Resource Billing in Mobile Agent Based Heterogeneous Distributed Information Systems

Martin Pirotta, Arkady Zaslavsky

School of Computer Science & Software Engineering
Monash University
Melbourne, Australia

e-mails: {martin.pirotta, arkady.zaslavsky}@csse.monash.edu.au

Abstract
The research of billing schemes for computational resources is fundamental to information economy. Software agents that roam the Internet are consumers of resources. In this paper we first outline the surgeoncy of information economy as a result of computational resources being made available to the general public and their software agents. We propose a billing scheme for the usage of such resources using e-wallets and Internet dollars. The proposed billing scheme is applied to a community of mobile agents that live, and roam across a number of nodes, which provide the necessary resources and services used by these agents. An integrated service is charged at a dynamically computed rate at the end of a provider and agent negotiation phase. The paper also puts forward ways of achieving secure and reliable payments. The prototype implementation of a simulator is also discussed.

Keywords
Mobile agents, resource billing and information economy

THE SURGENCE OF AGENT ECONOMIES
Billing for use of resources in an information economy is an important topic as a greater portion of the industrial countries’ gross national product (GNP) is based on information, communications and entertainment. It is expected that the contribution of information economy to the GNP will keep escalating and innovative billing techniques will certainly contribute to this trend. Electronic services over both private networks and the Internet include home shopping of goods such as food and books, electronic booking of travel and entertainment, electronic payment of electricity, gas and water bills and electronic banking (Clements et al 1997). In general, information economy is sustained commerce encompassing both mobile and wired transactions. The former is giving rise to M-Commerce (E-Commerce over mobile devices).

Information economy really started off with the introduction of the ARPANET, which eventually got adopted for the Internet implementation in the late 1960s. The Internet grew as a mechanism to distribute information and soon companies used it to advertise and sell products and services. Internet users log on to sites and look for needed services and products, which they purchase via credit card. This method of search and procurement is still widely used today and necessitates the involvement of humans in the business loop. It is an example of a business to customer (B2C) model. The next step was the development of fully automated systems that provide real-time processing of credit card payments. This automated approach has lead to an explosion of business-to-business (B2B) implementations.

A close look at the information economy reveals that the system supporting it is heterogeneous and distributed in nature (Hanson et al 2000). The amount of information and product variety on the web is extensive and users need the means of selecting the right information and product. Moreover the selection of information should ideally take place at the source in order to avoid the need to download large amounts of data. Note that the information overload syndrome does not only apply to web users but to e-business companies alike, as the former need to monitor their competitors pricing and product offering to survive in the market by formulating their competitive strategies accordingly.

The above challenges to the information economy and systems can best be solved using portable software that can independently perform tasks on behalf of the user at various sites (White 1996). This is a prime candidate for a mobile agent. Today software agents are used in both B2C and B2B, in each case each having specific characteristics (Brian Blake 2002). These agents are autonomous and mobile and accomplish there given objectives by consuming resources. One form of revenue generation is to bill the software agents for all the resources they use, that is, CPU, memory, bandwidth, data retrieved etc. Present day examples of agents on the Internet are the shopbots (Greenwald et al 1999, Greenwald et al 2001) employed by mySimon.com,
DealPilot.com and the BidClick at amazon.com. As the information economy expands it is envisaged that it will eventually be made up of billions of software agents.

One other form of revenue stream for an e-business is the hiring of software agents together with the service they provide. Thus, the agent itself becomes a resource generating revenue. Advertising can still occur in a different format, however. Providers can publish their product offerings and corresponding prices on bulletin boards, which agents can access to gather information first and then decide which node to visit to accomplish a given task.

**MOBILE AGENTS IN INFORMATION SYSTEMS**

Much has been written on mobile agents, which are autonomous intelligent software programs that can travel across a network to numerous nodes (also called either workplaces or marketplaces) searching for and interacting with services on the user’s behalf. They also allow processes to split into multiple instances that execute on different nodes (Cardelli 1999, Cardelli et al 1999, Cardelli et al 1997a, Cardelli et al 1997b). Note that the distributed system must be able to carry the agent’s code and its current state form one node to the next.

In distributed systems running distributed applications mobile code performs better than its stationary counterparts. Moving the computation code to a different location reduces the overall cost of communications not only by reducing the distance to the data source but moving the computational code to the data source eliminate the need to transmit large amounts of data. These mobile programs have a unique characteristic as they co-locate data and computation by bringing the computation to the data, this is in sharp contrast with the client server and remote procedure call models. When the agent is on the node that contains the required data, the agent can request for the needed data and it can then manipulate it to extract and produce only the relevant data.

The main issue in building agents is the coding of decision making and plan management algorithms needed for the agent to be independent and be able to reason for itself (Ferguson 1989, Ferguson et al 1989). Other issues relating to such a system include portability, security, messaging, accounting and billing. This paper focuses on accounting and billing schemes however reference (Das et al 1997) puts forward some suggestions on decision and management ideas. Agent software is built using a layered approach involving mobility, translation, collaboration, actions, reasoning, beliefs and sensory (Elizabeth A. Kendall et al).

In an information system, which is made up of millions of nodes there will be several types of agents. We propose a system, which makes use of various types of agents, each having specific tasks and objectives. A user injects a user agent to fulfil his/her objectives; this agent carries some electronic currency in its e-wallet and communicates with the system agent who is responsible for the smooth running of a workplace (node or marketplace). Money matters are handled by the bank workplace, which is run by a bank agent. An independent agent known as an arbitrator agent ensures that an agent has the necessary cash to cover a requested resource from a workplace while checking that the resource is actually available.

Consider figure 1 which shows a distributed system in which agents live. The system is too big to be controlled by one company alone and so is divided into enterprise zones (or markets) separated by enterprise boundaries. Agents cross these boundaries to fulfil their tasks. Agents, which belong to an enterprise owner do not pay any currency while dwelling in its’ own enterprise zone, but once the boundary is crossed the agent has to pay for all its resources. The figure below shows Enterprise farm A which provides a marketplace for its’ own yellow agents. Enterprise farm A also provides resources for any other agent in the system. A pair of mobile machines (depicted in green and blue) each has either own user agent, which has been injected into the system.

The user agent can be sent on an assignment to shop around for a product at the cheapest price. On the other hand, a workplace can send off a system agent to roam and check the prices of other workplaces. In such a scenario the user agent becomes a shopbot while the system agent is a pricebot. The combination of shopbots and pricebots is an important concept especially when in the field of automated pricing by sellers.

Today there are several toolkits, which can be used to build mobile agents and applications that support them. These tools include Telescript (Milojicic et al 1999), which supports the concept of place and agent as subclasses of the Telescript process. A place object represents a virtual space in which other objects including agents can co-exist, work together and communicate through local communication. Each node on the network can run a Telescript Engine, which is capable of supporting a number of places. Other toolkits include Java from SUN MICRO Systems. Agent systems can also be used including Agent Tcl developed by Dartmouth College, Agent Java, IBM Aglets SDK (originally known as IBM Aglets workbench) and Grasshopper (Milojicic et al 1999).
Messaging between agents and workplaces should be based on independent and standard languages such as the Knowledge Query and Manipulation Language and Protocol (KQML) (FIPA 2000), Interactive Data Language (IDL) and Electronic Data Exchange (EDE). This does not mean that when appropriate messaging using application-based structures and specifications cannot be used.

COSTING AMBIENT RESOURCES

There are several price charging schemes but in general they fall in one of two broad categories, either fixed pricing or dynamic pricing strategies (Zaslavsky et al 2002). The latter include negotiations between sell and buyer and auctions (Das et al 2001 and Tesauro et al 2001). One of the most successful seller-driven auctions is the sealed-bin second-price auction proposed by Vickery, where each bidder submits an offer and the auction winner is the one that offers the highest price (Bredin et al 1998).

The proposed simulator is based on a marketplace model, which consists of nodes. Service providers control an enterprise marketplace. We consider a service provider that has a presence on the distributed system with a marketplace, which makes available resources including data, applications, algorithms, memory, CPU, bandwidth etc. These resources can be divided into the three categories of physical, logical and communication.

Physical resources relate to the actual hardware of a compute system, which include disk, memory, CPU, display units, tape drives, cdrom drives and network controls etc. Logical resources consists of threads, databases, data files, data and programs, while communication resources include services (ftp, email, ntp etc), network sessions, channels and bandwidth. Let P denote a physical resource, L a logical resource and S a communication resource. Hence the total service offering at a node is defined as:

$$O = P(\text{total}) + L(\text{total}) + S(\text{total})$$

We propose a method of calculating the cost in Internet dollars for the resources used per unit time and present a single dollar figure based on pre-defined assumptions on the usage of the individual resources. Consider an agent, which migrates on to the node. The agent has a variety of choices for it can use all three types of resources, it can only decide to use a subset of the provider's offering and supplement the rest with its own. For example it may want to apply its own algorithms to the data provided by the service provider. As it is generally cheaper to manipulate the data at the source the agent will need CPU cycles, memory and network bandwidth to transmit its results. Hence in general an agent can utilise it's own data and service resources as well as that provided at the marketplace in which it currently lives.

The resource offering used by an agent can be expressed mathematically as follows:

$$[PLS] = \begin{bmatrix} 01001011 \\ 00101101 \\ 00010111 \end{bmatrix} = [0PLS(P + L)(L + S)(P + S)(P + L + S)]$$

Where a 1 indicates a consumed resource and a 0 indicates an unused offering in the allocation matrix.

The service provider has to adopt a profitable business model to remain viable and this can only be achieved by at least covering all its costs for hosting the node. In reality costs can be covered either by subsidy or earnings
and the latter is the most economical approach. The business model can be such that for each vertical tuple in the allocation matrix the cost varies such that an agent is encouraged to use most of the service offerings in the marketplace. However, the basic costs need to be known upfront and then any profit added on top of these costs depending on the competition, the state of the economy, etc.

One of the main difficulties in calculating the costs associated with hosting a node are the identification of all expenses. Expenses fall onto two categories: on-going costs and start-up costs. The latter are needed to set up the service itself while the former are present throughout the lifetime of the product.

<table>
<thead>
<tr>
<th>Set up costs</th>
<th>On going costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase (or hire) and installation of hardware</td>
<td>Annual hardware maintenance contracts with</td>
</tr>
<tr>
<td>Purchase (or hire) and installation of software</td>
<td>hardware manufacturing</td>
</tr>
<tr>
<td>Purchase (or hire) and installation of communication lines and equipment</td>
<td>Annual software licenses</td>
</tr>
<tr>
<td>Purchase (or hire) of software</td>
<td>Software bug fixes and upgrades</td>
</tr>
<tr>
<td>Upfront charges for software development</td>
<td>Communications costs</td>
</tr>
<tr>
<td>Upfront charges for the setting up of computer room facilities</td>
<td>Hiring of equipment</td>
</tr>
<tr>
<td></td>
<td>Air conditioning and power</td>
</tr>
<tr>
<td></td>
<td>Backup facilities</td>
</tr>
<tr>
<td></td>
<td>Data updates</td>
</tr>
<tr>
<td></td>
<td>Service offering advertising</td>
</tr>
<tr>
<td></td>
<td>Management and accounting fees</td>
</tr>
<tr>
<td></td>
<td>Equipment upgrade</td>
</tr>
<tr>
<td></td>
<td>Tax</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Ambient product costs

The unit quantity of charge for using a resource is based on the above expenses and the time period during which the capital costs are recovered completely. The number of time units covering the recovery period divides the capital cost giving the capital cost per unit time. On the other hand, the on-going costs are generally calculated monthly or annually. This cost can then be calculated per unit time. Let \( C_T \) denote the total cost per unit time.

\[
C_T = \sum_{N=1}^{q} P_N C_N + \sum_{M=1}^{r} L_M C_M + \sum_{K=1}^{h} S_K C_K
\]

Where \( P, L, S \) are equal to 1 or 0 and \( C \) is the cost (on-going plus set up) per unit time for using a particular resource and \( q, r, h \) are the number of different resources of each type. Each individual cost \( C \) is made up of the on-going and capital costs. \( C_T \) has the units of Internet dollars per unit time IS/sec. Figure 3 illustrates the process flow diagram for cost and resource allocation. As regards to pricebots the algorithms they employ to determine the cost of a resource can be based on business objectives such as complete buyer demand, competitor’s prices or its own previous history of prices, profits and defined period of investment return. Moreover, working with less information means less storage and less computational time for agent action. The total cost per unit time for a particular resource is not the selling price, but the absolute minimum-selling price at which the resource is advertised. The above equation is modified by a weighting factor to represent the applied profit. This figure is adjusted in dynamic pricing according to predefined business objectives. Hence the advertised cost is expressed as:

\[
C_{A} = \sum_{N=1}^{q} P_N C_N W_N + \sum_{M=1}^{r} L_M C_M W_M + \sum_{K=1}^{h} S_K C_K W_K
\]

Where \( C \) and \( W \) represent variability of cost \( C \) and weight \( W \).

A high selling price may attract fewer shopbots thus taking longer to recover the investment made. On the other hand a too low selling price will result in a financial collapse of the venture, while a just selling price will enable the business to remain afloat by attract a healthy amount of profitable shopbots. The business model can apply either the same across the board or can be specific for each customer. Moreover the model can be static or dynamic in nature.

**MODELLING MARKETPLACES**

One type of model that we have studied is that which encompasses bank and service nodes where user, system and arbiter agents live. Agents need to have currency to be able to acquire a resource in the marketplace. They...
can decide which resource (for example space and time) to use after having scouted around for competitive prices. A node owner may not wish to allocate CPU time to agents which are not his and hence discourages CPU usage by allocating a high premium cost to his CPU resources. On the other hand another marketplace may promote CPU usage as it has lots of free clock cycles, hence giving a substantial discount for this resource. The agent is intelligent enough to analyse the quality of the service provided and the cost difference between the two options and decide accordingly depending on its’ own encoded priorities.

One important aspect of the currency concept is that this can be used as a means of ensuring that an agent lifetime comes to an end and that agents don’t keep roaming around. When an agent’s currency falls to zero the agent is terminated, while when it is at a critical low level the agent can decide to either return to its owner or issue a death notification to its owner. A variation to this scheme is one that allows the agent to return to its owner once the agent’s level of currency is zero.

Currency is managed through a distributed system of banks, which are contacted by the agents every time a transaction needs to take place. In a real Internet based information economy, an accredited body regulates banks and banks employ bank agents to issue currency. As in real life money has to be authentic and individual notes or bills are unique and identifiable. Some applies to the currency used in an information economy. Identification of bills is carried by means of the bank’s signature and it’s public key. Secure protocols are used between banks and agents.

Bill transaction mechanisms have to ensure that bills are actually transferred between agents for services and goods provided. The possibility that agents procure services that they cannot afford has to be eliminated especially where large amounts of currency is involved. All these issues can be addressed by the concept of an arbitrator and bank agents. There are no direct currency transactions between a seller and a buyer only a service (or goods) transaction takes place between the two agents.

Currency transactions take place between a banker agent, representing a bank, and either a buyer or a seller agent as put forward by Bredin et al 1998. Consider an agent A (as illustrated in figure 2) who requires a service from agent B. The latter informs the bank how much his service costs, while the former hands over to the bank an amount of currency greater than or equal to that which was requested by agent B. Once agent A confirms that the service has been provided the correct amount of currency is handed over to agent B by the bank. This currency transaction also involves the validation of the bill(s) originated from agent A and the issuing of a new bill(s) to agent B. In the literature one finds that bills are only used once and when they are handed over to a bank they are not reused. The non-reuse of banked bills does not represent the best way of controlling the system finances. A different approach is to have no agents carrying any bills but register their bill amounts with a bank. When an agent starts its existence it has a deposit of bills in a bank and as it consumes resources and services it authorizes the bank to transfer funds to the service provider. All the buyer agents do is reconcile the transfers with the bills they have to pay, and seller agents need to ensure that revenue has been received for the invoices they have issued. Banker and user agents keep receipts of paid bills.

![Figure 2: A marketplace model using arbiter and banker agents to ensure proper running of an information economy](image)

An enhancement to the basic model needs to be made as banks only facilitate the transfer of currency and so another mechanism is required to ensure that agent B can provide the service while agent A can afford it. Hence, comes the creation of an arbiter agent, which collects collateral from both parties as a form of security. If the seller agent B does not provide the service then B’s collateral is passed on to A, at the same time if A goes not pay for the service consumed then its’ collateral is forwarded to agent B.
SIMULATOR DESIGN AND IMPLEMENTATION

We test our proposal using a simulator, which consists of a number of marketplaces providing resources such as CPU, memory and disk space. The agents themselves are mobile in nature but have different degree of intelligence. Agents negotiate for services and prices and this inherently necessitates agents to have some level of intelligence. Agents fall into three categories of zero-intelligent agents, zero-intelligent-plus agents and intelligent agents. The model we employ regarding the agent’s willingness to pay is based on the following strategies:

- **Specific Seller**: where shopbots (user agents) do not shop around but buy from a specific workplace in the belief that the workplace offers the best for quality in their opinion.
- **Random Seller**: where a user agent randomly selects a seller (workplace) completes a purchase as long as the price requested is lower that the agent's valuation.
- **Bargain Hunter**: where a shopbot checks the prices at various workplaces and selects the cheapest as long at the price is below the agent's valuation.

Each marketplace has its own housekeeping agents, known as system agents that track the status of resources. System agents use a polling and scheduling mechanism to keep track of resources. Periodically the resources of a marketplace are checked and a resource profile sheet is update accordingly. Also before a user agent can use a set of resources these resources have to be booked and the price for their usage agreed upon by both parties.

The process for price determination will depend on the status of the resource profile sheet together with the business objects and the business costs for providing the resource in first place. This in turn will dictate the value of the weighting attached to the cost of a particular resource. Note that the business objectives can vary from marketplace to marketplace. One marketplace can strive to achieve maximum profit in order to cover its business costs in the shortest period of time. On the other hand another marketplace can decide to adjust the price so that it has maximum use of its resources at all times, thus it will have a tendency of lowering the cost to a minimum value acceptable to the business. Another possibility is for an Enterprise to have several marketplaces with different objectives thus being able to provide an expensive high quality service while also catering for the lower end of the market.

The weighting function that we propose is expressed as:

$$ W = \sum_{N=1}^{M} \frac{B_N(R)}{N} $$

Where B is the business objective and M is the total number of objectives, thus the most important business objective has the most weighting, while the least important has its value reduced by M. R is the resource rating.

Figure 3 illustrates the process flow diagram for cost and resource allocation. As regards to pricebots the algorithms they employ to determine the cost of a resource can be based on business objectives such as complete buyer demand, competitor’s prices or its own previous history of prices, profits and period of return. Moreover working with less information means less storage and less computational time for agent action.

![Figure 3 Functional modelling of cost computation](image-url)
The simulator is currently being developed using Grasshopper version 2.2.4, which is a development platform that enables us to develop and deploy agents in different workspaces on different machines connected together a network via the Grasshopper Distributed Agent Environment (GDAE). Each machine has a region in which at least one agency is present. An agency is the actual runtime environment for mobile and stationary agents. A host can have multiple agencies and the functionality offered by an agency is encapsulated in a workspace. The local region registry maintains information and status of agents, places and agencies. While regions, agencies and places are stationary on a host mobile agents can migrate from agency to agency in different regions on various hosts. The agents themselves have been coded up using Java 2 SDK, Standard Edition version 1.4.2.

Once the price of a set of resources is presented to an agent the agent can either accept or decline the offering. We propose the function depicted below in figure 4, which forms the basis for an agent to accept an offer for a given quality of service. One other difficulty, which is associated with billing resources employed by mobile applications, is how to represent other factors apart from prices such as the case of information bundling, horizontal and vertical differentiation and performance (Kephart et al 2000a, Kephart et al 2000b and Kephart et al 1998). At this stage we are only concentrating on price and propose the following function for service acceptance by the agent. The formula below is used to simulate a random seller scenario, where the agent selects a provider at random and accepts its offering as long as the price requested is lower than the agent’s evaluation.

\[ C_1 \leq C_T \leq C_2 \quad \text{where} \quad C_T = ae^{by} \quad a < 0 \text{ and } 0 < y < 10 \]

\[ C_1 \leq \text{price} \leq C_2 \]

\[ \text{price too low} \quad \text{price too high} \]

\[ \text{QOS too low} \quad \text{QOS too high} \]

\[ \text{QOS} \]

\[ \text{cost} \]

\[ \text{acceptable cost} \]

\[ \text{rejected QOS} \]

\[ \text{QOS versus cost and agent’s acceptable limits} \]

**CONCLUSION AND FUTURE WORK**

Proper billing of resource utilization in a distributed agent community is one of the main objectives for a successful and mature information economy. Software agents running a free market economy can be susceptible to the emergence of unpredictable behavior. On the other hand proper allocation of resources brought about by the collective interactions of individual agents in a distributed system yields a positive economy.

We proposed a simple cost calculating mechanism that will present a single price to the agent requesting the resources and services. The proposed basic algorithm can incorporate several business objectives whose priorities can be adjusted according to the present agent economy. A simple algorithm has the main advantage that it can be used to track a large number of parameters including several competitors’ price lists. The main difficulty facing pricebots is the ability of frequently changing its prices to keep in line with competitors. This change has to take place in a reasonable time and the algorithm enables this to occur. We currently are gathering data via our simulator, which uses the proposed algorithm to calculate the selling price.

**REFERENCES**


Elizabeth A.Kendall, P.V. Murali Krishna, Chirag V. Pathak, C.B. Suresh Patterns of Intelligent and Mobile Agents . Computer Systems Engineering, Royal Melbourne Institute Of Technology, City Campus, GPO Box 2476V, Melbourne, Vic 3001 Australia.


Hanson, J.E. (2000) Cultivating the Agent Economy. The Fifth International Symposium on Autonomous Decentralized Systems. IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 USA.

Kephart, J.O., Greewald, A.R. (2000a) Shopbot Economics. IBM Institute for Advanced Commerce. IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 USA.

Kephart, J.O., Hanson, J.E. (2000b) Dynamic Pricing by Software Agents. IBM Institute for Advanced Commerce, IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 USA.


COPYRIGHT

Pirotta M. and Zaslavsky A. © 2003. The authors assign to ACIS and educational and non-profit institutions a non-exclusive licence to use this document for personal use and in course of instruction provided that the article
is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive license to ACIS to publish this document in full in the Conference Papers and Proceedings. Those documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Wed. Any other usage is prohibited without the express permission of the authors.