Service Learning in Internet Telephony

Xiaotao Wu
Department of Computer Science
Columbia University
New York, New York 10027
Email: xiaotaow@cs.columbia.edu

Henning Schulzrinne
Department of Computer Science
Columbia University
New York, New York 10027
Email: hgs@cs.columbia.edu

***Abstract*— Internet telephony can introduce many novel communication services, however, novelty puts learning burden on users. It will be a great help to users if their desired services can be created automatically. We developed an intelligent communication service creation environment which can handle automatic service creation by learning from users' daily communication behaviors. The service creation environment models communication services as decision trees and uses the Incremental Tree Induction (ITI) algorithm for decision tree learning. We use Language for End System Services (LESS) scripts to represent learned results and implemented a simulation environment to verify the learning algorithm. We also noticed that when users get their desired services, they may not be aware of unexpected behaviors that the serivces could introduce, for example, mistakenly rejecting expected calls. In this paper, we also did a comprehensive analysis on communication service fail-safe handling and propose several approaches to create fail-safe services.**

I. Introduction

One of the key advantages of Internet telephony is its ability of providing many innovative communication services, however, novelty is a barrier to entry. Many users may not be aware of what services are available and not know how to customize or create their own services. It will be a great help to users if there is a service creation environment that can automatically generate their desired services by learning from their daily communication behaviors. For example, if a user always reject calls from a telemarketing company, the service creation environment can infer that any calls from that company's domain should get rejected automatically. The service creation environment can then generate a service script to represent the auto-rejection, upload the service script to the user's communication user agent to prevent phone spams from the telemarketing company. We consider the learning process for automatic service creation is applicable in Internet telephony systems because of the following reasons.

In Internet telephony, call signaling messages usually contain much richer information than those in PSTN networks. For example, in a SIP [8] call, an INVITE message can contain the display name, SIP address, organization, preferred language, photo, and vCard information of the caller, and can contain the priority, and subject information of the call. The caller and the callee can know each other's presence status [9], activities [10], and location [5] information by supporting the SIP event notification architecture [7] in their SIP user agents. These information allows people to make sensible call decisions which are usually impossible in PSTN networks. Thus, in Internet telephony, call information and call decisions have a causal relationship. The causal relationship makes service learning valid. In addition, in Internet telephony systems, end systems usually have CPU and memory so they can easily collect users' communication behaviors for learning and use the learned results to provide communication services. The extended abilities of Internet telephony end systems make service learning practical and useful.

We need to handle four tasks for service learning: *representing communication behaviors, finding a learning algorithm, representing the learned results, and handling service fail-safe.* How to represent users' communication behaviors determines how easy the learning process can be. We choose to use decision trees to represent communication behaviors and detail the rational of the choice in Section II. Section III gives the criteria on how to choose a decision tree learning algorithm and introduces the Incremental Tree Induction (ITI) algorithm we use for service learning. Because the simplicity, safety, and the tree-like structure of the Language for End System Services (LESS) [15], we decide to use LESS scripts to describe learned results. Section IV shows how to convert a decision tree to a LESS script. In most cases, communication services bring great conveniences to users. However, sometimes they may cause unwanted behaviors and users may not be aware of. For example, a user can use a service script to block all calls from domain `ads.com` to prevent phone spams. If later on, one of the user's friends works in `ads.com`, the user might forget that he has a service blocking calls from `ads.com` and will never get calls from her friend. A call from the user's friend is an exception to the user's existing service and auto-rejecting the call is a misaction to handle the exception. In Section V, we give a comprehensive analysis of potential service exceptions and propose several approaches to create fail-safe services.

Section VI describes how we integrate the service learning functions in our SIP user agent, SIPC [14]. Section VII concludes the paper and discusses our future work.

II. Representing users' communication behaviors

There are many ways to represent users' communication behaviors, such as finite state machines, Use Case Maps [1], rule sets, decision trees, and Bayesian networks [3]. We consider binary decision trees are the most suitable way for the service learning.

We can clearly identify learning targets in a binary decision tree because the structure of a binary decision tree is simple. As shown in Figure 1, the learning process is to find all the non-leaf tree nodes that can best partition users' behavior data. It is hard to identify learning targets in a finite state machine or a Use Case Map because the structure of a finite state machine or a Use Case Map is too dynamic to learn.

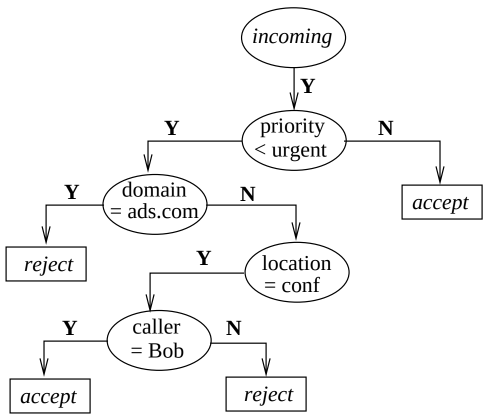

Fig. 1. Representing communication behaviors as decision trees

A binary decision tree is not Turing-complete so it cannot represent all possible communication behaviors. However, the goal of service learning is not to find all possible user desired services, instead, it only focuses on commonly used services to keep the learning process simple and efficient. Learning on complicated services is error prone and usually brings confusions to users.

Using rule sets to represent communication behaviors is another viable way. Since rule sets can usually be converted to decision trees, and decision trees are more efficient for service execution, we consider using decision trees is a better choice.

We also considered using Bayesian networks [3] to represent communication behaviors for learning. We found that Bayesian networks are not appropriate for service learning because when making a communication decision, different factors may not be independent to each other. For example, domain-based call rejection is not independent to caller-based call rejection. We cannot build a Bayesian network with co-related decision factors [3].

The training set for learning can come from several resources. Call logs can show the name, and the SIP URI of calling parties, and the calling time, the priority and the subject of calls. Event logs and calendars can provide calling parties' presence status, activities, and location information. Address books or LDAP servers can contain the affiliation of the calling parties. To enable communication service learning, a user agent must collect all related data and generate a training set. We detail how SIPC [14] handles training data collection in Section VI.

III. Decision tree learning

There are many existing decision tree learning algorithms [6], [12]. To choose an appropriate algorithm for our communication decision tree learning, we defined several requirements for the algorithm.

The decision tree should be learned incrementally. Collecting human communication behavior samples is an accumulating process. After a period of time, some new samples get collected, some new rules may get generated, and some old rules may get broken. Building decision trees in an incremental way reflect the dynamic changes of people's communication behaviors.

There should be an appropriate tree quality measurement mechanism [13] in the learning algorithm. As shown in Figure 2, two trees generate the same decisions on calls. Both trees have the same height. The left tree has fewer leaves and fewer nodes, but the right tree has a shorter path for most of the training samples (30 matches for rejecting calls from Bob with priority lower than urgent). In terms of simplicity, the left tree is better. In terms of efficiency, the right tree is better. Because the service learning is to help users to create and understand communication services, a simpler decision tree is more preferable. For example, in Figure 2, left tree is the desired result.

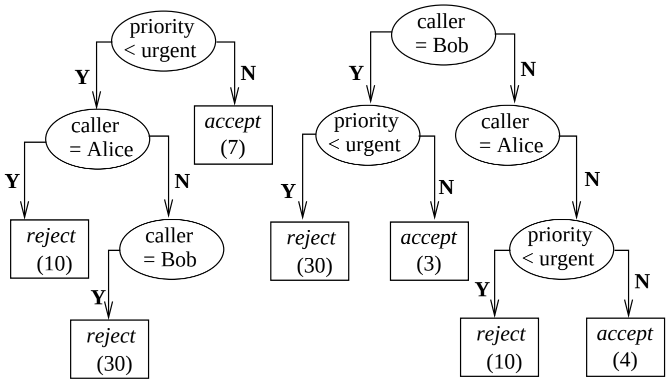

Fig. 2. Two decision trees with different tree node order

An important component of decision tree induction is to avoid overfitting the training data. Many random factors may affect communication behaviors. There should be an effective pruning method to filter decision errors (noise). In addition, because people's communication behaviors can change dynamically, with additional learning samples, pruned branches may get reactivated as valid communication services. Thus, the learning algorithm should also be able to save and reactivate pruned branches.

A. Incremental Tree Induction (ITI) algorithm

Based on the above requirements, we chose to use the Incremental Tree Induction (ITI) [12] as our service learning algorithm. ITI can map an existing tree and a new training example to a new tree based on several tree transformation mechanisms. This makes the average incremental cost of updating a tree being much lower than the average cost of building new decision trees from scratch each time.

ITI uses an algorithm called *direct metric tree induction* to map an existing tree to another based on the tree quality measurement. The algorithm introduces four tree quality measurement matrices, namely *expected number of tests, minimum description length*, *expected classication cost*, and *expected misclassication cost* [12]. We choose to use the *expected number of tests* matrix for our service learning process because that reflects the simpleness of a decision tree.

ITI also introduces a pruning technique named *virtual pruning*. The pruning technique uses one bit on a decision node to mark whether a decision branch should be pruned or not. It will not delete pruned branches. When presenting learned results to users, we can hide pruned branches based on the marks to avoid the overfitting problem, but when new learning samples are added in, ITI will use the unpruned tree to perform learning so the pruned branches can still get reactivated and generate useful services.

B. Accuracy of ITI algorithm

There are two ways to measure the accuracy of ITI algorithm for communication service learning: one is to perform real world usage testing and the other is to do simulation.

For real world usage testing, we need to collect real communication behavior data from users, perform service learning and generate services based on the data, send the services back to the users, gather feedback from the users, and evaluate the feedback. The testing requires a large deployment of VoIP systems with people using the VoIP systems for their daily communications. Currently, people are making efforts to deploy VoIP systems, such as SIP.edu [11] initiatives in Internet2 community and many companies have deployed VoIP solutions in their intranet, however, most of the deployment are still in their testing stage, not as primary communication means for people's daily usage. The real world usage testing is in our future work plan when our local VoIP deployment is available.

Because the real world VoIP deployment is not sufficient for performing accuracy measurement of the ITI algorithm, we have built a simulation environment to generate random simulated calls. In the default simulation setup, we use Poisson distribution to set call arriving time, uniform distribution multiplied by weights to set call priority and choose caller: 98% of the calls are in normal priority, 0.9% urgent calls, 0.1% emergency calls, and 1% non-urgent calls; 30% of the calls are from `user1`, 10% each from `user2`, `user3`, and `user4`, the rest from all the other users. The simulation environment also simulates people's daily life, such as time for meal and sleep, and can load calendars in iCal [2] format for meeting and appointment information.

To measure the accuracy of the learning algorithm, we randomly created several expected services and apply the services to the generated calls to simulate call handling process. For example, if we have an expected service `"reject all calls from sip:bob@examples.com"`, when we apply the service to the generated calls, all calls from `sip:bob@examples.com` get rejected. The other calls will be handled based on the simulation setup. With the default setup, if a call arrives when the simulated user is sleeping, 50% of the calls will not get answered; when the user is in meal, 20% of the calls will not get answered; if the user is in an appointment, 40% of the calls get rejected; in normal cases, 85% of the calls get accepted. The default setup will introduce reasonable noise for service learning. Once the simulation completes, we will get simulated communication behavior data saved in C4.5 [6] format. We can then use ITI algorithm to learn from the behavior data. The pruned trees generated by ITI algorithm should match the expected services. We tested 40 expected services, each applied to 300 simulated calls. The tests show that 80% of the learning results exactly match their expected services, 10% of the learning results represent the expected services in different ways, and 10% of the learning results do not match the expected services. The mismatch comes from the randomness of the simulation data. For example, if an expected service is `"accept all emergency calls"`, but there are not emergency calls in the 300 simulated calls since only 0.1% of calls are emergent, the learning algorithm cannot infer the expected service. Based on the simulation, we consider ITI algorithm fits our need.

C. Performance of ITI algorithm

Figure 3 and Figure 4 show the performance of ITI algorithm based on 700 simulated calls, running on an IBM ThinkPad laptop with Linux operating system, a 1GHZ Intel Pentium III Mobile CPU, and 256MB memory. The expected services in the testing make call decisions based on the priority of calls, caller's addresses, and callee's ongoing activities when receiving calls.

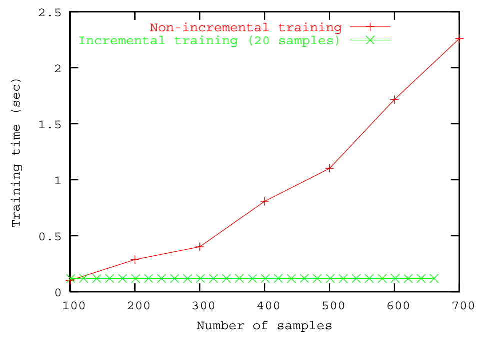

Fig. 3. Training time for fast training and incremental training

Figure 3 shows that the training time for non-incremental training increases quadraticly as more training samples are added in, while the training time for incremental training is a constant if only training on the new added samples. For 20 new added samples, the training time for incremental training is about 0.12 seconds, which is quick enough to provide automatic service creation to users.

Figure 4 shows that for incremental training, the training time is independent of the number of internal nodes in the expected services. This is because ITI algorithm uses the *virtual pruning* technique to handle overfitting problem so it always constructs complete decision trees. The number of

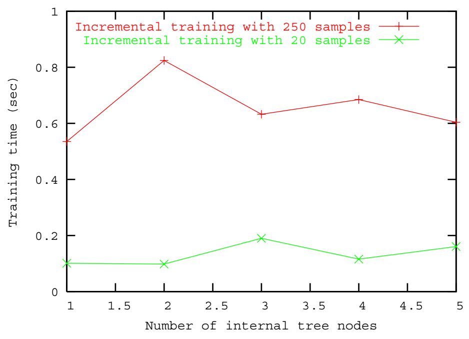

Fig. 4. Training time for trees with different number of internal nodes

nodes in a pruned tree will not affect the training time of building a complete decision tree.

IV. Using LESS scripts to represent learned results

Since the Language for End System Services (LESS) [15] has a tree-like structure, it is straightforward to convert learned results to LESS scripts. The non-leaf tree nodes can be converted to LESS `switches`, the leaf nodes can be converted to LESS `actions`, and the root nodes can be converted to `triggers`. For example, we can convert the decision tree in Figure 1 to the LESS script below:

```
<less><incoming>
 <priority-switch><priority less="urgent">
  <address-switch field="origin"
   subfield="host">
   <address subdomain-of="ads.com"><reject/>
   </address>
   <otherwise>
    <location-switch type="civil">
     <location LOC="confroom">
      <address-switch field="origin"
       subfield="display">
       <address is="Bob"><accept/></address>
       <otherwise><reject/></otherwise>
      </address-switch>
     </location>
    </location-switch>
   </otherwise>
  </address-switch>
 </priority>
 <otherwise><accept/></otherwise>
 </priority-switch>
</incoming></less>
```

The generated LESS scripts can be loaded into a user agent's service engine to automate call processing.

V. Creating fail-safe communication services

Once a user gets her desired services, the user may not be aware of the side-effect of the services. The service creation environment should alert the user about possible unexpected behaviors that the services may introduce and provide approaches to make the services fail-safe.

As we discussed in Section I, a communication service may have exceptions. For example, a user has a service blocking all calls after `11:00PM`, if later on she expects an important call at `11:30PM`, an exception to the service happens and the communication service will perform a misaction and cause unexpected results, in this example, the user will miss the important call. To help users better handle service exceptions, we first analyze what may cause service exceptions, we then propose several approaches to make communication services fail-safe.

A. Service exceptions

Telecommunication on one hand enables people to interact with each other, on the other hand, it reveals people's private information, distracts people's attention, costs people money, and may require people to change their environment, such as turning off a stereo, for better communication quality. We consider establishing connections, protecting private information, preventing people from disturbance, saving money, and keeping a comfortable environment are all required aspects for telecommunication. However, when we use communication services to automate call processing, actions in communication services usually favor some aspects of communication but break the others. For example, auto-accepting a call makes it easy to establish connections between users, but will reveal the callee's private information and cost the callee money without the callee's awareness. On the contrary, auto-rejecting a call can protect private information, save money, and prevent disturbance, but may lose important calls. A service exception happens when an action in the service breaks the aspects that a user wants to protect. Table I shows the required aspects of communication and the actions that may break them.

Communication aspects	Actions may break the aspects
establishing call connections	rejecting or redirecting calls, accepting calls on a wrong proxied branch
protecting private information	auto-accepting calls
preventing people from being disturbed	alerting, auto-accepting calls
saving money	auto-accepting calls, redirecting or transferring calls to a device with higher charge rate
keeping a comfortable environment	auto-controlling networked appliances

TABLE I
Communication aspects and actions that may break them

Several kinds of automated actions may cause users to lose connections. Automatically rejecting a call will terminate a connection request. Automatically redirecting a call or accepting a call on a wrong proxied branch may break the connection to user desired devices.

Communications can reveal calling parties' private information. When automatically accepting a call, audio and video streams will be sent to the remote party without users' awareness and reveal the information the users want to keep.

People sometimes will need to focus their attention on a specific task and do not like to be disturbed. For example, in a meeting, ringing a phone will disturb not only the owner of the phone but also other people in the meeting. When driving, auto-accepting a video call may distract the driver's attention away from the road and may cause danger. Different media types and different communication means have different disturbing characteristics, as discussed in Section V-B and Section V-C.

Different devices may have different call charges for communications. Auto-redirecting or auto-transferring a call may cost calling parties more money than expected. For example, to support session mobility, a service may automatically transfer an ongoing session from a fixed-line phone to a mobile phone and that may involve call charge changing.

Environment status can affect communication quality. For example, a dark environment will make video conferencing unpractical, a noisy environment will make audio conferencing non-preferred. Some services may automatically adjust environment, e.g., performing networked appliance control, to improve communication quality. However, changes to an environment may also affect the other people in the same environment and cause unwanted results.

Table I shows the actions that may cause service exceptions. To handle service exceptions, we need to further analyze disturbing characteristics and media availability of different communication means.

B. Disturbing characteristics of different communication means

Table II shows the disturbing characteristics for commonly used communication means, namely multimedia call, instant messaging, voicemail, and email, with different alerting style. The multimedia call with playing ring tone for alerting are the most disturbing way for communication. Usually, it requires the called party to pickup the call in less than one minute, otherwise, the called party will lose the call. The ring tone not only disturbs the called party but also the others in the same environment. The disturbance analysis can help us to create fail-safe communication services for disturbance prevention.

Action	Disturbed entities	Expected response time
Call/ringing	Callee and others	seconds - minutes
	Immediate attention otherwise lose the call	
Call/vibrating	Callee	seconds - minutes
	Immediate attention otherwise lose the call	
Instant messaging	Callee	minutes - hours
	Immediate attention but delay is ok	
Voicemail/Email with indication	Callee	minutes - days
	Immediate attention but delay is ok	
Voicemail/Email without indication	Callee	hours - days
	No immediate attention	

TABLE II
COMMUNICATION ACTIONS AND THEIR DISTURBING CHARACTERISTICS

C. Media availability analysis of multimedia communications

Table III shows the factors that affect users' media availabilities. If a user's sense organ is occupied or if the environment is not appropriate, some media communication should be considered unavailable. For example, when driving, hands and eyes are occupied, video, text, and whiteboard communications are not preferred; in a noisy environment, audio communications are not preferred. The analysis can help to improve some services by switching media types based on users' activity and environment status.

Media	Sense Organ	Preferred Environment
audio	ears, mouth	hearing in quiet environment
video	eyes	watching background not too bright sending in bright environment
text	hands, eyes	writing party at a stable place, watching background not too bright
whiteboard	hands, eyes	drawing party at a stable place, watching background not too bright

TABLE III
ORGANS AND PREFERRED ENVIRONMENT FOR MULTIMEDIA COMMUNICATIONS

D. Making communication services fail-safe

Based on the above analysis, we propose several solutions to make communication services fail-safe. The design rule of the solutions is to ensure that either the consequences of a misaction is trivial, or users can get alerted of misactions and the correction is viable. Because the most important task of telecommunication is to establish connections between users, our solutions are mainly focus on how to prevent people from losing connections. We will also consider how to protect the other aspects of communications, such as protecting people's private information.

1) Lowering disturbing level instead of rejecting: Some services help people to prevent disturbance by automatically rejecting incoming calls. This kind of services is very useful when users are in a meeting or sleeping. However, the services may also reject important calls. To save calls but still prevent people from being disturbed, instead of rejecting calls, users can choose to lower the disturbing level of incoming calls based on Table II, for example, in an unimportant meeting, change the alerting style from ringing to vibrating; in an important conference, forward the call to a voicemail and provide voicemail indication. If the user is doing a presentation in a conference and does not want to be disturbed in any way, he can forward the call to voicemail without providing any indication.

2) Changing media types instead of rejecting: Some services automatically reject calls when the user is unavailable or communication environment is not suitable. However, if we carefully distinguish partially available status and fully unavailable status, we will find communications are still possible in partially available status, but using different media types. As we discussed in Section V-C, different media types require different sense organs and have different environment requirements. For example, when driving, for a video conferencing call request, instead of rejecting it, the driver's communication services can automatically setup the connection as an audio only call. For an incoming audio call, when the called party on another audio call, his services can ask the caller to use instant messaging.

3) Transferring the disturbance instead of rejecting: Sometimes, disturbance can be transferred, for example, a boss can transfer calls to his secretary when he is in a meeting. The transfer ensures that calls get handled and the boss' attention still get protected.

4) Making a backup: When rejecting a call, it's better to make the called party be aware of the rejection, for example, by sending an email, or asking the caller to leave a voicemail. This allows the called party to correct unexpected rejections later on.

5) Setting appropriate initial media transmission status: As discussed in Section V-A, auto-accepting calls can provide convenience for establishing connections but will reveal users' private information. To protect users' private information, when automatically accepting incoming calls, it is safer to set the media transmission status as "receive only".

VI. Implementations

We have implemented a SIP user agent called SIPC [14]. SIPC has a built-in LESS engine that can execute LESS scripts and provide communication services. SIPC has integrated multimedia communication, SIP event notification, instant messaging, email, and web browsing into one application. It also supports networked appliance control, real-time multimedia streaming, networked resource discovery, third-party call control, Internet TV, location sensing, emergency call handling, and conference floor control. For SIP event notification, SIPC can retrieve remote parties' presence status [9], rich presence information (such as activities, privacy status, place types, and idle status) [10], location [5] information, and media availabilities [4]. SIPC also has a built-in address book. With all these functions, SIPC can collect a lot of information of calling party's status and has a rich set of actions for call handling. SIPC records all the call related information and user performed actions in a file in C4.5 [6] format, the same format as what we used in our learning algorithm testing simulation environment. Once the number of new call records reaches a threshold, e.g., 20 new call records, SIPC will use the integrated ITI algorithm to perform service learning. The learned result will be saved in two files, one is a LESS script that representing the pruned tree, the other is a binary file that representing the whole tree with virtual pruning marks. SIPC will then load the LESS script in its LESS engine to provide communication services. We are still working on building a user friendly interface to help users to handle communication service fail-safe.

VII. Conclusion and future work

This paper proposes a model for communication service learning. Because usually end users are not trained for communication service creation, they may not know how to customize or create their own services. Service learning can help them by generating communication services automatically based on their communication behaviors. We noticed that communication services may perform unexpected actions and proposed several solutions to make communication services fail-safe. Because people are still rarely using VoIP systems for their daily communications, we cannot perform real world testing for the ITI learning algorithm, but we had done accuracy and performance measurements for the ITI algorithm in our simulation environment and proved that it fits our service learning model. When we have a local deployed VoIP system and used by the people in our department, we will do a real world usage testing and to improve our service creation environment based on the real world feedback. We will also extend our current implementation to support more decision variables, such as rich presence information [10], and location information, for service learning.

References

[1] Daniel Amyot. Use case maps as a feature description notation. In *FIREwork Feature Constructs Workshop*, May 2000.

[2] F. Dawson and D. Stenerson. Internet calendaring and scheduling core object specification (icalendar). RFC 2445, Internet Engineering Task Force, November 1998.

[3] Finn V Jensen. Bayesian networks and decision graphs. In *Bayesian networks and decision graphs*. Springer, 2001.

[4] M. Lonnfors and K. Kiss. User agent capability presence status extension. Internet Draft draft-ietf-simple-prescaps-ext-00, Internet Engineering Task Force, February 2004. Work in progress.

[5] J. Peterson. A presence-based GEOPRIV location object format. Internet Draft draft-ietf-geopriv-pidf-lo-01, Internet Engineering Task Force, February 2004. Work in progress.

[6] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufman, 1993.

[7] A. B. Roach. Session initiation protocol (sip)-specific event notification. RFC 3265, Internet Engineering Task Force, June 2002.

[8] J. Rosenberg, Henning Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: session initiation protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[9] Jonathan Rosenberg. A presence event package for the session initiation protocol (SIP). Internet draft, Internet Engineering Task Force, January 2003. Work in progress.

[10] Henning Schulzrinne. RPID – rich presence information data format. Internet Draft draft-ietf-simple-rpid-01, Internet Engineering Task Force, February 2004. Work in progress.

[11] SIP.edu. Sip.edu cookbook. http://web.mit.edu/sip/sip.edu/.

[12] Paul E. Utgoff, Neil C. Berkman, and Jeffery A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44, 1997.

[13] S. M. Weiss and C.A. Kulikoswski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufman, 1991.

[14] Xiaotao Wu. Columbia university SIP user agent (sipc). http://www.cs.columbia.edu/IRT/sipc.

[15] Xiaotao Wu and Henning Schulzrinne. Programmable end system services using SIP. In *Conference Record of the International Conference on Communications (ICC)*, May 2003.