P2P M-Commerce in Pervasive Environments

SASIKANTH AVANCHA, PETER D'SOUZA, FILIP PERICH, ANUPAM JOSHI and YELENA YESHA
University of Maryland Baltimore County

Pervasive environments are characterized by the presence of wireless devices ranging from sensors to mobile PDA-phones to laptops. Peer-to-Peer (P2P) information sharing in such environments presents a tremendous opportunity for people and devices to exchange information such as music, pictures, documents and stock tickers with peers. Information that has a monetary value introduces a whole new set of problems for P2P interaction in pervasive environments. In this work, we present the design of a middleware that attempts to solve these problems using concepts of the Contract Net Protocol, Semantic Service Discovery and secure transaction protocols.

Keywords: P2P, M-Commerce, Pervasive, Ubiquitous

1. INTRODUCTION

In the information age we live today, the importance and usefulness of timely information cannot be overstated. The traditional notion of "timely information" is restricted to information related to dynamic events, such as political events, stock market information and weather information. In this work, we also consider static information, such as a clip art image, a song in MP3 format and a text document, that is required and available on-demand as "timely information". Currently, the primary repository of both dynamic and static information is the Internet. One can access CNN's website and obtain news, stock quotes and weather information. One can also issue queries in Google to search and obtain, say, a clip art image from the WWW. The most common method used to access the Internet has been, and will continue to be, a desktop computer running a full-featured operating system. Additionally, people are using devices, such as PDAs and cell phones, to run local applications and access the Internet. The mobile devices are enabled with some wireless technology, such as GSM, 802.11b and Bluetooth, which allows them to connect wirelessly to the Internet. The use of wireless devices that are untethered has lead to the concept of *anywhere, anytime* information access. In this paradigm, mobile devices establish a wireless connection to an "access point", which provides Internet connectivity. It is envisaged that people will use mobile devices in the future for more than simply browsing the web. Internet applications such as Gnutella have introduced the concept of information sharing in a *Peer-to-Peer or*

Authors' Address: Dept. of CSEE, UMBC, 1000 Hilltop Circle, Baltimore MD 21250
This work was supported by NSF awards IIS 9875433 and CCR 0070802.

P2P [Gerwig 2001] paradigm, where everyone on the Internet is equal in terms of being providers or consumers of information. It is straightforward to extend this model to the wireless world because most of the information continues to travel through the wired infrastructure. In addition to P2P information sharing, people are expected to engage in e-commerce using their mobile devices, i.e., engage in *m-commerce* [Varshney et al. 2000]. Provision for secure transactions is a very important component of an m-commerce system. As devices become computationally powerful, they can execute cryptographic algorithms in order to securely perform transactions with businesses on the web. Thus, "timely information" now possesses an additional attribute: monetary value.

The next logical step in m-commerce is *Peer-to-Peer m-commerce*. We combine the notions of P2P information sharing with the *anytime, anywhere* information access model and the m-commerce paradigm to enable P2P m-commerce in pervasive computing environments. We have designed and implemented a middleware prototype that uses concepts of Contract Net Protocol [FIPA 2001], Semantic Service Discovery [Avancha et al. 2002], and secure transactions to enable P2P m-commerce. The middleware provides a mechanism to link sellers of information to potential buyers. It targets transactions that range in cost from a few pennies to a few dollars. Any m-commerce transaction typically involves transfer of information in two stages – the transfer of payment from the buyer to the seller, and the transfer of the information (goods) from the seller back to the buyer. While the design of such transactions can be optimized in m-commerce systems, additional attention now needs to be paid to issues such as atomicity of transactions, and security of the goods/information that is received electronically. The system proposed here seeks to address these issues in a simple, efficient and reliable manner.

The rest of the paper is organized as follows. In section 2, we motivate the need for our middleware with a futuristic scenario. In section 3, we discuss the design of our prototype. Section 4 presents implementation details. We conclude and discuss future work in section 5.

2. A SCENARIO FOR THE FUTURE

Consider the following scenario. Bill is a student at UMBC, working hard to finish a class project. He is sitting in the Commons, working on a Microsoft PowerPoint presentation, which is due in class in 1 hour. At one point, while describing the motivation for his project, he decides to add the most appropriate clip art image to depict his motivation. His laptop is already connected to the Internet via the 802.11b based Wireless LAN in the Commons. He decides to surf the web to find the image. Unfortunately, all the images Google retrieved in the search prove to be inadequate. Bill then decides to use our middleware to *discover* the most appropriate image. He uses a simple GUI to fill in a form that describes what he wants and submits his query. Part of his input specifies how much he is willing to pay for the image and the quality of the image he requires. After this, Bill goes back to his presentation unaware of the inner details of the middleware's functioning. A few minutes later, the GUI flashes indicating that the required clip art image was successfully obtained, that Bill had to pay $2.00 for it, and that the image quality is 98%.

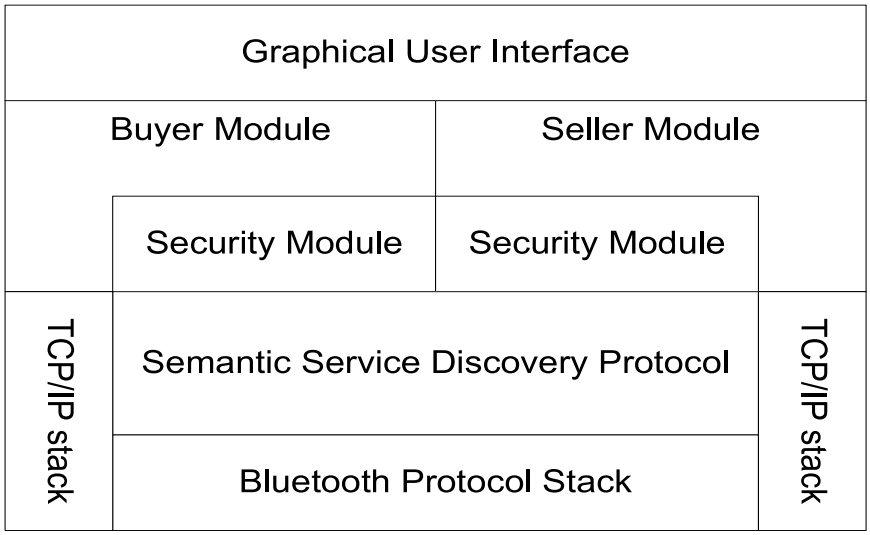

Fig. 1. Architecture of the middleware

In this scenario, the required information is static, but is associated with a definite time-bound. Bill's laptop used this temporal information in the query specification, so that other peer devices that possess the required clip art can determine whether they can meet this time bound or not. In addition, Bill's laptop used a description of the clip art image, consisting of a set of well-known attributes, such as format (GIF, EPS or JPEG), dimensions, size, and keywords. The laptop uses Bill's cost and quality specifications to evaluate potential responses and choose a response that most closely matches Bill's requirements. Finally, the laptop establishes a secure channel with the peer it has decided to buy the image from, sends an encrypted payment and receives the encrypted goods.

3. SYSTEM DESIGN

The middleware consists of four primary components – buyer, seller, security, and semantic service discovery. Figure 1 shows a schematic of the middleware stack consisting of these four components.

We describe each of these components briefly in the following sections. First, we describe the contract net protocol used by the buyer and seller components to engage in a successful transaction.

3.1 Contract Net Protocol

The contract net protocol used by the middleware is based on the protocol [FIPA 2001] specified by the Foundation for Intelligent Physical Agents (FIPA). Figure 2 shows the message sequence between the buyer and seller following our protocol. The initiator of the protocol broadcasts the Call for Proposal (CFP). One important parameter in the protocol, not shown in the figure, is the deadline specified by the initiator or the buyer. The buyer is allowed to specify a time deadline after which it will ignore all further responses (bids) to that particular CFP. This helps prevent

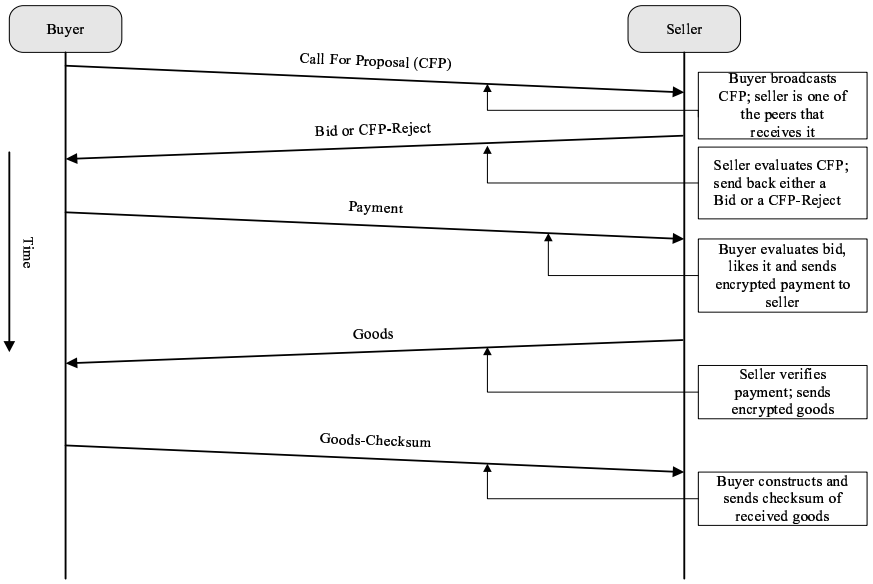

Fig. 2. Message sequence in the contract net protocol

the initiator from waiting indefinitely, if nobody decides to propose a bid. A critical component of the interaction is the exchange of payment and goods between the two parties in a secure manner. Thus, the two parties set up a secure session before exchanging encrypted data.

3.2 The Buyer

The buyer module is responsible for generating and transmitting CFP and payment messages. As shown in Figure 1, the buyer module is tightly coupled to the user input module. One of the main functions of the buyer module is receiving input from the user and creating a CFP marked up in the DARPA Agent Markup Language[1] (DAML+OIL). We discuss the reasons for using a semantic language and the issues associated with it in section 3.5. To create the CFP, the buyer module accepts user input in the form of key-value pairs and fills in a CFP template that is marked up in DAML+OIL.

This CFP is compressed and *broadcast* over the Bluetooth interface via the Bluetooth Service Discovery Protocol (SDP) layer. The advantage of using SDP as the transport layer is that it already contains the mechanisms to treat incoming messages as service discovery requests. Additionally, we use SeSDP, an enhanced version of Bluetooth SDP [Avancha et al. 2002]. We discuss SeSDP in greater detail in section 3.5. After the deadline set by the user has passed, the buyer module instructs the SeSDP module to evaluate all received bids. This ensures that the SeSDP module does not consider bids received after the deadline for evaluation. The buyer module then waits for the SeSDP to provide it with information related to the winning bidder, including the cost of the item. Subsequently, the buyer module

[1] http://www.daml.org

processes the information and constructs the encrypted payment using the security module. We explain the functionalities of the security module in detail in section 3.4. This encrypted payment is placed in a *payment* message, also marked up in DAML+OIL. The buyer module compresses the payment message and *unicasts* it to the seller winning the contract. Following the transmission of the payment, the buyer module assumes that the seller will send the goods in an encrypted form and, therefore, prepares to receive and verify the goods. The module then constructs a checksum on the item and unicasts it to seller, who can ensure that the item was received in order. Finally, it contacts the NetBill [Cox et al. 1995] server to obtain the decryption key and decrypt the item.

3.3 The Seller

Most of the functions performed by the seller module are complementary to those performed by the buyer module. The seller module has been designed to handle user input corresponding to a potential bid, and also to handle *payment* messages and *goods-checksum* messages. As explained in section 3.5.2, the SeSDP handles CFPs in terms of evaluation and transmission of the appropriate response. Thus, the seller module is mainly responsible for ensuring that potential bids are present in the local knowledge base, so that the SeSDP module can use them for CFP evaluation.

If the user wishes to store a bid in the local knowledge base for future use, she may present it in the form of key-value pairs which are converted to DAML+OIL using the appropriate Bid template. The seller module arranges for this bid to be parsed and converted into triples [Avancha et al. 2002] that are loaded into the knowledge base. If the bid sent by the SeSDP was successful, the buyer will unicast a *payment* message containing an encrypted payment. On receipt of this encrypted payment, the seller module first verifies it to identify the sender of the payment. This payment is encrypted, so it has to be transmitted to the NetBill server for decryption and crediting to the seller's NetBill account. Once the payment has been verified, the seller must construct the encrypted item (goods) using the security module. The encrypted item is placed in a *goods* message marked up in DAML+OIL. The message is compressed and unicast to the buyer that sent the payment. The buyer module sends this seller a *goods-checksum* message that contains a checksum computed by the buyer on the received goods. The seller module compares this checksum with the locally computed checksum before transmitting the encrypted goods. It does this to ensure that the buyer received the goods in order. If the local and received checksums do not match, the seller will attempt to resend the *goods* message.

3.4 The Security Module

The NetBill server functions as the governing entity in our security model. It's primary functions are:

—assignment of certificates to users,

—maintenance of account information of its members,

—tracking sessions and the associated key values used for encryption of the goods, and

—verification of financial information provided by buyers.

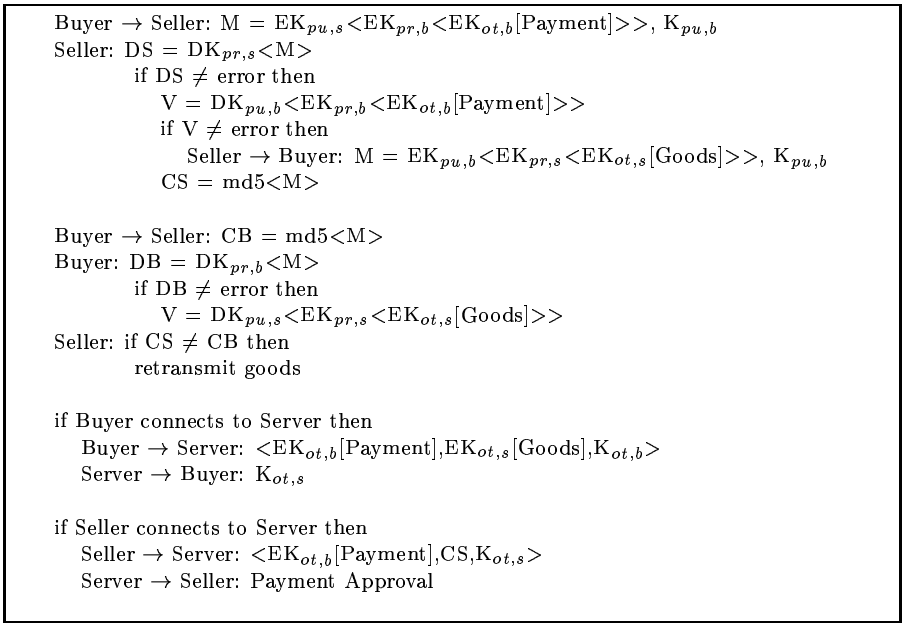

Fig. 3. Secure interaction between buyer, seller and server

The fundamental concepts underlying the security model have their roots in the *NetBill Protocol* developed at CMU, that addresses the issues related to the transfer of financial information and information goods over the network. It should be noted that the NetBill server is connected to the wired infrastructure. The peer devices are responsible for authentication, establishment of secure sessions and exchange of *encrypted* payments and goods. The process of verification and decryption of payments/goods is *deferred* until the peers connect to the wired network to access the NetBill server. It is not necessary for both, the buyer and seller, to connect to the NetBill server at the same time. The server possesses enough information to process the transaction partially until the other peer completes it. The steps followed by the buyer, seller, and server using the security protocol are shown in Figure 3. We use the following notations in this figure: $K_{pu,s}$, $K_{pr,s}$, $K_{ot,s}$ refer to the seller's public, private, and one-time key, respectively; $K_{pu,b}$, $K_{pr,b}$, $K_{ot,b}$ refer to the buyer's public private, and one-time key, respectively.

Both, the payment and goods, are sent encrypted and signed to prevent replay attacks. The secure session is established by the application of the final encryption on intermediate data using the peer's public key. Signatures ensure identification of the transmitting party at the receiver. The buyer transmits an encrypted, signed payment to the seller over the secure session. After receiving the encrypted payment and verifying the signature, the seller transmits encrypted, signed goods to the buyer over the secure session. It also computes and stores an MD5 checksum over the transmitted data. The buyer, on reception of this data, computes its MD5 checksum and transmits it back to the seller. The ensures correct reception of the goods at the buyer. The buyer cannot access the goods until it connects to

the NetBill server to receive the decryption key. Once the checksums match, the seller creates a digitally signed invoice consisting of the payment, checksum and decryption key for the goods. The seller transmits this invoice to the NetBill server for further processing. If the buyer has the necessary funds or credit in his/her account, the server debits the buyer's account and credits the seller's account, logs the transaction and saves a copy of the decryption key. This key can then either be sent to the buyer directly or be requested by the buyer. At the same time a digitally signed message containing an approval, or an error code indicating why the transaction failed, is sent back to the seller. It is clear from Figure 3 that the NetBill server is used only at the end of the transaction between the buyer and the seller. Both parties possess information, albeit unusable, at the end of the transaction, but before contacting the server.

3.5 The SeSDP

The SeSDP module handles all the six message types shown in Figure 2. It performs different functions depending on whether the message is for the buyer module or the seller module. The SeSDP consists of three main components – the Bluetooth SDP, the DAML+OIL parser called Wilbur [Lassila and Theran 2001], and the Prolog-based Reasoning Engine (RE) and Knowledge Base (KB). To understand and validate incoming messages, the SeSDP uses an ontology, marked up in DAML+OIL, that describes terms related to P2P m-commerce. Additionally, the ontology contains definitions of items that can be exchanged in a P2P environment. We assume that the ontology in the pervasive environment is shared by all peers in the environment, so that they can understand each others' messages. SeSDP handles service discovery requests constructed in either the Resource Description Framework (RDF) or DAML+OIL. Unlike other service discovery protocols, SeSDP attempts to match service requests to local information using semantics of service descriptions rather than their syntax. The combination of a semantic language that describes information in a detailed, structured manner, and a reasoning engine based on first-order logic for information discovery is ideal for pervasive environments. We have shown in [Avancha 2002] that such a combination helps conserve energy in mobile device by reducing the number of transmissions during service discovery. Figure 4 shows the algorithm used by SeSDP in determining the local description that most closely matches the description in the incoming request. The reasoning engine first validates the request, creates DAML+OIL specified constraints if required, and ranks attributes as specified in the description in the preliminary phase. In the next phase, it attempts to match incoming attribute values to locally stored values. Based on the specified constraints, the engine will return either exact, closest or no matches for a particular attribute. Due to space constraints, we cannot describe the functioning of SeSDP in greater detail. We refer interested readers to [Avancha 2002] and [Avancha et al. 2002] for detailed description and evaluation of SeSDP.

3.5.1 *The SeSDP on the Buyer.* The buyer-side SeSDP is designed to handle *bid*, *reject*, and *goods* messages. If the buyer module has broadcast a CFP, there may be multiple bid and/or reject messages in response to the CFP. If the received message is a bid, the SeSDP parses it using Wilbur, and loads it into the knowledge base. If it is a reject message, it stores information on the rejecting peer for possible

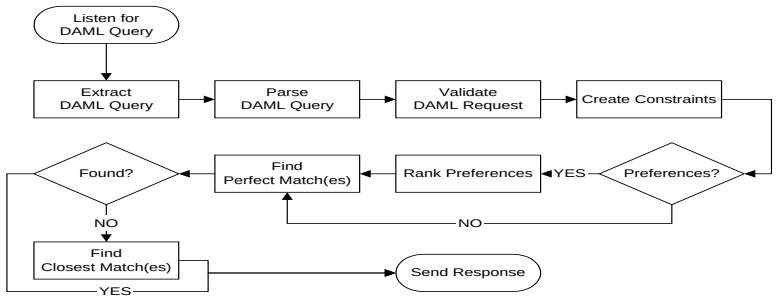

Fig. 4. Semantic matching algorithm in SeSDP

use later. After receiving the signal from the buyer module that the deadline has passed, the SeSDP evaluates all the bids by executing the reasoning engine. The result of this execution consists of information relating to the seller, such as *peerID*, and *cost of item*. The SeSDP module passes the seller information to the buyer module who acts upon it as described above. The next expected message in this sequence is the *goods* message. If this message is received, the SeSDP parses it and passes it to the reasoning engine, which extracts the encrypted goods and passes them to the buyer module.

3.5.2 *The SeSDP on the Seller.* The seller-side SeSDP is designed to handle *CFP*, *payment*, and *goods-checksum* messages, apart from the *local bid* messages sent by the seller module. If the device running the middleware is in range of the device that broadcasted a CFP, then the SeSDP module will receive it. First, it parses the message and passes it to the reasoning engine for validation and evaluation. Validation by the engine consists of determining whether the requested item is defined in the ontology or not. Requesting an undefined item results in a *reject* message to the buyer. Evaluation consists of determining if the value at least one of primary attributes of the specified item matches with that of the potential bid. An additional, important attribute to evaluate is the deadline or time limit specified in the CFP. A time limit value that is greater than the allowed limit also results in a *reject* message to the buyer. If the time limits match, then the matching bid message is compressed and unicast to the buyer. Information about the potential buyer is passed on to the seller module. In the case that the buyer-side SeSDP chooses this seller to buy the item from, then a *payment* message will be received. The seller module will wait for a fixed period of time for the payment, before reporting an error to the user. If the *payment* message is received before the timeout, the SeSDP parses it and passes it to the RE for validation and extraction of the encrypted payment. The encrypted payment is passed on to the seller module for further processing as described above. The next expected message in this sequence is the *goods-checksum* message. If this message is received, the SeSDP passes it directly to the seller module for verification of successful reception of goods by the buyer.

4. IMPLEMENTATION

We implemented the prototype of our middleware in the C language. We used laptops running Linux 2.4.7-10 and installed the Bluetooth protocol stack developed

by Axis Communications on each laptop. We included the enhanced SDP module with the protocol stack. We also downloaded and installed Allegro Lisp version 6.0, the DAML+OIL parser, and XSB (a Prolog variant) version 2.3 on each laptop. We executed the buyer, seller, SeSDP and reasoning engine as separate processes on each laptop. All modules communicated with each other using messages based on the message structure designed as part of the MoGatu framework [Perich et al. 2002]. In order to test the system, we provided a test file containing key-value pairs to the buyer module and thus initiated the entire protocol. We verified the system for correctness of execution in terms of order of messages, encryption of payment and goods, and verification of payment and goods.

5. CONCLUSIONS AND FUTURE WORK

We have presented the design of a flexible middleware system for m-commerce in pervasive environments. The protocol ensures security in transactions and integrity of the received goods. It also protects against misuse of information, both financial and goods, by all parties in the transaction. The architecture described here is well suited for supporting commerce in information goods; it can be extended to support other types of purchases like streaming video or software rental applications. Future work in this area consists of porting the system to smaller mobile devices, such as the iPAQ, and different network technologies, such as IEEE 802.11b. We also propose to conduct experiments to evaluate the performance of the system by varying information file sizes and information descriptions.

REFERENCES

AVANCHA, S. 2002. Enhancing the Bluetooth Service Discovery Protocol. M.S. thesis, Univeristy of Maryland Baltimore County.

AVANCHA, S., JOSHI, A., AND FININ, T. 2002. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, 96–99.

COX, B., TYGAR, D., AND SIRBU, M. 1995. NetBill Security and Transaction Protocol. In *First USENIX Workshop of Electronic Commerce Proceedings.*

FIPA. 2001. FIPA Contract Net Interaction Protocol Specification. *http://www.fipa.org/specs/fipa00029/XC00029F.pdf* .

GERWIG, K. 2001. Business: The 8th layer: Computing power to the people. *netWorker 5,* 1, 13–16.

LASSILA, O. AND THERAN, L. 2001. Wilbur RDF Toolkit. *http://wilbur-rdf.sourceforge.net.*

PERICH, F., AVANCHA, S., CHAKRABORTY, D., JOSHI, A., AND YESHA, Y. 2002. Profile driven data management for pervasive environments. In *3th International Conference on Database and Expert Systems Applications (DEXA 2002).* Aix en Provence, France.

VARSHNEY, U., VETTER, R., AND KALAKOTA, R. 2000. Mobile commerce: A New Frontier. *IEEE Computer: Special Issue on E-commerce.*