Mobile Agents and Intellectual Property Protection

Stephane G. Belmon* and Bennet S. Yee**

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114

Abstract. Technical enforcement of intellectual property (IP) rights often conflicts with the ability to use the IP. This is especially true when the IP is data, which may easily be copied while it is being accessed. As electronic commerce of data becomes more widespread, traditional approaches will prove increasingly problematic. In this paper, we show that the mobile agent architecture is an ideal solution to this dilemma: by providing full access to the data but charging for the transmission of results back to the user — *results-based billing* — we resolve the access versus protection conflict. We define new requirements for agent frameworks to implement results-based billing: "data-aware accounting" and "data-tight sandboxing", which, along with the common requirements such as authentication, authorization, agent self-monitoring, and efficiency, provide the mechanisms by which database owners can effectively grant users access to their intellectual property.

1 Introduction

The concept of intellectual property (IP) protection is usually viewed as necessary for innovation. It protects the IP creators from seeing their work used without remuneration, economically motivating research and development of new ideas and technologies. Unfortunately, it also often conflicts with freedom of use. Unlike academic circles where the only remuneration is "giving proper credit", IP protection in a commercial setting often means controlling all forms of access, especially when legal protection (copyright and patent laws) alone is deemed unsatisfactory. This tension between accessibility and protection is especially acute when the IP is in electronic form.

To use a simple bookstore analogy, the IP protection problem originates in the difficulty of electronically allowing "free, in-store, full-text browsing" while retaining the "buy-it-to-read-it" model. Traditionally, data is either made available or not to a customer's computer, and further use is beyond the provider's control. The usual solution is defensive, and the data is not made available in its entirety, but only by fragments supposed to be sufficient for the customers'

* sbelmon@cs.ucsd.edu
** bsy@cs.ucsd.edu

purposes (the table of contents, for example, hopefully enough to whet the customers' appetite so they will pay for physical goods, e.g., a CD-ROM or a book). While this solution might work for books, it severely limits usefulness. Furthermore, we will show that there are other common scenarios where it simply does not fit the customer's needs.

In this paper, we describe a solution to this conflict called *results-based billing* based on mobile agent technology. It allows a maximum freedom for the customer, retains strong intellectual property protection, and most important of all, charges a fair cost to the customer for almost any possible use.

In the next section, we present some application scenarios which can be readily solved by our agent-based IP protection scheme, motivating its use. In Section 3, we present "traditional", non-agent-based approaches to these IP protection applications, and discuss the limitations of these approaches. In Section 4, we provide a more detailed description of our vision of how an agent system can be used for IP protection. In Section 5, we compare our scheme with watermarking and fingerprinting, two methods also being developed for IP protection. Next, in Section 6, we examine the agent system infrastructure requirements needed to support our IP protection scheme. Finally, in Section 7, we make some concluding remarks.

2 Intellectual Property Applications of Agents

In this section we present a few application scenarios that involve intellectual property of data. The point of view adopted throughout this section is intentionally an end-user one. In other words, we are not as much concerned about *how* the system solves the problem as we are about *what* end-users see. By end-users, we refer to the customer and some individuals at the company providing the service — mostly, the accountants and the content-providing departments, hence, the tendency to emphasize billing and copyright issues.

We believe that these scenarios are typical, and we will later show the common pattern underlying them. However, it should be noted that we do not present any solution in this section. These cases can be solved in many ways, and though the choice of cases is arguably biased towards agent-based solutions, we hope that the way we express the problems themselves here is fair.

Our first application is a multimedia database, whose content is the whole set of films, videos, and sound-tracks owned by a major motion picture company. The company wants to let customers access this database as conveniently as possible while charging them in a manner consistent with their actual needs. The company also wants to keep the information systems costs under control. Let us give some examples of these customer needs:

– Basic video-on-demand, with advanced search for the cast, plot, etc. The cost to the user should be comparable to the purchase of a videotape with the film on it through a store. The resolution and sound quality are user-specified and should have an impact on cost.

- Clip retrieval. Given some rough lines of the script, the user wants to retrieve a specific clip in a film. The film and even the actors may be unknown to the user. The cost should mostly be a function of the clip duration and should be definitely less than that of the whole film containing the clip. More advanced searches could involve some pattern recognition or other advanced techniques.
- Poster editing. Maybe through the previous example, the user retrieves some clips and/or still images in low resolution. She mixes, blends and otherwise edits those to form a custom image, using an image editing tool. After having finished the work, the original, very high quality source images are used to build a final high resolution image. The cost should be comparable to that of a single high quality picture (and definitely less than that of the whole set of high quality images used in the process). The company might also charge for some computation costs, should the chosen solution involve significant calculations.

Our second application is a DNA database containing the complete sequencing of several species. This database is owned by a company trying to amortize the sequencing costs. The billing is primarily based on the length of the extracted sequences and the effort required for the search in terms of company resources (users are free to post-process the results at will). Beyond the usual pattern matching and regular expressions search, the company wants customers to be able to test new theories. The company wants to allow them to use the database with a freedom comparable to that of the case where they would actually have total access, and still be able to bill them in a fair manner.

An example is the search for the gene encoding for a protein in all known DNA sequences given some clues about its structure. This could involve inferring some information about the sequence, in order to restrict the search to a small subset, and then computing a likely structure for these remaining few sequences. The structure prediction process should be user-specified, as the field is still evolving (and as the general problem is likely to remain unsolved for some time). The cost should not be that of extracting the whole set of sequences from the database, but that of the computations required in addition to the price of the final answer (which could be mostly length-dependent) [9].

Other examples include the computation of various new metrics to measure the evolutionary distance between all species in the database. The fair cost for this computation should be reasonable, and very small compared to the price listed by the sequencing company to give total freedom of access to the whole database.

3 Traditional Solutions

In this section, we present the traditional solutions for the applications discussed. By "traditional", we refer to those implementations that do not involve the use of agents, for example systems utilizing a client-server or remote procedure call computation model. The solutions discussed here are what we believe to be representative of reasonable choices of implementation.

For the first set of applications involving services provided by a multimedia database, a traditional system configuration is that of a client-server model. The server is a multimedia database that serves the information contents to clients through the use of a querying service. Based on this model, possible implementations of the three applications in this domain discussed previously are described below.

- Basic video-on-demand: This service can be provided by a relational database server that can be queried using a database query language such as the Structured Query Language (SQL) [6]. The query language chosen must provide enough expressive power to allow the clients to specify various aspects of the desired data such as the resolution levels, compression, and formats (i.e. audio, video, etc). It must also be flexible enough to adapt to changes in technology, such as new compression techniques or data formats, so that it can continue to provide the services after the changes take place. In addition, an advanced accounting system must be provided so that appropriate fees can be charged, for example, for the price of the films and the queries.
- Clip retrieval: A similar configuration as described above can be used to provide a clip retrieval service. The video format must support an efficient retrieval of a specific clip, preferably without the needs to decompress the data before executing the queries. Other advanced search capabilities must be included in the database query language. In addition, the accounting system must be modified accordingly.
- Poster editing: We see two possible approaches for this application:
    - The first approach is to design a language that can describe sequence of actions performed on an image. This language will describe video editing actions in the same way a language such as Postscript describes a page description of a document. A sophisticated accounting system must also be designed to calculate the fees. Furthermore, both the language and the accounting system must be upgradable as the format and the set of possible actions change.
    - The second approach is to allow the client to download all the data needed to produce the final result. Since it will be more fair to charge the client only for the final result, care must be taken to ensure that the client will not be able to obtain unrestricted access to any part of the retrieved data. The client must be allowed to apply a certain set of actions to the data and to obtain only the final result. To fully satisfy the security requirements, the data must be encrypted and downloaded to secure hardware in the client's machine [11]. The encryption prevents the client from "understanding" the data. The secure hardware is the only component that can decrypt the data, perform the sequence of actions as requested by the client, and yield the final results. The accounting system in this approach will be responsible for monitoring the secure conversation between the server and the secure hardware.

We now consider a possible solution to the DNA database search and structure prediction problems. A relatively straight-forward implementation of the DNA database for searching can be realized by utilizing a search engine-type setup along with a language for the definition of DNA "regular expressions.[9]" Advantages of the use of a remote search engine system include the provision for a fair amount of flexibility limited only by the DNA expression language and the low bandwidth usage requirements for sending out the initial query. However, since this is an intellectual property application, the major drawback of this system is that the client has to be charged for whatever results are returned. The cost to the client is the charge for the total amount of data returned, regardless of whether or not the sequences were actually useful.

Protein and RNA structure prediction involves the search for a protein exhibiting a given spatial form. In order for the client to be able to give queries to the DNA database server, a special purpose language must be created to express these spatial representations. Similar to the general DNA sequence search application, this language must provide query expressions (in this case, expression of spatial orientations) that will allow the server to return a subset of sequences that exhibit a comparable spatial form. This query language must allow for additional specifications to be added at a later time to account for advances in structure prediction technology. Under this type of system, the queries will be large because of the large amounts of data needed to encode the spatial information. Again, the client must pay for all sequences returned by the search.

4 Agent-Based Systems

In this section, we will describe an IP protection solution based on mobile agents.

In an agent system with *results-based billing*, users pay for the answers that they receive back from the agent. By using agents in this way, we achieve simple, natural, and elegant solutions for IP protection.

Here is how we use agents. To the first approximation, users of a results-based billing system may send agents into a server to access the intellectual property arbitrarily. Once within the server, there are no access restrictions on the data or other server-side resources, and the agent may use whatever algorithms it needs to process it. The only restriction is in sending messages off the server — the agent must pay in order to send results back to the user. (Clearly, our first approximation is imperfect. Resource limits and nominal usage charges must exist, or the servers will be abused. We will see later in more detail what should be done in practice.)

Results-based billing is natural in many ways. The user learns only what is communicated back to him. The user's agent may chose to compress or otherwise pre-process the data in order to make the message size as small as possible, but even with perfect compression schemes the information theoretic limits provide a nice measure of the amount of IP disclosure. Furthermore, as noted for agent systems in general, unlike other systems such as RPCs or per-item billing such a scheme will provide a very flexible interface to the users and yet retain strong IP protection guarantees.

While results-based billing has great appeal, note that it should not be used when data of wildly varying values are available on a server. For example, a geographical database which contains both the exact latitude and longitude of Stuttgart, Germany and the latitude and longitude of an otherwise-undiscovered, gold laden sunken Spanish galleon should price the information differently.

Let's see how this scheme will work for our examples. For the video-on-demand and clip retrieval applications, a full featured database management system is not strictly necessary because an agent may choose to perform a brute-force scan of the data itself. This data scan incurs no network bandwidth costs since it is executing locally on the data server. The agent can process the data into any format it chooses, thus relieving the server of providing several specific video formats. An agent-based system provides accurate accounting costs because charges are only applied for the hardware resources used and for the data returned to the client. The poster editing application requires a similar agent-based solution with perhaps additional basic image editing primitives usable by the agent.

An agent-based implementation of the DNA database can be realized either by allowing a brute-force search or by providing some regular expression search mechanism for the agent. This search mechanism would be used just to reduce the brute force search to a manageable subset. The protein structure prediction is performed by the user code perhaps with the assistance of some basic blocks provided by a library. This library may be provided by a third party so that the server can simply link it to the agent when needed. Advantages of such an approach are that the server is not required to have any knowledge of the library implementation (libraries may be updated without any change to the server code) and the library itself may exist as compiled native machine code for improved efficiency.

5 Watermarking and fingerprinting

The solutions discussed so far tried to forbid any kind of uncontrolled use of data. Another relatively new approach is to try to deter users from misusing the data by embedded identifying information in it.

One simple idea is to hide some information, such as the provider's and user's names, within images, e.g., the low-order bits of the image pixel values. This is known as *watermarks* and *fingerprints*, respectively. Simply using low-order bits is not very secure, in that such an encoding can easily be removed without significantly affecting the image quality, even when the exact encoding scheme is unknown to the attacker. Other approaches [3], however, embeds information more securely: the data is placed in perceptually significant components of the data, and attempts to remove the information should significantly degrade the image — while the difference between the original image and the protected one will still be "invisible" to the naked eye. With watermarking (and fingerprinting), if the user claims the data as being his (or if he spreads numerous copies of it), the provider can prove the offense; and users, even while colluding in a group, have no way of removing the watermarks (fingerprints) without damaging the image.

Watermarking and fingerprinting are orthogonal to our proposal.

Fingerprinting cannot prevent uncontrolled private use. It relies on the assumption that the user will make and spread enough unauthorized copies to let the IP owner discover the misuse. Unfortunately, this assumption is not always true, such as our poster-editing example or limited redistributions to "close friends". Furthermore, there are media for which any modification is unacceptable or impossible — many forms of text and our DNA example are in this category. Fingerprinting and watermarking is also very media-specific. While agents do not provide complete control over private use, they give a much better level of control than traditional solutions.

On the other hand, fingerprinting can protect data in ways that are not possible in our framework. It can let users view a film and still deter them from making many copies of it, or from broadcasting it. In this respect, fingerprinting is clearly very useful. The two approaches can be used together to provide even more security, and still give maximum convenience to the users.

6 Requirements

In the previous sections we argued that our results-based billing scheme is an ideal solution to the IP protection problem. However, to utilize it, certain requirements must be met by the underlying agent system, and we will now present these specific requirements. We will only discuss those that are pertinent to our applications. The requirements that are applicable to agent-based systems in general can be found in [4].

6.1 Accounting

IP owners want to charge for access to their IP. To do this, the agent infrastructure must provide the mechanisms to monitor resource usage.

In our IP usage metering scheme, the central idea is that we charge the agent-owners for answers received. This implies that agent network communications must be measured and charged on a per-byte or per-bit basis.

Unfortunately, the scheme is necessarily more complex than just charging for data transmissions. Because an agent can chose to trade CPU time for communications by employing data compression, usage of the CPU should also be charged. By requiring payment for CPU time, we also limit frivolous or inefficient squandering of server-side resources. Similar to CPU time, memory usage should be monitored and "rented out" to the agents. In typical situations, the resource usage costs will be small compared to the data costs.

One might try to charge data transfers differently, depending on what is transmitted – such as a low price for images and a higher one for texts, arguing that text has "more information" per byte than images have. As shown in [1], it is extremely difficult to make sure that agents are not using the available channels to communicate "hidden" information. This means that one should use an uniform rate, charging "by the byte".

However, the price charged an agent may depend on the agent's execution history: if the agent has only read from the "public knowledge" portion of a server, it would be charged a low fee; if, on the other hand, the agent has accessed the "cutting-edge research data" portion of a server, the data transmission charges would be concomitantly higher. Similarly, other access controls may be placed on the IP data; for example, a medical database may allow full read and certain kinds of append access for patient records (and genetic information) to the patient and the patient's doctors, but only provide restricted views for medical researchers who should only have access to aggregate information.

Mobility and agent migration, including that of transferring per-thread invocation stacks[5], implies message transfers for which an agent would be charged. If an agent has accessed "cutting edge" data, the server would have to charge a high fee for migration as well. For many IP protection problems, the agents can send back (compressed) results rather than migrating home from the server, so the initial migration to the IP-providing server is a one-way trip. This also implies that greater resource co-location is needed; the agent must have everything it needs to compute its result, since migrating to external resources from the IP-containing server is expensive.

6.2 Secure and Data-Tight Sandboxing

The service provider must be able to prevent agents from bypassing the accounting system discussed in Section 6.1. As expected in any agent-oriented application, an agent must not be able to "harm" the server. Furthermore, it should not be able to gain access to data it is not entitled to. By *data-tight*, we mean that it is impossible for the agent to bypass the accounting system when exporting data.

As research on multi-level security systems have demonstrated, processes in different security domains do not have to only use the system-provided communication mechanisms to send messages to each other: they can use unintentionally provided mechanisms, or "covert channels" [7], to communicate with each other. When the server can run several agents simultaneously and variable pricing is used[1], covert channels between agents should be eliminated or bandwidth-restricted.

While complete elimination of covert channels is extremely difficult if not impossible, a simple pragmatic solution exists: as long as we can estimate the maximum bandwidth achievable for a given amount of computation, we can charge enough for the CPU or other resource usage to compensate for the data

[1] If uniform pricing is used and there are no other access controls on the data, then covert channels between agents are not an issue: as long as we can meter outgoing data transmissions, these covert channels do not benefit the agents.

leaked over any potential covert channels. The system needs only to be reasonably data-tight.

In contrast to preventing agents from harming the server or leaking information, another desirable property would be the protection of agents from hosts: ensuring that their computations and communications are not tampered with, and private data are kept confidential [12]. This is difficult to achieve, and may require the use of secure hardware [11], but may be important for proving that the paid-for solution is actually correct.

6.3 Authentication and Authorization

The service provider must be able to authenticate the agent using its service. It must ensure that the agent is who it claims to be so that the fees can be charged to the appropriate party. This authentication mechanism must also prevent a replay attack where a network eavesdropper obtains a packet that has been used by a legitimate agent to identify itself, and simply retransmits it some time later. This attack will cause a fee to be charged to the legitimate agent's account even though it has neither requested nor received a service.

Some service providers may want an agent to agree to a contract, such as a non-disclosure agreement, before providing services for it. An agent is said to have obtained an "authorization" for the service once it agrees to the contract. The service provider must be able to keep track of and only provide services for authorized parties. Furthermore, some agents may have more privileges than others. The company providing the service may have a contract with some other parties to disclose information that would normally be otherwise inaccessible.

6.4 Agent Self-Monitoring

An agent must be able to monitor its own spending to keep its expense under a limit allocated for it. In other words, it should know its "financial status" at any point in time to be able to make an informed decision whether to continue consuming resources. Knowing in advance the cost of certain operations might be useful, but not generally feasible — for example it would be difficult to predict the cost of executing a non trivial procedure. An agent should be allowed to stop before its expenditure exceeds a certain limit [10]. In this case, it is the agent's responsibility to keep track of its own expenses, whether by explicitly arranging with the server for usage limits and warning notifications (interrupts), or by periodic checks (polling) on the resource billing. Obviously, servers cannot charge for resource usage without providing a mechanism by which agents can determine resource costs and current charges.

Resource monitoring may have to be coarse-grained: it would be disastrous if, by making the metering and billing mechanisms fine grained, the cost of these mechanisms became a significant fraction of the cost of the agent's execution.

In addition, the server must be able to monitor and perhaps inform the agent just before it has exhausted its given limit so that it will have time to either migrate to a server with less expensive resources or ask its home site for more "money." This notification is analogous to the situation with the soft CPU time and filesystem usage limits in Unix, which causes the delivery of the `SIGXCPU` and `SIGXFSZ` signals [8]. The messages sent for this purpose must be pre-defined, and the fees, if any, for such notifications should be comparatively low compared to data transfer charges.

6.5 Efficiency

While the need for an efficient agent system for the IP protection application may not appear to differ much from that of a "normal" agent system, we believe that it places greater stress on the efficiency of the system.

In general, uses of mobile software agents for IP protection will be more resource intensive than other applications. IP is of value when it is difficult to re-create, and agents are most useful when index pre-computation is difficult or impossible — otherwise standard database techniques may be applied. Large scale searches through unstructured or minimally structured data — for example, DNA database searches where no pre-computed indices exist because searches are wildly different — will necessarily be data and computationally intensive: the searching agent may have to scan through vast amounts of data and build for itself indices or other auxiliary data structures.

7 Conclusion

Intellectual property protection requires balancing usability, accessibility, and data security. In this paper, we explored an hitherto unexplored region of the design space and presented a scheme that strikes a new balance: in lieu of fine-grained metering of data accesses, we proposed using mobile agent systems for *results-based billing*, i.e., where the user's agent is charged not for accessing the data, but for sending the data (or the result of some computation thereof) back to the user. We showed that such a system would be highly flexible, easy to administer, and very secure.

To optimally support results-based billing, the importance of some existing design criteria increases and new criteria arises. We discussed the needs for *data-tight sandboxing* and greater *resource co-location* compared to "conventional" mobile agent systems. Additionally, we contrasted results-based billing with several traditional intellectual property protection schemes, and discussed their relative strengths and shortcomings.

Acknowledgments

Portions of this paper are based on an earlier survey [2] written for Keith Marzullo's operating systems course. In that paper, the authors evaluated several agent systems for their ability to protect intellectual property using results-based billing, and it provided the impetus to finally write up our ideas here; without that work, this paper would likely still remain on the back burner. We would also like to thank Keith for his invaluable input and advice on the earlier paper.

References

[1] Ross Anderson. Stretching the limits of steganography. *Lecture Notes in Computer Science*, 1174:39–48, 1996.

[2] Stephane Belmon, Chanathip Namprempre, Kenji Onishi, Sule Ozev, and John Seng. Mobile agents and the intellectual property of data. Technical Report CS98-573, Computer Science and Engineering Department, University of California at San Diego, La Jolla, CA, February 1998.

[3] Ingemar J. Cox, Joe Killian, Tom Leighton, and Talal Shamoon. A secure, robust watermark for multimedia. In *International Workshop on Information Hiding*. Newton Institute, University of Cambridge, May 1996.

[4] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile agents: Are they a good idea? *Lecture Notes in Computer Science*, 1222:25–47, 1997.

[5] Matthew Hohlfeld and Bennet S. Yee. How to migrate agents. Technical Report CS98-588, Computer Science and Engineering Department, University of California at San Diego, La Jolla, CA, June 1998.

[6] C. J. Hursch and J. L. Hursch. *SQL: The Structured Query Language*. Tab Books; ACM CR 8812-0907, 1988.

[7] B. Lampson. A note on the confinement problem. In *Communications of the ACM*, pages 613–615. ACM, October 1973.

[8] Marshall K. McKusick, Keith Bostic, and Michael J. Karels. *The Design and Implementation of the 4.4 BSD UNIX Operating System*. Addison-Wesley, 1996.

[9] Joao Meidanis and Joao Carlos Setubal. *Introduction to Computational Molecular Biology*, chapter 8. PWS Publishing Co., 1996.

[10] J. E. White. Mobile agents. In J. Bradshaw, editor, *Software Agents*. AAAI Press and MIT Press, 1996.

[11] Bennet S. Yee. *Using Secure Coprocessors*. PhD thesis, School of Computer Science, Carnegie Mellon University, May 1994. CMU-CS-94-149.

[12] Bennet S. Yee. A sanctuary for mobile agents. Technical Report CS97-537, Computer Science and Engineering Department, University of California at San Diego, La Jolla, CA, April 1997.