Extended Version of Paper Accepted for Publication in IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2007), 13-15 June, Bologna, Italy

Call Management Policy Specification for the Asterisk Telephone Private Branch Exchange

G. Konstantoulakis[1], M. Sloman[2]
[1]*Morgan Stanley, London*
[2]*Department of Computing, Imperial College London 180 Queen's Gate, London SW2 2RH.*
gkonstantoulakis@gmail.com, m.sloman@imperial.ac.uk

Abstract

A VoIP PBX supports flexible call handling functionality for selective forwarding, cost based outward call routing, recording calls etc. Both users and administrators need a flexible and easy to use means of specifying call management policies which can take advantage of the features supported by these PBXs. This paper presents a case study of a system for specifying and implementing policies for the Asterisk VoIP PBX. The system is able to detect and resolve conflicts in a user's policies. The evaluation results of a prototype implemented in an enterprise PBX for both administrators and users indicates performance was reasonable, even without any optimization and users were easily able to specify call management policies.

1. Introduction

Voice over IP (VoIP) [11] is becoming an important means of supporting telephony to the desktop by integrating telephone headsets with desktop PCs and PDAs or using wired or wireless VoIP handsets via an internet port. The key reasons for VoIP's success are that it provides lower cost and increased functionality. VoIP calls can be free or cost only a small fraction of typical telecommunications service provider charges. VoIP telephony can easily provide a rich set of features such as cost based routing, selective call forwarding or rejection, call recording etc. at very low cost for businesses. It is possible to manage calls depending on a user's "context" – location, activity or time of the day.

Readily available open source, public domain software supporting VoIP private branch exchange (PBX) functionality on standard PCs has made VoIP telephony attractive for small businesses. The Asterisk Software IP-PBX [1] has been deployed across numerous enterprise installations. Although Asterisk is free and readily available, it is not easy to manage, configure and maintain. A full time administrator, skilled in VoIP technology, is often needed, especially when the users require frequent changes in the call management of their extensions.

Users want to be able to easily define policies relating to context-based handling of calls. For example:

- When I am busy, calls from customers should be forwarded to a team member, calls from colleagues should be informed of my current schedule, and calls from friends should be passed to voicemail.
- During office hours, when I am out of the office, all incoming calls should be forwarded to both my mobile and to the company's other office across town. Whichever phone answers will accept the call.

Administrators may also want to define policies relating to which types of communication links should be used in order to minimize costs, for example:

- Outgoing calls to mobile numbers are first told that the call will be limited to 5 minutes and after that time they are told that the call has been terminated
- Only staff in the research and sales department are permitted to dial international numbers, but they are told that the call should be limited to no more than 10 minutes.
- All calls to the cross town office must use the Session Initiation Protocol (SIP) over the datalink connecting the offices.

Although the Asterix software is capable of supporting policies such as those above, they are very complex and difficult to set up and it would not be practical for non-technical users to specify their own policies. For this reason an easy to use interface for both non-technical users and administrators is needed that facilitates the specification of flexible call management policies. Users typically want to selectively deal with incoming calls based on who is calling, where the call is coming from (internal, external or overseas), time of day, whether the user is in a meeting or already busy on the phone. Typical actions for a call could include playing a message, forwarding to one or more other numbers, forwarding to voicemail or generating a busy tone to reject the call. Administrators want to be able to route calls to specific links as well as permit or deny access to specific types of calls for specific users. They need to be able to define what statistics the system maintains about calls and possibly even record particular types of incoming or outgoing calls.

The paper describe a web-based user interface for specifying both user and administrator policies for an Asterisk VoIP PBX as well as the underlying implementation architecture. In section 2 we give an overview of the Asterisk software, while section 3 describes the architecture of the implemented prototype. Section 4 gives examples of how users and administrators actually specify policies via the web-based interface. In section 5 we discuss how conflicts between policies are detected and some possible resolution strategies. Section 6 evaluates the deployed system in terms of performance and usability. Section 7 discusses related work followed by the conclusions in section 8.

2. Asterisk

2.1. Asterisk Architecture

Asterisk is a public-domain, complete software PBX system written in the C Programming language [1]. It runs on Linux, BSD, Windows and OS X, supports Voice over IP using all the major protocols, and can interoperate with almost all standards-based telephony equipment using relatively inexpensive hardware. Asterisk is open source software under the GPL licence. Specific APIs are defined around a central PBX core system which handles the internal interconnection of the PBX, cleanly abstracting the specific protocols, codecs, and hardware interfaces from the telephony applications which are all defined as loadable APIs.

There are 2 main VoIP protocols, both supported by Asterisk. Session Initiation Protocol (SIP) [9] is the Internet Engineering Task Force (IETF) application-layer control (signalling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. H323 [7] is the International Tele-communications Union (ITU) VoIP protocol that is needed for interworking with the public switched telephone network e.g. via ISDN lines. It is more powerful but more complex than SIP.

Figure 1 shows a typical Asterisk PBX within an organisation, with both internet and ISDN connections. Asterisk has a simple embedded database, within the PBX machine, used to store name-value pairs and supports the implementation of simple scripts for the storage of runtime values such as last number dialed etc. In addition, Asterisk connects with an external database through an ODBC interface to log call detail records (CDR's), store configuration files such as the SIP configuration, the voicemail configuration and others. Asterisk typically interacts with the external database in real-time while handling calls. For example, it only loads the general section of the SIP configuration with the protocol attributes into memory, and every time it needs user account information (e.g. for authentication) it queries the database. This allows dynamic creation of user accounts while at the same time adding little or no latency to the protocol response times. Moreover it allows flexibility in the search and handling of user information.

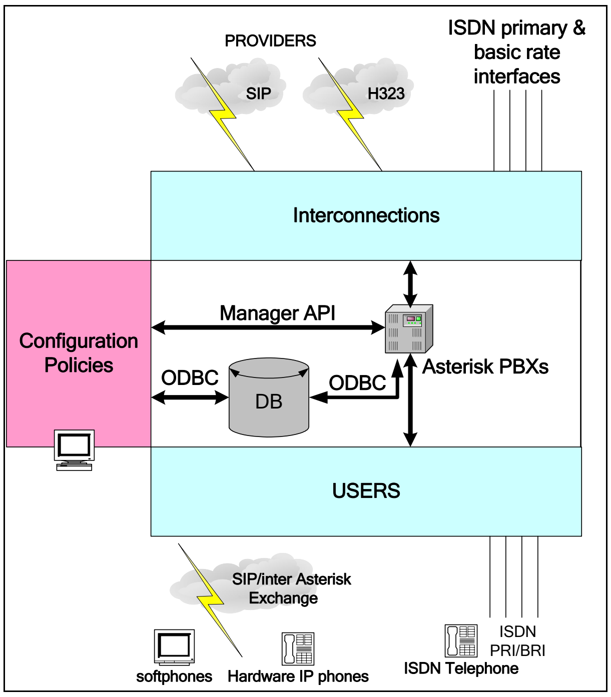


Figure 1 Typical Asterisk PBX

Typical features supported by Asterisk include:

Call Forwarding: redirection of an incoming/outgoing call to another number when the number matches a particular pattern or based on the current time.

Call Monitoring: call details logged in CDRs.

Call Recording: Record calls on demand

Call Transfer: to another number during the conversation.

Call Pickup: Pickup a call that is ringing to another number.

Call Waiting: Allows the use of a hook-flash to change between two simultaneous calls.

Call Blocking: Rejecting calls from particular users.

Calling ID: The calling party's ID can be shown, hidden or modified to be more meaningful to the user.

PBX Direct Inward System Access (DISA): Allows an outside caller to have full access to PBX functions, which is normally reserved for internal-only access. Common use of DISA is to enable mobile users or home-based workers to call into your PBX and make outbound long distance calls as though from work.

Do Not Disturb (DND): sends incoming calls to voicemail.

Database Store / Retrieve of data.

Distinctive Ring: The called phone has a distinct ring based on the identity of the originator eg., a specific number, internal or external calls.

Hunt Groups: Grouping extensions which ring in a specific order so that when an extension does not answer, the call is forwarded to the next in extension in the list.

Call groups: Grouping extensions so that a call to any extension rings all and any one can able take the call.

HTTP connectivity: Can access a web or remote database during call processing e.g. to obtain more information relating to the caller ID.

Queues and Agents: Allows the creation of queues of calls waiting to be handled with assignment of head of the queue to an idle agent. Music and announcements can be played for calls waiting in the queue.

Speed Dial: Assigning a short number to dial an external call e.g. pressing 1 dials 0777208366.

SMTP connectivity: Can send email to notify a user/administrator of specific events, e.g. someone is making a long distance call of over an hour.

N Way Calling: for conference calls.

Music on Hold: Allows user to listen to music (mp3, wav etc.) while waiting

Voice Mail: Asterisk's voicemail can send an email to the voicemail recipient, and can optionally attach a WAV file of the entire message.

2.2. Asterisk Configuration Files

The Asterisk configuration file extensions.conf controls how incoming and outgoing calls are handled and routed and so can be considered the implementation level for call handling policy. It is divided into 3 sections: [general] contains a few general settings, [globals] defines global variables (or constants) and their initial values and [dialplan] consists of contexts defining a set of actions relating to an extension. When Asterisk receives a call from an internal or external extension, it is matched with a context, depending on what channel the call came in on. Dial-plan extensions can be simple numbers, alphanumerical values or patterns that group more than one extension. Unlike a traditional PBX, where extensions are associated with phones, interfaces, menus etc, in Asterisk an extension is defined as a list of commands to execute. The commands are generally executed in the order defined by their priority tag, but some commands, such as the Dial command, have the ability to redirect somewhere else, based on some condition (the Dial result – Busy, Not Answering, etc). Execution steps for an extension are specified using the Asterisk Configuration Language (ACL) as follows:

```
exten => extension pattern,priority, Command(parameters)
```

When an extension pattern is matched, the command tagged with a priority of 1 is executed, followed by command priority 2, and so on until: the call is hung-up, a command returns a result code of -1 (indicating failure), a command with the next higher priority does not exist (note: Asterisk will not "skip over" missing priorities), or the call is routed to a new extension. Asterisk uses some extension names for special purposes:

i: Invalid: incoming extension matches no pattern
s: Start: Starting extension
h: Hangup: actions to be executed when call hangs up
t: Timeout: timeout after call starts ringing
T: AbsoluteTimeout: timeout since call initiation

A simple example of a context named **playNvoicemail** is shown below. The called number is 123. (In this example we do not differentiate on who the caller is). The caller will hear a message and then be forwarded to 123's voicemail.

```
[playNvoicemail]
 exten => 123,1,Answer()
 exten => 123,2,Playback(myMessage)
 exten => 123,3,Voicemail(123)
 exten => 123,4,Hangup()
```

2.3. Context awareness

It is very useful to be able to specify context aware policies with respect to a user's current state, availability or time of day. For example, a user may wish to define a context of "Relaxing" so that all incoming calls after 19.00 from the group work-colleagues are sent to voicemail. To increase flexibility and support the reuse of contexts we provided the facility to group them and establish a hierarchy with general contexts applying to many users, but deriving specific contexts from these by changing some of the parameters.

ACL variables can be used to define a context with many parameters e.g. relating to caller-id, time of day, if the call was forwarded, call duration, call type (local, long distance), call outcome(answered, failed, busy) etc. Moreover Asterisk has a large list of actions (see features in 2.1) that can be associated with these parameters. Finally Asterisk allows the inclusion of sub-contexts in a context thus allowing a notion of hierarchy or inheritance.

2.4. User-defined Actions

Asterisk can cater for user-defined actions in two ways. The first would be by examining dial-plan variables that hold the outcome of an action (Asterisk application result), and depending on that result, execute the appropriate actions. An example is to notify the user by email that his home number has exceeded a 30 min. limit for mobile calls (maybe his daughter is talking to her friends) and subsequently disallow any calls to mobiles of more than 2 min. Administration actions can be implemented using the manager interface and/or database connection and poll Asterisk at fixed intervals to read variables. This would allow administrators to define policies for groups of users more easily. For example, assume a company has two separate SIP accounts with different SIP providers for outgoing calls. The administrator can define a policy that queries the Asterisk Database every 2 min. and if the number of failed calls exceeds a certain threshold Asterisk should switch from using one provider to the other as shown below in the context called SipOutgoingCallsCheck.

```
[Globals]
SipOUT = SipFirst
FailedCalls = 0
…………………………..
[SipOutgoingCallsCheck]
exten => _X,1,Dial(${EXTEN}@${SipOUT})
;;; After the call check the result in the
;;; DIALSTATUS variable. If it is a failed call
;;; go-to step 101 otherwise exit
exten => _X, 2, GotoIf
    ($[${DIALSTATUS} = CONGESTION ]? 101:200)

;;; FAILED CALL
exten => _X,101,SetVar(FailedCalls = ${FailedCalls} +1)
;;; If failed calls are more than 10, switch providers
exten => _X,102, GotoIf($[${FailedCalls } = 10 ]? 103:200)
exten => _X,103, SetVar(SipOUT = SipSecond)
exten => _X,104, SetVar(FailedCalls = 0)
exten => _X, 105, Hangup()

;;;; OK EXIT
exten => _X, 200, Hangup()

This would require a database query to determine congestions
 from the call data records (CDR)
SELECT COUNT(NUM)
FROM CDR
WHERE DIALSTATUS = 'CONGESTION';
```

Defining policies using the ACL as indicated above is not very easy for ordinary users or administrators, so we defined an easy to use graphical interface which still provided many of the features expected of a Policy language but was specifically tailored to call handling.

3. ACME System Architecture

The Asterisk Configuration Management Engine (ACME) is the implementation which supports web-based policy specification and storage within a database. It retrieves and evaluates the relevant policies for incoming and outgoing calls. It provides the interface for both users and managers to interact with Asterisk.

3.1. ACME roles

ACME supports the following user roles.

Administrator may view the internals of the system, monitor it and initiate actions that need knowledge of telephony and IT.

Operator is authorized to perform a subset of the administrative actions requiring less technical knowledge such as create /delete users, assign services etc. This enables Administrators to *delegate* less skilled tasks to operators.

User may use the services assigned to him by an Administrator or an Operator

A user group attribute defines groups of users (regardless of the role) that may be assigned different Quality of Service (QoS) and different global policies by the administrator. Examples groups could include Executive, Employees, Customer-support etc. with a global policy that all incoming calls to Customer-support users are recorded.

3.2. ACME Components

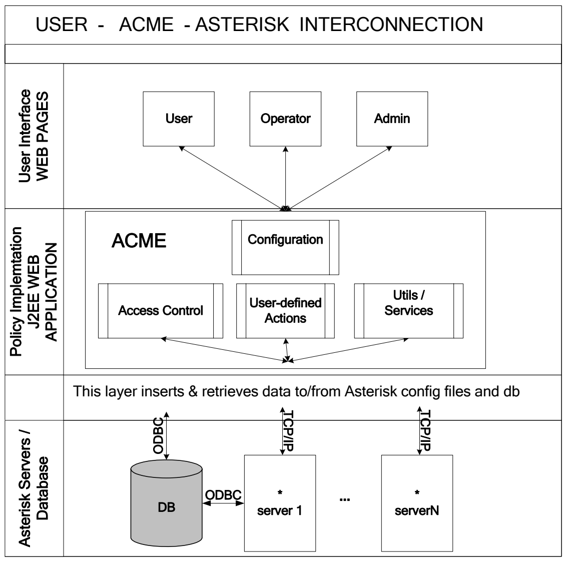

USER - ACME - ASTERISK INTERCONNECTION
User Interface WEB PAGES
User
Operator
Admin
Policy Implemtation J2EE WEB APPLICATION
ACME
Configuration
Access Control
User-defined Actions
Utils / Services
This layer inserts & retrieves data to/from Asterisk config files and db
Asterisk Servers / Database
ODBC
TCP/IP
TCP/IP
DB
ODBC
* server 1
...
* serverN

Figure 2 ACME Architecture

ACME is has the 3 level architecture shown in Figure 2. Administrators can either interact directly with Asterisk servers via TCP/IP through the Asterisk management interface or indirectly, by configuring the Asterisk database. ACME had to both provide a user-friendly flexible interface to define policies as well as enforcing these policies by interacting in real-time with Asterisk servers to handle call-control with minimal response times. To that end ACME has been designed as a modular architecture of interconnected modules that undertake specific tasks:

The **configuration module** is responsible for manipulating data received via the Web-based user interface and storing it in the appropriate DB tables. Moreover it is responsible for updating the corresponding text-based configuration files of Asterisk servers when necessary.

The **access control module** checks the access rights of users and provide the appropriate interfaces for them, as well as handling the call control intelligent routing in real-time and checking the access rights of users and allowing or disallowing the appropriate actions.

The **obligation and alarms** module monitors the system and reports pre-defined aspects of the system to the appropriate users either in real-time via Web consoles or by initiating notifications via email or phone calls.

The **user-defined actions module** enables the user to extend the system functionality with new actions. Typical services provided include a CDR-parser enabling users to easily browse all incoming / outgoing calls; a Web Dialer, to enable users to place calls to destinations retrieved from Microsoft Outlook entries, Excel spreadsheets etc

3.3. Outgoing call handling

The most critical aspects of ACME was to support flexible and efficient call handling using two contexts created for every user – <username>_OUT and <username>_IN in exten.conf (stored in the DB). These define the set of outgoing and incoming ACL call handling policies.

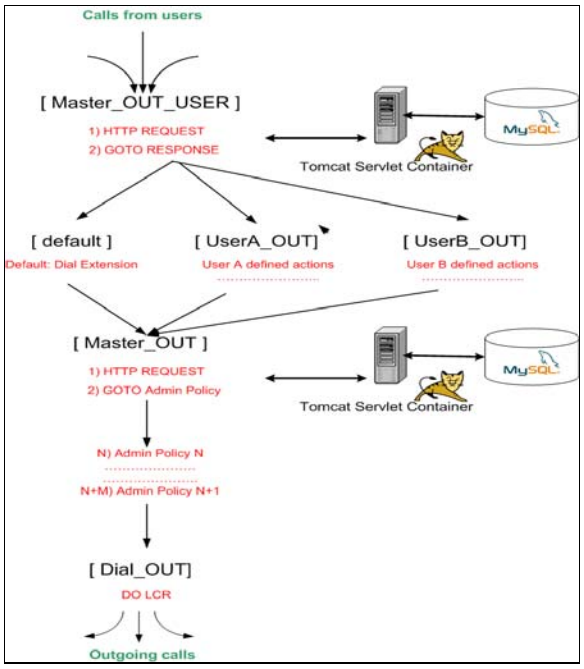


Figure 3 Outgoing Call Handling

Figure 3 shows the steps involved in handling outgoing calls. All extensions regardless of type are configured to first execute the Master_OUT_USER context to enable common handling of all calls independent of the underlying communication protocol (SIP, H.323 etc.) but rather based on user criteria matching such as time, destination or caller-Id. This context makes an HTTP request to ACME submitting caller-Id and destination. ACME identifies the User and Extension based on the caller-Id and retrieves the user's outgoing policies for that Extension if any. The policies are examined to determine if any match the user-defined criteria, such as destination or time context then:

1. If ACME finds a matching policy it responds by manipulating the appropriate data for Asterisk to execute an internal GOTO to another context. The data is set so that:
   - Context is set to <username>_OUT
   - exten attribute is set to <extension_internal_number>_<policy_name>
   - priority is set to 1

   If no matching policy is found, the data for the GOTO takes the following form
   - Context is set to default
   - exten is set to the dialled number
   - priority is set to 1

2. If the call has been routed to a user-defined policy it sequentially executes the Actions defined by the user, until it reaches a DIAL, a Hang-up action or the defined actions finish. In the first case it executes a "Local" dial, which effectively is another GOTO to the Master_OUT context with the destination number (possibly) changed. We use this local dial instead of a GOTO because the user may wish to change the dialled number or simultaneously call more than one number. The local dial creates new threads of execution thus allowing the user to dial out multiple numbers at the same time. Therefore the execution of all these outgoing calls continue independently and each executes the following actions as an independent outgoing call. If the call reaches a Hang-up action then the process is terminated. If the defined actions finish without a dial (the user may not want to change the destination number) the system has a predefined "Local" dial to Master_OUT with the original dialled number. This is also the case of the default context for cases where there are not user-defined outgoing call handling policies

The Master_OUT context implements administrator policies after the user defined policies and therefore take precedence over the user-defined policies. It queries the ACME DB with caller-Id and destination (which may have been altered by a user policy). ACME identifies the User and Extension based on the caller-Id, usergroup to which the caller belongs, and the type of outgoing call (local, international etc.) ACME uses this information to check if there are any matching outgoing administrative policies that apply (also taking under consideration the time-context). If one is found ACME manipulates the response data accordingly and sets:

- context is set to Master_OUT
- exten attribute to <admin_policy_name>
- priority is set to 1

The call executes all the actions of the Administrative policy until it reaches a Dial, Hang-up or the actions end. In the first case again the Dial is a “Local dial” to the “Dial-OUT” context where the actual call is made. Hang-up may terminate the call if the user does not have the rights to place international calls. If the admin policy actions finish without a dial the default case is to execute a local dial to the context “Dial-OUT” which executes the Least Cost routing (LCR) policies in which the administrator has specified the appropriate trunks for each type of outgoing call. These trunks are tried sequentially and if a trunk is congested it automatically tries the next one. If all attempts to place the call fail the interaction is terminated. The big advantage of the separation of administration policies and LCR is that the Administrative actions are executed regardless of the underlying protocol. It is the responsibility of this context to place the call on the outgoing trunk with the protocol defined by the system to correspond to the Trunk type (ISDN, SIP, etc).

3.4. Incoming Call Handling

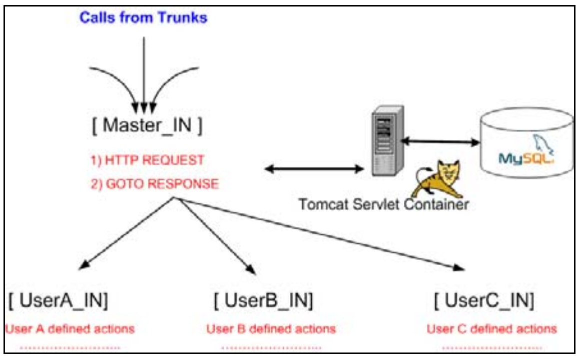


Figure 4 Incoming Call Handling

Figure 4 shows the various contexts related to handling incoming calls. All trunks regardless of their type evaluate the Master_IN context which queries ACME with caller-Id and destination. ACME identifies the User and Extension based on the destination which is a unique external number, and retrieves the user’s incoming policies for that Extension if any. It checks if the call matches any policy criteria, such as destination, time context and Extensions status i.e. BUSY, NOT ANSWERING, etc. The ALWAYS option denotes that the specified policy will be executed regardless of the extension’s status.

If ACME finds a matching policy it responds by manipulating the appropriate data for Asterisk to execute an internal GOTO to another context. The data is set so that:

- Context is set to <username>_IN
- exten attribute to <extension_internal_number>_<policy_name>
- priority is set to 1

If no matching policy is found the data for the GOTO takes the following form

- Context is set to <username>_IN
- exten is set to the internal number
- priority is set to 1

If the call has been routed to a user-defined policy, it sequentially executes the actions defined by the user depending on the status criterion. This means that in the case the status criterion has been set to BUSY or NO-ANSWER, Asterisk will first dial the extension using the proper protocol defined by the system corresponding to the Extensions type (ISDN, SIP, etc), and if it is busy/not answering it will only then start executing the actions for this policy. So in the case the status criterion is matched the user-defined actions are executed sequentially until a DIAL or a Hang-up action is reached or the defined actions finish. For DIAL, a “Local dial” is executed to the Master_OUT context to place the outgoing call as described previously. If the call reaches a Hang-up action then the process is terminated. If the defined actions finish without a dial (the user does not want to change the destination) the system has a predefined dial using the proper protocol as above.

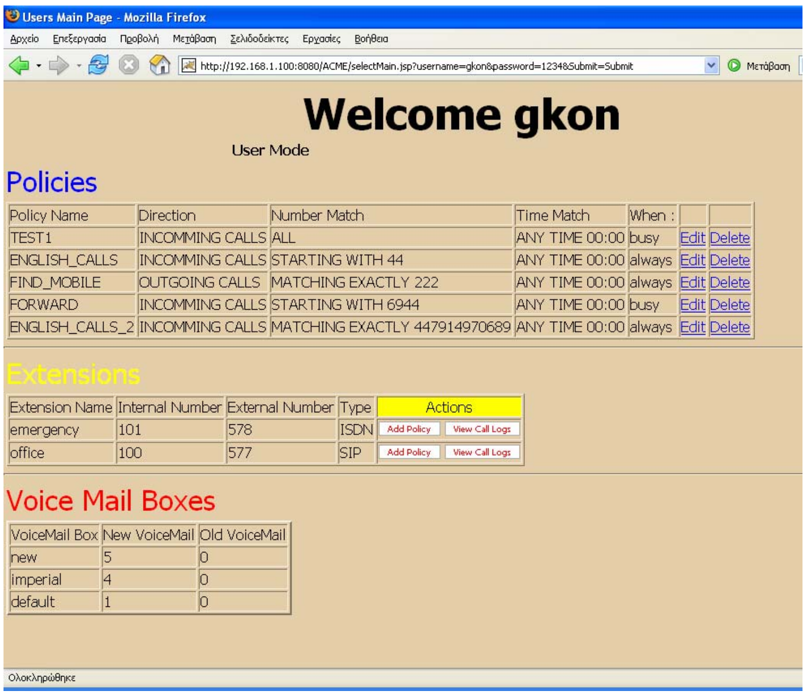
4. User Interface for Policy Specification

In this section we describe how the graphical interface can be used to define some policies. In Figure 5 the user’s main page shows he has been assigned 2 extensions and 3 VoiceMail boxes with their status, as well as various policies that he has already specified. From this main menu the user has the option to manage existing policies (edit/delete) and specify new ones for any of the extensions he owns. Moreover he may view the call detail records (CDRs) of the extensions he owns.

When specifying a policy a user must enter the trigger i.e. the direction of the call to that extension, the peer number expression (caller or called number starting with, matching exactly …), the time context and the status of the phone (busy, not answering, always).

Example Policy

During office hours (before 17.00 on week days) when my phone is not answering and the caller number is a mobile (starts with 6944 for Greece) I want to forward the call to my cell (6944564175) and to the company’s other office number (8077503). Whichever phone answers will take the call. Moreover I want the caller-id to be set to 100 so that I understand that it is a call forwarded by this policy, limit the call to 180 seconds (don’t want to pay too much). Finally I want to record the call in case it is important


Users Main Page - Mozilla Firefox
Αρχείο Επεξεργασία Προβολή Μετάβαση Σελιδοδείκτες Εργασίες Βοήθεια
Μετάβαση
Ολοκληρώθηκε

http://192.168.1.100:8080/ACME/selectMain.jsp?username=gkon&password=1234&Submit=Submit

Welcome gkon

User Mode

Policies

Policy Name	Direction	Number Match	Time Match	When :		
TEST1	INCOMMING CALLS	ALL	ANY TIME 00:00	busy	Edit	Delete
ENGLISH_CALLS	INCOMMING CALLS	STARTING WITH 44	ANY TIME 00:00	always	Edit	Delete
FIND_MOBILE	OUTGOING CALLS	MATCHING EXACTLY 222	ANY TIME 00:00	always	Edit	Delete
FORWARD	INCOMMING CALLS	STARTING WITH 6944	ANY TIME 00:00	busy	Edit	Delete
ENGLISH_CALLS_2	INCOMMING CALLS	MATCHING EXACTLY 447914970689	ANY TIME 00:00	always	Edit	Delete

Extensions

Extension Name	Internal Number	External Number	Type	Actions	
emergency	101	578	ISDN	Add Policy	View Call Logs
office	100	577	SIP	Add Policy	View Call Logs

Voice Mail Boxes

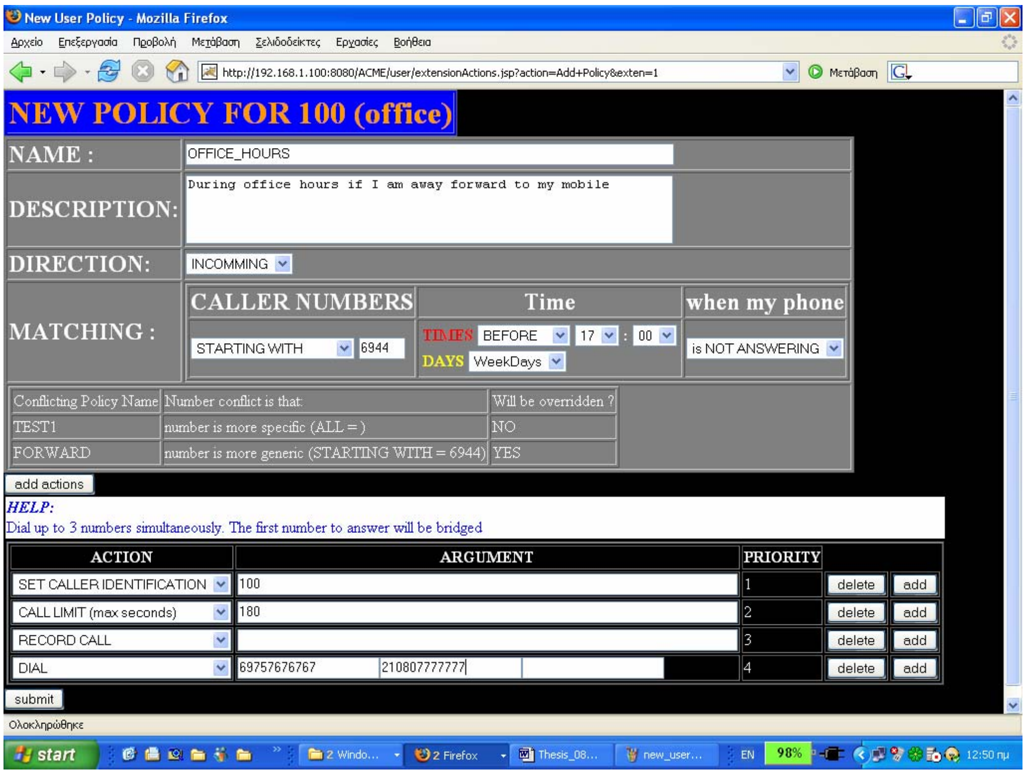
VoiceMail Box	New VoiceMail	Old VoiceMail
new	5	0
imperial	4	0
default	1	0

Figure 5 Users Main Page

Figure 6 shows how ACME users can easily specify such complex policies. First the triggering criteria is set to incoming, the caller number expression to “STARTING WITH 6944”, the time context to “BEFORE 17.00 on WEEKDAYS” and the phone status match to not answering. The add actions button is used to specify the desired action i.e. “SET CALLER IDENTIFICATION to 100”, CALL LIMIT duration to 180 seconds, RECORD THE CALL, and finally forward (DIAL) the call to the desired numbers. By clicking submit the policy is registered and in the main user menu we see a new entry.

As well as specifying policies, users can also view their CDRs showing destination, which policies were used, start time, duration and whether answered. Various sort criteria can be used for viewing the CDRs

The Administrator typically specifies global policies for user groups, usually applying to outgoing calls e.g. the default user group is not permitted to place international calls. In Figure 7, we can see a policy specifying that the users of the default group may place mobile calls up to 5 minutes (300 seconds) duration. If the users try to place an international call they must first be told that they cannot place such calls


New User Policy - Mozilla Firefox
Αρχείο Επεξεργασία Προβολή Μετάβαση Σελιδοδείκτες Εργασίες Βοήθεια
Μετάβαση
Ολοκληρώθηκε
start
2 Windo...
2 Firefox
Thesis_08...
new_user...
EN
98%
12:50 μμ

http://192.168.1.100:8080/ACME/user/extensionActions.jsp?action=Add+Policy&exten=1

NEW POLICY FOR 100 (office)

NAME :	OFFICE_HOURS
DESCRIPTION:	During office hours if I am away forward to my mobile
DIRECTION:	INCOMMING

MATCHING :

CALLER NUMBERS	Time	when my phone
STARTING WITH 6944	TIMES BEFORE 17 : 00 DAYS WeekDays	is NOT ANSWERING

Conflicting Policy Name	Number conflict is that:	Will be overridden ?
TEST1	number is more specific (ALL =)	NO
FORWARD	number is more generic (STARTING WITH = 6944)	YES

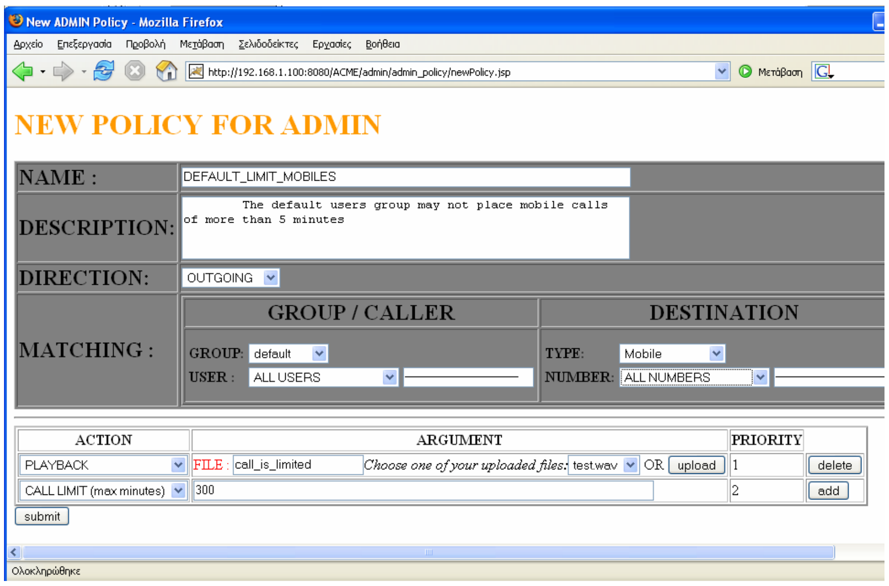
add actions

HELP:
Dial up to 3 numbers simultaneously. The first number to answer will be bridged

ACTION	ARGUMENT	PRIORITY		
SET CALLER IDENTIFICATION	100	1	delete	add
CALL LIMIT (max seconds)	180	2	delete	add
RECORD CALL		3	delete	add
DIAL	69757676767 210807777777	4	delete	add

submit

Figure 6 Example User Policy


New ADMIN Policy - Mozilla Firefox
Αρχείο Επεξεργασία Προβολή Μετάβαση Σελιδοδείκτες Εργασίες Βοήθεια
http://192.168.1.100:8080/ACME/admin/admin_policy/newPolicy.jsp
Μετάβαση
upload
delete
add
submit
Ολοκληρώθηκε

NEW POLICY FOR ADMIN

NAME :	DEFAULT_LIMIT_MOBILES
DESCRIPTION:	The default users group may not place mobile calls of more than 5 minutes
DIRECTION:	OUTGOING
MATCHING :	GROUP / CALLER: GROUP: default; USER : ALL USERS — DESTINATION: TYPE: Mobile; NUMBER: ALL NUMBERS

ACTION	ARGUMENT	PRIORITY
PLAYBACK	FILE : call_is_limited *Choose one of your uploaded files:* test.wav OR	1
CALL LIMIT (max minutes)	300	2

Figure 7 Example Administration Policy

(not_allowed.wav), then the call will hang-up. In the second case they will be told that the call will terminate in 5 minutes (call_is_limited.wav) and then the call will proceed, but hang-up after 5 minutes.

The administrator can also create, delete extensions, or reassign them to different users. He can also manage trunks which connect to the outside world. Least Cost Routing policies are used to select specific trunks based on the destination and how to deal with congestion.

It is not necessary to start specifying every policy from scratch. It is possible to create a copy of an existing one and edit this. Typically, new users will have a few default policies and can access some example ones to edit for their own use.

5. Policy Conflict Handling

Conflicts may arise as a result of users specifying polices relating to numbers at different granularity relating to the caller or destination. For example a policy for extension 101, for incoming calls at any time, starting with 0030 (calls from Greece) specifies that the call will be forwarded to his cell and recorded. Moreover it changes the caller-id in order to identify the call is coming from Greece. The user's second policy for when his mother calls him from Greece (matching exactly 003021080777777), forwards the call simultaneously to his mobile and home number to see which is answered first. Obviously a call from his mother will match both the first and second policies and the actions are not consistent. ACME detects the conflict and informs the user with a pop-up box and by adding a new line in the table with the conflict details. It states that the second policy conflicts with the first, the reason (this policy is entering a more specific number) and that this policy will NOT be overridden by the first one. The current default approach to resolve conflicts is to give precedence to the more specific one, as we consider that allocating priorities to policies increases the complexity of the interface without significant gain.

Policy conflict detection in ACME is done when the user enters the policy triggering criteria and presses the "add actions" button. Then a JavaScript function is called that submits all this criteria to a conflict analyser which first converts the received criteria, into a Policy object. It retrieves from the DB all existing user policies for the same extension and the same direction (incoming / outgoing) as a vector of Policy objects. It then does the following checks:

Time conflict check: If the candidate policy specifies any time there is no need to check time constrains of the other policies. Otherwise it iterates through the policies and removes those that do not overlap.

Peer number expression check: Each of the remaining policies not removed previously is checked against the candidate policy. The purpose is not only to detect conflicts but also state the reason to the user and suggest whether it overrides the existing ones. The combination of the peer expression match (ALL, STARTING WITH, MATCHING EXACTLY) and the peer number are checked. As mentioned before in the policy conflict resolution the most specific policy will be chosen. The hierarchy is as follows expressions with MATCHING EXACTLY override those with STARTING WITH which override those with ALL. If both polices use the STARTING WITH expression then the one with the longer peer number value will be chosen.

6. Evaluation

The ACME system was installed and evaluated at InAccess Networks which is a telecommunications consultant and small service provider in Athens. The deployed system is shown in

Figure 8.

Local users and administrators were asked to evaluate the system for ease of specifying policies and usability. Most specified policies relating to incoming calls, in particular to forward their calls to voicemail or to their cell phones when their extension did not answer or was busy. Outgoing policies were mostly used to dial multiple destinations at the same time. The administrator specified a policy to record phone calls to a particular number, which is used by the company as a conference server. He wanted to record all conferences in which he participated. Most of the feedback related to requests for increased functionality e.g. specification of more elaborate time criteria, ability to specify actions before the status criterion is checked for incoming calls, better colors in the web interface. ACME does not have the full functionality of Asterisk and some very complex policies could not be defined through ACME. Generally the feedback was very positive with users appreciating the ease of policy specification and the ability to view CDRs.

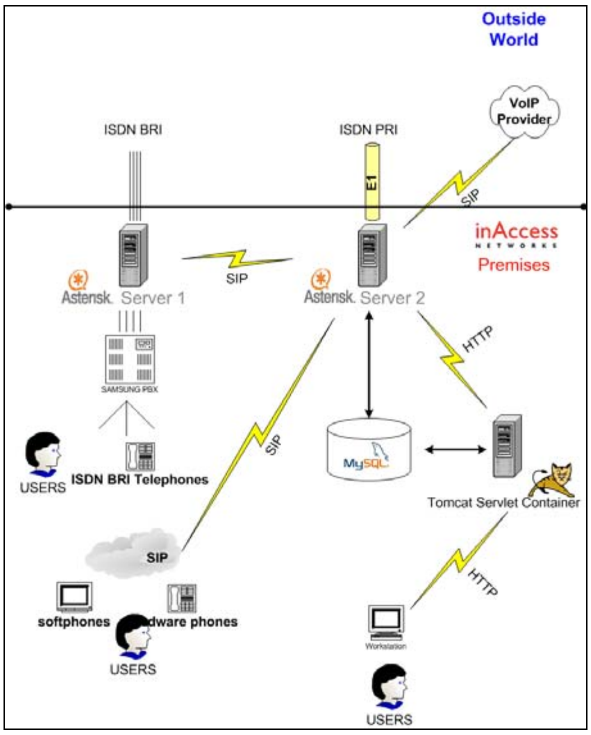


Figure 8 InAccess Evaluation System

An indication of performance is given from the additional overhead that ACME introduces in the call setup for an outgoing call. This overhead consists of 2 HTTP requests to the ACME server which we neglect as this takes place over a LAN so overhead average should be a negligible constant. In our setup ACME and the Asterisk server are implemented on the same computer so there is no network interaction. We measured the overhead related to the time needed by the Tomcat web server to execute the routing scripts, as this is the dominating factor. A dummy user account was created with 5 user policies for outgoing calls and the administrator created an additional 5 Admin policies for outgoing calls for that particular user. ACME had to retrieve and choose the correct user policy and also the appropriate administrative policy. A script was written on another Asterisk server that placed outgoing calls every second for a time frame of little over an hour i.e. around 4000 calls. The relevant overheads are shown in Table 1.

As expected the dominating factor is the connection of ACME with the DB and not the time needed by the script logic to choose the correct policy. This can be seen in the table above for the user latency where the DB connection time is 5.776 milliseconds while the Policy selection time is 0.238 milliseconds and therefore negligible. Note that the Overall Admin Policy latency is more than double the Overall User Policy latency, because the queries needed to retrieve the admin policies are more complicated and need to join more tables. Additional data about the user has to be retrieved, such as user group and call type. This is not an important issue because due to lack of time no effort was spent on optimizing ACME and it would not take much effort to improve the handling of database connection so that the overall Admin latency could be made similar to the overall User Latency. Therefore it is safe to say that an overall outgoing latency of 20 ms in call setup time is totally acceptable and can be improved further.

LATENCY	Ave. time in ms
User DB Latency	5.776
User Policy Latency	0.238
Overall User Latency	**6.01**
Admin DB Latency	14.124
Admin Policy Latency	0.407
Overall Admin Latency	**14.531**
Overall Outgoing latency	**20.546**

Table 1 : Outgoing Call Overheads.

7. Related Work

The graphical policy specification tool is implicitly based on event-condition-action rules similar to those found in many management policy approaches e.g. obligation policies in Ponder, call policies in Appel and PDL [6][8][10]. The graphical approach limits the policies which can be defined. Events can only be incoming or outgoing calls; Actions are limited to dial, hangup, record, play precorded file, limit call duration etc. and Conditions relate to caller numbers, time, current context, caller ID or group, and destination type. A policy specification language can be more flexible, but specifying specific policies has to be done within the context of a system model with predefined events, actions, conditions for the specific application. These are explicit in the graphical interface which effectively constrains the policies which can be specified so makes it much easier to use by non-experts of the application domain. A language based approach is better suited to network management where the potential range of events, conditions and actions is very large.

Obviously previous work with Ponder [6], influenced the approach taken and we tried to support some of the inheritance ideas of Ponder by supporting reuse of Policy Templates. It would have been possible to generate Ponder from the Graphical Interface, but there was no point, as Asterisk already provided configuration files and an implementation framework for performing all the actions related to call management.

The Accent Project Policy Environment Language (Appel) [8] is an XML based call management policy specification language. It is not specifically focussed on VoIP as is Asterisk and caters for signalling events between different exchange units so has a larger predefined 'vocabulary' particularly for triggers (events) and also for conditions and actions. Events can include disconnect requests, no-answer signals, invalid address signals etc. Appel, being XML based is even more user-unfriendly than Ponder and so they have also developed graphical user interfaces.

The UCS Router [2] was also aimed at policies for call handling but the call could be any form of message – email, SMS or phonecall arriving at a workstation or portable device. It introduced the concept of a Universal Communications Identifier for a user which could be used to route calls to whatever device the user was currently using e.g. work phone, home phone, mobile, PDA or laptop. It also has a very restricted vocabulary of events, conditions and actions related to call handling applications.

PDL was used in the SARAS Softswitch [10] but more to implement flexible management functionality of the switch e.g. to collect data relating to failures, alarm handling etc., than to implement user defined call handling policies. It is assumed that users are technical managers, capable of writing PDL policies.

Our approach to dealing with conflicts only caters for conflicts within the policies defined by a single user or with some administrator defined defaults. This was quick and easy to implement within the time constraints of the project. It does not cater for more complex conflicts that could arise when one user's policies conflict with those of another user on a different machine. For example a call forwarding policy may forward a call to a destination whose policies bar receiving calls from the source who made the original call. Appel defines both the conflicts and policies to resolve them in the Appel language [3]. However even that approach would not work for complex policies which can arise in policy languages which have conditions relating to system state. Particularly if the policies were generated by multiple users. [4][5] uses the event calculus as a logic based formalism to represent both policies and the conditions which indicate a conflict for Quality of Service Management applications. This is more generalized but more complex than our simple approach.

8. Conclusions

This work was a Masters level individual project at Imperial College London with about 6 man-months of effort. It was implemented in a real Asterisk system used by InAccess in Athens and evaluated by both users and administrators working for that organization. The feedback on ease of use was very positive. Ordinary users were able to easily write policies that they would never have attempted using Asterisk's ACL. Considering that there we did not try to optimize performance, the measured overheads were acceptable.

Although there has been much emphasis in the literature on policy languages, it is unlikely that non-technical users will be able to use these, and application specific graphical interfaces will be needed. This paper presents a typical graphical interface tailored to call management but similar ideas could be used in other application areas.

The potential for future work is vast. ACME currently only supports the VoiceMail service. The web dialer service was not fully implemented. Other services could include conferencing. In order to provide better response times for call handling requests, ACME could be separated in two applications. One that handles the call logic and another that handles only system configuration and user interfaces. The two applications would share a common database in order for the call routing application to apply the correct user choices set by the interface application. Performance could also be improved by using a database connection caching mechanism that would reuse connections with the database thus eliminating the overhead of establishing and tearing down JDBC connections.

9. References

[1] Asterisk the Open Source PBX http://www.asterisk.org/

[2] Bertino E. , M. Cochinwal, M. Mesiti. UCS-Router: A Policy Engine for Enforcing Message Routing Rules in a Universal Communication System. *Proc. 3rd. Int. Conference on Mobile Data Management*, 2002.

[3] Blair L., Turner K. . Handling Policy Conflicts in Call Control. In S. Reiff-Marganiec and M. Ryan, editors, *Proc. 8th. Int. Conference on Feature Interaction*, pages 39-57, IOS Press, Amsterdam, June 2005

[4] Charalambides M. et. al., Policy Conflict Analysis for Quality of Service Management, *Proc 6th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2005)*, Stockholm, Sweden, June 2005.

[5] Charalambides M. et. al Dynamic Policy Analysis and Conflict Resolution for DiffServ Quality of Service

Management, *Proc. 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)* Vancouver, Canada, April 2006.

[6] Damianou N., N. Dulay, E. Lupu, M Sloman, The Ponder Specification Language *Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, 29-31 Jan. 2001, Springer-Verlag LNCS 1995, pp. 18-39

[7] H.323 specification: http://www.itu.int/rec/T-REC-H.323/en

[8] Huang T., and Turner K. . Policy Support for H.323 Call Handling, *Computer Standards and Interfaces , Dec. 2005*.Vol. 28, No. 2, pp. 204-217

[9] SIP protocol RFC3261: http://www.ietf.org/rfc/rfc3261.txt

[10] Virmani A, J. Lobo, M. Kohli, “Netmon: Network Management for the SARAS Softswitch”, *IEEE/IFIP Network Operations and Management Symposium, (NOMS2000),* Hawaii, May 2000, pp803-816

[11] VoIP History: http://www.voipreview.org/news.details.aspx?nid=51

10. Acknowledgements

We would like to thank InAccess for providing the facilities for deploying and evaluating the ACME system as well as the staff for their help and advice. In addition Naranker Dulay and Emil Lupu of Imperial College who provided many suggestions.