A Subscribe/Push Framework: Periodical Push with Mobile Agents

Tzone I. Wang *Kun H. Chung*

Laboratory of Intelligent Network Applications
Department of Engineering Science
National Cheng Kung University
Taiwan
wti535@mail.ncku.edu.tw

Abstract

Conventional web technology distributes information in a client-server style. You have to connect to the Internet, search for what you want, and download what you find. The explosion in the number of information resources on the Web makes it difficult for people to find what they really need. Collecting bookmarks easies the pain for search but not for the rest tedious works. The Push technology is seen as the one for taking these pains away. Though still under client-server model, the perspective is not quite the same. A client subscribes to a service once, and the server serves when necessarily. While the Push technology is promising, the realization of it needs careful thoughts. Implementations that use multicast or broadcast techniques might worsen the network traffic further. On the other hand, mobile agents have earned the reputation as a network traffic reducing technology. They are now engaged in many application fields.

The purpose of this paper is to advocate the use of mobile agents in realizing the Push technology. The main advantage of doing so is that a single mobile agent, carrying a piece of information, may visit many client sites without worsening the network traffic. This technique will be helpful for many application areas, such as information appliances, news service, commercials, and etc.

A prototype implementation is based on the MACE (Mobile Agent Carrier Environment). Consequently, the push framework of the prototype is designed according to the service protocol of the MACE.

Keywords:Push Technology, Mobile Agent, Agent System, Push Server

1 Introduction

Conventional web technology distributes information in a client-server style and is sometimes referred to as "pull technology". Clients must connect to the Internet servers, search for what they want, and download what they find. Web surfing may be an enjoyable and productive adventure to some people. But, for those who just need regularly updated information from a same place, many laborious works seem unnecessary. The explosion in the number of information resources on the Web also makes it difficult for people to find what they really need. More often than not people will compromise and accept the content they have found in the first few rounds. They are just too tired to surf the Web and get more suitable information. Collecting bookmarks easies the pain for search but not for the rest tedious works. What is worse is that one might miss critical information because the next update happens before the access.

To one extreme, there are situations when information consumers are so occupied that they do not have any chance to issue complicated commands. For example, in battlefields or ambulances that use highly mobile and low bandwidth data receivers soldiers or doctors need to be fed with vital information whenever necessary. They are simply too busy to summon complex queries. In such cases, the client should simply turn on the data receiver and the server would automatically send it all the necessary information; data is delivered in much the same way as radio broadcasting or household power supply.

The Push has been advocated as the right technology for solving all these problems. Though still under client-server model, the perspective is not quite the same. A client subscribes to a service once, and the server serves when necessarily. For example, after a client registered to a news site for some kind of news updated on a daily basis, the server will push specific articles to the client every day.

While Push technology is promising, the realization of it needs careful thoughts. Broadcast or multicast seems to be the only way for the final journey of bit streams if information receivers are roaming in a mobile network. For such radio style transmissions, the major concerns would

be privacy and security if Push were used in a sensitive area such as a battlefield. Cryptography will play an important role here. In a LAN, periodical push of information via broadcasting or multicasting is still bearable as long as the frequency is reasonable. Continuous broadcast or multicast might become disturbances to network administrator and other users. In addition to the security issues, it might also worsen the network traffic further.

On the other hand, mobile agent paradigm has also been advocated as a network traffic reducing technology in various application domains. Agents are created to roam in a heterogeneous network and access distributed resources [KK1]. In contrast to the traditional client/server model in distributed computing paradigm, which ships data to remote application codes, the computation model of mobile agent paradigm migrates application codes to remote data. The motivation is to reduce network traffic by reducing interactions and data transfers between distributed components. Interactions between client and server modules are mostly localized at the server side. The aim is to gain more efficient bandwidth utilization and higher system availability. This code mobility greatly benefits those distributed applications that compute with simple logic and huge remote data. Therefore, mobile agents are particularly suitable for developing those kinds of distributed applications. There have been academic and industrial studies, implementations, and prototype applications of mobile agents [CTB], [KGN], [KLO].

This paper describes how mobile agents can be used to realize Push technology. A framework with two types of mobile agents is proposed and implemented. The subscribe agent is for consumers to register themselves to providers and the push agent is for dispatching information in the reverse direction. The framework is designed specifically for applications that need to dispatch information actively and periodically. Its prototype implementation is based on the Mobile Agent Carried Environment - MACE [WAN2] [WAN1], developed at LINA[WAN3].

The rest of this paper is organized as follows. Section 2 outlines the current status of Push and the way mobile agents can be used to realize such technology. Section 3 introduces the MACE and section 4 presents a mobile agent framework and the functions of its components to support Push. Section 5 discusses some of related works and section 6 concludes this paper along with

some further works.

2 Push Technology and Mobile Agent

When PointCast announced it in 1996, PointCast Network soon became extremely popular. It was the first Internet software system labeled with push technology. Since then, a number of similar products have been announced and deployed on the Internet. They claimed to have the superiority in narrowing your information to specific subjects by using channels and, most importantly, in delivering timely information to you automatically. However, take a closer look at many of those so-called push technology solutions and you soon realize that they are more of smart-pull than real push services. Most of them use a client program running on your machine to request updated information from the server at intervals that you can define.

This smart-pull approach suffers from two sever problems. Firstly, although you can schedule the client program to use your computer's idle time to retrieve information, the load on the server due to the client messages and server responses becomes heavier as the number of enrolled clients grows. And, the traffic to the Internet soon becomes saturated when dozens of users in a company have individual push applications running. Secondly, timely information may become useless because the delay due to a client's longer requesting period. In addition to the bandwidth problem, yet another shortcoming of the smart-pull approach comes from its gargantuan appetite for disk space. The client program may cache information, such as news, images, and advertisements that you might never use. This drawback is fatal to the low bandwidth, highly mobile, and diskless hand-held devices. It does have the advantage that the client base is small on the server because the channel subscription details can be store locally on the clients' own nodes.

There is the oldest version of push technology, though, the E-mail. In addition to plain text, E-mail is also being used to distribute multimedia files and web pages. So many websites now are using E-Mails to send electronic newspapers and advertisements to registered surfers. Although E-Mail is much closer to the real push, those disadvantages mentioned above still remain. Multiple messages have to be delivered to multiple users and that will obviously jam the Internet traffic.

Mobile agents, as mentioned above, are a relatively new technology that is attracting more and more researches. Because mobile agents consume fewer network resources, especially the bandwidth, they are therefore particularly suitable for developing applications to run on bandwidth-limited mobile devices. There have been studies, implementations, and prototypical applications of such systems [WAN1] [WAN2] [KGN].

To support mobile agents to roam freely, interact with each other, and access available resources in a heterogeneous network, agent systems that can create, interpret, execute, transfer and terminate agents must be deployed in the network. Agents can then be transferred between execution environments, called places that can provide functions such as access control, in the agent systems. The source place and the destination place can reside on the same agent system, or on different agent systems that support the same agent profile.

Before being transferred out of an agent system, a mobile agent must have its code, state, and context (if in a strong migration case) encapsulated as a whole unit - the agent. After being accepted to an agent system, the agent is de-capsulated before being handed to the execution environments and interpreted. Just like a packet is routed on special routes in a Tunneling process, a mobile agent is also transferred on special routes that may be planned before the journey.

Therefore, if the communication infrastructure is based on TCP/IP protocol, mobile Agent technology, in a sense, is an IP Tunneling technology synonym. Agents are encapsulated (carried as payload) within IP datagrams and transmitted between different agent systems. These agent systems, similar to the encapsulator and decapsulator in the *IP multicast*, are then considered to be the "endpoints" of the tunnel; the agent system sending out agents is referred to as the "entry point" of the tunnel, and the agent system receiving agents is referred to as the "exit point" of the tunnel.

In contract to the requirements of complex software and special hardware in IP Multicast, mobile agent implementation induces only a moderate software complexity and uses only matured network infrastructure. Of course, a well-developed mobile agent push system should retain the

flexibility for adapting to IP Multicast-enabled network infrastructure in the future. Besides, it may also take advantage of the current IP Multicast technology; that is to multicast in a LAN without causing much more network traffic. This can be achieved by an agent system proxy capable of multicasting in a LAN. With a proper deployment of agent duplicator, agents can be cloned and distributed to different channels. And this superiority in multicasting to several channels in the WAN by a single mobile agent also justifies the preference.

One thing seems conflicting to the original sprit of mobile agents is that, for Push, agents carry not only code but also information data. This may at first confuse many researchers, but one should always remember not to limit the types of mobile agent's state to reduce its flexibility.

2.1 Push by Mobile Agents

Many technologies must be integrated to efficiently realize the real Push that demands minimal consumer initiative. These include publication, collection, aggregation, routing, caching, delivery, subscription, notification, presentation, and etc. This paper concentrates on designing a mobile agent framework for a provider to effectively collect subscription from and push information to customers. Using this framework, a push system with an open architecture can easily integrate successively developed modules to enhance its functionality. An exemplary implementation of the framework is based on the MACE system that was designed to have an open architecture [WAN3].

2.1.1 The framework

The whole framework contains two major subsystems, one is the Subscribe Subsystem, and the other is the Push Subsystem. The former, residing at a subscriber's host, sends out Subscribe Agents to register with push servers for those information channels or web pages the subscriber is interested. The latter, residing at a push server, is to deliver information to the subscribers by Push Agents. Figure 1 shows the overview of the framework. Also shown in the figure is a Push Cache Server that may be deployed in a LAN that has a lot of subscribers.

In addition to the subscribing requests for channels or web pages, a Subscribe Agent carries

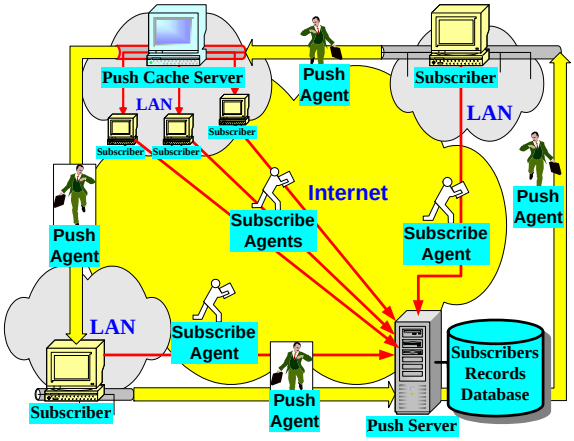

Figure 1: A Framework for Information Push Systems

the IP address of the subscriber's host, the IP address of the Push Cache Server if there exist one in the LAN, and other optional information such as hobbies, occupations, etc. All these will be recorded into the Subscriber Records Database in the push server for later reference. When new information is added to a channel or a web page is updated, the push server is triggered to start the whole push process. First, the database is consulted to find out those who subscribed to the channel or the web page. Then a Push Agent (or several if necessary) is created with the new information or the content of the web page as its payload. The IP addresses of all the subscribers' hosts are collected into a Place Set with those hosts that are in a same LAN and has a same Push Cache Server replaced by the Push Cache Server. The Place Set is then put into the navigation part of the Push Agent and the agent is launched into the network.

When a Push Agent arrives at the agent system of a subscriber, it duplicates its payload and drops the clone. The Push Agent then heads for the next agent system in the Place Set. The agent system is supposed to take major role in notification and presentation, or at least to initiate modules for these tasks. When a Push Cache Server accepts a Push Agent, it must duplicate as many Push Agents as the number of subscribers in the LAN and push these duplicators to the subscribers' agent systems. These cloned Push Agents will carry the same payload but with the Place Set changed to a single subscriber address. Or, as mentioned above, it may take advantages of the available IP Multicast capability and multicast the received information or web page to

a specific channel for all the subscribers. Another function of a Push Cache Server is to store temporarily those cloned Push Agents for the subscribers who were off-line when the Push Agent arrived.

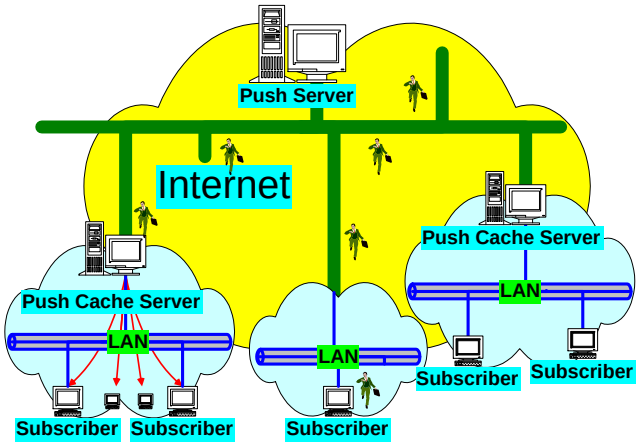

Figure 2: A Hierarchical Information Push System

Using this framework, an information push system can be built hierarchically as shown in Figure 2. When IP Multicast capability is available, Push Cache Servers can be deployed to take the most advantages of it. In the next section, MACE will be briefed before an exemplary implementation of the framework is described.

3 MACE: the Mobile Agent Carrier Environment

Most of the mobile systems to date offer an integrated environment for agent programming, transportation, communication, execution, and more. Trying to offer a total solution may not be appropriate for service providers interested in the *service-on-demand* applications. Service providers may be willing to program their services and business logic in a familiar language. Or, they would just like to quickly plug in legacy services. To them, a system for delivering services easily and quickly may be much more appreciated. They may not be interested in programming and tedious debugging in a complex system with agents interacting heavily.

When providing a service, the service provider may not count on the users to code agents that will interact correctly with the service station. Thus, giving an entire mobile agent developing

system to the users of a service seems inadequate too. What a service provider most likely does is to offer the users some information collecting programs with an attractive user interface. When a user calls for a service, its associated program is invoked to take from the user only these necessary arguments for instantiating the service program at the service station. The business logic of the service is totally soft-coded in the information collecting and the service programs by the service provider. In this case, only the list of arguments, not the entire program, has to migrate to instantiate the service program. This observation establishes the important principles behind the design of the **Service Protocol** in the **MACE** system.

3.1 The Service Protocol

MACE adopts the weak migration notion, which transfers no context of execution of an agent [PTV1]. In fact, it uses an even weaker migration policy. The **Service Protocol** realizes the *service-on-demand* concept and abstracts distributed resource accesses and management into services. In the Service Protocol, a service is fulfilled by the execution of two closely related components, an **Agentlet** and a **Serverlet**. In general, a service provider will code both of them for a service. The Agentlet is distributed to some dedicated servers that provide directory services for downloading and the Serverlet is stored at a service station where the service will be actually carried out. Agentlets of frequently used services may be cached locally in a user's local directory. Users invoke Agentlets via an **Agent Creator**. Once invoked, an Agentlet may produce one or more **Service Items**, each of which corresponds to an invocation of a Serverlet at a service station. The information in a **Service Item** may be the key attribute of a distributed database table to be sorted, or it may be the business logic of a transaction for some electronic commerce, and so on. Several Agentlets may be activated in the course of creating an agent. In other words, many **Service Items** may be packed into a single agent, and each of them with its own service information. Thus, an agent carries **Service Items** instead of codes, and a **Service Item** is actually the tight connection between an Agentlet and a Serverlet.

The great flexibility of a **Service Item** comes from its variable content; it can range from

as simple as a list of arguments to as complex as a segment of code written in a general purpose language or a customer-defined script language. If a **Service Item** is a segment of code, the associated Serverlet becomes an interpreter. To one extreme, the Agentlet becomes a compiler and the Serverlet serves as a virtual Machine. In such a case, MACE is in reality as well in name a Mobile **Agent** Carrier Environment. Therefore, the Service Protocol is simple yet powerful, confined mainly in the rules for transferring information among three parties - Agentlet, Agent Creator, and Serverlet.

The design of the Service Protocol in MACE frees users from coding agents by themselves. Service provides may use their own familiar language for developing Agentlet and Serverlet as long as they are confined to the user-defined information exchange protocol. Though designed in allusion to *service-on-demand* applications, MACE retains the flexibility of a general mobile agent system. It may as well support many other domains of applications without any difficulty.

A prototype of MACE has been built and is operational. It has been used to build an information retrieving system, in which mobile agents autonomously roam through a network (ultimately the Internet) to find sources and collect desired information. Another project was to build a distributed database system, using MACE as its infrastructure. Agents carrying SQL queries migrate among distributed database sites to consult tables and to perform on these tables operations such as projection, production, and more. The same project has also been extended to support mobile computing and mobile information retrieve.

4 The MACE Architecture

The whole MACE system is divided into four major parts as shown in Figure 4. The **Agent Launch Module** creates and launches agents into the network. Launched agents will roam through the network and reach some **Agent System Module** where services are carried out. Agent Launch Module is the implementation of **Agent Creator** in the Service Protocol. Before creating agents, users can consult the **Directory Service Module** to find out preferable or favorite services and download the associated Agentlets.

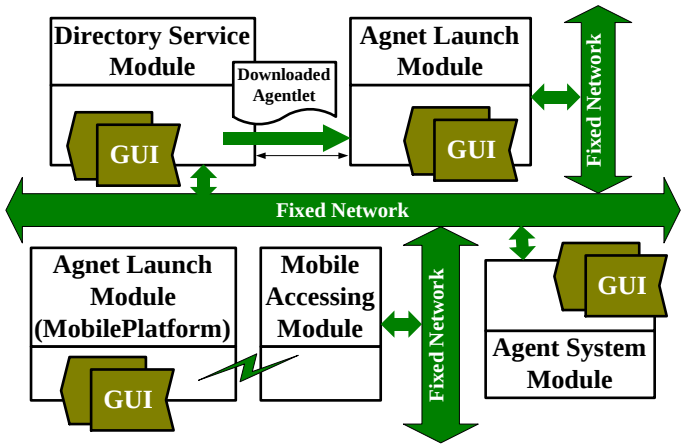

Figure 3: The MACE System

There are two types of Agent Launch Module, one is for hosts with permanent connection to the network, and the other is for mobile platforms. They have almost the same components except the physical links to the permanent network.

The **Agent System Module** plays the major role in serving the **Service Item** in a received agent. It resides at a service station that provides different kind of services. The **Directory Service Module**, as its name suggests, helps a user to find specific services. It is an important part of the implementation of Service Protocol in MACE. A Directory Server keeps and manages Agentlets of a variety of services. To support mobile computing, MACE includes a component named **Mobile Accessing Module**. This module is a bridge for mobile platforms to launch and collect agents to and from the network. In addition to the operations of normal wireless network connections, this module supports the so-called *disconnected operations* for mobile platforms [GRA1].

4.1 An Exemplary Implementation

As mentioned in section 2.1.1, the entire framework includes two subsystems - the Subscribe Subsystem and the Push Subsystem. Each of them must be able to both launch and receive agents. To implement these subsystems in MACE, the subscriber and push server will have to install both the Agent Launch Module and the Agent System Module in their MACE system. In

addition to the existing modules in MACE, the prototype push system requires only some extra Agentlets and Severlets. There is no need to modify any part of the MACE system.

4.1.1 The Push Subsystem

Figure 4 shows all the internal components of the Push Subsystem. The major function of this subsystem is to delivery information to each subscriber by using mobile agents. It also records the registering information of subscribers into a Subscriber's Records database. The **Information Push Manager** with its two databases are the only extra modules implemented; other components are existing modules in MACE. An Agentlet-Serverlet pair and another single Serverlet actually constitute the **Information Push Manager**.

The Agentlet of the pair is triggered by the insertion of a new information page to the Information Database or by the information administrator via GUIs to generate a Push Agent. This Push Agent has a special service item that will be served locally right away to invoke its associated Serverlet. The Serverlet will pack the new information page into the Push Agent's result area. It then retrieves the sucscribers, who registered with the channels that the new information page belongs to, from the Subscriber's Records database and has them processed before putting them into the navigation part of the Push Agent. The Push Agent is then put into the network by the Agent Launcher to roam to its destinations. In this exemplary implementation, the to-be-pushed information page is packed in a Push Agent's result area instead of as a service item. For rapid prototyping, this approach was chosen mainly to avoid modifying the format of the agents in MACE.

The other Serverlet is invoked when a Subscribe Agent from a subscriber is received. Its simple task is to record those subscription requests carried in the Subscribe Agent into the Subscriber Records database. In this ptototype, the acknowledgement process is skipped, but the implementation of it should be quite straight.

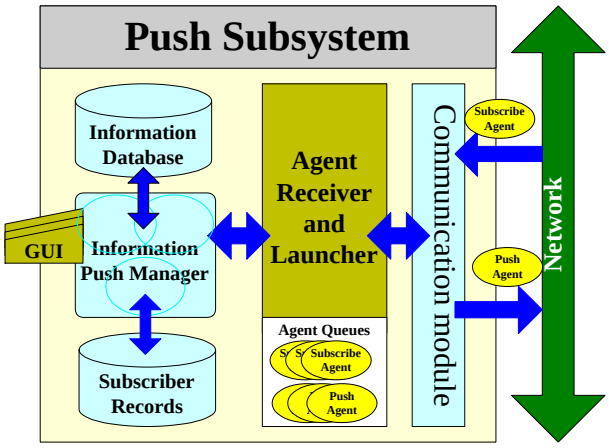

Figure 4: The Push Subsystem at a push server

4.1.2 The Subscribe Subsystem

The internal components of a Subscribe Subsystem is shown as in Figure 5. The major function of this subsystem is to accept information pages carried by Push Agents from a push server. It also sends out Subscribe Agents carrying registering requirements for some specific channels at some push servers. Same as in the Push Subsystem, the **Information Manager** with its two databases are the only extra modules implemented; other components are existing modules in MACE. The **Information Manager** is composed of an Agentlet and a Serverlet; they are not associated with each other, however.

The Agentlet is triggered when a user trys to browse thoses channels cached in the Chanel Database. Subscribe Agents are created if the user decides to register with some specific channels. Unlike a Push Agent, a Subscribe Agent is put into the network by the Agent Launcher, and is transfered to the push server immediately. On the other hand, the Serverlet is invoked by the **Agent Receiver** when a Push Agent is received. It extracts the information page(s) in the received agent and inserts it into the Information Spool, which is an indexed database for storing all the received and classified information pages. For the time being, the Serverlet is equiped with a simple presentation component capable of displaying MIME types of pages. However, other mush more sophisticated programs can be plugged in owing to its open architecture. The notification is now based on a simple pop-up window system, but is also extensible.

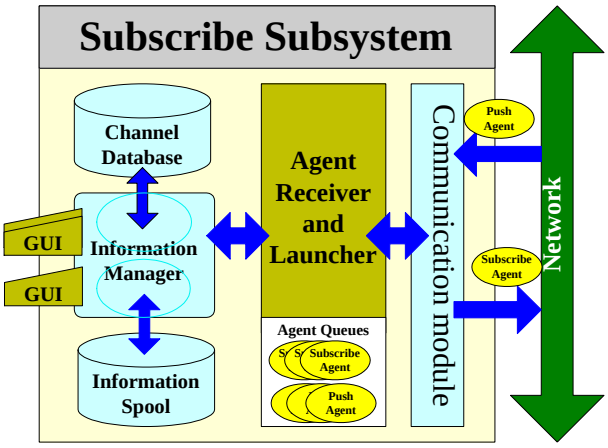

Figure 5: The Subscribe Subsystem at a subscriber's site

4.1.3 Other Subsystems

Other subsystems, such as the Push Cache Server and Agent Dorwarding Module, have also been prototyped. Modules for supporting Push in mobile platforms are also available. Limited by the paragraphs, they are not described here. Their implementations based on MACE, however, are quite straightforward.

4.2 Related Works

In [KST], a new push system that uses the Internet for simultaneous web content delivery to multiple users was developed. The system is claimed to be more efficient than traditional push systems and facilitates the delivery of content in a more timely fashion by using reliable multicasting. The multicasting mechanism is based on IP Multicasting too, thus assumes the availability of M-Bone.

[ROD] also proposed the use of continuous multicast push (CMP) to deliver extremely popular and frequently updated web pages. The benefits of CMP, as claimed in the case of documents with great demands, are highly efficient uses of network resources, reduction of the load on the server, shorter response times, and scalability for an increasing number of information receivers. The work did present a quantitative evaluation of the continuous multicast push for a wide range of parameters. However, the availability of IP Multicast-enabled devices is also assumed.

[HAU], from another perspective, presented a communication and component model for push systems. In constrast to the traditional push systems, the model is a client-server and event-based systems. It is consists of producers and consumers, broadcasters and channels, and a transport system. The component model, they claimed, provides a basis for comparison and evaluation of different push systems and their design alternatives. The paper does compare several prominent push systems using our component model based on the concerns of each of these components. Compared systems include pure pull systems such as Castnet, WebCasting, and Intermind, pull and push combined system such as PointCast and BackWeb, and a real push system WebChannel. The final claim in the work is that the lack of payment models are the most important deficiencies among several open issues that challenge the widespread deployment of push or any other system on an Internet-wide scale.

4.3 Conclusion

The drive to real Push does not mean that the old-fashioned pull style Web browsing will disappear forever one day. For many situations, such as doing one-time researches, ferreting out a specific fact, or just surfing randomly to kill time, pull technology will remain the best and perhaps the only choice. At least, before you decide to send a subscription you might still have to do a lot of browsing on a variety of channels, and that might just be realized purely by pull. What is likely to happen is that Push will become the core technology in many web application arsenals and be accepted unaware and nearly invisibly in many parts of the Web. As one comment ever says that "In some ways the definition of the Web and the definition of Push is getting a little gray, and ultimately it really doesn't matter anyway."

In this paper, we advocate the use of mobile agents in realizing the Push technology and propose a reliable and real push framework that uses mobile agents as it transport system. By also taking advantages from the gradually maturing IP Multicast and the Tunnelling technologies, it will reduce network traffic, use bandwidth efficiently, and require no sepcial routers for the time being. The last characteristic reveals a great chance to advance software techniques before the

associated hardwares are available. It also provides chances to accelerate the widespread of many new products, such as of Information Appliances.

References

[AGL] Aglets: Mobile Java Agents, IBM Tokyo Research Lab, URL = http://www.ibm.co.jp/*trl/projects/aglets*

[CTB] S. Covaci,; Zhang Tianning; I. Busse, "Java-based intelligent mobile agents for open system management", In Proceedings of Ninth IEEE International Conference on Tools with Artificial Intelligence, Page(s): 492 -501, 1997.

[FGS1] W.M. Farmer, J.D. Guttman and V. Swarup, " Security for Mobile Agents: Issues and Requirements, ". Proceedings 19th National System Security Conf. (NISSC 96), 1996, pp.591-597.1997.

[FOR] Forrester, Leslie Ann. "Push Technology" in Legal Assistant Today. May/June 1998. p36-37.

[GRA1] Robert Gray, David Kotz, Saurab Nog, Daniela Rus and Geoge Cybenko. "Mobile Agents for mobile computing.", Technical Report PCS-TR96-285, Department of Computer Science, Dartmouth College, May 1996.

[GRA2] R.S. Gray. "Agent Tcl: A Flexible and Secure Mobile-Agent System," in Proc. Fourth Annual Tcl/Tk Workshop (TCL 96), 1996.

[GEN] General Magic, "Agent Technology: General Magic's Odyssey", http://www.genmagic.com/html/agent_overview.html, 1997.

[HAU] M. Hauswirth, M. Jazayeri, "A Component and Communication Model for Push Systems", in Proceedings of the ESEC/FSE 99 - Joint 7th European Software Engineering Conference (ESEC) and 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-7), September 6-10, 1999, Toulouse, France.

[KLO] G. Karjoth; D. B. Lange; M. Oshima, "A security model for Aglets", IEEE Internet Computing, Volume: 1 4 , Page(s): 68 -77, July-Aug. 1997.

[KK1] Keith D. Kotay and David Kotz, "Transportable Agents ". In Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management, Gaithersburg, Maryland, December 1994

[KGN] D. Kotz; R. Gray; S. Nog; D. Rus; S. Chawla; G. Cybenko, "AGENT TCL: targeting the needs of mobile computers", IEEE Internet Computing Volume: 1 4 , Page(s): 58 -67, 1997.

[KOA] G. Kunito, Y. Okumura, E. Aizawa, M. Hatori, "Tracking agent: a new way of communication in a multi-agent environment" , Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on , Volume: 2 , 1997 Page(s): 903 -907 vol.2

[KST] S. Kinoshita, T. Shiroshita, T. Nagata, "The RealPush Network: A New Push-type Content Delivery System using Reliable Multicasting", IEEE Transaction on Consumer Electronics, Vol. 44, No. 4, pp. 1216-1224.

[LAN] D.B. Lange and M. Oshima, Java Agent API: Programming and Deploying Aglets with Java, published by Addison-Wesley, Fall 1997

[LWS] S. Lazar, I. Weerakoon, D. Sidhu, "A scalable location tracking and message delivery scheme for mobile agents" , Enabling Technologies: Infrastructure for Collaborative Enterprises, 1998. (WET ICE '98) Proceedings., Seventh IEEE International Workshops on , 1998 Page(s): 243 -248

[MBO1] "MBONE:Multicasting Tomorrow's Internet". URL=http://www.savetz.com/mbone/.

[ORD] J.J. Ordille, "When Agents Roam, Who Can You Trust?" Proc. Fitst Conf. on Emerging Technologies and Applications in Communications(etaCOM), May 1996.

[PTV1] A. Puliafito; O. Tomarchio; L. Vita, "MAP: Design and implementation of a mobile agents' platform", In Journal Of Systems Architecture, Vol 46, Issue: 2, pp. 145-162, 2000

[ROD] P. Rodriguez and E. Biersack,"Continuous multicast push of web documents over the internet", IEEE Network Magazine, vol. 12, 2, pp. 18–31, Mar-Apr 1998.

[TAR] J. Tardo and L. Valente. "Mobile Agent Security and Telescript," Proc. IEEE CompCon 96, IEEE Computer Society Press, Los Alamitos, Calif., 1996

[TFM] C. Thirunavukkarasu, T. Finin, and J. Mayfield. "Secret agent: A Security Architecture for the KQML Agent Communication Language," Proc. Intelligent Information Agents Workshop held in conjunction with Fourth Int'l Conf. Information and Knowledge Management CIKM 95, Baltimore, Dec. 1995.

[WAN1] T. I. Wang, "A Mobile Agent Carrier Environment with Mobile Computing Facilities", IIP: International Conference on Intelligent Information Processing, The 16th IFIP World Computer Congress.21-25/08, 2000, Beijin.

[WAN2] T. I. Wang, "A Mobile Agent Carrier Environment for Mobile Information Retrieval", 11-th International Conference on Database and Expert Systems Applications - DEXA 2000, 05-08/09, 2000, Greenwich, London.

[WAN3] T. I. Wang, "A Mobile Agent Carrier Environment", ICS2000, 6-8, December, 2000, Chiayi, Taiwan, R.O.C.

[WHI] J. E. White, "Telescript technology: The foundation for the electronic marketplace." General Magic White Paper, 1994.