internet dramatically increase, thanks to browsers diffusion.

In order to pave the way for interoperability, and in order to make that kind of communication available to as many users as possible, researchers start to work on the definition of a standardized architecture.

The idea is to define a W3C API for Web applications running on any device that makes them able to send and receive real-time media streams and data in a peer-to-peer fashion between browsers.

While the W3C WebRTC WG is more focused on the definition of the application-level local responsibility, the IETF RTCWeb WG concentrates on the protocols interactions between remote nodes. However, at the time of writing, there is no clean separation between the two standardization efforts and they proceed hand in hand.

The to-be-defined W3C API allows browser and client-side scripting languages to interact with the local media capture devices (mics, web cameras, and the like), with encoders and decoders and with local transmission functions. It will likely expand HTML5 specification which already represents an enabling technology for real-time multimedia streams flowing from servers to browsers.

The architectural model is the browser RTC trapezoid reported in Figure 4.1 [54].

The upper part is the representation of application layer, exploiting the so-called Open Web Platform technologies (HTML, CSS (Cascading Style Sheets), the DOM (Document Object Model) convention, JavaScript, and scripting APIs). Application providers deliver HTML pages via HTTP or via Web sockets. Browsers execute the embedded JS code and perform the ren-

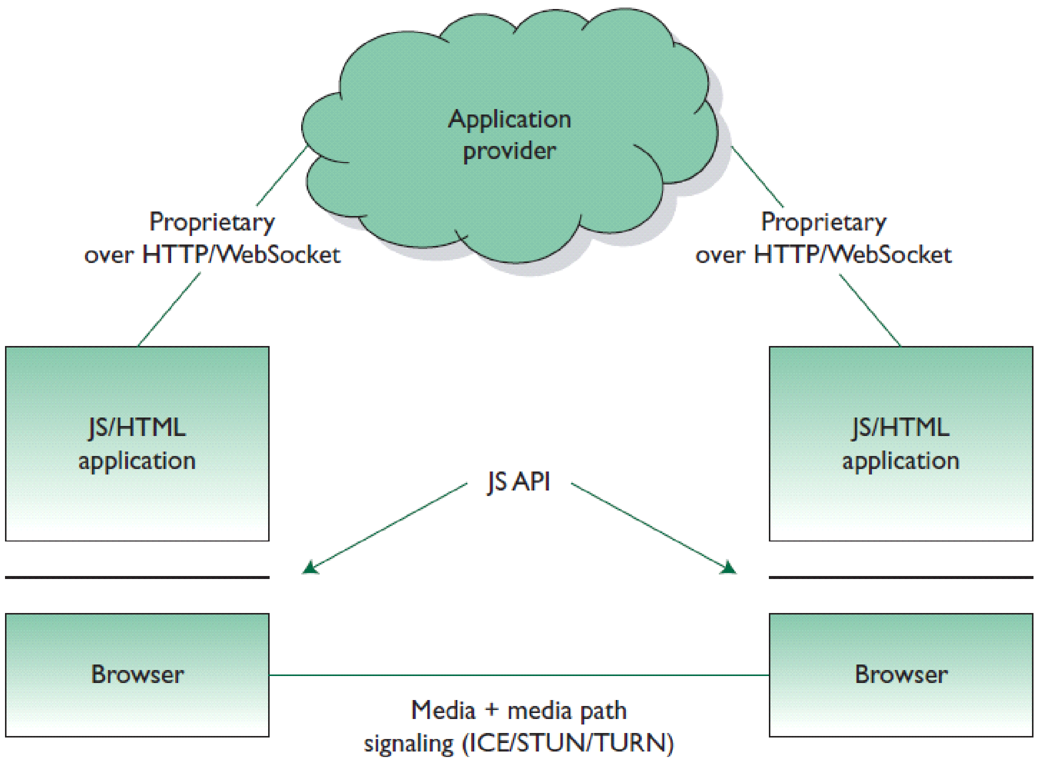

Application provider
Proprietary over HTTP/WebSocket
Proprietary over HTTP/WebSocket
JS/HTML application
JS/HTML application
JS API
Browser
Browser
Media + media path signaling (ICE/STUN/TURN)

Figure 4.1: The RTCWeb architecture

dering as indicated by the CSS stylesheets associated with the page. Scripting APIs let application programmers exploit browsers capabilities. Indeed, as soon as new functions are added to a browser, the W3C devices novel APIs to expose such functions to developers.

The lower part interests browser-to-browser communication both for the media streams, flowing directly between browsers, and for the signaling needed to control the media path.

The media path instantiation goes through complex interactions. First, the caller browser interacts with the caller JS application via the JS API. Then, the caller JS application interacts with the application server. The application servers contacts the callee JS application, which, in turn, interacts with the callee browser via the JS API again.

4.2.1 Signaling: the JSEP protocol

The RTCWeb philosophy envisions to standardize how to control the media plane while leaving the signaling plane to the application layer. In that way, applications can use the signaling protocol they prefer.

This approach has been formalized into the JSEP protocol (Javascript Session Establishment Protocol) [84].

According to such a model, the key information that needs to be exchanged between the browser and the controlling JS application is the media session description, specifying transport and NAT traversal information, the involved media and related metadata (type, format and configuration parameters). The negotiation of the media session between browsers will be handled with the chosen signaling protocol on the application layer. The signaling state machine runs on the application server side, rather than on the browser. In that way, if the user reload the web page, the server can simply push the current state back down to the page and resume the call where it left off.

JSEP provides the standard interface an application needs in order to (i) deal with the session descriptions and (ii) to interact with the ICE state machine [1] for the NAT traversal runned into the web browser.

4.2.2 The WebRTC API

The WebRTC API [19] has to provide to web applications several functionality, starting from connection management, negotiation and control of media streams, encoding and decoding capabilities, and NAT-traversal. The aforementioned functions need to be implemented in next generation browsers, but the specifications have not reached yet a stable state. For example, there isn't yet a decision regarding which audio and video codecs the browser must adopt. The API is being designed around three main concepts:

- PeerConnection: the abstraction of the peer-to-peer channel between web browsers. Once the calling browser establishes a peer connection, it can send MediaStreams objects directly to the remote browser.
- MediaStream: the abstract representation of a media stream. It serves as a handle for managing elaboration on the media stream such as displaying, recording, or transmission. A MediaStream can be extendend to represent a "LocalStream", i.e., the representation of audio or video streams that have been locally captured, or a "RemoteStream", i.e., a stream arriving from a remote peer.
- DataChannel: the abstraction of a generic peer-to-peer transport service between browsers for non-media data types.

All media and data streams will always be encrypted. Media transmission is envisioned to be carried upon SRTP (Secure Real-time Transport Protocol) and by using DTLS (Datagram Transport Layer Security) as an SRTP key and for association management. On the other hand, SCTP (Stream Control Transmission Protocol) over DTLS will be used to handle non media data types.

[1]ICE (Interactive Connectivity Establishment) is a standard NAT traversal protocol. Next-generation browsers are supposed to implement it in order to overcome NAT traversal issue.

4.2.3 RTP multiplexing

In a multimedia session, each medium is typically carried in a separate RTP session with its own RTCP packets. However, to overcome the issue of opening a new "hole" for each stream used, the IETF is working on possibly reducing the number of transport layer ports that RTP-based real-time applications consume by combining (that is, multiplexing) multimedia traffic in a single RTP session [47].

Such a mechanism is currently under analysis. It will affect also the approach to be used to translate media session description information into the selected signaling and negotiation protocol.

4.2.4 Security considerations

The RTCWeb approach and related architecture definitely represent a new challenge in the world of telecommunications because they allow Web browsers to directly communicate with little or no intervention from the server side. This is extremely challenging because it requires that we consider several issues, among which trust and security play a fundamental role. With respect to this last point, a new set of potential security threats comes to the fore when we allow direct browser-to-browser communication.

First, security should be handled in the same way as with other network protocols (such as SIP) that allow for direct communication between any two endpoints. Second, mechanisms must exist that let users verify consent before the actual call starts. Consent verification should be directly enforced by the browser aiming at initiating communication with a potential peer.

Finally, RTC scenarios through the Web clearly call for the browser to interact on a deep level with the node on which it resides to access local audio and video devices.

Access policies must be defined that potentially involve some form of user consent, to avoid privacy issues.

4.3 Telepresence

Telepresence refers to the set of technologies allowing for a real-time communication among several remote participants which gives participants the sensation of being in a same phisical shared environment. The main aim of the involved technologies is the one of providing users with the effect of being present at a place other than their true location. This is what is called a “being-there” experience. Users have to be immersed in a properly equipped room providing them with multi-sensorial stimuli replicating in space and time what is going on in the remote location they are connected to. The equipment within the room itself must be able to capture and send all the information needed to allow a high-fidelity reproduction at a remote site of local participants’ movement, voice, appearence, background sounds, and the like. This can be accomplished by using special designed room with, for example, multiple displays permitting life size image reproduction, multiple cameras, microphones and loudspeakers.

4.3.1 Differences between telepresence and plain web conferencing

Video conferencing can be seen as a basic way of doing telepresence. However, it is not conceived at the outset to provide an immersive experience to participants. Participant’s high quality video is certainly one of the main media stream a telepresence session involves. Nonetheless, telepresence sessions ends up to be made of more media streams than a typical video conference. Telepresence rooms are equipped with several capture devices wisely distributed to best capture the local scene from multiple view points. Each of these streams should be sent to the remote site in order to make it able to chose the set of stream that fits best with its rendering capabilities, i.e., the displays and loud-speakers installed at the remote room. Among an increased number of received media streams, remote sites need to be able to understand the content of the media streams in order to consciously select the most suitable combination they desire to be provided with. Moreover, the spatial distribution of the capturing devices at the sending site should be communicated to the receiving site in order to let it better duplicate the source environment. Spatial information about the issued media streams need to travel along with them. These highlighted needs are those that mainly differentiate plain web conferencing from telepresence applications.

4.4 Overview of the IETF CLUE framework for telepresence

Similarly to the case of conferencing, the need for standardization activity in the field of telepresence arises to address interoperability issues. Several proprietary telepresence systems exist by now, but they are not able to easily interoperate. Although they are mainly based on open standards (SIP, H.264, the H.323 suite, RTP), their interoperability is achievable only by means of operator assistance and expensive additional equipment which translates from one vendor to another. That is mainly due to the lack of a common way of describe and negotiate media streams addressing the needs mentioned above. In 2010 the IETF CLUE (ControLling mUltiple streams for tElepresence) WG was born to solve the task. The goal is producing new specifications for SIP-based conferencing system for the communication among sending systems, receving systems, and intermediate systems in order to properly set a telepresence multi-stream conference and optimize the user's experience.

As usual within the IETF WG, the kick-off phase of the project is the definition of use cases and requirements. Reference use cases that have been identified vary from a simple point-to-point symmetric telepresence session, where two equally-equipped rooms communicate, to more complex scenarios involving more differently equipped sites. Several telepresence scenarios are related to communication among conferencing rooms for business purposes, but also educational applications are considered. This is the case of the telemedicine scenario, where a fully-fledged surgery room is connected with two remote sites, one with the surgeon remotely managing the surgery, and one with students passively assisting the operation. The surgery room is endowed with different kind of devices capturing multiple local video presentation streams to help the surgeon monitor the status of the patient during the surgery (an endoscopic monitor, an X-ray output device, a cardiogram generator, and the like).

Among the major interoperability requirements detected, there are those related to the description of the available media streams a transmitting site can send to remote sites. The description should include both the spatial arrangement of the capturing devices, since it enables a satisfactory reproduction at the receiver, and further metadata about the content of the issued streams, in order to help the receiver in the selection process.

Media streams in a multimedia session are usually described in type and available encodings by means of the Session Description Protocol (SDP) in SIP-based conferencing. Nonetheless, SDP and its extensions do not address the telepresence session description needs. First, SDP does not envision the possibility of provide spatial information about media streams. Second, the "content" SDP extension [46], designed in order to specify the content of the media stream to a more detailed level than the media description line, has been considered unsatisfactory to represent the whole semantics of the streams involved in a telepresence session, since it does not cover all the desired facets. Moreover, its intrinsic ambiguity makes it not suitable to be machine-interpreted.

Given such limitations, the media streams negotiation carried on by means of SDP within SIP cannot fulfill completely telepresence needs. That's why it is on the table of discussion the introduction of a novel protocol enabling a media provider (transmitting site) and a media receiver (receiving site) agree on the streams to be exchanged during a telepresence session. Such a protocol basically envisions an ADVERTISEMENT message, sent from a media provider to a media receiver, and a CONFIGURE message, flowing in the opposite way as a response to the received ADVERTISEMENT message. By means of the ADVERTISEMENT message, a media provider exhibits to the media receivers all the available media streams it can send, and attach to them all the describing metadata. Within the ADVERTISEMENT message, all the needed information about the provider's capabilities (available encodings, bandwidth constraints, and simultaneity constraints - i.e, restrictions on the streams that can be sent simultaneously) are provided to the receiver. The advertised media streams are properly arranged in "capture scenes", that are set of streams representing a semantically complete part of the telepresence room (participants, the overall room, the presentation part of the conference). This media flows organization has been conceived in order to help the receiver in the streams selection process. On the other hand, the CONFIGURE message allows the receiver to manifest to the sender the results of its internal selection process, by indicating to the media provider the only streams of interest.

4.5 Contribution to the ongoing standardization work

CLUE activities are still in an embrional phase, even if some milestones have been pointed out in terms of requirements and main matters. The WG is stuck on the very basic issue of defining the inner workings of CLUE, in terms of protocol mechanism, protocol messages, call flows and main actors and entities involved in telepresence scenarios. At the time of writing, all such issues are included at different levels of deepening in [37]. We get into CLUE activities in this context. We propose to organize the work flow in ordinary documents, by taking inspiration from the effective experience gained within the XCON WG. The lesson learned from the centralized conferencing standard specification is that it is very useful to split the efforts among different kind of deliverable drafts, namely:

- a framework document, containing high-level descriptions of the overall architecture, protocol, messages, data model and call flows. The reader of this document should come out with a clear idea of what CLUE means and does, while being redirected to the companion documents for the details;

- a datamodel document, identifying and formally defining the main entities considered in a standard telepresence environment, as well as their relationships;
- a signaling document, defining and showing by means of detailed call flows how the SDP Offer/Answer negotiation and CLUE protocol messages concur in effectively setting up a CLUE session.
- a protocol document, defining the ADVERTISEMENT and CONFIGURE messages, as well as their semantics;

We are directly involved in the drafting of the last documents, as well as in the revision of the first one. In the following state of the art about such works is provided.

4.6 A data model for the CLUE framework

Data model documents describe the entities managed within the application context in order to better pointing out which they are and how they are characterized. Such entities have to be managed in the framework by means of the protocols involved to realize the envisioned application scenarios.

We support the telepresence standard definition by drafting a data model reflecting information and constraints currently contained in [37] and [42].

In [67], we have tried to encode such ideas in a formal language, which is XML Schema, to the aim of building non-ambiguous definitions of the concepts defined in the aforementioned documents. This effort is important to foster discussion on a common basis of understanding. It allows to identify problems and weaknesses in the concepts and their natural language definitions used in the cited drafts.

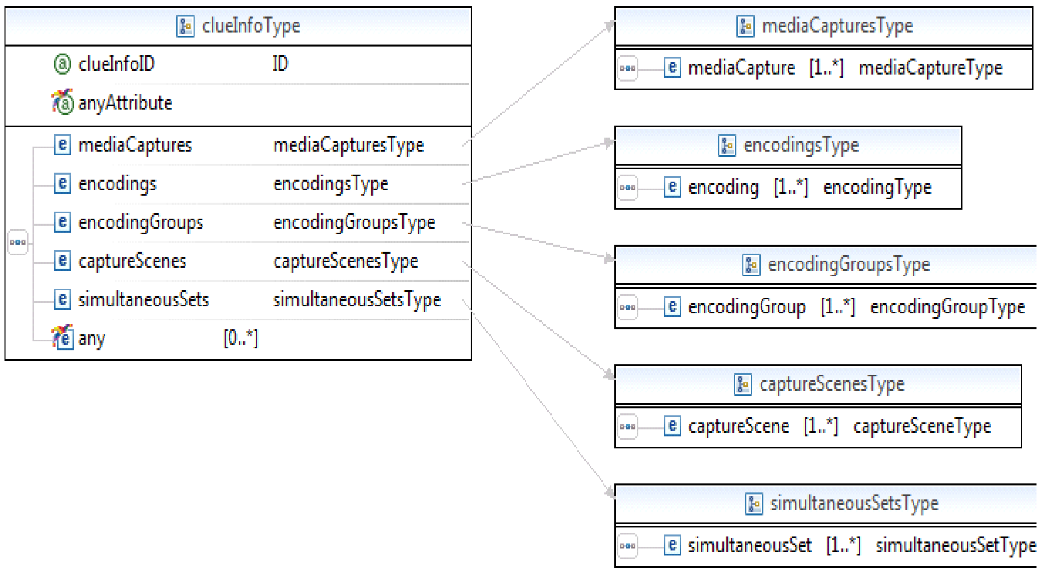

clueInfoType
clueInfoID ID
anyAttribute
mediaCaptures mediaCapturesType
encodings encodingsType
encodingGroups encodingGroupsType
captureScenes captureScenesType
simultaneousSets simultaneousSetsType
any [0..*]
mediaCapturesType
mediaCapture [1..*] mediaCaptureType
encodingsType
encoding [1..*] encodingType
encodingGroupsType
encodingGroup [1..*] encodingGroupType
captureScenesType
captureScene [1..*] captureSceneType
simultaneousSetsType
simultaneousSet [1..*] simultaneousSetType

Figure 4.2: Overview of the CLUE XML Schema datamodel

We started from the formal description of the telepresence capabilities of a telepresence room in terms of available streams and encodings. Such set of information, indeed, consitutes the basis of the information carried into the ADVERTISEMENT messages sent from the endpoint representing a telepresence room to the other endpoint(s) involved in the telepresence conference.

Figure4.2 presents a schematic representation of the clue info type. The clue info type is the type of the root XML element named `<clueInfo>` which contains all the information that are needed to describe the CLUE capabilities of a telepresence room. It comprises the following subelements:

- mediaCaptures: the list of media captures available. Within the CLUE framework, media captures are the fundamental representations of streams that a device can transmit. They can represent the immediate output of a physical source (e.g. camera, microphone) or "synthetic" source (e.g. laptop, computer, DVD player), but also concepts such as "the loudest speaker stream".

- encodings: the list of the individual encoding appliable on media captures, characterized through a set of maximum values (e.g., maximum bandwidth, maximum frame rate, ...).

- encodingGroups: the list of encodings which have been grouped together. An encoding group includes a set of one or more individual encodings, plus some parameters that apply to the group as a whole, as for example the maximum bandwith totally exploitable by the encodings of the group. That last kind of parameters represents the total encoding capability of the provider which is sub-divided across the individual encodings composing the group.
- captureScenes: the list of scenes, that are aggregates of captures. Different scenes are semantically and spatially independent from each other.
- simultaneousSets: the list of group of capture that can not be sent simultaneously by the media provider. Captures can not be sent simultaneously when they are generated from the same device or for other motivation of the media provider deriving from its internal policies. A way for expressing this kind of constraints is provided through simultaneous set.

In the following further details about the main components of the data model are provided.

4.6.1 Media captures

Media Capture type is the base class for describing a media capture. Video capture type, audio capture type, text capture type are specialization of that class, i.e. they derive from that class. We design it as an abstract class in order to force using specific-media type definition. The result is that all of such specialized media type share a common part, which is constituted by the elements composing the media capture type (seeFigure 4.3). Several elements compose a media capture.

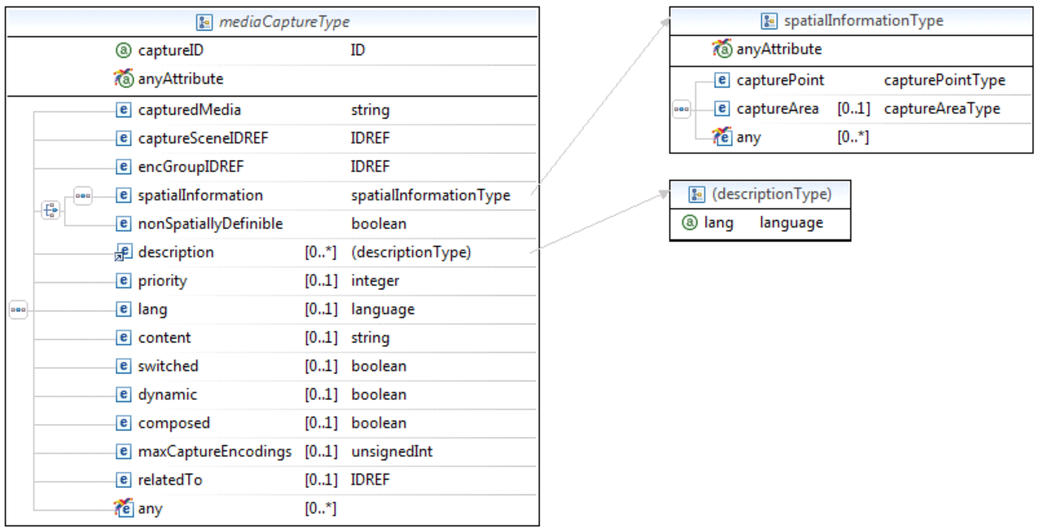


Figure 4.3: XML Schema description of a CLUE media capture

The first three of them are mandatory and describe, respectively, the media type, the capture scene the media capture refers to, and the encoding group the capture is associated with. A media capture can be spatially definible or not. If it is, then a `<spatialInformation>` element must appear in its XML representation, otherwise the capture is marked with a `<nonSpatiallyDefinible>` element set to true. Non spatially-definible captures are for example pre-recorded media streams sent in the conference, but not spatially related to the space of the transmitting site. On the other hand, the `<spatialInformation>` element contains the capture point, i.e, the position of the capturing device in the space of the scene, and, optionally, a capture area representing the area captured by the originating device. A capture area is represented by means of four points, each of them made by three coordinates in the space of the scene (seeFigure 4.4). The capturePoint definition envisions, in addition to the position of the capture device, an embedded further point that determines the pointing direction (seeFigure 4.5).

Media captures are further described by optional fields. Among them, there are one or more human-readable textual description, that can be expressed in different languages. `<priority>` indicates through an integer num-

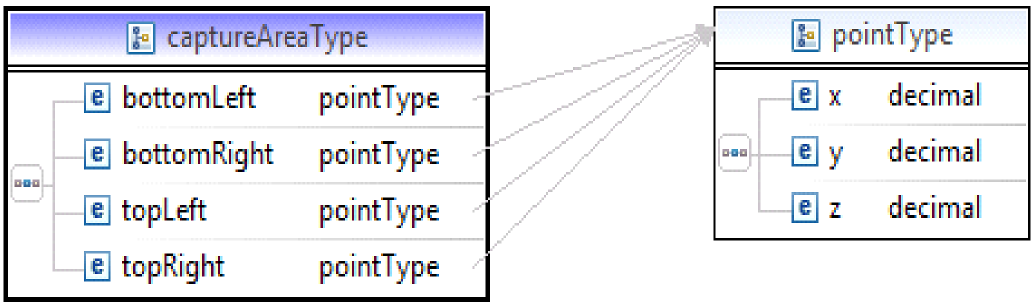


Figure 4.4: Details on the definition of the capture area

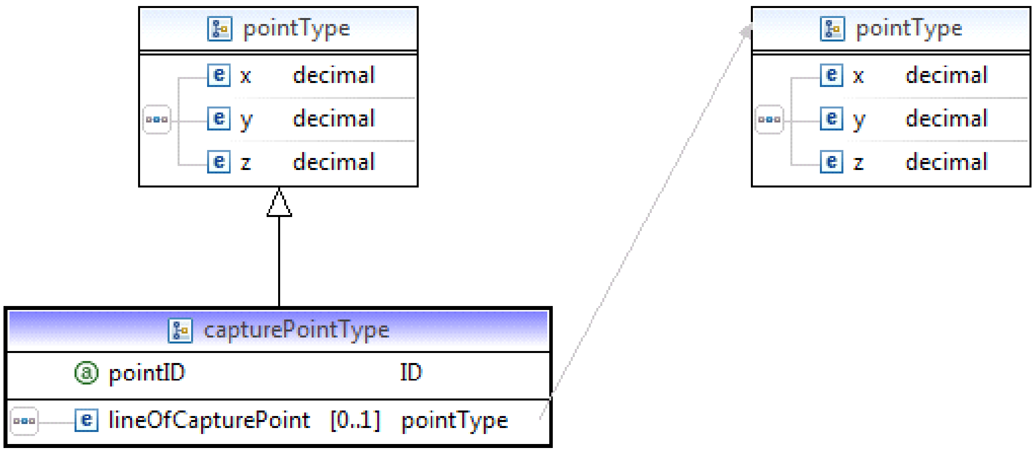


Figure 4.5: Details on the definition of the capture point

ber the importance associated by the media provider to a certain capture in the context of the conference in order to recommend capture to the consumer in the selection process. The language used in the capture is encoded into the `<language>` parameter. Some indication about the content of the capture can provided through the `<content>` element, which contains metadata about the capture as envisioned for the SDP content attribute (RFC4796). Actually, there is an ongoing work about the definition of more specific keywords defining the subject of a capture to be used in place of (or besides) `<content>`.

Then, there are a list of boolean element specifying, respectively, if the capture device originating the capture may move during the conference (`<dynamic>`), if the capture can switch to another subject of another site during the conference (`<switching>`, and if it is the result of a mixing process (`<composed>`). Finally the `<maxCaptureEncoding>` element indicates the maximum number of individual encodings that can be associated with a capture at the same time.

In Figure4.6 we have reported the audio capture type example. This type derives from the media capture type and adds media-specific information that are those about the audio channel format, which can be “mono” or “stereo”, and the mic pattern, defining the kind of mic that has captured the audio signal (i.e., “figure8”, “shotgun”, “cardioid”, and so on).

4.6.2 Capture scenes

A media provider arranges media captures in a capture scene to help the media consumer choose which captures it wants. Capture scenes are an organized structure of captures, describing a certain scene. For example, we can have a scene capturing the participants that appear in the telepresence room and a scene capturing a side presentation. The captures listed inside a capture scene can be spatially related to each other: indeed, the spatial information associated with the media captures of a same capture scene must share the same coordinate space and the same scale, that are determined

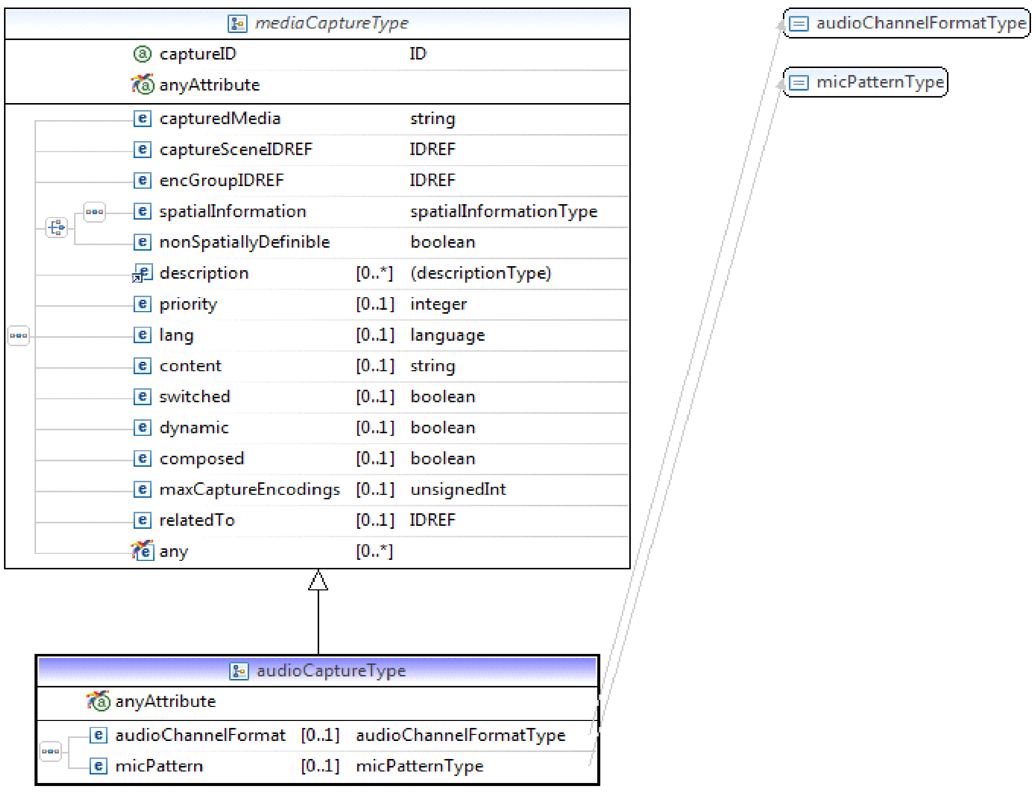


Figure 4.6: Model of an audio capture

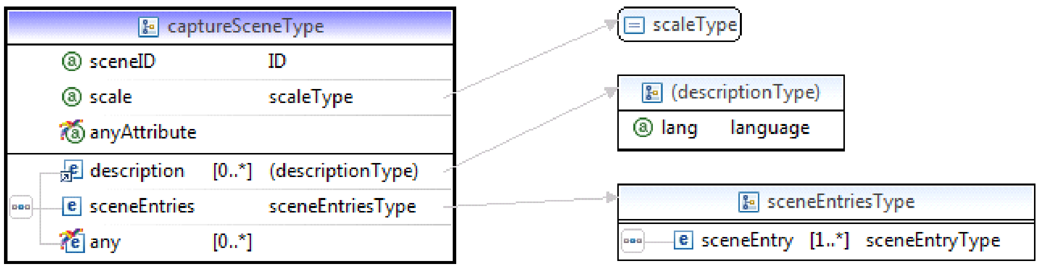


Figure 4.7: XML Schema representation of a CLUE capture scene

from the involving capture scenes. Within a capture scene, media captures are organized in "capture scene entries", where each entry is a set of possible alternatives of capture of the same media type representing the scene.

Given such semantics, the XML representation of a capture scene is the one depicted in Figure 4.7. There is a "scale" attribute indicating the unity of measure of point coordinates used in the media captures' spatial descriptions, and a `<sceneEntries>` element, containing the list of different capture scene entries in the form of `<sceneEntry>` objects.

4.7 CLUE signaling: open issues

Call flow documents are aimed to document how the architecture works. Within the context of CLUE, call flows have the burden of clarifying how signaling is conducted in the course of CLUE sessions. This includes how SIP/SDP signaling is applied to CLUE conferences as well as defining a CLUE-specific signaling protocol that complements SIP/SDP and supports negotiation of CLUE application level data.

We promote a drafty solution in [53]. Figure4.7 depicts the main ideas behind it.

```
+----------+                      +-----------+
|   EP1    |                      |    EP2    |
|          |                      |           |
+----+-----+                      +-----+-----+
     |                                  |
     |                                  |
     | INVITE (BASIC SDP+COMEDIA)       |
     |--------------------------------->|
     |                                  |
     |                                  |
     |       200 OK (BASIC SDP+COMEDIA)|
     |<---------------------------------|
     |                                  |
     |                                  |
     | ACK                              |
     |--------------------------------->|
     |                                  |
     |                                  |
     |                                  |
     |<################################>|
     |  ?? BASIC SDP MEDIA SESSION  ??  |
     |<################################>|
     |                                  |
     |                                  |
     |TCP CONNECT (CLUE CTRL CHANNEL)   |
     |================================>|
     |           ...                    |
     |<================================>|
     |   CLUE CTRL CHANNEL ESTABLISHED  |
     |<================================>|
     |                                  |
     |                                  |
     | ADVERTISEMENT 1                  |
     |*********************************>|
     |                                  |
     |                                  |
     |                  ADVERTISEMENT 2 |
     |<*********************************|
     |                                  |
     |                                  |
     |                                  |
     |                      CONFIGURE 1 |
     |<*********************************|
     |                                  |
     |                                  |
     | CONFIGURE 2                      |
     |*********************************>|
     |                                  |
     |                                  |
     |                                  |
     | REINVITE (UPDATED SDP)           |
     |--------------------------------->|
     |                                  |
     |                                  |
     |             200 OK (UPDATED SDP)|
     |<---------------------------------|
     |                                  |
     |                                  |
     | ACK                              |
     |--------------------------------->|
     |                                  |
     |<################################>|
     |   UPDATED SDP MEDIA SESSION      |
     |<################################>|
     |                                  |
     |                                  |
     |                                  |
     |                                  |
     |                                  |
     v                                  v
```

Basically, the two communicating endpoint, EP1 and EP2, start negotiating a basic audio-video SIP-conference with in addition a negotiation of a further channel to be used for the CLUE signaling. Such a channel is negotiated by following the COMEDIA approach, as already envisioned within MEDIACTRL for the creation of the control channel between the AS and the

MS. When the channel is established, the two endpoints can properly set the telepresence multi-stream session by issuing ADVERTISEMENT and CONFIGURE messages over it.

While that can be considered the general reference behavior, the standardization is still far from a stable and complete solution.

Indeed, many issues have still to be faced.

First of all, the RTP bundle mechanism (seePar. 4.2.3) that is currently under definition, will affect the association mechanism between media captures and the identifiers used within the m-lines of the SDP description.

Second, CLUE protocol messages have not been formally defined yet. As we already mentioned, basically they will be an ADVERTISEMENT message, which makes a sender able to announce its capabilities in terms of available streams, and a CONFIGURE message, which makes a receiver able to select the preferred streams advertised by a sender. Such messages will probably use or refere to entities defined in the data model. Even if the CLUE channel is reliable, it has to be discussed if an application acknowledge is still needed to indicate that messages have been received and correctly parsed. That could lead to the introduction of new CLUE protocol messages. Moreover, it is on the agenda the research of a proper mechanism to comunicate variation with respect to an ADVERTISEMENT already issued. Such variations can be related to the deletion of a capture of a change in the offer, for example. A solution can be the one of issuing only the updates (a "diff") and an alternative can be that of sending a brand new ADVERTISEMENT message: both have to be evaluated, also in terms of overhead.

Chapter 5

Real-time applications and infrastructure part III: a security perspective based on social behavior analysis

5.1 Introduction

Both conferencing and telepresence system are complex real-time multimedia applications built on top of VoIP networks.

A VoIP network is a system engineered on top of an IP network which is able to support real-time voice communications among network-connected parties. Enabling technologies of such a system are: (i) an application protocol for singaling, i.e, dealing with instantiating, managing and terminating calls, and (ii) a transportprotocol devoted to carry multimedia streams across the network in a timely manner. The most widespread VoIP technologies are SIP, as the signaling protocol, and RTP, for media transfer. Both of them are standard IETF protocols.

Basic VoIP services nowadays are offered by most telecom providers and, by looking at their growing diffusion, they will probably completely replace the circuit-based telephony in the future. This is also due to their flexibility accomplishing the creation of a wide range of sophisticated value-added services to customers, including collaborative conferencing and telepresence applications.

Since customers would rely on VoIP networks as they did on PSTN telephony, securing VoIP platforms is a task of major concerns. Indeed, nowadays voice communications are deemed as a critical service offered to the society and need to be as reliable and secure as possible. Compared to old telephony systems, securing VoIP platform is a much more challenging task, since they are exposed to both IP network attacks and new application-specific threats. Social attacks are a particular kind of threat belonging to the last category. Such attacks are performed leveraging the trust VoIP customers put in the telephony service. Social attackers directly contact the victim with a VoIP call for advertisement or to steal critical information by using social engineering techniques. This kind of phenomena, when reaching massive proportion, can become very annoying to VoIP clients and cause economical damages to service providers because of customers' loss.

Given the social nature of VoIP communications, we investigate the possibility of leveraging users' social behavior profiling to detect social attackers in VoIP networks in [30]. Social attackers are indeed expected to exhibit anomalies in their behavior both as caller and as callees. For example, they would perform much more calls than an ordinary VoIP user, since they should aim to reach the widespread audience as possible. On the other hand, they should receive far fewer calls than a normal client, since their social networks should not be made of "friends" interested to call them. These considerations led us to think that studying users's habits and social networks, by looking at the calls they perform and receive, can help securing VoIP platforms from social attacks. We propose a behavioral model for VoIP users that allows for the detection of anomalous social behaviors in VoIP networks by using unsupervised clustering techniques. The study is conducted on real-world data, in the form of VoIP Call Detail Records (CDRs), made available to us by an Italian telecom operator in the framework of a sponsored research project called CAST (*Countering Attacks and Social Threats in VoIP Networks*). The profiling system we herein describe has been developed in the context of such a project, so we will refer to it as to the CAST system. We show how we can identify behavioral patterns associated with the most common anomalous behaviors of VoIP users. We also discuss how we can exploit the expressive power of relational graphs in order to both validate the results of the unsupervised analysis and ease their interpretation by human operators.

The chapter is structured as follows. In Section 5.2 we present a brief taxonomy of the most well-known security threats currently affecting VoIP networks, with a focus on social threats. In Section 5.3 we will present the current state of the art in the field of VoIP security, with an eye on the most interesting proposals related to social attacks detection. Section 5.4 presents an overview of the CAST architecture. The main details associated with its implementation and performance are provided in section 5.5, whereas section 5.6 illustrates the results we obtained by applying the CAST profiler to the analysis of the real data about VoIP calls which were made available to us. As a further contribution to the analysis, we also provide in section 5.7 more information about the application of the theory of social graphs to the analyzed data set.

5.2 VoIP security threats

VoIP (Voice over IP) communications are becoming more and more widespread nowadays, thanks to their positive impact on both capital and operational expenses. However, the possibility of leveraging the existing Internet infrastructure in order to provide such kind of services also entails a number of new challenges that the operators must face, with special regard to the unavoidable issues associated with the exploitation of 'open' solutions, both at the architectural and at the protocol layer. In the depicted scenario, security definitely represents one of the most critical aspects. VoIP threats have been since long studied. An interesting taxonomy related to them can be found in [3]. Basically, such a taxonomy divides threats in several macro-categories, which include the following major representatives:

- Social threats: such attacks have the end users as final targets; *vishing* (i.e., voice phishing), theft of service and SPIT (SPam over Internet Telephony) belong in this category, which will be discussed in greater detail later in the chapter;
- Eavesdropping: these attacks typically take place when an unauthorized malicious user becomes capable of intercepting a VoIP call and, e.g., reconstruct an entire conversation between two users;
- Intentional interruption: this category of attacks mainly refers to Denial of Service, which aims at letting a service become unavailable to the end users. It includes both the traditional attacks to the IP networking infrastructure and a whole set of novel threats specifically targeted to VoIP systems and applications;
- Service abuse: this particular kind of threats mainly concerns all those situations in which a VoIP service is provided in a commercial context. Billing frauds typically fall into this category.

This work mainly focuses on social threats, which have been gaining more and more momentum in the last years. Among such threats, SPIT and vishing definitely play a major role, since they represent a fatal combination of low implementation cost and high effectiveness. SPIT refers to the application of spam (which we all know thanks to its widespread diffusion in the e-mail service) to VoIP communications. It basically consists of the delivery of undesired information to unprotected VoIP users. The first place in SPIT ranking is indisputably owned by the so-called tele-marketeers, i.e., people trying to sell something over the phone. With respect to the specific attack techniques, most of them try and exploit SIP (Session Initiation Protocol) vulnerabilities, since this protocol is undoubtably the most widely deployed among Internet users. Coming to *vishing*, this term refers to an activity which is typical of social engineering, aimed at gathering sensitive data and personal information with the final objective of arriving at a theft of identity. Back in 2002, D. Mitnick [62], one of the most well known hackers in the world, paved the ground for social attacks carried out over the phone line. He demonstrated how such means make it easier for the attacker to gain trust from the attack target thanks to human psychology, which seems to irrationally enhance the level of trust we place in the others when they contact us in a 'warm' way, like it happens with voice communications.

5.3 State of the art in social attacks detection

In this section we briefly present the most interesting research studies aimed at countering SPIT [50]. As it will become apparent in the following, most of the literature embraces approaches which belong to the wide field of Intrusion Detection, ranging from artificial intelligence algorithms to the exploitation of the typical parameters characterizing VoIP networks, through the study of the connections of the social network associated with this kind of communication infrastructures. This brief survey, besides representing a good knowledge base related to the problem at hand, will also prove fundamental to clearly understand the theoretical foundations of the novel solution we propose in the following sections.

A first noteworthy approach is represented by the spectral analysis of the audio signal parameters. Authors of the work presented in [68], starting from the assumption that most SPIT attacks rely on pre-registered audio messages, have developed a software which is capable of performing speech recognition of the caller's voice. The software in question computes a sort of digital fingerprint of the caller, namely the *Spectral Flatness Measure* (SFM), which allows to create a database of undesired interlocutors that have been black-listed in the past.

A different approach relies on the detection of SPIT attackers ("spitters") through the so-called Turing test. A group of researchers at NEC Laboratories in Heidelberg have based their work [69] on the assumption that malicious users are able to send fake signaling messages and can easily bypass non-intrusive controls and countermeasures. They propose to adopt a proactive approach whereby the caller is preliminarily presented with a series of small tests aimed at telling automatic callers, as well as botnets, apart from normal users. Such tests are mainly based on the observation that a typical conversation between human parties follows well-defined interaction patterns.

In [34] the authors present a multi-stage approach combining different solutions in an integrated architecture. The core of such system includes a bayesian learning module, as well as a reputation module based on the theory of social networks. A similar approach has also been embraced by the authors of the work in [13], who use social networks in conjunction with the results of a classification work based on the analysis of call duration. The basic assumption, in this case, resides in the intuitive observation that the longer a conversation between two users lasts, the higher their mutual trustworthiness (which is called, in such context, *call credentials*). The authors use the computed trustworthiness values in combination with a cluster-based analysis of the social graph among users, which helps track down their social habits and arrive at a more reliable identification of potentially anomalous activities.

Another interesting approach is based on the application of semi-supervised clustering techniques [89]. Basically, with such approach the data space is divided into distinct sets, the former containing legitimate conversations and the others related to potential SPIT calls, each characterized by the presence of well-defined call *features*. The system is called semi-supervised since only part of the data is provided with a label extrapolated from users' feedbacks about the performed conversations.

The authors of the project in [51] propose to adopt the so-called DEVS (Discrete Event System Specification) formalism in order to properly model the behavior of malicious users. Such formalism, first introduced by Bernard Ziegler, can be thought of as a sort of extension to the classical Moore finite state machine, with the difference that each state can be assigned a well defined life cycle, together with a *coupling* operation capable of inducing a hierarchical structure into the overall model. Once done with the modeling phase, each new incoming call is analyzed by comparing its specific pattern to the ones embedded in the model. A SPIT level is computed and associated with the user.

We conclude this section by citing one further proposal, made by researchers at Telecom Italia [36]. The SAD system has been designed with the aim of detecting anomalies in a SIP-based VoIP network through the analysis of the spread between the traffic generated by its users and a well-defined (and acceptable) entropy level. The architecture includes a component called VoIP security subsystem which is in charge of performing the core of the mentioned analysis. The so-called Call Detail Records (CDR) associated with the managed conversations are first parsed and then stored in a database, from which they are further analyzed through the application of an algorithm based both on the computation of the entropy level of the network and on a metric capable of providing an indication of the distance between the data under analysis and what is deemed to be the 'normal' behavior of the network itself. Such a distance is computed through both average and standard deviation values allowing for the definition of an interval of variation which is considered to be acceptable with respect to a pre-defined *normality* threshold.

5.4 CAST approach to behavior profiling

CAST (Countering Attacks and Social Threats in VoIP Networks[1]) mainly focuses on social aspects associated with user interaction patterns, with the final goal to extract the typical profiles and thus be capable of identifying potential anomalous behaviors in the analyzed data. The mentioned objective can be achieved through the application of artificial intelligence techniques allowing for the classification of the multiple instances of the problem at hand, especially in those cases when such instances show an intrinsic degree of separation. The actual implementation of the above techniques requires that a good set of *features* be identified by carefully looking at the specific

[1]Seminal work related to the CAST project can be found in [58]

characteristics of the system under study. With a good metric associated with the chosen set of features, the applied algorithms will aim at dividing the data set in clusters, where each cluster contains data points which are close enough in the problem space, while different clusters tend to maximize the distances between each pair of respective elements.

Coming back to the set of discriminating features, we tried and identified a number of behavioral characteristics which are general enough to allow for a clear identification of the social attitudes of the system users. Such features will enable us, during the analysis phase, to not only tell well-behaving users apart from potentially misbehaving ones, but also discriminate among different profiles of malicious behaviors.

We propose to analyze data flowing through the core of the network of an Italian telecom operator. The approach we propose is not invasive at all, since we use data collected by the operator's monitoring and management system, therefore no change or addition to the operator's core network is required. The collected data are then analyzed in a second step, without any interference with the service operation.

We develop and test a framework based on a model of user behaviors, and describe how it can be capable of discriminating between several user types.

By design, CAST has to preserve data confidentiality and be as transparent as possible with respect to the end users. This constraint highlights a feature of the system, concerning the specific nature of the information sources that can be analyzed in order to detect potential anomalies. A system like the one we envision would certainly guarantee good performance if it had as input raw traffic data. Unfortunately, this is not feasible when dealing with large Internet Service Providers (ISPs), both because of the huge quantities of traffic to be analyzed and because of data confidentiality, which is for them a major concern. Most such systems have to live with the non-optimal choice of analyzing pre-processed summarizing information stored inside the already mentioned Call Detail Records (CDR).

5.4.1 Clustering in CAST

We have adopted a well-known and widely-used unsupervised clustering technique to group users on the basis of the behavioral model we designed, i.e., the K-Means algorithm [56]. K-Means is one of the fastest approaches to partitive clustering. Given a specified data set, K-Means is capable of creating up to K different clusters, for which the following two properties hold: (i) intra-cluster (i.e., between each pair of nodes in a cluster) distances are minimized; (ii) distances between data points belonging to different clusters are maximized. Each cluster is characterized by its own centroid; the value of K, as well as the particular metric adopted, are fixed a priori. The algorithm proceeds as follows: K initial centroids are preliminarily randomly chosen inside the data set; iteratively, each point in the data set is assigned to the cluster holding the centroid which is at a minimum distance from the point in question. Once the modeling phase is complete, centroids are re-computed in order to better represent the "center of gravity" of the points that have been assigned to them, and then the algorithm goes back to the previous phase. The procedure continues until centroids locations stop changing, being guaranteed that this will happen in a finite number of steps if the Euclidean distance is adopted, as in our case.

Although K-Means is very appealing thanks to its simplicity and speed of execution, it provides a clustering solution which can be arbitrarily far from the optimal one. It has been demonstrated that, by introducing an appropriate selection of the initial centroids, the resulting clustering Êis O(log(k))-competitive with the optimal partition. Such an enhanced K-Means is named K-Means++ [11]. According to this algorithm, the selection of the K centroids is not random anymore, but is determined by the following rules. Only the first centroid is randomly selected among the dataset points. Then, the square distance between the first centroid and the other points is computed. The second centroid is elected according to a probability distribution that is determined by the square distance (i.e., the next centroid is chosen among the farthest points from the previous centroid). A new square distance distribution is calculated by considering the distances between each point and its nearest centroid among those that have already been elected. The next centroid can be selected according to that new probability distribution. This process continues until all the K centroids are established. After this initialization phase, the K-Means clustering algorithm can start without any further modification. In view of the above considerations, we have actually used K-Means++ in our system.

5.4.2 System architecture

CAST is made of different components, each with a specific function, as showed in Figure 5.1. First, a number of *Probes* are spread across crucial points in the operator's network, with the task of capturing user traffic and storing it in the form of call detail records. In our case, we were able to work on the CDRs logged by 32 such probes located inside the operator's VoIP infrastructure. Each probe produces, at the end of each observed call, a CDR file containing information about the identity of the active users, as well as about the data they exchanged, both on the signalling and on the data plane. CDR files are actually parsed by a dedicated module, whose task is to store therein the information contained inside a MySQL database, which is then used for all further elaborations of the captured data. The core of the architecture is represented by the *Preprocessor* module which is in charge of computing the features we selected for the description of the behavior of the system users. Given the approach we described previously, such features are computed by observing VoIP callers' behavior within a fixed-size time window that shifts forward in time. The output of the preprocessor is stored back in the MySQL database and eventually processed by the last component of the architecture, namely the *Analyzer*. This module, apart from some potential optimization such as a proper selection of the most appropriate subset of features needed to describe the problem space, is devoted to the application of the hierarchical clustering to the selected feature set. In a nutshell, the Analyzer studies the behavior of a certain user starting from

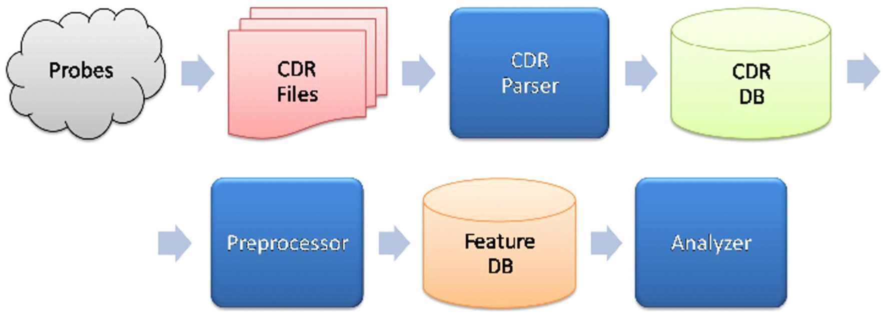


Figure 5.1: CAST system architecture

the observation of the behaviors of similar users in the past. Behavioral differences are computed through the clustering process which is at the basis of the classification algorithm. For each analyzed time frame, in fact, the clustering algorithm assigns a reference cluster to each user, representing her/his peculiar behavioral attitudes. From an in-depth analysis of the output of the Analyzer (and thanks to the semantic meaningfulness of the cluster centroids) it is also possible to arrive at a first, coarse-grained classification of the nature of the identified groups, by highlighting those which have been populated with potentially malicious users.

As to the process of selection of the time windows, each of them is confronted with an homologous one, that is to say a window corresponding to the same time of day, as well as the same day of the week. If we take, for example, the time interval from 10 to 11 am of a Friday, it will be compared to exactly the same interval of past Fridays and this is done in order to avoid that the analysis be affected by the differences that are intrinsic to either different periods of a 'typical' day or different days of a 'typical' week (or both).

5.5 CAST implementation

Given the general architecture described above, in the following section we will focus on the details related to the critical aspects we had to consider during the implementation of the system, namely feature selection and interpretation of the output of the classification process through the application of the theory of social graphs.

5.5.1 Features of the user profile

The main goal we had in mind when defining the set of users features to be computed by our preprocessors was to be able to discriminate between an acceptable (i.e, *normal*) behavior and the set of potentially malicious ones. Thus, the features we eventually selected are highly biased towards such a goal and often rely on the existence of threshold-based mechanisms, like, e.g., in the case of an activity threshold used to differentiate the least active users from the most active ones (being the degree of activity of a user one of the most intuitive yet coarse grained indicators of a potentially malicious behavior). Given the above assumptions, we list below the six features we have used for our analysis of the data set:

1. $$\begin{cases} \frac{\#InboundCalls}{\#OutboundCalls} & \text{, if } (\texttt{\#OutboundCalls>}\Delta_1) \text{ and } (\texttt{\#InboundCalls<\#OutboundCalls}); \\ 1 & \text{, otherwise.} \end{cases}$$

2. $$\begin{cases} \frac{\#OutboundCalls}{\#InboundCalls} & \text{, if } (\texttt{\#InboundCalls>}\Delta_2) \text{ and } (\texttt{\#OutboundCalls<\#InboundCalls}); \\ 1 & \text{, otherwise.} \end{cases}$$

3. $$\begin{cases} \frac{\#UniqueCallees}{\#OutboundCalls} & \text{, if } (\texttt{\#OutboundCalls>}\Delta_1) \text{ and } (\texttt{\#InboundCalls<\#OutboundCalls}); \\ 0 & \text{, otherwise.} \end{cases}$$

4. $$\begin{cases} \frac{\#UniqueCallers}{\#InboundCalls} & \text{, if } (\texttt{\#InboundCalls>}\Delta_2) \text{ and } (\texttt{\#OutboundCalls<\#InboundCalls}); \\ 0 & \text{, otherwise.} \end{cases}$$

5. $$\begin{cases} \frac{\arctan(\frac{TotalCallDuration}{\#TotalCalls}-\mu)+\frac{\pi}{2}}{\pi} & \text{, if } (\texttt{\#OutboundCalls>}\Delta_1) \text{ or } (\texttt{\#InboundCalls>}\Delta_2); \\ 1 & \text{, otherwise.} \end{cases}$$

6. $$\begin{cases} \frac{\#SuccessfulCalls}{\#TotalCalls} & \text{, if } (\texttt{\#OutboundCalls>}\Delta_1) \text{ or } (\texttt{\#InboundCalls>}\Delta_2); \\ 0 & \text{, otherwise.} \end{cases}$$

where Δ_1, Δ_2, and μ represent three thresholds that are actually used in order to make a sort of preliminary partition of the users set. It can be easily recognized, in fact, that all of the above formulae are used for those cases that can be defined as potentially anomalous: users who don't raise any concern with respect to their behavior will be all assigned fixed feature values after the feature computation phase. In this way, we allow for a first coarse-grained classification step aimed at clearly separating normal users from potentially malicious ones. This approach entails that the subsequent clustering phase will most expectedly assign all normal users to a single cluster, while populating all other clusters with malicious users having common behaviors. This will be much better highlighted in the following, when we discuss the results of our experimental campaign.

Coming back to the listed features, their respective meaning is briefly illustrated below:

1. With the first feature, users having a number of issued calls higher than the fixed threshold Δ_1, as well as much higher than the number of received calls (which is a clear indicator of a potentially malicious behavior related to a so-called *telemarketer*), will get a value close to 0. On the other hand, users with either a low number of issued calls or an approximately equal number of received and issued calls will be assigned a feature value close to 1.

2. The second feature is quite similar to the first, but with a focus on inbound activity. We remark that, while in the previous case the identified users have a high probability of being malicious, the kind of discrimination we obtain with this second feature is not necessarily associated with anomalies. Suffice to say that a 'user' with a high level of inbound activity might easily be a call-center, which is by no means classifiable as malicious. We keep such feature both for the sake of completeness and because it allows to detect potential victims of Denial of Service attacks.

3. The third feature allows us to identify potentially anomalous users who have issued a significant number of calls towards different parties. It

is in fact assumed that a user with a normal behavior typically calls a well-defined set of destination numbers.

4. Once again, the fourth feature mimics the behavior of the third one, with the difference that it looks at the inbound traffic (it in fact computes the ratio of unique callers to the total number of received calls). Thus, the same considerations as above apply to those users who are found to be over the threshold.

5. The fifth feature looks at the average duration of a user's calls along the analysis period. The introduction of the variable μ (associated with a duration level which is deemed acceptable) allows us to properly shift the normalized ratio in order to let it vary between 0 and 1. With such a feature it becomes possible to discriminate between users having very brief conversations (for which the feature will approximate 0) and those with average duration beyond the threshold (for which the feature in question approaches a value of 1). If the threshold is set sufficiently low (like, e.g., in our tests, where it had a fixed value of 30 sec), the former class of users will be considered potentially malicious: such a reduced duration is in fact quite unlikely in a 'normal' conversation between two human beings. Please notice that the condition related to the evaluation of the constraints is this time an inclusive `OR` rather than an `AND`, so to guarantee that the formula is applied to very active users, from both inbound and outbound perspectives. For those users who do not match the mentioned condition, the feature has a value of 1, which is consistent with a classification as 'well behaving' users.

6. The sixth and last feature analyzes the number of calls successfully issued. This feature has been added after realizing (through a preliminary inspection of the data set) that many of the users with a significant traffic, either inbound or outbound or both, happened to have a reduced number of successfully completed calls. Such a situation probably relates to one of the following two cases: (i) users who

have tried multiple times to reach the desired callee before succeeding, due to temporary unavailability of the network; (ii) users who are just making reachability tests, e.g., for network management purposes. The existence of these categories of users risked to distort the interpretation of the data set, since the described behaviors are not necessarily classifiable as anomalous. As usual, the feature has a value of 0 for all users with a number of outbound calls below the threshold.

5.5.2 Output interpretation

Classification results are stored back in the database, inside two distinct tables called, respectively, *classification* and *cluster*. The former table contains, for each active user, the cluster she/he belongs to both within the current time window and within the two preceding epochs, provided that she/he was present also in those time frames. This allows us to compare the evolution in time of the behavior profile of a specified user. The latter table stores the raw output of the clustering algorithm. With respect to the interpretation of such output, which is far from trivial given the huge quantity of data at hand, we have investigated the possibility of applying the theory of social graphs [14, 23] to the input data set, in order to have an easy and intuitive way of comparing the results of the analysis with the actual behavioral context logged in the CDRs. As we are going to show in the next section, an informal yet reliable way of assessing the effectiveness of the proposed automated approach to classification is definitely represented by this comparative analysis between the graphical representation of the social relations embedded in the input data set and the results of the hierarchical clustering procedure. As an example, a relational graph associated with the interactions between caller and callee within a specified time window can be used to catch at a glance the potential anomalous behaviors of some of the nodes and then verify whether the clusters to which such nodes have been assigned after the classification phase are actually associated with anomalous behaviors.

For our work, we have made a deep use of a graphical analysis of the input data set, by relying both on social graphs like the ones mentioned above and on more classical yet useful representations, as described below: (i) chart of the time trend of active users, with the aim of identifying potential patterns in the generated data; (ii) chart of the partitioning of the instances produced by the clustering algorithm, capable of providing a first visual representation of the classification made by the clustering algorithm; (iii) chart of the time distribution of the calls issued by a suspect user, with a clear intent of highlighting potential suspicious behaviors like, e.g., the presence of a telemarketer issuing a high volume of calls in a restricted and unusual time frame; (iv) chart of the distribution of the average call duration for a suspect user, which also provides a useful means for isolating anomalous behavioral patterns.

5.5.3 Tuning of the relevant parameters

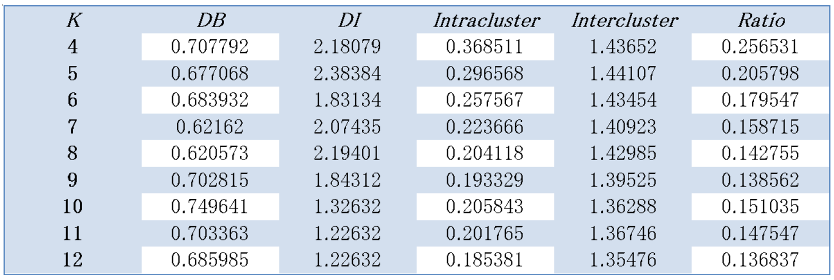
Whenever a learning-based approach is embraced, several parameters come into play during the tuning phase.

K-Means needs the clusters number K as an input to the algorithm. Such a value corresponds to the number of behaviors that we can detect by analyzing the dataset of VoIP users' features. In order to discover the best value of K for our purposes, we have made some experiments and considerations. We pick the most populated analysis window, i.e., the one hour long window with the highest number of active VoIP users. The selected window is the one from 3:00 p.m. to 4:00 p.m. of Tuesday, 22 April 2008, and presents about 240.000 active VoIP users. We chose the most populated window since we expect it to provide us with the widest spectrum of detectable behaviors. We then evaluate clustering results by varying the parameter K. We remark that we cannot perform any clustering evaluation in terms of classification error rate with reference to a real-world dataset since we have no a-priori knowledge about it. Nonetheless, we can evaluate the effectiveness of the clustering results by means of internal criteria. Indeed, data are well grouped if instances belonging to the same cluster are very similar and instances belonging to different clusters are very dissimilar according to a certain similarity (or distance) measure. We can then look at such properties of the resulting clusters by varying K and decide accordingly which value of K is the most suitable one for our purposes.

We first evaluate the intra-cluster distance as the mean value of the Euclidean distances between each point and the centroid of the cluster it belongs to. This value indicates the cohesion of the cluster, as well as the centroid's property of properly representing the cluster members. The lower the intra-cluster distance, the higher the cluster compactness. Then we calculate the inter-cluster distance as the mean value of the Euclidean distances between each pair of centroids. This quantity expresses the diversity degree among centroids, as well as the degree of separation among clusters. The higher the inter-cluster distance, the more each centroid has a distinctive behavior.

Both intra-cluster distance and inter-cluster distance are shown in the table in Figure 5.2. Such distances decrease as long as the parameter K increases. However, as shown in the last column of the same table, the ratio between intra-cluster and inter-cluster distances decreases as well, meaning that the intra-cluster distance decreases at a faster pace than the inter-centroid one.

We also consider two further indexes expressing the clustering effectiveness: the Davies-Bouldin index (DB) [35] and the Dunn index (DI) [20]. Such indexes represent different aspects of the clustering appropriateness. The DB index envisions that an internal cluster dispersion be calculated for each cluster (S_i, i = 1,...,K). The dispersion is computed as the root mean square of the Euclidean distances between a point of the cluster (a six-dimensional vector of features representing a user) and the centroid of the cluster itself. The higher the dispersion, the less the cluster cohesion. Then, the inter-centroids Euclidean distances are computed ($D_{i,j}$). For each cluster, the nearest cluster is elected according to the minimum inter-centroid distance, $D_{i,min}$. Then, for each cluster it is evaluated the ratio between (i) the sum of its dispersion

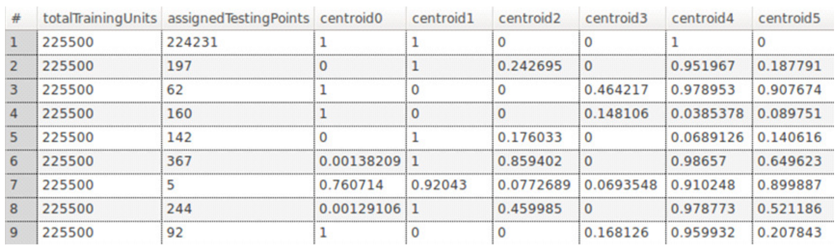


K	*DB*	*DI*	*Intracluster*	*Intercluster*	*Ratio*
4	0.707792	2.18079	0.368511	1.43652	0.256531
5	0.677068	2.38384	0.296568	1.44107	0.205798
6	0.683932	1.83134	0.257567	1.43454	0.179547
7	0.62162	2.07435	0.223666	1.40923	0.158715
8	0.620573	2.19401	0.204118	1.42985	0.142755
9	0.702815	1.84312	0.193329	1.39525	0.138562
10	0.749641	1.32632	0.205843	1.36288	0.151035
11	0.703363	1.22632	0.201765	1.36746	0.147547
12	0.685985	1.22632	0.185381	1.35476	0.136837

Figure 5.2: Clustering performance indexes, $\Delta_1 = \Delta_2 = 10$, $\mu = 20$

and of the dispersion of the nearest cluster and (ii) their inter-centroid distance. The bigger this quantity, the lower the cluster separation. The DB index is calculated as the mean of these values, then it expresses a sort of mean degree of similarity between clusters. The lower the DB index, the better the clustering. For the DI index evaluation, we first select the maximum dispersion among the S_i values, S_{max}. The DI index is calculated as the minimum values among the ratios between $D_{i,min}$ and S_{max}, then it expresses a sort of minimum degree of separation. The higher the DI index, the better the clustering. Clustering performance, as measured with the above indexes, is comparatively showed in the table in Fig. 5.2.

The last three columns in the table seem to indicate that the higher the K, the better the clustering results. However, by inspecting the resulting clusters composition it is possible to notice that starting with K greater than 8 new clusters tend to be formed by just a few members (less than 5) and singleton clusters begin to arise. This means that the centroids we extract from our analysis are not representative of a family of behaviors, but rather coincide with outlier users having peculiar behavioral characteristics. This situation alters the clustering performance indexes we consider, since, when clusters are made by only one member, the intra-cluster distance is zero. As a further (and even more important) consideration, we observe that, for K $>$ 8, some of the centroids are meaningless according to the behavioral model we propose. As an example, Fig. 5.3 shows that such centroids present the first and the second feature both varying within the interval $[0, 1[$. It can be easily verified, by looking at the features description in section 5.5.1, that



#	totalTrainingUnits	assignedTestingPoints	centroid0	centroid1	centroid2	centroid3	centroid4	centroid5
1	225500	224231	1	1	0	0	1	0
2	225500	197	0	1	0.242695	0	0.951967	0.187791
3	225500	62	1	0	0	0.464217	0.978953	0.907674
4	225500	160	1	0	0	0.148106	0.0385378	0.089751
5	225500	142	0	1	0.176033	0	0.0689126	0.140616
6	225500	367	0.00138209	1	0.859402	0	0.98657	0.649623
7	225500	5	0.760714	0.92043	0.0772689	0.0693548	0.910248	0.899887
8	225500	244	0.00129106	1	0.459985	0	0.978773	0.521186
9	225500	92	1	0	0	0.168126	0.959932	0.207843

Figure 5.3: Clustering results with $K = 9$

such situation would mean that the users represented by the centroids make, at the same time, a number of outgoing calls greater than the number of incoming calls, and a number of incoming calls greater than the number of outgoing calls, which is clearly a nonsense.

The above considerations explain why we introduced a further termination condition for the K-means algorithm, which is actually associated with the mentioned ‘meaningfulness constraint’. In the following, we will hence focus on the results obtained for $K = 8$.

For what concerns the thresholds Δ_1 and Δ_2, associated, respectively, with outbound and inbound calls, they should vary in time in order to best suit the typical pattern of phone traffic during the course of the day, with peaks in the morning and in the afternoon, as well as significant drops during night hours. For our exploratory analysis, we have nonetheless used constant threshold values. The thresholds have been leveraged as an ‘elastic’ means to properly tune the detection granularity level: higher thresholds help spot out those behaviors that are undoubtedly anomalous, whereas lower thresholds allow us to enlarge the set of users under the magnifying glass and reduce the set of users that are not deemed anomalous. We conducted experiments with call threshold values varying form 10 up to 50, with a step of 10; though, for the sake of brevity, we herein report just the results we obtained with both thresholds equal to 10.

Similarly, the third threshold μ, associated with call duration, has also been set to a constant value of 20 seconds.

5.5.4 Time performance

In order to provide the reader with an evaluation of the time required to analyze call data records with our approach we measure the computational time needed to analyze a whole week of traffic by using (i) a window interval of one hour, (ii) a forwarding step of one hour, and (iii) $K = 8$. The analysis is performed on a desktop PC with the following characteristics: CPU Intel®Core^{TM}i5 CPU M 480 @ 2.67GHz x 4, with $7,6$ GiB SODIMM DDR3 Synchronous 1067 MHz RAM and a hard disk Toshiba 2.5” 5400RPM 8MB SATA, equipped with an Ubuntu 12.04 LTS 64bit Operating System.

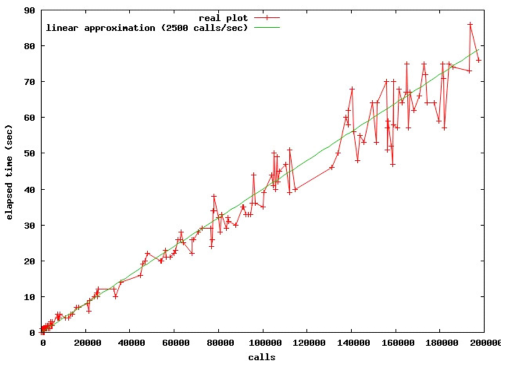
The overall analysis of about 12 million calls and about 15 million users took about 4 hours (1 hour for the preprocessing phase, 3 hours for the analysis). The most populated window (193.000 calls, 238.598 users) takes about 4 minutes to be preprocessed and analyzed. As witnessed by the linear approximation curve in Fig. 5.5, the overall processing speed (involving both the preprocessor and the analyzer) is of about 850 calls per second (corresponding to about 1.100 users per second).

5.6 CAST in action: user profiling based on social habits

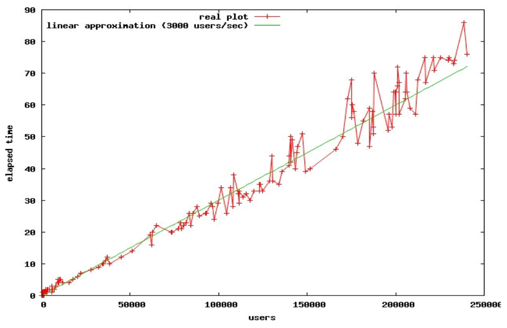
In this section we will finally see CAST in the field, by showing the results of the analysis of a very large data set. More precisely, as announced in the previous section, after a description of the main patterns embedded in the input data, we will first illustrate the clustering results and then assess their validity through a comparison with the most representative charts produced by the graphical analysis of the data set.

5.6.1 Key features of the data set

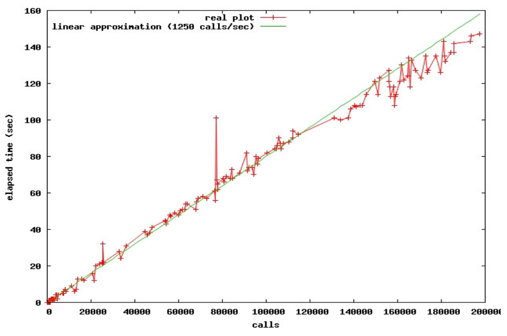
The data set we used for system testing has been extracted from an Italian operator’s VoIP infrastructure and refers to the period from April to May 2008.



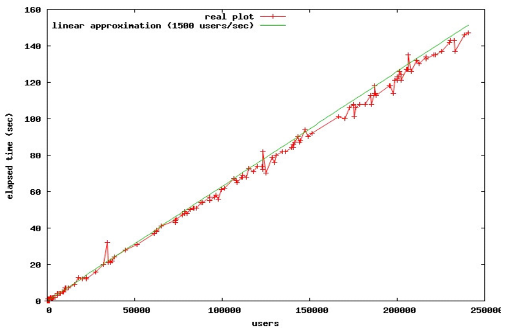
(a) Preprocessor: elapsed time vs calls



(b) Preprocessor: elapsed time vs users



(c) Analyzer: elapsed time vs calls

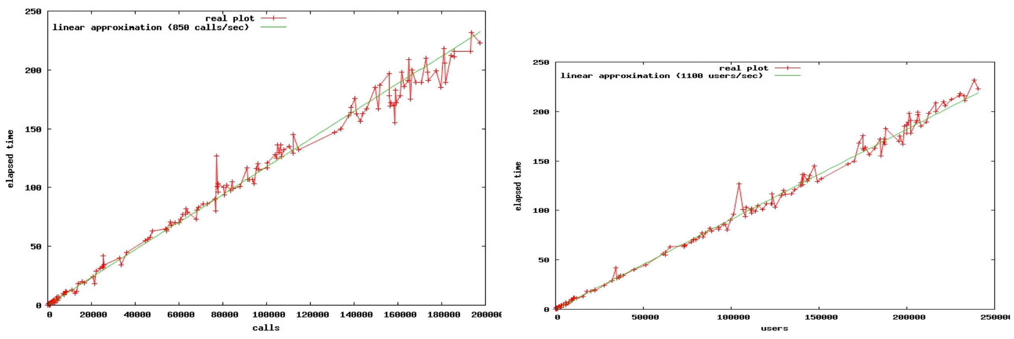


(d) Analyzer: elapsed time vs users

Figure 5.4: Time performance: CAST components

This very large data set comprises more than 20 million users who have issued a total number of more than 100 million calls. A preliminary analysis of the data at hand indicates that regular users (i.e., users whose frequency of appearance in the logs is above a predefined low-activity threshold) of the monitored service are less than 1 million. This aspect was taken into account when we chose the thresholds aimed at isolating the most active users inside the network. With respect to these users, another interesting feature is definitely represented by their usage behavior over time, as depicted in Figure 5.6.

The picture shows the typical trend of human-generated phone traffic. During weekdays it in fact has two peaks which correspond to the busy hours, in the morning and in the afternoon, and a central saddle at around 12:30. Users' activity is lower during week-ends. In any case, it drops drastically



(a) CAST system: elapsed time vs calls (b) CAST system: elapsed time vs users

Figure 5.5: Time performance: overall CAST system

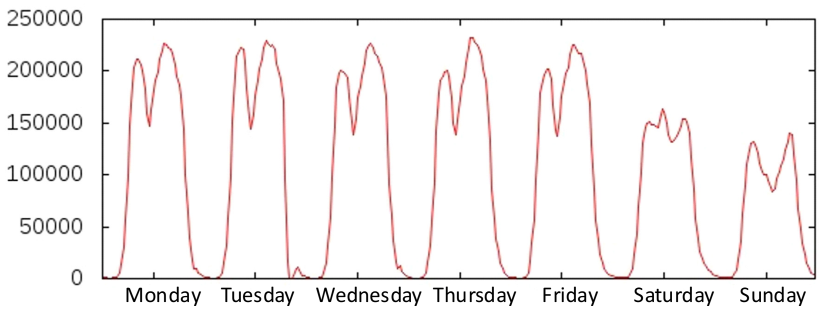


Figure 5.6: Active users in the first analysis week

during night hours, when it reaches almost zero. The same drop happens on holidays, as demonstrated, for example, by the chart in Figure 5.7, which contains Friday April 25^{th} 2008, the *Liberation Day* in Italy.

5.6.2 Clustering results

Based on the above considerations, in order to perform a reliable analysis of the results produced by the clustering algorithm, it is advisable to focus on a peak-time analysis slot, so as to work with a training dataset that is as large and as heterogeneous (with respect to the embedded behavior profiles) as possible. For the sake of brevity, but without losing in generality, we will herein focus on a significant time windows (225.500 users).

As a first example, let us have a look at Figure 5.8.

Here we report an excerpt from the clustering results table. The at-

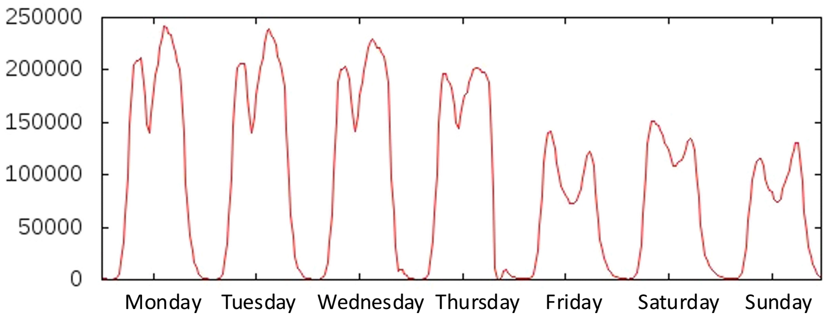
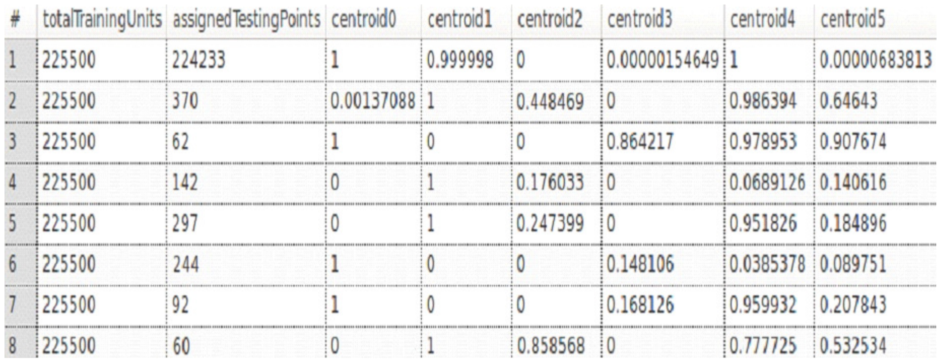


Figure 5.7: Active users in the week from April 21st to April 27th 2008



#	totalTrainingUnits	assignedTestingPoints	centroid0	centroid1	centroid2	centroid3	centroid4	centroid5
1	225500	224233	1	0.999998	0	0.00000154649	1	0.0000683813
2	225500	370	0.00137088	1	0.448469	0	0.986394	0.64643
3	225500	62	1	0	0	0.864217	0.978953	0.907674
4	225500	142	0	1	0.176033	0	0.0689126	0.140616
5	225500	297	0	1	0.247399	0	0.951826	0.184896
6	225500	244	1	0	0	0.148106	0.0385378	0.089751
7	225500	92	1	0	0	0.168126	0.959932	0.207843
8	225500	60	0	1	0.858568	0	0.777725	0.532534

Figure 5.8: Example of clustering results with $K = 8$

tributes of the table correspond to, respectively, the timestamp related to the window under analysis, the clusters identifiers, the number of users ascribed to each cluster, the total number of users in the window, and the centroid components, i.e., the values of the six features for each cluster centroid. The observation window in this case corresponds to a Tuesday at 2:00 p.m.: the reported picture indicates the most significant data stored by the CAST system inside the *cluster* table of the database. The column called 'assignedTestingPoints' contains the number of instances populating the various clusters. A first observation concerns the fact that cluster number 1 accumulated more than 99% of the users that have been analyzed during the considered time interval. By looking at the feature values in the first row of the clusters table, it becomes apparent that the mentioned users are those for whom the chosen feature threshold values have not been trespassed and who have thus been classified as 'normal'. We will then focus on the users who have on the contrary been assigned to the other clusters. We recall that such clusters account for 1% of the total classified samples, which should intuitively represent the entire set of potentially malicious users.

In order to assess the correctness of such assumption, let us make an in-depth study of the specific features of the involved clusters, with the help of the values of the cluster centroids (also reported in the results table), which can be assumed to be representative of the average behaviors of the identified user groups:

- Cluster 2: users who have mainly issued calls (greater in number than the threshold) towards a high number of callees. Most of the calls have been both successful and long lasting;
- Cluster 3: users who have only received calls from a high number of parties. Most of the calls have been both successful and long-lasting;
- Cluster 4: users who have only issued calls (greater in number than the threshold) towards a very low number of parties. Just a few such calls have been successful, with a low average duration as well;

- Cluster 5: users who have only issued calls (greater in number than the threshold), towards a low number of parties. Very few calls have been successful, but the average duration of the successful calls is quite above the predefined threshold.
- Cluster 6: users who have only received calls (greater in number than the threshold) from a very low number of parties. The number of successful calls is also very low, as well as their average duration;
- Cluster 7: users who have only received calls (greater in number than the threshold) from a restricted number of parties. Just a few calls have been successfully completed, but the successful ones show quite a long duration;
- Cluster 8: users who have only issued calls (greater in number than the threshold), towards a high number of parties. About 50% of the calls have been successful, with an average duration.

Some of the patterns described above clearly represent anomalous behaviors; others, instead, though indicating an intense activity, can be ascribed to particular, innocuous users. This is the case, for example, of the nodes with a significant number of inbound calls and with an almost constant presence in the network, which can most likely be associated with call centers (Cluster 3). A similar observation can be drawn with respect to the presence of automated testing systems, easily recognizable by the significant volume of unsuccessful calls towards a unique destination number; for such nodes, by making a cross-analysis involving the other information stored in the database, it is also possible to assess that the time elapsing between any pair of subsequent calls is so short as to allow to skip the hypothesis that the calls themselves were issued by a human operator (Cluster 4) .

5.6.3 Noteworthy behaviors

We will herein make an in-depth analysis of the various behavior profiles coming out of the classification process, by showing that they can be ascribed to the following macro-categories: (i) users with a significant number of inbound calls, most probably identifiable as call centers (such users are classified as belonging to Cluster 3); (ii) users with a high number of unsuccessful calls towards a single callee; such a number is so high that it hints at the utilization of an automatic tool in charge of performing some connection tests towards the callee, so we refer to these users as to “testers” (such users are classified as belonging to Cluster 4); (iii) spitter users, with a high volume of average duration calls towards different callees and along the entire observation interval (such users are classified as belonging to Cluster 8); (iv) intermittent spitter users, that is to say users behaving like spitters, but just during sporadic intervals of time within the entire observation window (as in the previous case, such users are classified as belonging to Cluster 8)[2]; (v) normal users, i.e., those users who are active during a significant part of the observation window, but with an innocuous behavior.

We start by looking at the total number of calls (either issued or received) per day, along the first month of observation, as reported in Figure 5.9. For what concerns the first user (call center), we can easily acknowledge the presence of a very high average number of received calls (up to 4.000), with significant drops on Sundays and public holidays. A different situation holds for the tester user, for which there is almost no activity, except for those days in which we have revealed an anomalous behavior, most probably due to the conduction of the tests (with a very high number of issued calls). Coming to the spitter, such user shows a behavior characterized by a high volume of issued calls (up to 500), with drops on Sundays and public holidays. The intermittent spitter has a similar profile, but just during some days along the entire observation period. In both cases, we might easily think of the presence of so-called telemarketers making an intensive use of a corporate phone network in order to contact potential clients.

[2]The difference between a spitter and an intermittent spitter actually resides in the fact that the former acts like a spitter both in the clustering observation window and in contiguous timeframes, whereas the latter does not always show a malicious behavior in timeframes other than the illustrated clustering window.

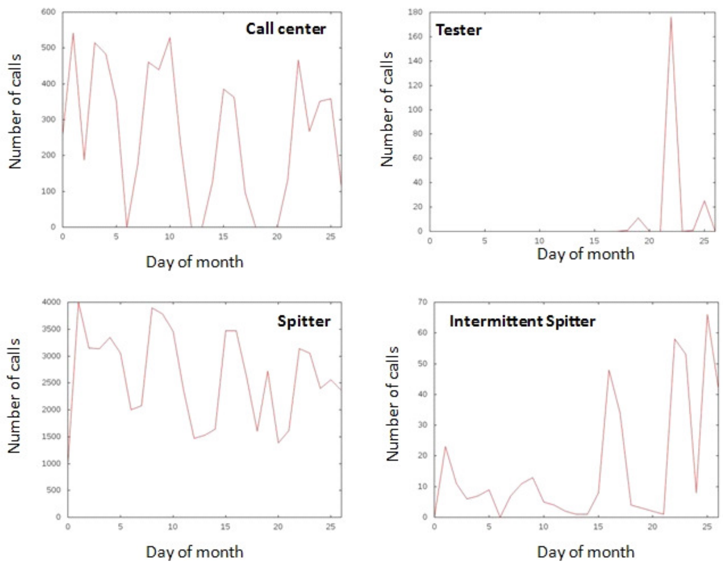


Figure 5.9: Calls per day during the first observation month

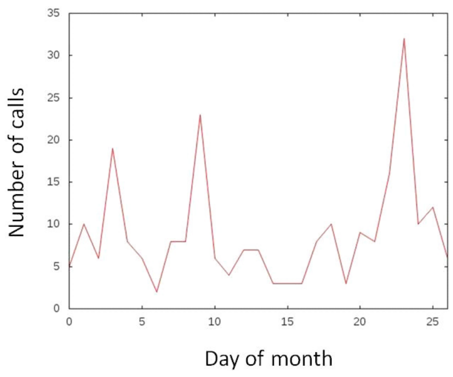


Figure 5.10: Calls per day during the first observation month, for a 'normal' user.

Finally, for the sake of completeness, we report in Figure 5.10 the typical profile of a user who was classified as 'normal' by the CAST system, during the observation period in question.

A different analysis can be performed by building charts associated with the distribution of calls during a single day of observation, as shown in Figure 5.11. With such charts it is possible to spot the presence of anomalies related to those users who do not adhere to the expected pattern, which is characterized by peaks in the middle of both morning and afternoon hours. From now on, we will no longer distinguish between the constant and the in-

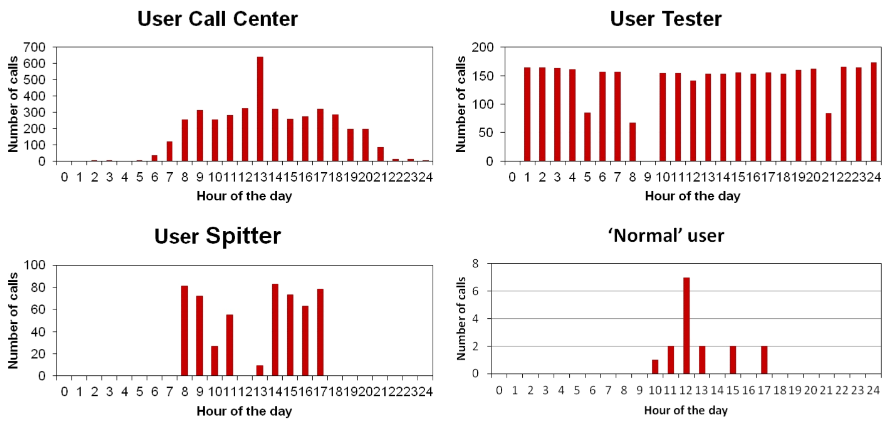


Figure 5.11: Calls distribution per time slots

termittent spitter, since the same considerations hold for both of them. If we look at the call center, we can observe a significant number of calls at all times during the day, with peaks corresponding to busy hours. As to the tester, it clearly shows a widely anomalous behavior, with an almost constant (and high) number of calls at all times (both in the daytime and at night). Coming to the spitter, the chart shows that her/his activity is mainly concentrated during work hours. If we recall from the previous charts that such activity was mainly conducted during workdays, it is legitimate to think that the intense use of the corporate phone network is ascribable to the presence of a telemarketer in action. The last user is the one that has been classified as 'normal' by CAST. The distribution of calls per day actually confirms this verdict, since the issued calls are quite low in number and mostly concentrate around busy hours, both in the morning and in the afternoon.

A further graphical tool we used for the analysis is represented by the distribution of the average call duration: the histograms in Figure 5.12 group calls (either received or issued) in six different clusters, ranging from those whose duration has been less than 30 seconds, up to the ones that have lasted more than 5 minutes. The call center shows a various set of durations (for the received calls). Calls with a duration of more than 5 minutes in fact prevail, but there are also several calls which lasted much less: such behavior is indeed fully compatible with the hypothesis made by the classifier. The same holds,

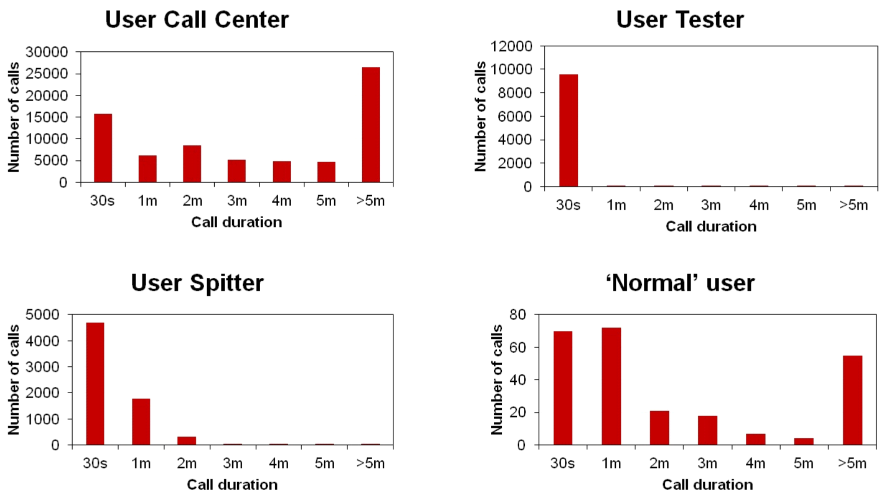


Figure 5.12: Distribution of calls duration

once more, for the tester user, for which almost all of the (unsuccessful) calls have terminated in less than 30 seconds. As to the spitter, as one would expect (since people tend to refuse undesired calls from telemarketers), most of the issued calls, each towards a different callee, have a very short duration. Finally, the normal user has issued both short and long duration calls, with a distribution which quite closely follows the typical usage patterns of the phone network.

5.7 An in-depth analysis based on relational graphs

As we mentioned earlier, a very useful analysis tool is definitely represented by relational graphs highlighting the relationships among callers and callees within a predefined observation window. A relational graph is a graph where each node represents a user and each edge represents a relation (a call, in this context) between users. As an example, such graphs can help identify at a glance the presence of so-called hub nodes, i.e., those users attracting the highest number of calls, both inbound and outbound. Typical behavioral templates that graphically represent the four user categories we have

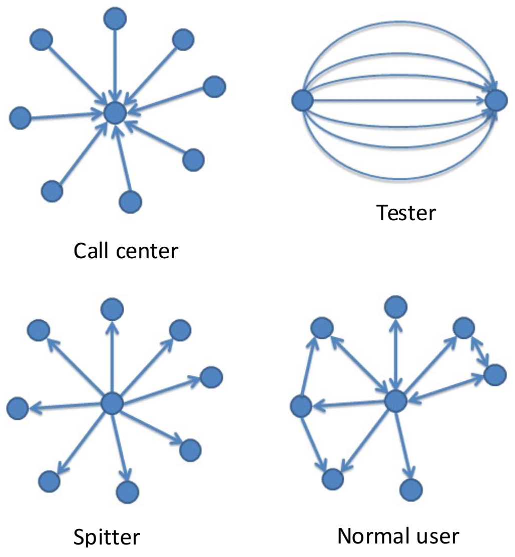


Figure 5.13: Relational graphs – behavioral templates

discussed so far are reported in Figure 5.13, whose interpretation is quite intuitive. We will now show how such patterns can actually be found in the data set we have studied. Figure 5.14 reports the relational graph associated with the users that were involved in VoIP calls during the analysis window we accounted for in the previous section and which covers the peak hour of a typical workday.

In the picture, it is possible to identify the mentioned hubs, which can be referred to as either call centers, if the edges are all incoming, or contact centers (i.e., telemarketers) in the opposite case. Pairs of nodes interconnected by a very high number of edges (outgoing from one node of the pair and incoming into the other) are clearly linked to the so-called testers we described before. It is also possible to observe that most of the highly active users (concentrated around the center of the figure) keep up relationships that can be defined "island-based", since they usually involve one user with a high degree of activity (either inbound or outbound) and many 'satellite' parties. Such phenomenon is peculiar to the theory of social graphs: it is not by chance, in fact, that one of the most interesting features that can be extracted from relational graphs is the so-called *clustering coefficient* allowing

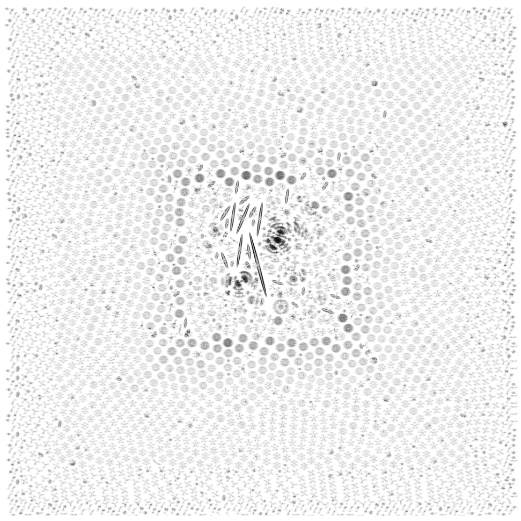


Figure 5.14: Relational graph – peak hour of a workday

to evaluate the relationships existing between pairs of nodes that are in turn linked to a third node.

The graph we show in Figure 5.15 refers to the peak hour of a holiday time window. It can be easily noticed that the hub nodes associated with call centers look less active than in the previous picture; tester nodes can also be clearly identified.

Before concluding and with the aim of further validating the results of the previous analysis, we show in Figure 5.16 a zoom of Figure 5.14 devoted to the graphical analysis of spitter nodes. All of the focus nodes of the reported clouds have just outgoing edges (higher in number than the defined threshold); furthermore, the nodes they connect to have non mutual connection, which contradicts the mentioned theory of social relations.

A further zoom-in detail is reported in Figure 5.17, where the typical shape of both testers (i.e., nodes from which a high number of edges towards the same destination departs) and a call center (very dense edge clusters, departing from different callers and all directed towards the same destination node) appear.

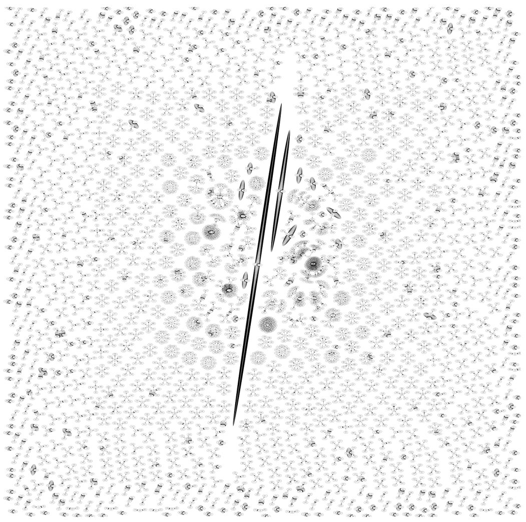


Figure 5.15: Relational graph – peak hour of a public holiday

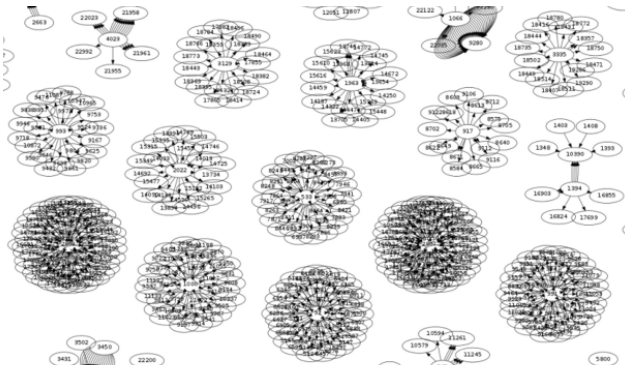


Figure 5.16: Relational graph – a zoom on spitter nodes

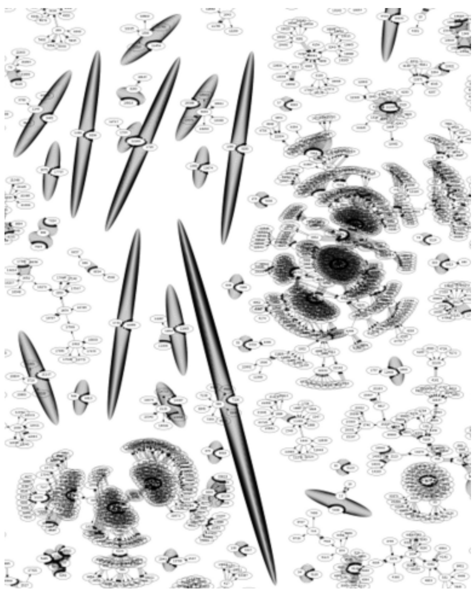


Figure 5.17: Relational graph – a zoom on tester and call center nodes

Chapter 6

Content-aware applications and infrastructure part I: a novel architecture for behavior-aware content management

6.1 Introduction

The multimedia revolution first, and the mobile revolution then, have totally upset the Web landscape.

The real-time entertainment traffic dominates in North America, overtaking the peer-to-peer one. In Europe, it keeps growing steadily, since users are starting to appreciate on-demand entertainment applications [75]. Web contents vary among static, dynamic, live and registered multimedia data (i.e., audio, video, images, besides documents and web pages), and they are more and more accessed by mobile consumers. Different studies [32, 4] published amazing numbers about mobile data traffic forecasts. The mobile-only Internet population will grow 56-fold from 14 million at the end of 2010 to 788 million by the end of 2015 and it will ask for more and more multimedia contents: two-thirds of the world's mobile data traffic will be video by 2015 [32]. By then, there will be nearly one mobile device per capita and this huge amount of devices will generate as much traffic as the entire global mobile network in 2010.

With such an explosion of mobile access to multimedia web contents, there is an unmet demand for a higher quality and speed in multimedia content and service delivery to the connected devices.

Further networking issues arise in this context.

As a first observation, contemporary trends show a drastic change in how the network is used for. Internet was designed to address the connection of remote endpoints thorugh host-to-host channels, in a host-centric scenario. Nowadays it has mainly become a bridge used by users to access contents and services, totally disregarding their physical location. Its role is not only distributing contents to users, but also allowing them to publish their own-made contents. Indeed, Internet users are nowadays called "prosumers" since they both produce and consume contents on the network. Internet has become a dynamic space of contents, populated by users, and explorated by them. In such a perspective, the network has the social functionality of interconnecting groups of people on the basis of the shared interests and social relationships. The use of the network is then switched from host-centric to content-centric.

Given the above considerations, Future Internet research projects are starting to re-consider and design the Internet as a Content-Centric Network. There are different kind of proposals of Content-Centric Network architectures. A first kind of solution envisions an overlay infrastructure over the existing host-centric platform. A more disruptive type of proposal is aimed to integrate in the core network content-centric facilities so that they become natively supported [52, 49]. Both of them realize the paradigm of the Content Centric Networking, which envisions publish/subscribe mechanisms where users directly access or push contents on the network, independently from contents' physical location.

The mobile revolution also poses new challenges to content centric networks design. The increased temporal and spatial variability of mobile users' contents demand should be taken into account both in the design phase and in the operational phase of content delivery infrastructures.

In this chapter, we first investigate how to exploit social behaviors and community analysis in content distribution networks to optimize content delivery. The research work herein presented is named BACON (Behavior Aware Content Oriented Networking) and has been financed by Telecom Italia within the context of a national initiative aimed to support innovation (Working Capital). We propose a behavior-aware content-oriented framework exploiting users' preferences for contents expressed by ratings.

Users' preferences are not the only aspect to be taken into account when exploiting the knowledge of users' behavior to enhance the content delivery. Mobility and request patterns should also be considered to this aim. We propose in Section 6.5 an overview of the state of the art achievements in these fields.

6.2 Content Delivery Networks: State of the Art and Architecture design

Content Delivery Networks are a key technological overlay solution that have supported up to now the Internet as a content-centric network. CDNs are mainly targeted to Web applications, and provide intelligent replication services for Web sites. Geographic replication of Web sites is indeed fundamental for Web content providers in order to make their contents available across the whole globe. By outsourcing replication to CDNs, they have not to provide their own replication infrastructure. A CDN is typically owned, deployed and maintained by an individual operator, that charges content providers for its services. CDNs are totally transparent to users. They deploy redirection services by way of which a user requesting a page is automatically (and transparently) directed to the replica site that can optimise content access. To this end, CDNs also deploy refined (and effective indeed) monitoring systems to understand where to dynamically redirect each user request taking into consideration the current status of the network and of the replica sites.

In [66] the main building blocks of CDNs are illustrated (see Fig. 6.1).

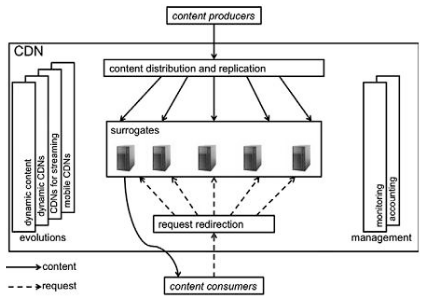


Figure 6.1: CDN building blocks

Fundamental building blocks are those devoted to distribution and replication of contents and the one devoted to request redirection. The distribution and replication module is the interface between the CDN operator and the content provider. It deals with three main issues:

- server placement, i.e., strategies for the positioning of replica servers;
- content selection, i.e., strategies for chosing the contents that have to be replicated on surrogates;
- content outsourcing, i.e., algorithms for the replication of contents on replica servers.

The redirection block on the other hand is the interface between the CDN and the final user. It handles users' request and their redirection towards the CDN's surrogates containing the requested contents.

Other conceptual blocks are related to monitoring and accounting functionality. The monitoring is responsible of controlling CDN resources' state (servers' workload, network workload, and so on). The accounting module controls how contents belonging to each CP client overload the CDN network by looking at parameters such as the traffic generated by a given replicated web site.

Since their birth in the 90s, CDNs have been facing several challenges due to the evolution of the usage patterns of the Internet and new architectural modules have been added besides the aforementioned basic ones. They deal with:

- dynamic content distribution, i.e., the replication of contents that have to be generated ad-hoc according to the received request;
- dynamic resource deployment, in order to accomplish overload conditions and flash crowd;
- users' mobility support
- multimedia streaming support

6.2.1 Content selection

When deciding the CDN replication policy for a certain CP's site, there are two main alternatives:

- performing a full-site replication: in this case, the whole site is replicated on CDN surrogates;
- performing a partial-site replication: here, only a subset of objects is replicated.

Partial-site strategies are preferred since more scalable when dealing with lots of bulky objects. However, a further complexity resides behind. Indeed, a proper content selection criterion has to be identified and applied. Typically, web objects are clustered according to a certain similarity metrics that can be based for example on object's popularity, number of incoming hyperlinks, or the reduction in the access latency that could be obtained by replicating the considered object.

6.2.2 Content outsourcing

Content outsourcing strategies can be "push-based" or "pull-based".

Push-based mechanisms envision content pre-fetching on CDN surrogates, i.e., the replication is performed in a pro-active manner without waiting for final users' requests. Strategies of such a kind are usually strictly interconnected with partial-site replication policies of content selection. Push-based outsourcing is not exploited in commercial CDNs since it can be effectively applied only if the spatial and temporal distribution of users' request can be known a-priori. That hypotesis is very interesting but at the moment it is also very hard that it can be satisfied in practice.

Pull-based strategies, on the other hand, replicate contents on a surrogate server only after a "miss event". A miss event happens when a request reaches a server that does not hold the requested content. They can be further divided in "cooperative" and "non-cooperative" solutions.

Non-cooperative pull-based strategies correspond to conventional chaching. Only contents already requested by users are replicated on surrogate servers. This policy realizes a user-driven automatic content replication on the CDN. This solution is the simplest one and is often employed in commercial CDNs as Akamai.

Cooperative pull-based solutions are different only in the management of the miss event: the missing content is requested to the other cooperating CDN servers rather than to the origin server, by this way reducing the load on the origin server and possibily enhance time performances.

6.2.3 Server selection

A server selection algorithm is used for selecting the surrogate server to which redirect a certain request among those that can satisfy that request. Server selection algorithms can be divided in (i) server-side and client-side and in (ii) single-server and multiple-server .

There are two algorithm families of server-side algorithm:

- adaptive: they do not take into account server load, but apply static policies. A typical example is a round-robin strategy.

- non-adaptive: they consider some information in the selection process, such as server load, the user proximity, and the like. Akamai adopts algorithm of such a kind.

Client-side algorithms envision the user selects the preferred replicas among the available ones. The CDN operator provides a list of information associated with the available replicas and the user can use this information in the selection process.

In single-server strategies, the content is taken from a single selected server, while in multiple-server the content is divided into multiple blocks and each block resides on a different server. Single-server strategies are preferred in case of non-bulky contents; in this last case, which is typical for multimedia contents, the object can be splitted among different replicas.

6.2.4 Dynamic content management

A web application generating dynamic contents is developed into different levels:

- front-end: the interface with the final user, i.e., the web page dynamically generated;
- application: the level responsible of generating the web page on the basis of the application logic;
- back-end: the database level containing data exploited by the application logic;
- user-profile: the database level containing user profiles enabling the contents personalization.

The replication on CDNs of sites of that kind can be performed at different levels. Replication at the first level corresponds to static contents replications, which is the one considered up to now. The replication of the application level can be useless if the bottleneck is the access to the database mantained by the origin server. Database replication can be performed in three ways: (i) context-blind caching, (ii) context-aware caching, (iii) full-replication. The full replication is quite onerous if it has to be performed on several surrogates dispersed across the network. Content-blind caching memorizes on replicas only the content resulting from previously issued requests and it is the typical commercial option. In the context-aware caching strategies, surrogates have a database which is popolated in a pro-active manner through a partial replication of the original database. The pro-active replication is driven by the surrogate users' access patterns. Even if more flexible, such an approach calls for a centralized control for the distribution of the requests among the multiple copies of the database and for the coordinated query execution. This strategy is typically employed for user profiles replication, since they are accessed from a single surrogate at a time in a localized manner.

6.2.5 Handling multimedia contents

CDNs evolve in time to better support multimedia streaming applications. Multimedia contents of bulky dimensions are typically divided into sub-parts, both in time domain and quality domains, when progessive encoding algorithms are applied. The full replication of such file on surrogate servers can be unefficient, since it generates a lot of traffic on the network and since users can be interested only in limited portions of the content. Typically, a CDN solution for the distribution of multimedia contents is a two level architecture. At the higher level, there is the transport of the content between the origin server and the surrogate server by means of typical file transfer protocols. At the lower level, there is the transport of the content between the surrogate server and the final user via RTP, controlled in case through protocols such as RTSP (Real Time Streaming Protocol) that enable the final user to perform typical VCR commands. The preferred content outsourcing strategy is a non-cooperative pull-based approach where surrogates possibly pre-fetch part of the contents that will be requested by the user to the aim of avoiding undesired interruptions during the reproduction. Live streaming (also known as webcasting) is more challenging on the architectural design plane. Live contents can not be pre-fetched on surrogates and the content distribution must be performed in real-time. When CDNs are largely expanded across the globe, webcasting of popular contents can really stress the content distribution network and the origin server.

6.3 A case study: behavior-aware delivery of User Generated Contents (UGC)

In this section we describe an original approach designed to optimize UGC content distribution in CDN according to users' attitudes.

We design a content outsourcing algorithm exploiting users' preferences to make replica servers able to keep in memory contents that their users like.

We realize such algorithm with the aim in mind of demonstrating that analyze user behaviors can be useful also in content distribution platform, both for the performance of the delivery service and for the network resources efficiency.

The scenario we consider is the one where a Content Provider (CP) uses a CDN for distributing its contents across the network. This is the case of typical UGC platforms such as YouTube, Metacafe, and so on. Further information about the envisioned scenario can be found in section 2.

User behaviors are exploited by the CDN nodes in the content management. Thanks to them, according to the designed caching algorithm, each node is able to determine the most convenient set of objects to be kept in memory. Such set is the group of object that will be requested by the users served by the node with the highest probability. If the strategy is effective, latency in content access and network traffic needed to solve the requests are both reduced, and the overall service performance increases.

6.3.1 Rationale and Motivation

Video-sharing platforms such as YouTube, DailyMotion and Metacafe are the application we focus on. In [76] these platforms are analyzed both from architectural and performance perspectives. Even if in different proportions, all the above-mentioned applications rely on CDNs. Performance analysis show that users proximity to content sources (CDN replica nodes) has a significative impact on the content delivery delay. Morever, the higher the last mile connection quality, the more influent is proximity on latency performance. Content dimensions must also be taken into account. The more it increases, the more the proximity become a critical factor. Indeed, making a bulky content follow a long path increases the network congestion probability and consequently the packet-loss probability, by this way worsening the overall delivery service performance.

Since video-sharing services are very popular nowadays, CDN nodes have to reserve lot of bandwith and memory space to deal with the contents of that applications. CPs pay CDN providers on the basis of the geographical coverage the want but also on the basis of the amount of the reserved resources.

A possible optimization is then trying to keep contents in the geographical area whithin they can be considered interesting by users. In other words, it can be useful to determine if a content will be popular only in a delimited area or on a global scale. In the first case, a content should be kept only in the replica servers placed in the interested region, since the users of such region would be the ones requesting it. Otherwise, the content should be replicated as much as possible on the distribution network.

Correlation between contents and geographical area has been discovered in recent studies. In [77], content request workload of YouTube is correlated with social cascades[1] detectable on Twitter. Nowadays, indeed, social cascades generate flash peak of requests directed to video-sharing platforms. The research work shows how social cascades have typically a delimited range of action, with sporadic exceptions that have a more global echo. The authors exploit the localization information that Twitter users push on their profiles to determine the geographical scope of social cascades.

[1]Social cascades are "words of mouth" performed on OSN such as Twitter and FB.

Given this big picture, we are motivated in investigating if (and how much) combining content popularity with user preferences analysis can be an effective solution to optimize content distribution.

6.3.2 A behavior-aware content caching policy

We consider N CDN servers distributed across a geographical network. A UGC platform CP engages the aforementioned CDN. We assume user requests are directed to the nearest CDN server, since this is a widespread approach adopted by CDN providers. For the same motivation, we envision that content distribution is implemented through a pull-based cooperative approach (see Par. 6.2.2). Indeed, this solution is quite diffused since CDN nodes are typically connected by means of a dedicated network and it is more convenient in terms of performance moving contents over that network instead of requesting them to the origin server.

CDN nodes act as caches. They keep in memory copies of the requested contents in order to immediatly satisfy following requests of them. Nonetheless, server memory is a constrained resource. When it reach saturation some contents need to be removed and new contents to be memorized according to a certain cache replacement strategy. Cache replacement policies [12] basically order objects according to their importance expressed on the basis of a certain criterion. When a content removal is needed, the least-priority content is deleted. Object priority can be updated on defined events, such as a new request of that content, a miss event for that content, or on periodic intervals. Classical cache replacement algorithms are LRU (Least Recently Used) and LFU (Least Frequently Used), that remove respectively the oldest contents and the least popular contents. The best strategies combine those criteria, bilancing both the content aging and their frequence of access. Utiliy-based caching is the most flexible formulation of a caching policy. It envisions the definition of a utility function that, according to some specified parameters, assigns a utility score to contents. Contents are prioritized in the cache according to their utility score.

We select a utility-based caching approach. We assume each node of the CDN perform a utility-based caching algorithm where content priority is assigned taking into account local users' preferences besides content aging and frequency of access. Given a certain CDN node, "local users" are users that usually get contents from the considered node, i.e., they are users who orbit around the geographical area served by that node. Indeed, server selection strategies usually tend to direct users' requests to the nearest CDN node. The idea is that if a node is able to keep in memory contents that like to local users, there could be a fewer number of miss events and of origin server interrogations, by this way increasing both the quality of service (smaller latency in content access) and the network efficiency.

We envision that utilities are computed by a profiling system which is able to analyze local users' attitudes and contents. The profiler divides local users into communities according to their content preferences. Users with similar taste fall into the same community. Contents are evaluated within the context of the discovered communities. The local utility of a content is defined as follows:

$$L = \sum_{c=1}^{N} w_c g_c \tag{6.1}$$

Where N is the number of the detected communities, w_c is the normalized weight of the c^{th} community, and g_c is the interest score the content gains within the c^{th} community. The normalized weight w_c is calculated as the ratio between the number of the c^{th} community members and the total number of the local users. The interest score g_c is evaluated as the interest of the content according to the most representative community member.

6.3.3 Providing interesting contents to interested users: exploiting clustering and recommendations

Recommendation algorithms perform the evaluation of the interest score of a certain content for a certain user, according to its preferences and behavior.

They then smoothly fit our needs. Recommendation algorithm are typically exploited to find content suitable to a certain user, in order to extend the user visit on a web site or in order to advertise purchasable contents. Interest score within the community scope can be calculated by considering the most representative user and considering the recommendation score the content would have for that user by using a recommendation algorithm.

Different types of recommender exist. They differ on the information about contents and users they exploit, and on the similarity metrics they employ. One of the most approach is the "collaborative filtering" where the information is represented by the ratings users express for the contents managed by a web platform. UGC platforms are typically ratings enabled, independently from the specific goal of the application (e.g., social networks vs. video sharing). Considering a rating dataset as the preliminar information base to build knowledge about users' attitudes is then a quite generical (and then acceptable) hypotesis. We then select a collaborative filtering recommendation algorithm for the computation of the g_c score. More information about the recommendation is provided in 6.3.4

In order to perform community detection, a similarity metrics is needed to determine users with similar tastes. Since we assume the set of exploitable information be represented by the ratings dataset, we consider a similarity metrics operating on such an information base. The chosen similarity metric is the Jaccard similarity, expressed as follows:

$$Jaccard(i,j) = \frac{commonRatings(i,j)}{unionItems(i,j)} \tag{6.2}$$

where $commonRatings(i,j)$ is the number of times when user i and user j express the same rating for the same content, and $unionItems(i,j)$ is the cardinality of the set of items obtained by the union of the items rated by user i and the items rated by user j. The Jaccard similarity is a real number varying between 0 and 1. The more users are similar, the more the Jaccard similarity value tends to 1. It expresses both the affinity between the ratings performed by the compared users and the number of common contents explored. A similarity matrix containing the similarity between each couple of users is calculated by using the Jaccard similarity. That similarity matrix is exploited both for the recommendation score calculation and for the community detection.

In order to perform community detection, we compute a distance matrix by simply melt the similarity metric into a distance metric:

$$Dist(i,j) = 1 - Jaccard(i,j) \tag{6.3}$$

On such a distance matrix, we apply the DBSCAN (Density Based Spatial Clustering of Application with Noise) algorithm to perform a clustering analysis and then discover the desired communities. The algorithm has been sligthly modified to obtain also the election of the most representative user of each group, or “centroid” (the user characterized by the highest density of members in his neighborhood) and the computation of the community weight. The most representative community member is chosen as a sort of centroid of the group. More information about the clustering and representative approach is provided into Par. 6.3.5.

Some results about the profiler testing are provided in the next chapter.

6.3.4 Collaborative filtering

In collaborative filtering (CF) recommendation algorithms there are three main entities involved: (i) items, (ii) users, and (iii) ratings, typically integers number ranging from 1 to 5. CF techniques do not require a semantic knowledge of the application domain, nor complex tracks about the past interactions between the user and the content platform. The only user’s past history needed is the set of his ratings. CF algorithms based on the users’ similarity, as the one we considered, build a NxN similarity matrix where the (i,j) element is the similarity between user i and user j. In order to evaluate the recommendation score of a given item for a certain user:

- the K most similar users to the considered user are selected by looking at the similarity matrix. Their similarities with the considered user are s_1, ..., s_K;
- the ratings performed by such K users for the considered item are retrieved (r_1, ..., r_K);
- the goal rating score is computed as:

$$rating = \sum_{k=1}^{K} s_k r_k \tag{6.4}$$

6.3.5 Community detection with DBSCAN

We aim to dividing users that usually connect to the same CDN node into interest-based communities. Dividing these users into communities allows to consider each group independently, i.e., to evaluate the utility of a certain content in separated groups of users having similar interests. The problem of community detection, when the number of the communities and the different kinds of communities are not known a-priori, can be faced by using unsupervised partitional clustering algorithms. Considered a metrics expressing the similarity among users, clustering algorithms can group users so that users similar to each other belong to the same group, and users dissimilar to each other belong to different groups. Among the most exploited partitional clustering algorithms, there are the K-Means algorithm and the DBSCAN algorithm. In particular, K-means shows a lower complexity than DBSCAN, but the data to be grouped (i.e., the users' representations) should be provided in the form of vectors of data features. On the other hand, DBSCAN works directly on the distance matrix among users. We have selected the DBSCAN algorithm since, while we are not able to build a feature vector for each user, we are able to build a distance matrix D starting from the similarity matrix S built by using the Jaccard similarity: $D = 1 - S$.

The inner working on the DBSCAN can be briefly described as follows. As the name suggest, DBSCN groups data in different clusters on the basis of the density. Considered a certain user, all the users having a distance from the considered user which is lower than "epsilon", are searched. If the number of the found users is higher than "minpoints", a cluster is created and all the users are ascribed to that cluster. Then, for each of the novel users ascribed to the cluster, the epsilon-neighborhood is explored. If other users are found within it, they are ascribed to the aforementioned group. The procedure stop until there are no other users to add to the group. When the group is completed, the left-out users are considered. A new user is taken as a new starting point, and the procedure is played again to find a new cluster.

There are two key parameters that need to be set: the radius of the density interval, epsilon, and the minimum number of members needed to form a group, minpoints. As for all clustering algorithms, input parameters need to be tuned according to the data to be analyzed. We have made some experiments both with artificial dataset and with real dataset, as described in the next chapter.

Detection of representative users

The DBSCAN algorithm has been slightly modified in order to allow the extraction of the most representative user of each community (centroids). Centroids are the nodes with the highest density among the community members, i.e., the nodes having the highest number of neighbors within the epsilon-neighborhood.

Weighting communities

In order to determine the local utility of a content in a cache node, the community utilities of that content have to be weighted according to the local weight of each community. Each community is weighted on the basis of its population (i.e., the number of community members), so that its weight corresponds to the number of points ascribed to the associated cluster determined through DBSCAN.

6.4 Dealing with mobility: Mobile CDNs

With such an explosion of mobile access to multimedia web contents, there is an unmet demand for a higher quality and speed in content delivery to the connected devices. Further issues, due to users mobility, arise in this context. First of all, it is harder to route contents to mobile users since they are continuously moving, i.e., changing the cell from which they connect to the global network. Moreover, cell switching increases the possibility to run into a “cache miss”, which is the case of a content not available on the CDN cache server. Cache misses typically cause the request to be forwarded across the complete path to the origin server, which increases both the response delay and network costs in terms of bandwidth utilization.

The increased temporal and spatial variability of mobile users demand should be taken into account both in the design phase and in the operational phase of a CDN. Mobility-aware CDNs are briefly called “mobile CDNs”. Given the complexity of the above scenario, content delivery optimization in mobile CDNs is clearly a hard task. We firmly believe that leveraging knowledge on users mobility, as well as on their favorite contents, can prove useful in the design of optimized cache and content placement strategies. In such context, new content placement policies, aimed to increase cache hits and minimize network costs, should be developed and tested. To quantify the deriving benefits, both in terms of user perceived Quality of Service (QoS) and ISP resource efficiency, simulation must be employed. Thus, we need to model and reproduce both people mobility and their request patterns in order to exploit a realistic request workload in simulations.

While CDN issues in wired networks have been widely explored so far, and still attract lots of research efforts, to the best of our knowledge only few works have been devoted to CDN enhancements specifically tailored to mobile networks. Some architectural solutions aim to improve the provisioning of multimedia streaming services to mobile clients. Indeed, streaming sessions are typically long-lived and can be affected by QoS degradation because of user mobility during the multimedia content fruition.

Yoshimura et al. specify in [91] a patented CDN architecture named "Mobile Streaming Media CDN" envisioning request redirection on the basis of client movements. A central Content Location Manager makes use of a SMIL (Synchronized Multimedia Integration Language) file to provide mobile clients with information about the best replica server from which they can gather content segments according to their current position, server load and network conditions.

Changing replica server during a streaming session according to client's movements (hand-off) causes network traffic overhead and can provoke service disruption or degradation. Tariq et al. propose in [81] a hierarchical architecture of CDN surrogate servers. Lower-layer servers are the nearest ones to mobile clients and can provide contents with low delay and jitter. Higher-layer servers, on the other hand, present larger coverage areas. Server selection is performed in a distributed manner, aiming to minimize the number of hand-offs while preserving client's QoS level agreement. Estimation of the time the client will spend in a server's coverage area is taken into account in the selection process.

As to content distribution strategies, Aioffi et al. [2] design a mobility-aware content placement algorithm aimed to minimize network traffic in the presence of a high variability level of content demand due to both content popularity and user mobility. The algorithm is based on the estimation of the future spatial demand distribution according to the past history recorded by each replica server for each content.

6.5 Mobility and request patterns

Mobile devices originating content requests are carried by humans. The mobile communication is therefore affected by users movements in urban and suburban spaces. Making predictions on users position is very useful to trace content request origin distribution in space. A human mobility model is employed to this purpose. Mobility models are engineered and used to represent mobile users movements by capturing how their space position, speed and acceleration change over time.

Up to now, different mobility models have been considered to simulate human mobility. We provide a brief overview of the main ones, respectively the Random Walk (RW) and the Lévy Flight (LF). Their main drawback is an insufficient degree of adherence with reality, as we motivate below. The model in [41], on the other hand, is the most realistic one. We then provide more details about it, as well as describe how we implemented it.

6.5.1 The random walk model

The random walk (RW) model is the mathematical representation of a sequence of random steps [80]. A random walk can be modeled as a Markov chain, i.e., a memoryless stochastic process, in which the next step depends only on the current position and not on the past steps. Given a two-dimensional grid of places, the random walk of a user according to a uniform distribution can be viewed as a jump from a place to the nearest location along one of the cardinal directions taken with the same probability (i.e., she/he can move with the same probability of 1/4 to North, South, East, West).

This model is the simplest one among those currently available for mobile users. It has been widely adopted in simulation studies related to mobile adhoc networks (MANETs).

6.5.2 The Lévy flight model

A Lévy Flight (LF) is a random walk in which the step length is a random variable having a heavy-tailed probability distribution[2]. A user moving according to a Lévy flight makes often “small” steps and sometimes “big” steps.

The Lévy flight model turns out to be suitable to represent some animals’ movement patterns. Works [85, 70] showed that, respectively, albatrosses and

[2]A heavy-tailed distribution is a probability distribution whose tails are not exponentially bounded.

spider monkeys move according to LF when searching for food. Likewise, sharks and other ocean predators switch to move according to LF when hunting, as demonstrated by an extended measurement campaign reported in [79].

LF has been associated also to human patterns. Authors in [71] monitored 44 volunteers by GPS for 100 hours in different outdoor environments (i.e., college campuses, a metropolitan area, an amusement park and a state fair) and concluded that users movements resemble Lévy flights within a displacement range of 10 km. These results agree with a previous study on human mobility, carried out by tracing bank notes [24]. In such study, the authors demonstrated that the distance between two consecutive bank notes sightings shows a fat-tailed probability distribution.

6.5.3 The González-Hidalgo-Barabási model

RW and LF mobility models are the most popular in literature. Nonetheless, they are far from being realistic in describing human behavior. The RW is widely used since it is simple to be implemented; though, it is quite generical. The LF, on the other hand, is not satisfying for modeling humans as it is for animals. In fact, human mobility measurements have been carried out on a small set of samples, as for the case in [71], or with a non-reliable trajectory detection, as for the case of bank notes tracking in [24].

Unlike the aforementioned models, the González-Hidalgo-Barabási (GHB) model has been derived from a study conducted over a huge amount of real data [41]. The authors followed 100.000 anonymized mobile phones to gain information about user mobility habits for a period of six months. Users positions have been tracked within a geographical area of the order of magnitude of 10^6 km^2 with a granularity of about 3 km^2. Each time a user initiated or received a call or a text message, the location of the antenna routing the communication was recorded, thus allowing the reconstruction of the user's trajectory. Mobile phones, carried by individuals during their daily routine, are the best proxy to capture individual human trajectories.

Moreover, thanks to the large size of the covered area and to the extended duration of the analysis period, the analyzed dataset can be considered to be both time- and space-independent.

The main results of the analysis is the finding that human trajectories show a high degree of temporal and spatial regularity. According to the derived model, in other words, it is possible to model human mobility regardless of time and space.

González et al. found people tend to return to places they visited before. They then associate each user with a certain number of visited locations, ranked by the number of time she/he returns to them (this number being called "mass" of the location). Given such set of places, the mobility pattern of each user is characterized by: (i) a center of mass, and (ii) a radius of gyration. They demonstrated users spend more than 80% of their time within an area identified by the circle whose center is the user's center of mass, and whose radius is the user's radius of gyration.

More precisely, by ranking each visited location according to the number of user's visits within the analysis period, they show that the probability of finding such user in the l^{th} place is about $1/l$, thus it approximates a Zipf distribution. The center of mass represents the user's average position during the tracking time. The center of mass (x_{cm}, y_{cm}) is computed as:

$$x_{cm} = \frac{1}{M}\sum_{l=1}^{L} m_l x_l \qquad y_{cm} = \frac{1}{M}\sum_{l=1}^{L} m_l y_l \tag{6.5}$$

where L is the number of visited locations, m_l and (x_l, y_l) are, respectively, the mass of the l^th location and its coordinates, and $M = \sum_{l=1}^{L} m_l$ is the total mass of the system. On the other hand, the radius of gyration can be expressed as:

$$r_g = \sqrt{\frac{1}{M}\sum_{l=1}^{L} m_l((x_l - x_{cm})^2 + (y_l - y_{cm})^2)} \tag{6.6}$$

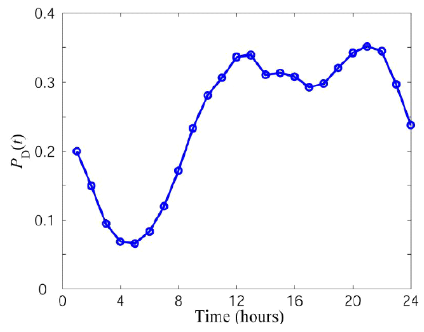


Figure 6.2: Users displacement probability over time

González et al. also analyzed the distribution of displacements of all users. They found that by rotating and scaling users trajectories, all users show the same step distribution, i.e., such distribution is independent from the users radius of gyration. Thus, despite the diversity of their 'traveling' histories, humans follow simple reproducible patterns.

To move or not to move

Further information on human mobility is provided in [88]. Wang et al. investigated how the probability of a user movement from a place to another varies over the day hours. On the basis of the same dataset employed in [41], they evaluated the ratio between the number of displacements in each hour over the total number of recorded positions, discovering the trend shown in Fig. 6.2.

An example

On the basis of the aforementioned studies, we implemented the GHB model to simulate mobile users movements. We herein provide a simple example to make the reader better understand the main principles of the model we implemented. Fig. 6.3 traces the trajectory of a user moving between 4 places on a two-dimensional grid. The starting place corresponds to point $(7, 6)$. Each arrow represents a movement. Since the user crosses both places $(7, 6)$ and $(3, 5)$ twice, such places have a mass of 2. Being crossed only once, the other places have a mass of 1. The mass of each location is reflected by their

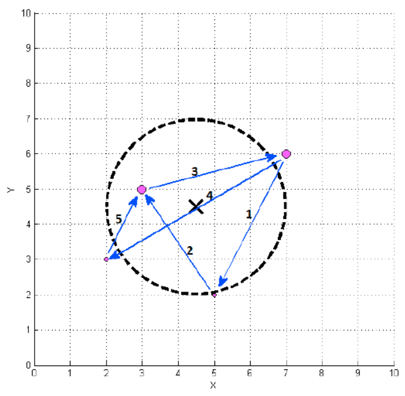


Figure 6.3: Describing mobility by the González model

bullet size. By using equation (6.5), we evaluate the user's center of mass, corresponding to point $(4.5, 4.5)$ - the black X in figure. By equation (6.6), we derive the user's radius of gyration, $r_g = 2.48$ km, thus detecting the main user's movement area - the one within the dotted circle.

6.5.4 Request patterns

In order to simulate a realistic workload for a mobile CDN both mobility and content request models are needed. While a mobility model provides information on where people are located, a content request pattern describes user's requests from both a quantitative and a qualitative perspective. Such a pattern tells how many requests a user would issue over time (the quantitative aspect), and what kinds of contents she/he would ask for (the qualitative aspect).

Content popularity has to be taken into account while modeling the quality aspect in a content request pattern. In the last years, web users behavior has dramatically changed, due to the success of User Generated Content (UGC) platforms such as Facebook, YouTube, and the like. Thanks to them, people are able not only to read, but also to produce and share contents over the web. Before the ascent of the UGC paradigm, content popularity was in some way controllable through professional marketing campaigns. Now, the scale, the dynamics, and the decentralization of user generated contents make conventional popularity evaluations unsuitable, since content popularity is much more ephemeral and unpredictable.

In the depicted scenario, deriving empirical conclusions from the analysis of real world data, rather than exploiting theoretical assumptions, is the only reasonable solution. This is the approach chosen by several content popularity studies.

An example is provided in [29], where the authors collected and analyzed a large amount of data coming from two outstanding video UGC platforms (i.e., YouTube and Daum[3]). Among the properties they discovered, they showed that the popularity distribution is heavy-tailed and that around 10% of top popular videos accounts for nearly 80% of the overall requests.

The work in [87] seems to confirm such qualitative trend and provides additional information on the quantitative plane. Wallenta et al. collected users public messages ("posts") from two noteworthy content aggregation sites, namely Digg[4] and Reddit[5], to understand how post popularity evolves over time. They relied on users vote counts as indicators of post popularity. In both cases, the vote distribution roughly follows the 80/20 rule, i.e., 80% of requests are directed to 20% of the most popular posts, being in agreement with the previous study. Moreover, they derived a trend for the number of requests over time. The obtained distribution presumably follows web users behavior, since it shows: (i) a global minimum at 4 a.m., when people are sleeping, (ii) a peak between 9 a.m. and 12 a.m. during weekdays, when people are working, and (iii) a peak between 11 a.m. and 13 a.m. in the weekend, when people enjoy their spare time.

We took into account both such experimental results in the implementation of the content request pattern in our simulations. The last two above cited works ([81, 2]) show the benefits deriving from the adoption of the proposed mobility-aware solution by means of simulation experiments, while the first ([91]) just presents a prototype implementation of the proposed system

[3]www.daum.net

[4]www.digg.com

[5]www.reddit.com

without providing performance results in the presence of a realistic workload of content requests. The mobility information datasets exploited in simulations are hand-crafted on the basis of common sense assumptions and don't derive from the scientific analysis of realistic data related to human mobility patterns. We do believe, instead, that in this kind of experiments a reliable model of user mobility is needed, in order to perform realistic movement estimation and obtain trustworthy performance results.

Chapter 7

Content-aware applications and infrastructure part II: testing and performance aspects

7.1 Introduction

In this chapter we present the preliminar results of our ongoing research on content-aware applications.

In the previous chapter we propose to leverage recent achievements coming from three main research areas in order to optimize the performance of content distribution networks: user mobility patterns, user request patterns, and user content preferences.

Regarding users content preferences, we draw a content caching mechanism taking into account local users preferences. In Section 7.2 we report some experimental analysis of the interest-based community detection needed to apply the behavior-aware content caching policy described in the previous chapter. We test the analysis on different datasets, described in Par. 7.2.1. We provide details on the analysis output in Par. 7.2.2, as well as on the validation criteria we design to evaluate the goodness of the proposed approach to the computation of contents' utility. Results and final considerations are given in Par. 7.2.3.

Then, we illustrate how we developed and validated a CDN simulation framework able to reproduce web users behaviors. The ad-hoc simulator is conceived at the outset as a general framework for assessing the effectiveness of different content distribution strategies in presence of realistic request workload. More precisely, we properly combine the Barabasi's time- and location-independent user mobility model with a content request pattern derived from the analysis of real data in order to derive a realistic load model for CDN servers. The simulator allows to configure and observe the behavior of the CDN infrastructure in terms of delivery costs, bandwidth requirements, server load and latency between content request and delivery phases. Section 7.3 is devoted to the main features of the simulator we developed, providing details on its architecture and on the algorithms we implemented to reproduce user behaviors. Finally, we report validation results in Section 7.3.3 and briefly draw conclusions in Section 7.3.4.

7.2 Community detection in rating-based web platform

The anayzer we developed within the BACON project is aimed to detect groups of users that have expressed similar ratings for the contents of a web platform. For each detected community, a reference user named “centroid” is extracted. He exhibits the typical preferences of the members of his cluster. Web contents are firstly evaluated in each community by considering the interest score they gain for each community centroid. Such interest score is computated leveraging a collaborative filtering recommendation algorithm. We develop a utility program that allows for running different test type for the above-mentioned analyzer. Tests are different both for the type of the test and for the type of the selected input dataset.

7.2.1 Dataset types

We consider four different type of dataset:

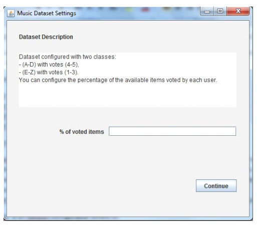
1. Music dataset

2. Artificial_1 dataset

3. Artifical_2 dataset

4. MovieLens dataset

Each dataset is made of Users, Ratings (where each rating is an integer number ranging from 1 to 5), and Items (the rated contents). The first three dataset are “artificial” since, for construction, they are made of pre-configured, separated communities on the basis of content preferences. Such datasets are useful since they can be used as a ground truth to verify and quantify how much the analyzer is able to detect different groups. Artificial dataset are configurable in different parameters (the number of users, item, percentage of rated item for user, and and so on - see Figure 7.4). That allows to perform testing on dataset characterized by different properties. The last dataset, MovieLens, is a realistic ratings dataset publicly provided by the GroupLens Research group of the Minnesota University[1].

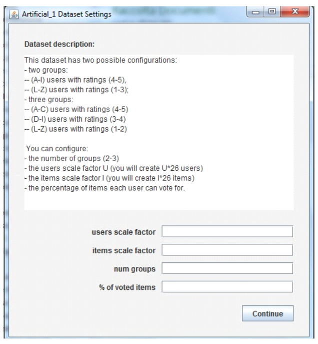

Music Dataset Settings

Dataset Description

Dataset configured with two classes:
- (A-D) with votes (4-5),
- (E-Z) with votes (1-3).
You can configure the percentage of the available items voted by each user.

% of voted items

Figure 7.1: Configuring the Music dataset


Artificial_1 Dataset Settings

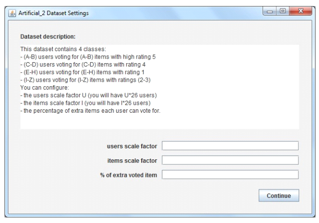
Dataset description:

This dataset has two possible configurations:
- two groups:
-- (A-I) users with ratings (4-5),
-- (L-Z) users with ratings (1-3);
- three groups:
-- (A-C) users with ratings (4-5)
-- (D-I) users with ratings (3-4)
-- (L-Z) users with ratings (1-2)

You can configure:
- the number of groups (2-3)
- the users scale factor U (you will create U*26 users)
- the items scale factor I (you will create I*26 items)
- the percentage of items each user can vote for.

users scale factor
items scale factor
num groups
% of voted items

Figure 7.2: Configuring the Artificial_1 dataset


Artificial_2 Dataset Settings

Dataset description:

This dataset contains 4 classes:
- (A-B) users voting for (A-B) items with high rating 5
- (C-D) users voting for (C-D) items with rating 4
- (E-H) users voting for (E-H) items with rating 1
- (I-Z) users voting for (I-Z) items with ratings (2-3)
You can configure:
- the users scale factor U (you will have U*26 users)
- the items scale factor I (you will have I*26 users)
- the percentage of extra items each user can vote for.

users scale factor
items scale factor
% of extra voted item

Figure 7.3: Configuring the Artificial_2 dataset

Figure 7.4: Dataset settings

Music dataset

The Music dataset is a small configurable artificial dataset characterized by:

[1]http://www.grouplens.org/node/73

- 23 users characterized by realistic names
- 19 contents constituted song names

This dataset has two pre-configured set of users: a first group which likes contents (users randomly rate contents with a 4 or 5 score) and a second group which does not (random rates varying among 1, 2, 3). Users belonging to the first group are recognizable since the first letter of their names ranges from A to D ("A-D users"), while the others have names beginning from E to Z ("E-Z users"). It is possible to set the percentage of the rated items randomly chosen from the dataset for each user (Figure 7.1).

Artificial_1 dataset

The Artificial_1 dataset can be created in two different modalities:

1. two-groups modality: in this case, the dataset contains two groups of different interests, similarly to the Music dataset case;
2. three-groups modality: in this case, the dataset is made of three groups, (i) A-C users that express a strong interest (a score of 4 or 5), (ii) D-I users that moderately appreciate contents (a score of 3), and (iii) L-Z users that don't like contents at all (a score of 2 or 1)

In both modalities, it is possibile to configure the number of users, the number of items, and the percentage of items that each user rates (Figure 7.2).

Artificial_2 dataset

The Artificial_2 dataset presents four groups of users:

- The first group of users are A-B users that rate only A-B contents with a high rate (5).
- The second group of users are C-D users that rate only C-D contents with a discrete rate (4).

- The third group of users are E-H users that rate only E-H contents with a very low rate (1).
- The fourth group of users are I-Z users that rate only I-Z contents with a low-medium rate (2 or 3).

In this dataset, people belonging to a same group tend to vote for the same contents with the same rating. For the sake of simplicity in the results interpretation, we have decided that names of the users and the names of the contents begin with the same letter. It is possible to introduce entropy in the group of voted items by setting the percentage of "extra items" (i.e., items with a different beginning letter) each user can vote for. If such percentage is null, than the user votes for the envisioned set of contents only; otherwise, a number of random extra contents are selected and the user rates them according to the rate envisioned to the group he belongs to. Other configurable parameters are the number of users and the number of items, as already seen above for the Artificial_1 dataset (Figure 7.4).

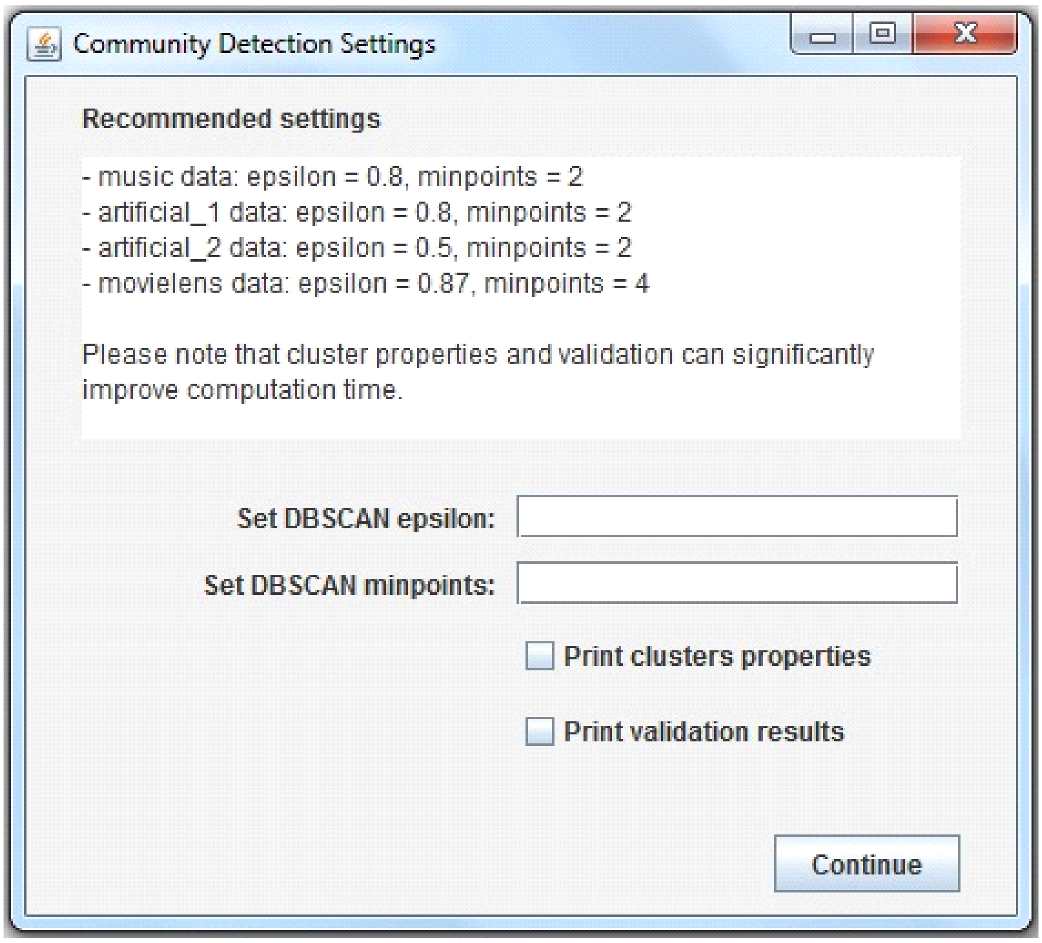
MovieLens dataset

The MovieLens dataset is made of 100.000 ratings performed by 943 users for 1682 contents, where contents are movies. It is publicly provided by the MovieLens recommendation platform[2], which is a free web service conceived to give recommendation of movies to users on the basis of their ratings history. The recommendation service is based on collaborative filtering techniques and is freely provided by the GroupLens research group of the Computer Science and Engineering Department of the Minnesota University. Users' information is properly anonymized.

7.2.2 Test types

There are two possible tests: the basic one, which is conceived to show the main functionality provided by the analyzer, and the tuning and performace

[2]http://www.movielens.org/html/tour/index.html


Community Detection Settings

Recommended settings

- music data: epsilon = 0.8, minpoints = 2
- artificial_1 data: epsilon = 0.8, minpoints = 2
- artificial_2 data: epsilon = 0.5, minpoints = 2
- movielens data: epsilon = 0.87, minpoints = 4

Please note that cluster properties and validation can significantly improve computation time.

Set DBSCAN epsilon:

Set DBSCAN minpoints:

☐ Print clusters properties

☐ Print validation results

Figure 7.5: Community detection settings

test, which is aimed to identify the better tuning parameters.

Basic cycle

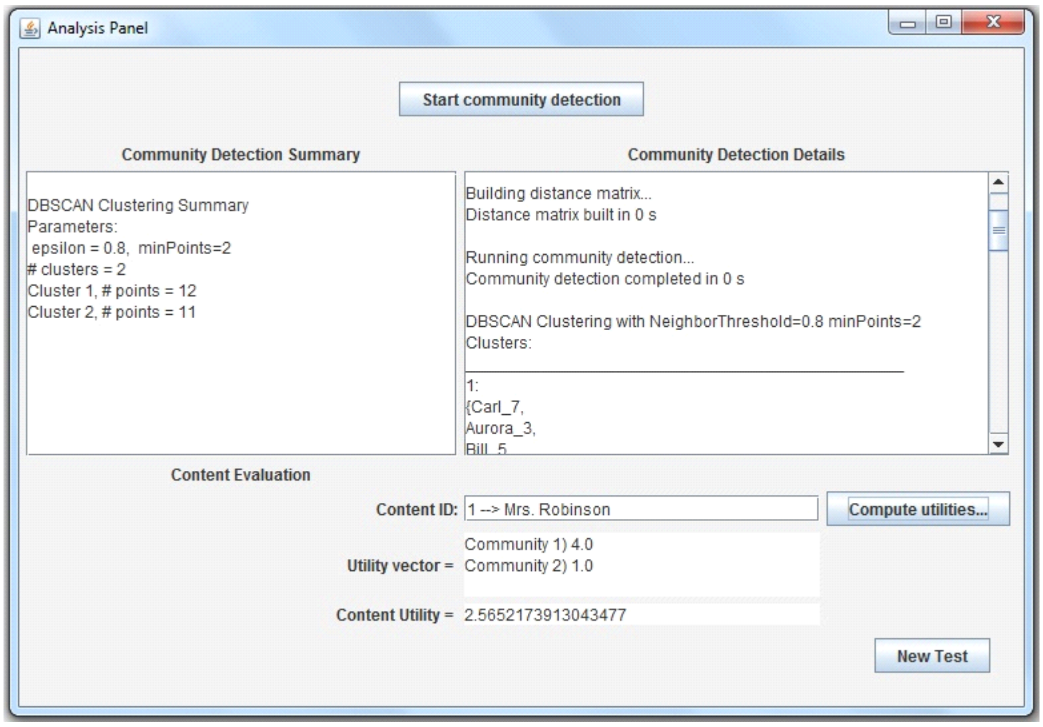
In the basic cycle, it is possible to select and configure the dataset used as input for the analyzer, as well as set the analyzer with the desired DBSCAN parameters (Figure 7.5). For the aforementioned dataset, the best DBSCAN parameters are suggested.

It is also possible to request the computation of some cluster properties and of the validation results .

In particular, the cluster properties that the analyzer evaluates are:

- the number of cluster members (community population, or the community weight)
- the mean distance between members of the same cluster ("average internal distance")
- the maximum distance between members of the same cluster ("max internal distance")
- the minimum distance between two users belonging to different cluster ("single link")
- the mean distance between two users belonging to different cluster ("average link")
- the maximum distance between two users belonging to different cluster ("complete link")

The validation is the computation of the percentage of how many times the centroid of a cluster represents effectively the members of his community. Such property is verified each time the difference between the recommendation score of a certain content for the centroid and the recommendation score for a member of his cluster is the lowest one among the difference computated between the recommendation score for the other centroids and the considered memeber. For each content and for each community we count the number of times such a properties is verified. The counters associated with each community are then added and the result is divided for the number of the discovered communities, in order to have a percentual value which is called "accuracy" and it is showed as the result of the validation test. The nearer is the accuracy to 100%, the better each centroid represents his community members in terms of content preferences. The validation output shows two kinds of accuracy: the first one is computed by ignoring the "noise points", i.e. points that are not ascribed to any cluster, while the second one is called "accuracy with noise" and it is computed by considering the noise points forming a community for which a centroid is also elected.


Analysis Panel
Start community detection
Community Detection Summary
Community Detection Details
Content Evaluation
Compute utilities...
New Test

```
DBSCAN Clustering Summary
Parameters:
 epsilon = 0.8,  minPoints=2
# clusters = 2
Cluster 1, # points = 12
Cluster 2, # points = 11
```

```
Building distance matrix...
Distance matrix built in 0 s

Running community detection...
Community detection completed in 0 s

DBSCAN Clustering with NeighborThreshold=0.8 minPoints=2
Clusters:
_____________________________________________
1:
{Carl_7,
Aurora_3,
Bill_5
```

Content ID: 1 --> Mrs. Robinson

Utility vector = Community 1) 4.0
Community 2) 1.0

Content Utility = 2.5652173913043477

Figure 7.6: An example of analysis panel

When cluster properties and validation results are requested, the analysis elapsed time can increase significantly, since both of them call for an onerous inspection of the detected communities.

When the desired analysis parameters and options has been configured, it is possible to start the community detection algorithm. When the analysis is completed, communities information is provided. It relates to the communities' composition (i.e., the members' list), population and centroids, as well as the time needed to calculate the similarity matrix(one of the most time-consuming operation) and to execute the clustering algorithm.

After both communities and centroids are determined, it is possible to chose an item and evaluate the score of it according to the utility model presented in the previous chapter, as it can be seen in Figure 7.6.

Tuning and performance

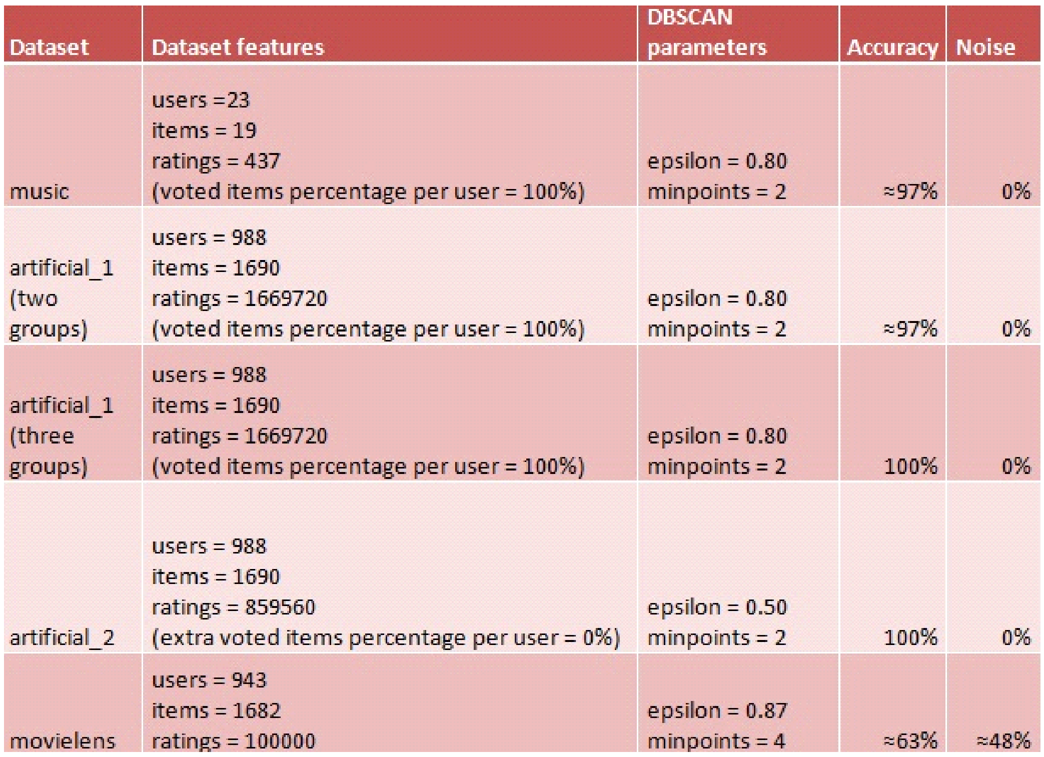
With the tuning and performance test, it is possible to perform multiple clustering tests with different tuning parameters (epsilon and minpoints) in order to assess the clustering goodness under different configurations. The clustering goodness is expressed in terms of the aforementioned clustering properties and accuracy results. It is possible to configure a range of values for each parameter and an incremental step, in order to automate tests execution. This kind of test allows for the identification of the couple of parameters values that, for example, lead to a community partition without noise points or that guarantees certain clusters properties as well as a certain level of accuracy.

7.2.3 Results and considerations

We perform tuning and performance tests for the artificial datasets and for the MovieLens dataset. We set the artificial datasets in a way that the number of users and items is similar to those of the MovieLens dataset. The best values for epsilon and minpoints we obtained are reported in Figure 7.7. Therein are reported also the accuracy percentage and the noise percentage we obtain in the case of optimal configurations.

For the artificial datasets, we select parameters values allowing for a correct detection of the pre-configured communities. For the reported parameters values, we reach a 100% correct detection.

For the tuning of the parameters in case of the realistic dataset, we observe how the noise percentage varies with the parameters. For the realistic dataset it has been discovered a minor robustness of the parameters values: for little variations, we obtain very different results. Looking at Figure 7.8 only for epsilon varying between 0.8 and 0.9 we can have a noise percentage different from the 100% and from the 0%. When epsilon tends to 0.9, the noise tends to disappear since the epsilon radius is so big that all users fall into the same cluster(see Figure 7.9), then when we lack the capacity of discriminating different interest-based groups.



Dataset	Dataset features	DBSCAN parameters	Accuracy	Noise

music	users =23
	items = 19
	ratings = 437
	(voted items percentage per user = 100%)
		epsilon = 0.80
		minpoints = 2
			≈97%	0%

artificial_1 (two groups)	users = 988
	items = 1690
	ratings = 1669720
	(voted items percentage per user = 100%)
		epsilon = 0.80
		minpoints = 2
			≈97%	0%

artificial_1 (three groups)	users = 988
	items = 1690
	ratings = 1669720
	(voted items percentage per user = 100%)
		epsilon = 0.80
		minpoints = 2
			100%	0%

artificial_2	users = 988
	items = 1690
	ratings = 859560
	(extra voted items percentage per user = 0%)
		epsilon = 0.50
		minpoints = 2
			100%	0%

movielens	users = 943
	items = 1682
	ratings = 100000
		epsilon = 0.87
		minpoints = 4
			≈63%	≈48%

Figure 7.7: Analysis results

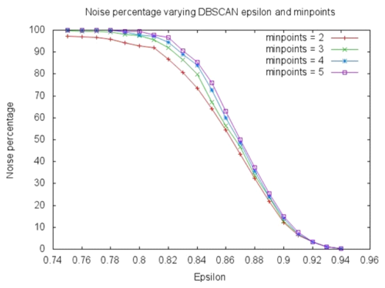


Figure 7.8: Noise % varying epsilon

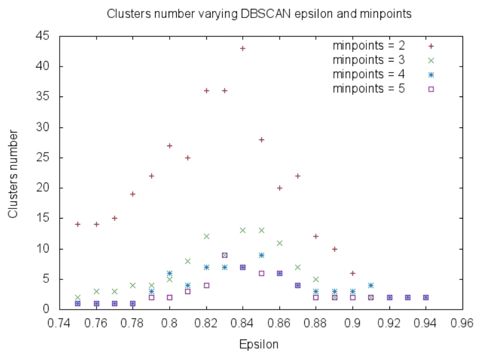


Figure 7.9: Clusters number varying epsilon

In Figure 7.7 are reported the mean accuracy and the mean accuracy with noise we obtain with 10 basic cycle tests. While results for artificial datasets are very good, using the MovieLens dataset we can only reach an accuracy of 63%. Moreover, in this last case, we have the 48% of the users classified as noise, i.e., they are not ascribed to any community. These results show that further clustering and similarity measures need to be tested in order to improve.

7.3 The mobile CDN simulator

In this section we present a modeling framework which was designed to provide a realistic simulation of a mobile CDN, including challenging functions like the agents mobility patterns [40]. The framework makes available an automated environment for conducting experiments and can be used as a testbed for mobile CDN evaluation and experimentation. It differs from other available CDN network simulators because it also reproduces a mobile environment. For this reason it has some innovative components enabling it to perform new kinds of tasks and which can be briefly categorized as follows:

- CDN topology: In the simulator there is a hierarchical structure for the network representation. The network consists of a core network, some edge networks and several antennas. It is possible to see the core network as a connected graph. For each node of the core network there is an edge network. Every edge network is connected to a single node of the core network. An edge network is a tree and its root is connected to the core node from which it stems. The edge networks are connected to antennas through their leaves.

- Caching: The surrogate servers are represented by caches that handle the distribution of contents.

- Content: Different content types (e.g., web content, streaming media and real-time video) are represented by a common data structure which can be specialized based on the dimension of the specific content to be represented.

- Agent: The agent is the linchpin of all simulations. It moves across the simulated space according to its mobility pattern and asks for contents according to its request pattern.
- Simulated space: Agents move in a rectangular area filled with antennas. Antennas are not homogeneously placed, so there are zones with different antenna densities. For this reason there is a special mapping policy allowing to link each leaf of the edge network trees to a specified set of antennas.
- Cost evaluation: Different cost evaluation metrics can be used to test the appropriateness of a content placement strategy. The output of the cost evaluation system can also be used at run-time as a feedback for the definition of better optimization strategies.

The simulator allows us to devise and deploy different antenna placement strategies. It performs an automatic mapping between the simulated space and the network topology and makes it possible to rely on different kinds of caches, located either in the core network or in one of the available edge networks and reachable via a simple Dijkstra routing algorithm. Agents have their own customizable mobility and request patterns. It is possible to create and use different content placement strategies, as well as different cost evaluation metrics.

Fig. 7.10 provides a high level view of the architecture of our simulator.

As it comes out from the picture, the CDN simulator has a structure made of an internal core and a number of plug-ins. The core deals with all the primary functions including the dynamic loading of the plug-ins and of the configuration parameters. A plug-in is created by inheritance: it is a subclass, derived from an abstract class of the core, that must implement a defined set of abstract methods of the superclass. In this way it is possible to

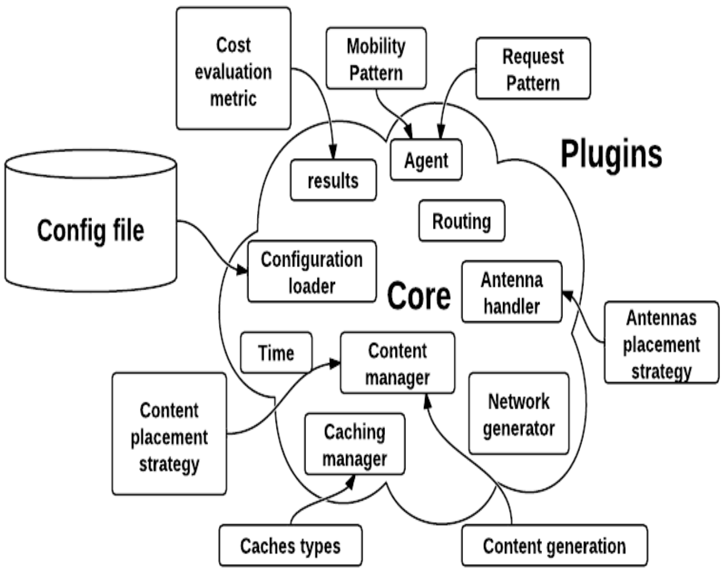


Figure 7.10: The mobile CDN simulator: architecture overview

create new plugins by simply following an easy formal grammar. There are currently seven kinds of plug-ins associated, respectively, with: (i) mobility pattern; (ii) request pattern; (iii) antenna placement strategy; (iv) cache types; (v) content generation; (vi) content placement strategy; (vii) CDN cost evaluation.

7.3.1 Inner workings of the simulator

The simulator makes many complex calculations and is composed of a large number of classes that interact with each other. We herein sketch the main steps characterizing its operation. There are four macro-phases: loading, creation, simulation and results collection. First of all, the simulator parses the configuration file to retrieve the needed information and creates the data structures to be used later. This is the loading phase. Once done with this step, the simulator creates the CDN by allocating a mXn grid representing the area to simulate and which hosts the antennas on the basis of a configurable antenna placement strategy. Both core network and edge networks are then created and interconnected. Each edge network will be linked to a pre-defined number of antennas. A soon as the network setup is over, the last two creation steps concern the positioning of caches within the CDN tree and, finally, the initial placement of contents according to the chosen strategy. Then, the actual simulation takes place, which consists of a preliminary configuration phase (related to the setting of the cost evaluation strategy, together with the request and mobility patterns of choice), followed by a number of iterations (whose overall duration depends on the chosen simulation time) during which the created agents are activated on the basis of the chosen model and the selected dynamic content placement strategy is applied, if needed. At the end of the main simulation loop, results are collected and analyzed. As an example, it is possible to see, for each antenna, the related cache miss rate together with its total cost, which is computed based on the requests it has helped to serve.

7.3.2 Mobile CDN simulation algorithms

In this subsection we will briefly discuss the main algorithms we chose for both the simulator core and the associated plugins. We will also try and illustrate the design choices which led us to selecting specific solutions over potential alternatives.

The core of the operation of the simulator resides in the agent animation loop showed in Fig. 7.11. To decide whether or not to move an agent we use the moving probability distribution proposed in [88]. To calculate the new position, we use the masses of the locations: each mass represents the probability to go to that location. The path is computed through the shortest path algorithm. A critical point here concerns the behavior at the borders of the simulated area (i.e., what happens if some agent locations are off the board). We decided to use a wraparound world. This means that if, for example, the agent leaves the area from the south, he will arrive in the north; if it leaves from the east, it will arrive in the west. This is done with a very simple technique. If the area is a grid of dimension mXn, there is a normalization of every location (x, y) such that $(x_{norm}, y_{norm}) = (x \mod m, y \mod n)$. The

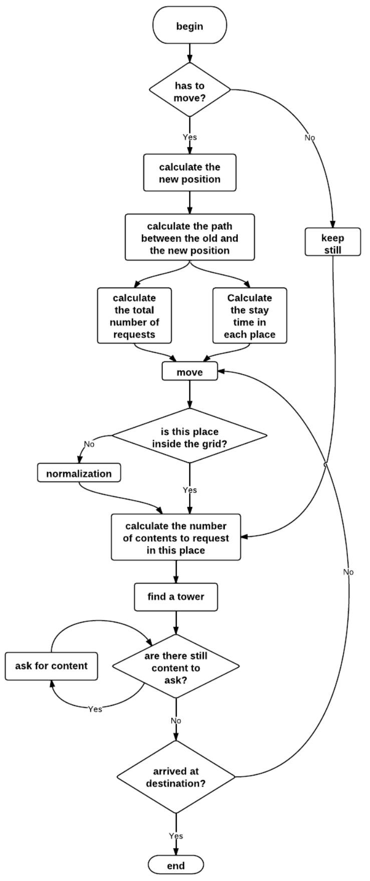

begin
has to move?
Yes
No
calculate the new position
calculate the path between the old and the new position
keep still
calculate the total number of requests
Calculate the stay time in each place
move
is this place inside the grid?
No
normalization
Yes
calculate the number of contents to request in this place
No
find a tower
ask for content
are there still content to ask?
Yes
No
arrived at destination?
Yes
end

Figure 7.11: Flow chart of the agent animation loop

new normalized location is the one that we use for all the computations.

As to the request pattern, we designed a model based on the work from Wallenta et al. [87], with some small changes and simplifications. Such model is dynamic in nature and depends on the time of the simulation. To decide 'how many' requests to make in a certain time period, a peak value is multiplied by a factor that depends on the time of the day. The distribution of the factors comes from a weighted merge of two distributions, associated, respectively, with weekdays and weekends. Finally, to determine 'which' requests to issue (i.e., which contents to download), a proper content popularity system is adopted. Such system works as follows. Each content belongs to a popularity class with some probability. We derived the content popularity distribution from an analysis of raw data coming from Digg and also contained in the above cited work from Wallenta. Such distribution is basically computed by: (i) grouping contents with the same number of votes; (ii) merging the groups with more than 100 votes; (iii) assigning a probability to each group according to the number of contents that it contains; (iv) properly normalizing the obtained score. The resulting distribution has been demonstrated to be heavy tailed, according to the results presented in [29, 87]. For each request, the agent chooses, according to the popularity distribution, both a class and (randomly) a content inside that class.

Finally, with respect to antennas management two main issues have to be faced: mapping and selection. We implemented two different strategies in our simulator. The former fills the simulated space with an antenna every kilometer, whereas the latter randomly places the antennas. This means that there can be areas with different antenna densities, which calls for a mapping and selection strategy that is independent of the antennas topology. The mapping strategy is based on the knowledge of the network topology and works as follows. If there are N core nodes (and hence N edge networks), it divides uniformly the simulated area horizontally in N pieces. Each piece is assigned to a single edge network. Afterwards, for each edge network: (i) it computes its own number of leaves; (ii) it determines how many antennas fall in its own piece of the overall area; (iii) it divides uniformly the antennas between the leaves. In this way each edge network manages an area of the simulated space independently of the quantity of antennas in that area and all its leaves are connected to a number of antennas that is the same for each such leaf. This means that antennas are distributed uniformly only inside the edge networks while among different edge networks there can be different antenna quantities.

7.3.3 Validation

A crucial part of our work consisted in an assessment of our implementation choices, with special regard to the employed GHB mobility model. To this purpose, we collected data by simulating 100.000 agents moving within an area of 10.000 km^2. For each such agent we logged its position at every simulation step (whose granularity was set to a value of one hour) and then computed both the frequency of locations it visited during its simulation lifetime and the agent's radius of gyration (using the equations (6.5) and (6.6)). In this way we created a dataset similar to the one used by Gonzalez et al. in their work and we compared the two mentioned datasets. Fig. 7.12 shows, in an overlapped fashion and with a log-log scale graph, how our implementation of the GHB model closely fits the graph in [41] for what concerns the radius of gyration distribution. Indeed, the picture shows that the two distributions have exactly the same shape, with just some slight distortion on our side towards the tail of the plot (i.e., for $r_g > 50$). This can be explained because of the low probability of having big radii. A sharper plot might actually be obtained if more agents were taken into account during the simulation, which nonetheless would affect the overall performance of the simulator itself.

With respect to the visited locations frequency we proceeded as follows. For each agent we counted how many locations it visited during the observation interval. We also recorded how many times it visited each single location. We ranked locations based on the above numbers, by associating

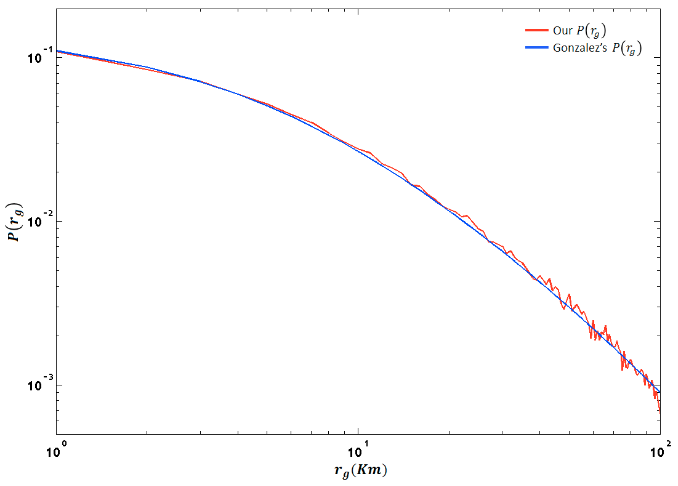


Figure 7.12: The distribution of the radius of gyration

a rank of L to the L^{th} most-visited location. Fig. 7.13 shows the results in a log-log scale with a small linear scale insert on the top right corner of the picture. The plot highlights that the obtained location frequency distribution follows a Zipf law ($P(L) \sim 1/L$), as expected from the Gonzalez et al. distribution. Finally, we conducted one more test using a different dataset of only 25.000 agents in the same area, just to measure the distance between agents' positions at consecutive displacements. Fig. 7.14 shows in a log-log scale our travel distance distribution overlapped with the one showed in [41]. We found out that there is a small gap between the two distributions due to the fact that, differently from the real data in the cited work, we currently use, in our simulator, the same location selection algorithm for each simulated agent.

7.3.4 Conclusions and future work

We defined an integrated architecture for a mobile CDN, by using both a mobility model and a content request model. We implemented in the mobile CDN simulator state-of-the-art proposals with respect to the two above mentioned models: the Gonzalez-Hidalgo-Barabasi model for users mobility and the content request pattern resulting by the Digg platform analysis. As

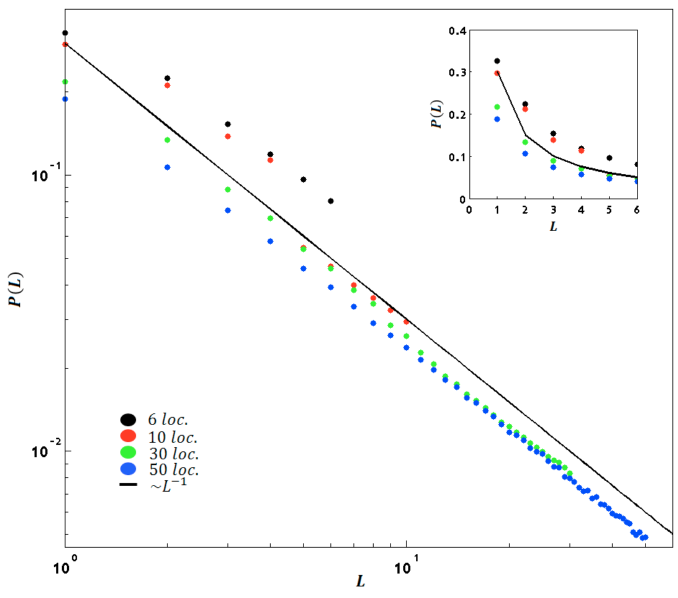


Figure 7.13: The Zipf plot associated with the visited locations frequency

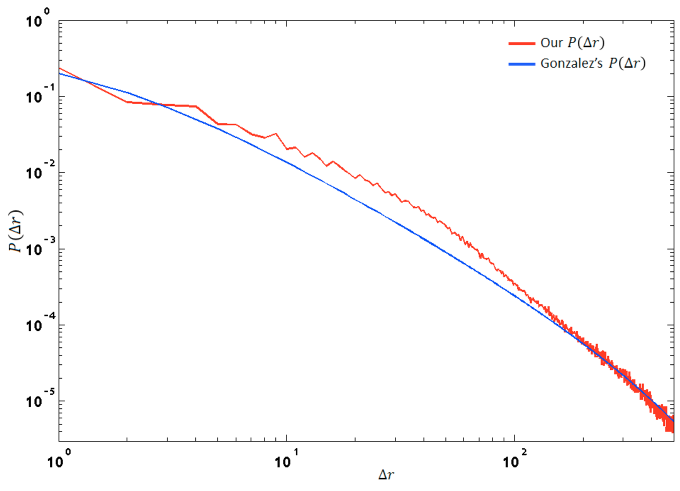


Figure 7.14: Probability density function of travel distances

to the mobility model, we validated our simulator by comparing its results with those published in [41]. Our future work will concern the use of the simulator in order to: (i) evaluate potential real-world deployment scenarios; (ii) show the effectiveness of content placement strategies aimed at moving contents as close as possible to end-users (i.e., in the edge of the network).

Chapter 8

Conclusions and Future Work

In this thesis we present a comprehensive study on Internet architectures devoted to real-time multimedia collaboration and content-aware web objects distribution. Our contribution includes design and implementation tasks, performance evaluation, as well as the definition and application of behavioral analysis to manage security and content distribution issues.

We have designed and developed innovative standard solutions for real-time multimedia communications over the Internet [18]. The thesis has described our active role in the design and implementation of the Meetecho architecture, which is currently one of the most advanced multimedia conferencing platforms based on standard protocols and providing enhanced collaboration functionality [8]. This allowed us to investigate several issues related to the centralized conferencing model, including complex protocol interactions and mechanisms to enhance scalability (e.g., the separation of responsibilities between business logic and media manipulation for CPU-intensive functionality like video processing). We showed in this work how we effectively supported the development of standards within the IETF Real-time Applications and Infrastructure area by providing meaningful insights on the practical issues unveiled by the actual implementation of research ideas. The experience gained in the centralized conferencing control field is currently presented as a best current practice in an Internet standard document [17]. The results we arrived at do represent practical examples of the IETF philosophy according to which the developed standards have to work both on paper and in practice, i.e., they should allow for rapid prototyping and deployment. We are currently bringing the same approach (based both on the definition and on the implementation of standards) to the most recent standard proposals currently under discussion within the international research community, dealing with real-time communication over the web and telepresence systems.

We have conducted the scalability analysis of the Meetecho conferencing platform by leveraging formal methods. In particular, we selected Stochastic Activity Networks as the most suitable modeling formalism for the study of the performance of complex systems. The resulting work represents a successful example of cross fertilization between two extremely active (yet often uncorrelated) research communities: (i) the networking community on one side, which mainly focuses on real-world implementations of the designed architectures, by embracing a pure engineering approach; (ii) the "performability" community on the other side, which usually opts for a more structured approach, whereby the designed architectures are mostly studied through the application of formal methods. Thanks to such cross fertilization, we were able to reach the twofold objective of both validating the suitability of the formal characterization of our real-world implementation of the distributed conferencing scenario and assessing the validity of the measurements performed on the field. By implementing a library of reusable SANs specifically devised to model the behavior of the various components of the conferencing framework, we were able to analyze the main performance figures associated with our current implementation of the conferencing framework through the proper composition of SAN templates. The results of the experiments we conducted clearly demonstrate that the proposed approach can be fruitfully exploited in order to: (i) easily compose and build the complete system model; (ii) analyze in an agile fashion the system's behavior; (iii) simplify the very first phases of the entire life cycle of a real system (i.e., before its actual deployment). The promising results obtained so far, encourage us to keep on investigating the proposed approach, by extending the modeling power through the adoption of multi-formalism techniques. Our aim is to further investigate model templates by moving the focus to the study of advanced functionality of the real system, with special regard to non functional requirements associated with its dependability properties. We have already started to work on the analysis of the system's behavior in the presence of faults (either accidental or due to malicious users' behaviors) and we plan to present the results of this further study as part of the dissemination activities related to our research efforts.

We then explore the application of user behavior profiling for protecting VoIP platforms from social threats [30]. Thanks to an unsupervised clustering approach, VoIP users are partitioned into different groups, each characterized by the compliance with a common behavioral pattern. User social activity is characterized by a set of features, which we have defined with the aim of expressing the user's call frequency, as well as the heterogeneity of her/his contacts, and the duration trend of the calls she/he is involved in. Each group turns out to be represented by a typical model of behavior showing peculiar characteristics. A magnifying glass has been put on behavior patterns that substantially differ from the normal one. Significant exemplars of each cluster have been searched out along the timeline and have been visually inspected by tracing their call activity exploiting, for visualization purposes, the expressive power of social graphs. The feature based model, as it is right now, needs to go through further validation steps, including a reduction process. We'd rather not use feature extraction in this phase, because keeping the semantics of each feature certainly gives more insight about each decision taken by the classifier. We aim to find the minimal set, by looking for a tradeoff between the effectiveness of the model and the computational power required to use it in a real-life scenario. The model will also be tested with more clustering, classification and tuning techniques, in order to prove its robustness.

We finally propose to apply behavioral analysis to the field of content distribution in order to improve the performance of content delivery. We design a behavior-aware content distribution architecture within the context of the BACON (Behavior-Aware Content-Oriented Networking) research project. Behavioral analysis is exploited to drive a placement strategy aimed at putting contents as close as possible to the most probably interested users. In this context, we design and test an approach to find interest-based communities among the users that rate web contents. We also indicate and study further behavioral directions that need to be explored in order to address the new content distribution challenges recently arisen with the boom of mobile and multimedia entertainment traffic: users' mobility and request patterns. We provide for a Content Delivery Network simulation framework able to reproduce realistic user behaviors by simulating the Gonzalez-Hidalgo-Barabasi model for users mobility, as well as the content request pattern resulting from an analysis of the well-known Digg platform. We perform a validation analysis of our simulated mobility model. Our future work in this field will concern the use of such a simulator in order to both evaluate potential real-world deployment scenarios and show the effectiveness of content placement strategies aimed at moving contents as close as possible to end-users (i.e., in the edge of the network).

Bibliography

[1] 3GPP. Conferencing using the IP multimedia (IM)core network (CN) subsystem-ts 24.147. Technical report, March 2008.

[2] W. Aioffi, G. Mateus, J. de Almeida, and A. Loureiro. Dynamic content distribution for mobile enterprise networks. *IEEE Journal on Selected Areas in Communications*, 23(10):2022 – 2031, oct. 2005.

[3] VoIP Security Alliance. Voip security and privacy threat taxonomy, 2005.

[4] Allot. Allot mobiletrends: Global mobile broadband traffic report h2, 2010.

[5] A. Amirante, A. Buono, T. Castaldi, L. Miniero, and S. P. Romano. A framework for distributed conferencing. draft-romano-dcon-framework-05.txt, June 2009.

[6] A. Amirante, A. Buono, T. Castaldi, L. Miniero, and S. P. Romano. Requirements for distributed conferencing. draft-romano-dcon-requirements-05.txt, June 2009.

[7] A. Amirante, A. Buono, T. Castaldi, L. Miniero, and S. P. Romano. Requirements for the xcon-dcon synchronization protocol. draft-romano-dcon-xdsp-reqs-05.txt, June 2009.

[8] A. Amirante, T. Castaldi, L. Miniero, R. Presta, and S. P. Romano. Standard multimedia conferencing in the wild: the Meetecho architecture. *Multimedia Tools and Applications*, 61, 2012.

[9] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano. Improving the scalability of an IMS-compliant conferencing framework through presence and event notification. In *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm), New York City, NY, USA*, July 2007.

[10] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano. Improving the Scalability of an IMS-Compliant Conferencing Framework Part II: Involving Mixing and Floor Control. In *Lecture Notes in Computer Science - IPTComm 2008*, pages 174–195. Springer-Verlag, 2008.

[11] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[12] A. Balamash and M. Krunz. An overview of web caching replacement algorithms. *IEEE Communications Surveys and Tutorials*, 6(1-4):44–56, 2004.

[13] V. Balasubramaniyan, M. Ahamad, and H. Park. Callrank: Combating spit using call duration, social networks and global reputation. In *CEAS*, 2007.

[14] A. Barabasi. *Linked: the New Science of Networks.* Perseus Publishing, 2002.

[15] M. Barnes, C. Boulton, and O. Levin. A framework for centralized conferencing. RFC5239, June 2008.

[16] M. Barnes, C. Boulton, S. P. Romano, and H. Schulzrinne. Centralized conferencing manipulation protocol. RFC6503, March 2012.

[17] M. Barnes, L. Miniero, R. Presta, and S. P. Romano. Centralized Conferencing Manipulation Protocol (CCMP) Call Flow Examples. RFC6504, March 2012.

[18] M. Barnes, L. Miniero, R. Presta, S. P. Romano, and H. Schulzrinne. CCMP: a novel standard protocol for conference management in the XCON framework. In *Proceedings of the 4th International Conference on Principles, Systems and Applications of IP Telecommunications (IPT-Comm), Munich, Germany*, August 2010.

[19] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan. Webrtc 1.0: Real-time communication between browsers. W3C Working Draft 21 August 2012, 2012 (work in progress).

[20] James C. Bezdek and Nikhil R. Pal. Cluster Validation with Generalized Dunn's Indices. In *ANNES '95: Proceedings of the 2nd New Zealand*

Two-Stream International Conference on Artificial Neural Networks and Expert Systems. IEEE Computer Society, 1995.

[21] C. Bo, C. Junliang, and D. Min. Petri net based formal analysis for multimedia conferencing services orchestration. *Expert Systems with Applications*, 39:696–705, January 2012.

[22] A. Bondavalli, P. Lollini, and L. Montecchi. Analysis of user perceived QoS in ubiquitous UMTS environments subject to faults. In *Proceedings of the 6th IFIP WG 10.2 international workshop on Software Technologies for Embedded and Ubiquitous Systems*, SEUS '08, pages 186–197, Berlin, Heidelberg, 2008. Springer-Verlag.

[23] P. Oscar Boykin and Vwani P. Roychowdhury. Leveraging social networks to fight spam. *Computer*, 38(4):61–68, April 2005.

[24] D. Brockmann, L. Hufnagel, and T. Geisel. The scaling laws of human travel, May 2006.

[25] A. Buono, S. Loreto, L. Miniero, and S.P. Romano. A distributed IMS enabled conferencing architecture on top of a standard centralized conferencing framework. *IEEE Communications Magazine*, 45(3):152–159, March 2007.

[26] G. Camarillo and M. A. Garcia-Martin. *"The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds"*. Wiley, iii edition, September 2008.

[27] G. Camarillo, J. Ott, and K. Drage. The binary floor control protocol (bfcp). RFC4582, November 2006.

[28] G. Camarillo, S. Srinivasan, R. Even, and J. Urpalainen. Conference event package data format extension. RFC6502, March 2012.

[29] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 1–14, New York, NY, USA, 2007. ACM.

[30] S. Chiappetta, C. Mazzariello, R. Presta, and S.P. Romano. An anomaly-based approach to the analysis of the social behavior of VoIP users, 2013, Communication Magazine, to appear, http://dx.doi.org/10.1016/j.comnet.2013.02.009.

[31] S. Chiaradonna, F. Di Giandomenico, and P. Lollini. Definition, implementation and application of a model-based framework for analyzing interdependencies in electric power systems. *International Journal of Critical Infrastructure Protection*, 4(1):24 – 40, 2011.

[32] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2010-2015.

[33] T. Courtney, S. Gaonkar, K. Keefe, E. Rozier, and W.H. Sanders. Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models. In *DSN*, pages 353–358, 2009.

[34] R. Dantu and P. Kolan. Detecting spam in voip networks. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, SRUTI'05, pages 5–5. USENIX Association, 2005.

[35] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(2):224–227, February 1979.

[36] P. De Lutiis and D. Lombardo. An innovative way to analyze large isp data for ims security and monitoring. In *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, pages 1 –6, oct. 2009.

[37] M. Duckworth, A. Pepperell, and S. Wenger. Framework for telepresence multi-streams. draft-ietf-clue-framework-08, December 2012 (work in progress).

[38] R. T. Fielding. Architectural styles and the design of network-based software architectures. Technical report, 2000.

[39] G. Franceschinis, M. Gribaudo, M. Iacono, S. Marrone, F. Moscato, and V. Vittorini. Interfaces and binding in component based development of formal models. In *Proceedings of the 4th International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09, pages 44:1–44:10. ICST, 2009.

[40] E. Fusella. Design and implementation of a tool for the optimization of CDN performance, MSc thesis, a.a. 2010/2011.

[41] M. Gonzalez, C. Hidalgo, and A. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[42] C. Groves, W. Yang, and R. Even. Clue media capture description. draft-groves-clue-capture-attr-00, September 2012 (work in progress).

[43] M. Gudgin, N. Mendelsohn, M. Hadley, J. Moreau, and H. Nielsen. Soap version 1.2 part 1: Messaging framework. World wide web consortium first edition rec-soap12-part1-20030624, W3C, June 2003.

[44] A.P. Guimarães, P.R.M. Maciel, and R. Matias. Quantitative analysis of dependability and performability in voice and data networks. In *Advances in Networks and Communications*, volume 132 of *Communications in Computer and Information Science*, pages 302–312. Springer Berlin Heidelberg, 2011.

[45] P. Gupta and V. Shmatikov. Security analysis of voice-over-ip protocols. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 49–63, Washington, DC, USA, 2007. IEEE Computer Society.

[46] J. Hautakorpi and G. Camarillo. The Session Description Protocol (SDP) Content Attribute. RFC 4796 (Proposed Standard), February 2007.

[47] C. Holmberg and H Alvestrand. Multiplexing negotiation using session description protocol (sdp) port numbers. draft-ietf-mmusic-sdp-bundle-negotiation-02, February 2013 (work in progress).

[48] W. Huaxu, S. GuiPing, and D. Yanlan. *SIP Modeling and Simulation*, pages 397–431. CRC Press, 2008.

[49] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[50] Angelos D. Keromytis. A comprehensive survey of voice over ip security research. *IEEE Communications Surveys and Tutorials*, 14(2):514–537, 2012.

[51] H. Kim, M. J. Kim, Y. Kim, and H. C. Jeong. DEVS-Based modeling of VoIP spam callers' behavior for SPIT level calculation. *Simulation Modelling Practice and Theory*, 17(4):569–584, April 2009.

[52] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, August 2007.

[53] P. Kyzivat. Clue signaling. draft-kyzivat-clue-signaling-02, February 2013 (work in progress).

[54] S. Loreto and S.P. Romano. Real-time communications in the web: Issues, achievements, and ongoing standardization efforts. *Internet Computing, IEEE*, 16(5):68 –73, October 2012.

[55] A. Luo, C. Lin, K. Wang, L. Lei, and C. Liu. Quality of protection analysis and performance modeling in IP multimedia subsystem. *Comput. Commun.*, 32:1336–1345, July 2009.

[56] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[57] Stefano Marrone, Nicola Mazzocca, Roberto Nardone, Roberta Presta, Simon Pietro Romano, and Valeria Vittorini. A SAN-based modeling approach to performance evaluation of an IMS-compliant conferencing framework. *T. Petri Nets and Other Models of Concurrency*, 6:308–333, 2012.

[58] C. Mazzariello, P. De Lutiis, and D. Lombardo. Clustering NGN user behavior for anomaly detection. *Information Security Technical Report*, 16(1):20–28, 2011.

[59] S. McGlashan, T. Melanchuk, and C. Boulton. An interactive voice response (ivr) control package for the media control channel framework. RFC6231, May 2011.

[60] S. McGlashan, T. Melanchuk, and C. Boulton. A mixer control package for the media control channel framework. RFC6505, March 2012.

[61] T. Melanchuk. An architectural framework for media server control. RFC5567, June 2009.

[62] Kevin Mitnick. *Ghost in the Wires: My Adventures as the World's Most Wanted Hacker*. Hachette Book Group, 2011.

[63] F. Moscato, V. Vittorini, F. Amato, A. Mazzeo, and N. Mazzocca. Solution workflows for model-based analysis of complex systems. *IEEE T. Automation Science and Engineering*, 9(1):83–95, 2012.

[64] O. Novo, G. Camarillo, D. Morgan, and J. Urpalainen. Conference information data model for centralized conferencing. RFC6501, March 2012.

[65] P. Owezarski and M. Boyer. *Modeling of Multimedia Architectures: The Case of Videoconferencing with Guaranteed Quality of Service*, pages 501–525. ISTE, 2010.

[66] Andrea Passarella. A survey on content-centric technologies for the current internet: Cdn and p2p solutions. *Computer Communications*, 35(1):1–32, 2012.

[67] R. Presta and S. P. Romano. An XML schema for the CLUE data model. draft-presta-clue-data-model-schema-03, March 2013 (work in progress).

[68] C. Pörschmann and H. Knospe. Analysis of spectral parameters of audio signals for the identification of spam over ip telephony. In *CEAS*, 2008.

[69] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald. Detecting spit calls by checking human communication patterns. In *ICC*, pages 1979–1984, 2007.

[70] G. Ramos-Fernández, J. Mateos, O. Miramontes, G. Cocho, H. Larralde, and B. Ayala-Orozco. Lévy walk patterns in the foraging movements of spider monkeys (Ateles geoffroyi). *Behavioral Ecology and Sociobiology*, 55(3):223–230, January 2004.

[71] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong. On the levy-walk nature of human mobility. *IEEE/ACM Trans. Netw.*, 19(3):630–643, June 2011.

[72] J. Rosenberg. A Framework for Conferencing with the Session Initiation Protocol (SIP). RFC4353, February 2006.

[73] J. Rosenberg, H. Schulzrinne, G. Camarillo, et al. Sip: Session initiation protocol. RFC3261, June 2002.

[74] J. Rosenberg, H. Schulzrinne, and O. Levin. A Session Initiation Protocol (SIP) Event Package for Conference State. RFC4575, August 2006.

[75] Sandvine. Global internet phenomena report - fall 2011.

[76] M. Saxena, U. Sharan, and S. Fahmy. Analyzing video services in web 2.0: a global perspective. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, pages 39–44, New York, NY, USA, 2008. ACM.

[77] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. WWW '11, pages 457–466, New York, NY, USA, 2011. ACM.

[78] R. Shankesi, M. AlTurki, R. Sasse, C.A. Gunter, and J. Meseguer. Model-checking DoS amplification for VoIP session initiation. In *Proceedings of the 14th European conference on Research in computer security*, ESORICS'09, pages 390–405, Berlin, Heidelberg, 2009. Springer-Verlag.

[79] D. W. Sims, E. J. Southall, N. E. Humphries, G. C. Hays, C. J. A. Bradshaw, J. W. Pitchford, A. James, M. Z. Ahmed, A. S. Brierley, M. A. Hindell, D. Morritt, M. K. Musyl, D. Righton, E. L. C. Shepard, V. J. Wearmouth, R. P. Wilson, M. J. Witt, and J. D. Metcalfe. Scaling laws of marine predator search behaviour. *Nature*, 451(7182):1098–1102, February 2008.

[80] F. Spitzer. *Principles of Random Walk*. 2001.

[81] B. Tariq, M. Mukarram, R. Jain, and T. Kawahara. Web content caching and distribution. chapter Mobility aware server selection for mobile streaming multimedia content distribution networks, pages 1–18. Kluwer Academic Publishers, 2004.

[82] K. Tiassou, K. Kanoun, M. Kaâniche, C. Seguin, and C. Papadopoulos. Modeling aircraft operational reliability. In *Proceedings of the 30th international conference on Computer safety, reliability, and security*, SAFECOMP'11, pages 157–170, Berlin, Heidelberg, 2011. Springer-Verlag.

[83] P. Truchly, M. Golha, F. Tomas, G. Radoslav, and M. Legen. Simulation of IMS using current simulators. In *ELMAR, 2008. 50th International Symposium*, volume 2, pages 545 –548, sept. 2008.

[84] J. Uberti and C. Jennings. Javascript session establishment protocol. draft-ietf-rtcweb-jsep-02, February 2012 (work in progress).

[85] G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev, E. J. Murphy, P. A. Prince, and H. E. Stanley. Lévy flight search patterns of wandering albatrosses. *Nature*, 381(6581):413–415, May 1996.

[86] V. Vittorini, M. Iacono, N. Mazzocca, and G. Franceschinis. The OsMoSys approach to multi-formalism modeling of systems. *Software and System Modeling*, 3(1):68–81, 2004.

[87] C. Wallenta, M. Ahmed, I. Brown, S. Hailes, and F. Huici. Analysing and modelling traffic of systems with highly dynamic user generated content. Technical report, 2008.

[88] P. Wang, M. González, C. Hidalgo, and A. Barabási. Understanding the spreading patterns of mobile phone viruses. *CoRR*, abs/0906.4567, 2009.

[89] Y. Wu, S. Bagchi, N. Singh, and R. Wita. Spam detection in voice-over-ip calls through semi-supervised clustering. In *DSN*, pages 307–316, 2009.

[90] D. Yon and G. Camarillo. Tcp-based media transport in the session description protocol (sdp). RFC4145, September 2005.

[91] T. Yoshimura, Y. Yonemoto, T. Ohya, M. Etoh, and S. Wee. Mobile streaming media cdn enabled by dynamic smil. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 651–661. ACM, 2002.

[92] W. J. Zhang, Q. Li, Z. M. Bi, and X. F. Zha. A generic Petri net model for flexible manufacturing systems and its use for FMS control software testing. *International Journal of Production Research*, 38(50):1109–1131, 2000.