Printing in Ubiquitous Computing Environments

Athanasios Karapantelakis[1,2], Alisa Devlic[1,3], Mohammad Zarify[4], and Saltanat Khamit[1]

[1] Royal Institute of Technology (KTH), Stockholm, Sweden
[2] Ericsson AB, Stockholm, Sweden
[3] Appear Networks AB, Kista Science Tower, Sweden
[4] University of Twente, Enschede, Netherlands
{athkar,devlic}@kth.se, M.Zarifi@utwente.nl,
saltanat.khamit@radio.kth.se

Abstract. Document printing has long been considered an indispensable part of the workspace. While this process is considered trivial and simple for environments where resources are ample (e.g. desktop computers connected to printers within a corporate network), it becomes complicated when applied in a mobile context. Contemporary mobile devices have the computational resources required for document processing and are affordable enough for an increasingly large number of users. Therefore, document printing using mobile devices is now both technically feasible and relevant to users' needs. In this study, we present an infrastructure for document printing using mobile devices. In order to realize the vision, we utilize an existing set of network protocols, a set of common programming languages, standard concepts of ubiquitous computing, and machine learning, in order to automate the printing process.

Keywords: Ubiquitous printing, SIP/SIMPLE, SNMP, PJL.

1 Introduction

Over the last decades printing devices have continuously improved, keeping pace with customer expectations. Monitoring of annual print outputs from various academic institutions [1] [2] indicates a trend towards increased printer utilization. Wright suggests that printing will continue to increase as public interest shifts towards the enormous volume of information available on the World Wide Web [3].

To realize the customer demand for printing services, printer manufacturers and operating system authors have tried to improve interoperability between printers (hardware) and workstations (software) [4]. Although such efforts have been successful in solving technical issues, such as a common page layout description (i.e. page description languages such as PCL and PostScript), standardized communication interfaces (e.g. IEEE 802.11, Ethernet, Bluetooth, parallel, USB ports), and a consistent way for the user to parameterize a print request (printing protocols and APIs), there has been little improvement in processing and interpretation of printer *semantics*. Traditionally, the user visually monitors the progress of their print request on a display. If the printing process halts due to an error, (for example need for paper

D. Zhang et al. (Eds.): UIC 2009, LNCS 5585, pp. 83–98, 2009.
© Springer-Verlag Berlin Heidelberg 2009

refill/ink replacement), the responsibility for fixing the issue and resuming printing is the user's. Another common case is that certain document types may require special hardware capabilities (e.g. a color, high resolution printer) for optimal results. The lack of a resource aware infrastructure wastes time, a critical factor for academic and corporate environments.

Additionally, in office environments increasingly users use their personal mobile devices for common tasks such as reading/composing e-mails, accessing the corporate directory, browsing the web, etc. These devices are also used for document editing and storage, but generally lack the software to interface with printers: due to their heterogeneous nature (different processor architectures, hardware specifications), printer vendors have not written device drivers for all of these mobile devices.

Our solution addresses mobile printing by providing a universal printer access interface based on *open* protocols. Additionally, exploiting awareness of all the printers in the environment it chooses the most suitable printer to serve the current print request, thus making printing more time-effective and transparent for users.

Our implementation approximates the definition of the *Ubiquitous Computing Environment*, as defined by McGarth *et al.* [5]. We consider users with mobile devices as autonomous, heterogeneous entities, with no prior knowledge of the infrastructure. Similarly the printing infrastructure has no knowledge of potential users.

This paper is organized into five sections. Section 2 provides an overview of published relevant work. In section 3, we describe the theoretical scope of our work, introducing data models and decision algorithms. This section also includes a technical overview of the system. Section 4 presents and evaluates a set of measurements. We conclude in section 5, with a recapitulation of the key aspects of our work, and suggest some future work.

2 Related Work

Mobile printing is a relatively new concept since mobile devices have only recently acquired the necessary network interfaces (e.g. Bluetooth and IEEE 802.11) to interact with printers – other than via a point-to-point IRDA link. Burke specifies two general methods for mobile printing, namely serverless printing i.e. directly through the utilization of device drivers, and print by reference [6].

Mobile device drivers render the files locally on the device using a page description language, and subsequently send this data to the printer over a network interface. Printer manufacturers (such as HP [7]) and mobile device manufacturers (such as Nokia [8]) have developed printing drivers for specific models of devices. SonyEricsson's camera phones have adopted the PictBridge standard [9], which allows images to be printed directly to the printer without the presence of a computer. Additionally, third party developers have implemented their own solutions for mobile printing [10]. In an effort to disambiguate the complex issue of implementing serverless mobile printing, the Mobile Imaging and Printing Consortium (MIPC) publishes technical documentation in a developer guideline format [11]. Their document distinguishes between different network interfaces that mobile devices may potentially be equipped with, and addresses each case separately (see figure 1). However, although serverless printing can function as an integrated solution on

specific pairs of mobile devices and printers, it does not scale well for environments where the hardware specifications are not fixed *a priori*.

One constrain is that there must be a network path from the network interface (typically via a WLAN, USB, and/or wide area wireless network interface) of the mobile device to the printer. Statistically, the majority of mobile devices today have a wireless network interface, whereas most network printers still use Ethernet as their network interface of choice. However, MIPC's serverless printing implies a direct connection between the mobile device and the printer. An additional disadvantage is the vertical implementation of the current serverless printing solutions, which leads to poor code reusability and increased costs for maintenance.

In order to render a file, the corresponding application has to decode its contents, encode it (in a page description language), and send it to the printer. Although most mobile devices support common file formats (e.g. for text, spreadsheets, and images), this software lacks features available in desktop applications. Therefore, files edited with specialized desktop applications (e.g. architectural drawings, 3d models, mathematical workspaces, etc), are unlikely to be able to be directly printed, since the mobile device does not know how to decode the document and encode it for printing.

Printing by reference is another method that moves the computational complexity, file format decoding and page layout from the mobile device and delegates this responsibility to the printer [6]. In this case, a mobile device transmits only the URI of the document to a printer. Subsequently, the printer fetches the document and prints it. The rendering resources required can be implemented by a server attached to the network or software in the printer itself. Since the mobile device only provides a link to the document there is no requirement for printer drivers on the mobile device. However, there is need for a network attached document repository, where documents will be stored (after being sent from the mobile device prior to printing or already stored).

We propose a third method that improves on concepts from both the previously mentioned methods while effectively addressing their disadvantages (see section 3.1) for a networked printer environment with heterogeneous sets of mobile devices. Specifically we seek to exploit device mobility and context in our design of a mobile printing infrastructure. The service provided by our solution can be summarized in a sentence: For a user trying to print, this infrastructure automatically selects the *nearest suitable* printer.

A distributed algorithm that accepts printing requests handles the selection process. It uses the user's location, printer status, and document type as input parameters. The algorithm quantifies the possible printer states according to the severity of their impact on the selection process. Similarly the relative location of each printer with respect to the user is considered. Thus, for each printer in the infrastructure, the algorithm calculates a printer suitability index. Subsequently, these indexes are compared and the largest value indicates the most suitable printer for this request.

3 System Overview

Figure 1 illustrates the operating environment of the system we deployed in our lab. Printers are scattered around an indoor workspace, consisting of different rooms and corridors. Printing Point Services (PPS) provide printer context (see section 3.2).

Additionally, user location context is provided from RFID tag readers, when available[1]. The context broker is the centerpiece of our architecture, and plays a double role: It functions both as a context repository and also implements the aforementioned decision algorithm (see section 2). Each of the system components is described in detail in the following subsections. This section concludes with a description of the operating phases of the system.

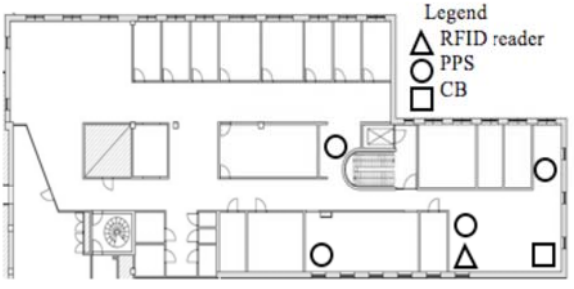

Fig. 1. Deployed system topology. A context broker ("CB") acts as a repository of context information coming from Printing Point Services – "PPS" (printer status and capabilities), as well as RFID readers – "R" (user location). The CB is also responsible for handling printing requests from mobile users by designating suitable a suitable printer.

3.1 Context Broker

The context broker is a host with network connectivity to every PPS and mobile terminal. Our software solution builts upon the open source SIP Express Router (SER) [12] and is responsible for registering new mobile terminals and handling print requests. We extended the functionality of SER by implementing three modules for context handling, user presence , and resource discovery.

The context broker incorporates the Service Locator, a separate resource discovery mechanism which uses the Service Location Protocol (SLP) [13] in order to discover available Printing Point Services in the infrastructure (see section 3.2). Resource discovery takes place not only during bootstrapping, but also during normal system operation. This allows for installation of new PPSs at any time. Pairs of <PPSid, IPaddress> are stored inside a "printer pool" table in a MySQL database (The PPSId is a unique identifier of a given PPS and is assigned by the Service Locator).

The context aggregation module receives and stores location updates from the location service infrastructure, as well as printer status updates from the PPSs. We are using SIP/SIMPLE messages to transfer event updates to SER [14]. These events are transferred as a PUBLISH message payload and are modeled after the Presence Information Data Format (PIDF) XML Schema [15]. SIP messages, as well as the functional role of the presence server, are defined in the SIP/SIMPLE protocol [16].

The user presence module registers mobile terminals with the infrastructure. Registration can be automatic (if the devices are detected by the RFID readers), or

[1] The user's location could be provided by any location system, with room level resolution.

by an explicit registration request. A mobile terminal initially subscribes to the printing service by sending a SUBSCRIBE request message to the context broker. SER processes these requests and registers the mobile terminal in a database (see figure 2).

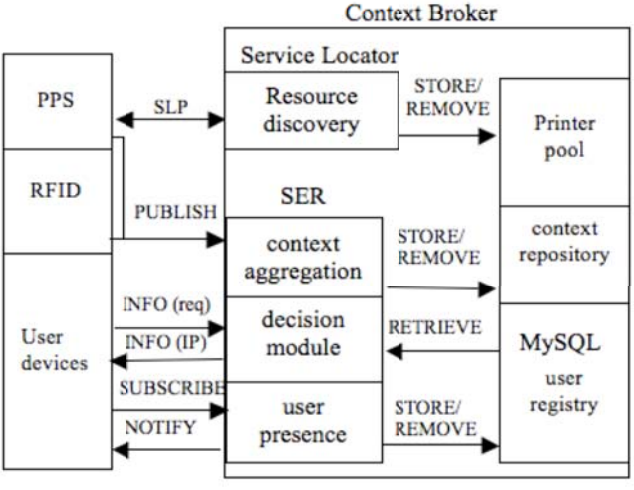

Fig. 2. Architecture of the Context Broker and interoperability with the user terminals and context providers. Context providers (PPSs and RFID readers) transmit PUBLISH messages to the aggregator, with XML-encoded context as payload. Users register with the CB using SIP SUBSCRIBE messages. After subscribing, clients are eligible to place print requests using INFO messages. The decision module uses the context stored by the context aggregator to deduce a suitable printer and returns its IP address to the requestor. The Context Broker automatically discovers printing resources using the Service Location Protocol.

In order to renew their registration, mobile terminals have to periodically transmit SUBSCRIBE requests. In addition to handling subscriptions, the presence module also transmits NOTIFY messages to subscribed devices. These responses concern feedback about printing sessions (i.e., documents remaining in the queue and current printer status) or location information (see section 3.3). The presence module interoperates with the context aggregation module, in order to provide this information to the terminals. Together these modules implement a *presence server*, a logical entity capable of acquiring, storing, and transmitting context.

After establishing a subscription to the context broker, a mobile terminal is able to send printing requests. This is done by sending SIP INFO messages to the context broker [17]. Acceptance and processing of printing requests is handled by the decision module, which collects relevant context stored in the database, and responds by returning the IP address of a suitable printer to the requestor (see figure 3).

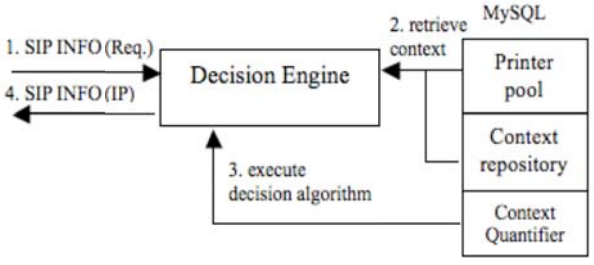

Fig. 3. Handling of printing requests in the decision module of the context broker. An incoming request triggers the execution of the decision algorithm, which in turn uses information stored in the database to select a PPS. Subsequently, it sends a response with the PPS's IP address to the requestor. Responses and requests are carried as SIP INFO payloads.

The module includes a decision algorithm, which estimates the *suitability index* (SI) of each of the PPSs, based on the retrieved context information. Our solution is based on printer status quantifiers, where each state is assigned a *weight*. The larger the value of the weight, the greater its contribution is to the SI. The same approach is employed for user location. In this case, there is an inverse relationship between the value of the weight and the SI. For our proof of concept system, we have hardcoded

Table 1. List of parameters contributing to suitability index

Location Status	$L(p_i)$	Description
Distance from PPS	$\frac{1}{d_i}$	d_i: Distance from PPS in meters (hardcoded into the database).
Insufficient RFID reader data.	0	No RFID footprint of the user's device on any of the readers for the last 60 seconds.

User Preferences	$U(t_i)$	Description
Color printout	$c_i = \begin{matrix} 0.5 \text{ supported} \\ 0 \text{ not supported} \end{matrix}$	d_i: Distance from PPS in meters (hardcoded into the database).
Resolution	$d_{res} = 0.25 \cdot \frac{a_{dpi}}{b_{dpi}}$	a_{dpi} resolution of printer, r_{dpi} requested resolution
Double-sided printing	$ds_i = \begin{matrix} 0.5 \text{ supported} \\ 0 \text{ not supported} \end{matrix}$	True/false operation

Printer Status	$S(p_i)$	Description
Busy	$\frac{0.5}{q_i}$	q_i: Number of pending print jobs in the PPS.
Idle	0	True/False operation
Offline – Permanent	-1	Errors can be: network connection (PPS does not respond to keepalive requests), paper jam (communicated from the PPS).
Offline - Temporary	-0.5	Printer in PPS may require user intervention to resume printing (no paper, low toner)

the weights into a database table called *Context Quantifier*. The SI for a given PPS is the sum of the status, location, and user preference weights. The algorithm returns the IP address of the PPS with the largest SI.

Table 1 gives some example weights for user location and printer status. It is important to note that the formulae in this table are representative of the operating environment of our system (see figure 2), coupled with the average user requirements for this environment. Future versions of the broker will allow for customization of the Context Quantifier. Note that these weights are fixed at the time of system initialization.

When referencing the table of weights, the SI for each printer p_i, given a print request from a terminal t_i, is computed using the equation $SI = S(p_i) + L(p_i) + u(t_i)$.

For example, assume that a user wants to print a color photograph. Initially, the user configures their printing requirements via the client application in their mobile terminal (see section 3.4). This information is propagated to the Decision Engine, as payload in a SIP INFO request message (see figure 4). The decision engine retrieves printer and location context from the local database, aggregates the user preferences and executes the decision algorithm. The algorithm consults the list of available printers and evaluates their SI, returning the IP address of the printer with the largest SI value (see table 2 for this example).

Table 2. Example of calculated weights for printer, user and location context

Printer 1		Printer 2	
Context	Weight	Context	Weight
$S(p_i)$: Printer status: Busy (1 document pending)	0.5	$S(p_i)$: Printer status: Idle	1
$L(p_{ii})$: 15m from printer (recorded 20 seconds ago)	0.066	$L(p_i)$: 30m from printer (recorded 20 seconds ago)	0.033
$u(t_i)$: Printer supports color, 1200 dpi resolution (requested 2000dpi)	0.65	$u(t_i)$: Printer black and white, 2000 dpi resolution (requested 2000dpi)	1
SI	1.216	SI	2.033

3.2 Printing Point Services

Printing Point Services combine a Service Agent (SA) with a physical connection to a printer. This SA is software that acts as an intermediary between users and printers. This software runs on a computer that is connected to a printer, and has network reachability from the various clients (i.e., the applications run by users to print files). This SA plays a double role: (1) processing requests to print documents (from the users) and (2) acquiring and providing printer context to the context broker. The SA is highly configurable and scalable and may support more than one network interfaces, a wide range of printers (either network-attached printers, or printers directly connected to the SA).

The SA builds on standards, both for printer context aggregation, and for printing. We have defined a simple communication protocol for negotiating with devices. In our architecture, we propose to use *tight coupling* between a service agent and a printer (see figure 4). We believe that a single interface to the printer enhances the role of the service agent as a gateway for providing print services (e.g. it allows for policy-based control of printing and is more secure).

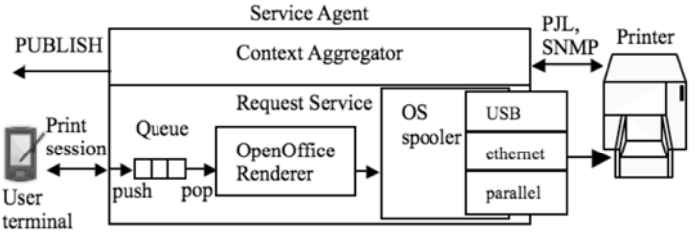

Fig. 4. Block structure of a Printing Point Service. Tight coupling of printer and service agent allows for control of printing requests. In addition, support of multiple interfaces allows for use of a wide range of printers.

In order to successfully support printer awareness, a service agent needs to successfully retrieve static and real-time information from a printer (see section 2.2). In this study, we have used HP's Printer Job Language (PJL) to get static information about the printer [18]. PJL was developed in order to support printer status reading from the printer to the host computer, and has been adopted by all major printer manufacturers. In our system, static information inquiries are only done during start-up of the SA. As soon as the necessary information has been gathered, an SLP message is sent to the context broker, informing it of the URI address of the service agent, as well as the retrieved printer information as attributes (see table 3).

Table 3. Construction of an SLP message using information obtained through PJL commands

PJL command, as issued from Service Agent	Reply from the printer	SLP Message
PJL INQUIRE RESOLUTION	1200	URI:service:printer:lpr://192.168.1.12 Attributes:(printer-model=*LJ4M*),(printer-document-format-supported=*application/postscript*),(printer-color–supported=*false*), (printer-resolution-supported=*1200*)
PJL INFO ID	LJ 4M	
PJL INQUIRE RENDERMODE	GREYSCALE	
PJL INFO LANGUAGES	POSTSCRIPT	

Although PJL has a status command, tests in our lab indicated that some printers are slow to respond to PJL commands. However, response time is a critical factor when retrieving printer status, therefore, in order to collect real - time information,

we make use of managed objects (defined here as printer information resources). Such objects are stored inside a database known as Management Information Base (MIB). Inside each MIB the objects are organised hierarchically and are accessed using a network - management protocol such as SNMP. The identity of each object is known as an object identifier. RFC1759 indicates that proper combination of information from printer objects, leads to an accurate printer status inference [19]. We distinguish between four printer states: busy, idle, alert, and offline. Alerts (low toner and low paper) indicate that future print requests are likely to generate error messages. An offline state conveys irreversible errors, requiring human intervention (out of paper, jammed, paper tray open, and network error).

The status is encoded in an XML document and periodically transmitted to the context broker. The document is modeled after the Printer Working Group (PWG) semantic model, with a collection of documents extending the XML Schema and describing printer characteristics (The PWG schema is currently a candidate standard.) [20]

In addition to the printer's status, information about the printer's location and name are transmitted, as well as the IP address of the SA. Additional information necessary for the communication between the mobile clients and the SA is appended. We chose to transmit this information in real-time, because it may change (for example, a SA might change its IP address, the printer may be relocated, a new printer might replace an existing printer, etc.). In many of these cases, a restart of the SA is not required, as the broker will be notified about the changes automatically.

The negotiation process for printing a document occurs between the device and the service agent, after the discovery phase, on a per-request basis. Note that this process starts immediately after the user terminal receives the IP address of an available PPS, from the context broker (see section 3.1). For this negotiation we implemented a simple UDP protocol. After receiving an IP address from the context broker (see section 3.1), the client sends a UDP request to print to the PPS. Based upon the relevant policies, the identity of the user, the status of the printer, etc., the service agent chooses whether to allow or deny the printing request. If the request is allowed, the client starts an FTP session with the service agent in order to transfer the document.

The printing queue of the service agent is a queue: a process monitors the queue and while there are pending jobs, it extracts the first job from the queue and sends it to the printer. Note that in some cases, it is permissible to prioritize one job over another based on criteria other than the time of arrival of the document. Therefore, the queue also includes a prioritize function, which is used by the decision module.

Once a document is dequeued it is ready to be sent to the printer. Note that contemporary printers generally expect Adobe's PostScript®, a page description language [21]. In order to convert the file to this format (if necessary), we use Open Office, an open source office suite, which can print all major file formats (documents, spreadsheets, images, etc.). Therefore, the document to be printed is sent to the corresponding open office application (i.e. there is a different application for spreadsheets, presentations and documents), which in turn outputs the document in PostScript format.

The service agent makes use of printing profiles. These profiles correspond to different settings for the same printer. Parameters that can be set are the number of

pages per sheet, number of copies of the document, printing quality, colour printing (if supported), and others. The idea is that printing profiles fine-tune the printing, thus enhancing the user's experience. Profiles can be customized to specific users and are created by the service agent, based on the feedback from the previous choices of each user. We plan to show that in the long term, such profiles can accommodate the printing preferences of the users, without users having to tweak their printing preferences every time that they want to print something (see section 3.4).

3.3 RFID Infrastructure

Acquiring location information in a mobile computing environment is not a trivial task. Hazas, et al. overview different location sensing technologies and show that they differ in accuracy and deployment feasibility [22]. Based on this study, we chose RFID as a technology for inferring user location, as it addresses our accuracy, scalability, and availability requirements.

RFID readers in proximity of the RFID tag can read the tag and report having seen this tag to the context broker. Such readers are strategically placed throughout the context-aware environment.

The TagMaster AB LR-6 readers used for our proof of concept system were running an embedded version of the Linux operating system. This enabled us to create our own software which enables the reader to act as a context provider, i.e., the reader sends a PUBLISH message to the context broker, containing the MAC address of the mobile device. The reader locally stores an internal table of <tagID, MAC> pairs, which is used to match authenticated tags against the MAC address of the device they belong to. This mapping between tag number and device ID is done locally on the TagMaster device in order to avoid transmitting the tag ID, thus protecting the privacy of the users.

3.4 Mobile Application

A proof-of-concept application was written in C# and runs on Windows Mobile 2003 platform (both PDAs and smartphones). It requires some form of IP connectivity (e.g. Wi-Fi) and reachability to the service agents and the SIP proxy (in the case of the subscription-based approach). Since printing is usertriggered, the user starts the application, selects a document, and waits for feedback from the system.

The application itself is designed to proactively adjust to the user's preferences. Initially the user browses the filesystem of their mobile device and selects a file to print. The application automatically makes a rudimentary match of the file to one of three categories: Presentation, Document, or Photo. This match is currently based on the extension of the file's name. However, future versions of the application should allow for recognition of the file type, based on a file header signature [23].

Depending on the category and the capabilities of the printer, different printing options are available. For example when printing presentation slides, the user selects how many slides per page he/she wants to print. There is also an option for double-sided printing. For documents, additional options are color printing, and a printing speed/quality adjustment. For pictures, the user gets a preview of the picture he/she

wants to print (the default choice in this case is to print the picture using the highest quality settings, so there is no adjustment for printing quality, as there is on the documents). All categories include a number of copies setting.

The long-term objective is to minimize user input; therefore the application stores a history of selected printing options, aggregating them into a dataset. This implies that there is a training period, in order for the application to have sufficient data. During this period the user is presented with all the printing options. We found that the number of samples in the dataset required to reach a statistically safe decision may vary depending on the user (see section 4).

Using past behavior and the category of the file to print, we apply a probabilistic algorithm to make an informed decisions about the printing options to present to the user. The algorithm is based on the principle that *given a new print request, a print option "O", "n" number of previous requests, and "d" times of previously selected option "O", calculate the likelihood that the user chooses option "O" for this request.* Mathematically, this can be expressed as the probability $P(O)= d/n$. On every iteration, the algorithm calculates P(O) for every printing option, then classifies the options in order of previous occurrence. This step also determines which options will be visible to the user (see section 4.1) for hiding these options. Options are presented to the user in such a way so that the ones more likely to be chosen are shown in a more visually prominent position in the client's application window (see figure 5)

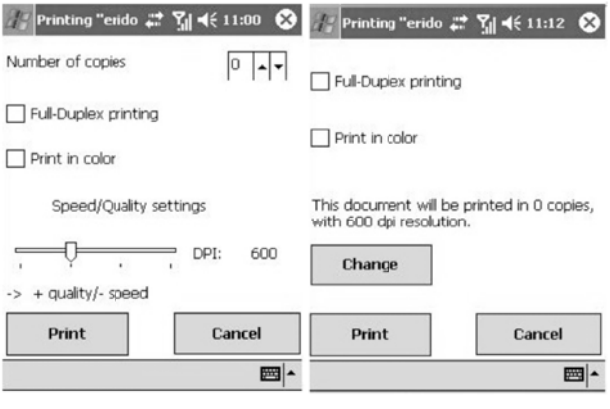

Fig. 5. Options presented to the user from the client application when printing a document: Each option O_i is presented in its own visual space (i.e. a different "line") on the client window. When $P(O_i)$ is above a certain threshold, the designated option is visible. Upon initial use of the application, all options are visible (a). However after a period of use, it is possible that some options are preselected automatically, depending on the frequency they are chosen by the user (b). This limits the amount of input required, therefore accelerating the printing process.

4 Measurements

In order to properly monitor the system in real-world operating conditions, we deployed a prototype in our lab and monitored the system for a period of five consecutive (working) days. In an effort to promote user diversity (as it would occur in real-world situations), six different clients were given access to the system, during working hours. In order to establish the efficiency of the decision algorithm of the context broker, we measure the ratio of "true positive" versus the total number of print sessions. A true positive result occurs when the decision algorithm works as expected, i.e. the user is able to print to a printer best approximating a preset, prioritized set of requirements (see section 3.1).

Figure 6 illustrates the performance of the decision algorithm as a histogram. The results represent the collective percentage of true positive occurrences for the period of five days. The true positive occurrences are at least 80% of the total. This 20% error margin is attributed to false parameter estimations due to synchronization errors: as some service agents did not report the current status of the printer in time to the context broker.

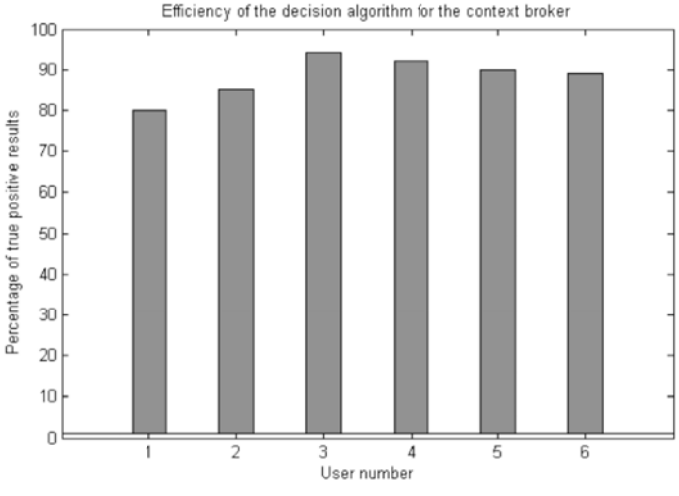

Fig. 6. Efficiency of the decision algorithm in the Context Broker. Efficiency is determined by the percentage of correct results from the algorithm (i.e. true positive outcome)

When evaluating the performance of the decision algorithm in the user application, the question was how large should the training set be for this algorithm, i.e. how many print sessions are required a priori, in order for the algorithm to generate statistically safe decisions.

Figure 8 illustrates an experiment in which we executed the decision algorithm for every client, with a different dataset size each time. In this case, a true positive result indicates that all desired options were visible to the user beforehand (i.e. the user did not have to push the "change" button as seen in figure 7). Overall we conducted 8 different measurements, with an increase in 5 print sessions per step. The results

indicated a large improvement for a dataset size up to 35 (which corresponds to an approximate 83.8 % average of true positive occurrences). After this point, the gains are not substantial enough to justify a further increase in the number of samples; therefore the default size was set to 35. It is important to note that in these empirical measurements, we are not concerned about absolute values but rather simply observing trends in user behaviour.

We have conducted a series of measurements concerning various aspects of performance of the context broker, the centrepiece of this system's architecture. Initially, we monitored the average response time, from when an incoming SIP INFO message is received, until the response is sent in another INFO message. Note that this value is also influenced by external factors such as printer availability, user preferences, and location (in these measurements the propagation delay over the carrier network is not taken under account, since we assume that there is sufficient capacity to serve all the requests). In practical terms this is an important measurement because it indicates how long the user has to wait in order to access a printer. In a typical working environment, prompt handling of print requests is vital.

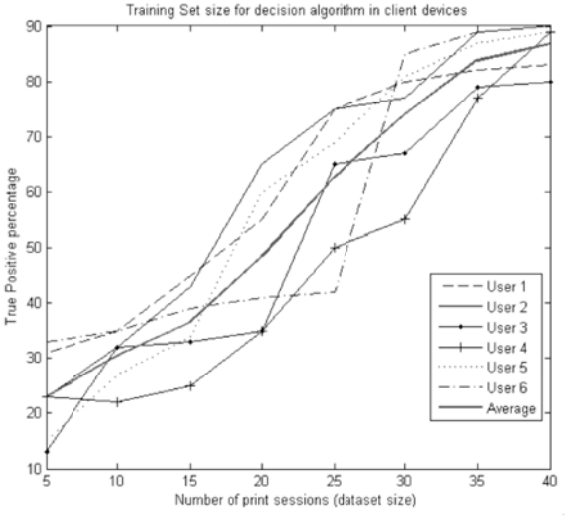

Fig. 7. Determining the optimal dataset size for statistically safe decisions

In general, we found that the demand was higher during working hours (typically in the middle of a day), and that the response times were between approximately 0.5 and 0.8 seconds (see figure 8). This can be interpreted as a positive outcome, since there were no prominent delays observed.

We also subjected the system to a number of stress tests. This helped to identify the thresholds of stable operation. This is important when the system is deployed on a large-scale, with many resources and users. Our test involved testing the context aggregation module (see section 3.1). In particular we wanted to learn the maximum frequency of PUBLISH messages that this module is able to accept and successfully

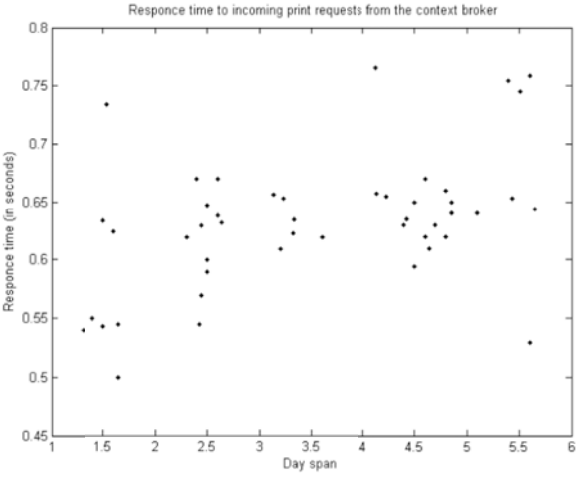

Fig. 8. This scatter plot showing responce times to print requests made from the users to the Context Broker for all 5 days of our experiment

store the updated presence information in the database. We created a virtual RFID reader, which sends PUBLISH updates to the presence server. Figure 9 illustrates the error rate observed for a given frequency of incoming PUBLISH updates. Beyond 400 messages/sec, we observe a gradual increase in the error rate, which is very prominent after 1200 messages/sec mark. This means that our CB in its present form can sustain up to 400 presence user agents (assuming that these agents send periodic updates every second); larger systems have to distribute the handling of PUBLISH messages between multiple CBs.

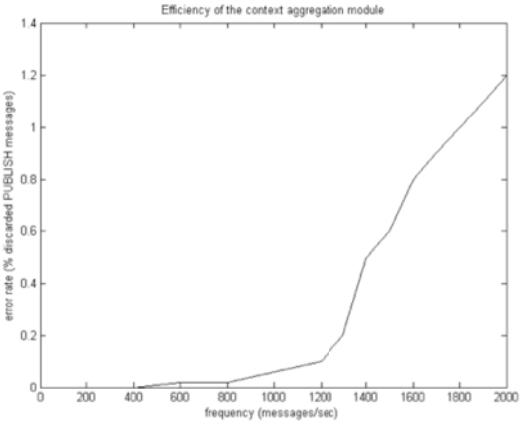

Fig. 9. This plot showing the percentage of discarded PUBLISH messages from the CB, depending on the volume of incoming messages

5 Conclusion

In this paper we proposed a mobile printing solution for workspace environments. This solution implements a policy-based decision algorithm for forwarding print requests from mobile devices to appropriate printers. The objective was to provide the user with a transparent printing service allowing him/her to print the files on his/her mobile device. We have currently tested the solution in our research lab, but plan to extend testing to a corporate environment.

We are also working on a security module for the context broker which works both for the service agents (which first have to obtain a digital certificate from the module, in order to communicate), and for the mobile terminals (which first have to form an SSL tunnel to securely interface with a service agent). Note that much of the proposed infrastructure is compatible with the 3GPP IP Multimedia System (IMS), so an area of future work is to further evaluate the integration of such a context aware approach with IMS.

Finally, although this paper presented a printing service, our ambition is to use the same infrastructure (i.e. the context broker and RFID readers) to provide more services in the future. Examples of such services include automatic presentation arrangement, meeting room reservations, information guides, etc.

References

[1] Blando, P.: Printing statistics from University of Davis in California (annual from 1996 to 2004), http://clm.ucdavis.edu/pubs/labrep/summer2004/stats/print.html (visited July 8, 2008)

[2] Minnesota State University Technical Services Group, Annual statistics from various sites at Minnesota State University (annual from 2004 to 2007), http://mavdisk.mnsu.edu/bryan/mavprintstats07.htm (visited July 8, 2008)

[3] Wright, D.: The ubiquity of print. Lexmark Technical Report, http://www.w3.org/2006/02/slides/wright.pdf (visited July 8, 2008)

[4] Microsoft Knowledge Base, Agreement between Microsoft and IHVs (Independent Hardware Vendors), http://support.microsoft.com/kb/156082 (visited July 8, 2008)

[5] McGrath, R.E., Ranganathan, A., Campbell, R.H., Mickunas, M.D.: Incorporating semantic discovery into ubiquitous computing infrastructure. In: Proceedings of System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp 2003) (October 2003)

[6] Burke, P.: Mobile Printing. In: W3C Mobile Web Initiative Workshop, Barcelona, Spain, November 18-19 (2004)

[7] HP mobile printing SDK for mobile devices (discontinued), http://h20000.www2.hp.com/bizsupport/TechSupport/Home.jsp?lang=en&cc=se&prodTypeId=18972&prodSeriesId=352517&lang=sv&cc=se (visited July 8, 2008)

[8] Nokia, Xpress print, http://europe.nokia.com/xpressprint (visited July 8, 2008)

[9] Camera and Imaging Products Association (CIPA), Digital Photo Solutions for Imaging Devices, whitepaper (February 2003), http://www.cipa.jp/english/pictbridge/DPS_WhitePaper_E.pdf (visited July 8, 2008)
[10] WestTek, JETCET Print application for Windows CE, http://www.westtek.com/pocketpc/jetcet/ (visited July 8, 2008)
[11] Mobile Imaging and Printing Consortium: Implementation Guidelines for Printing with Mobile Terminals, version 2.1, http://www.mobileprinting.org/developers/guidelines/ (visited July 8, 2008)
[12] Iptel. org., SIP Express router - high-performance, configurable, free SIP server licensed under the open source GNU license, http://www.iptel.org/ser/ (visited July 8, 2008)
[13] Eslami, M.Z.: A Presence Server for Context-Aware Applications. Masters Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden (December 2007)
[14] Niemi, A. (ed.): RFC3903, Session Initiation Protocol (SIP) Extension for Event State Publication (October 2004), http://www.ietf.org/rfc/rfc3903.txt (visited July 8, 2008)
[15] Sugano, H., Fujimoto, S., Klye, G., Bateman, A., Carr, W., Peterson, J.: RFC 3863, Presence Information Data Format (PIDF) (August. 2004), http://www.ietf.org/rfc/rfc3863.txt (visited July 8, 2008)
[16] Rosenberg, J.: RFC 3856, A Presence Event Package for the Session Initiation Protocol (SIP) (August 2004), http://www.ietf.org/rfc/rfc3856.txt (visited July 8, 2008)
[17] Donovan, S.: RFC 2976, The SIP INFO Method (October 2000), http://www.ietf.org/rfc/rfc2976.txt (visited July 8, 2008)
[18] HP, PCL/PJL reference - Printer Job Language Technical Reference Manual, http://h20000.www2.hp.com/bc/docs/support/SupportManual/bpl13208/bpl13208.pdf (visited July 8, 2008)
[19] Smith, R., Wright, F., Hastings, T., Zilles, S., Gyllenskog, J.: RFC1759, Printer MIB, section 2.2.13.2, http://www.ietf.org/rfc/rfc1759.txt (visited July 8, 2008)
[20] Printer Working Group (PWG) Semantic Model – Candidate standard, ftp://ftp.pwg.org/pub/pwg/candidates/cs-sm10-20040120-5105.1.pdf (visited July 8, 2008)
[21] Adobe Systems Incorporated, Postscript Language Reference, 3rd edn., http://www.adobe.com/products/postscript/pdfs/PLRM.pdf (visited July 8, 2008)
[22] Hazas, M., Scott, J., Krumm, J.: Location-Aware Computing Comes of Age. IEEE Computer 37(2), 95–97 (2004)
[23] Kessler, G.: File signatures table, http://www.garykessler.net/library/file_sigs.html (visited July 8, 2008)