

Pervasive Messaging

Jan-Mark Wams, Maarten van Steen

Vrije Universiteit, Amsterdam
Department of Computer Sciences
{jms,steen}@cs.vu.nl

Abstract

Pervasive messaging is the part of pervasive computing that enables users to communicate with each other. Many of today's electronic messaging systems have their own distinct merits and peculiarities. Pervasive messaging will have to shield the user from these differences. In this paper, we introduce a taxonomy for electronic messaging systems, providing a uniform way to analyze, compare, and discuss electronic messaging systems. With this taxonomy, we analyze the current practice, demonstrating its shortfalls. To overcome these shortfalls, we introduce a novel messaging model: The Unified Messaging System. This system can, in fact, mimic any electronic messaging system, thus providing powerful unified messaging.

1 Introduction

Pervasive computing is coming. We believe *pervasive messaging* evolving, in part, from current Electronic Messaging Systems (EMSs) like e-mail, i-mail, USENET News, Instant Messaging (IM), paging, UNIX talk, Web Logging, and Short Message Service (SMS), will be on the forefront of pervasive computing.

The pervasiveness of an EMS increases with the number of users that are connected through it. Linking EMSs together is an effective way to increase their pervasiveness. Linking could, for example, enable a GSM cell phone to send an SMS message that will be received as an e-mail message.

Currently many EMSs are connected to Internet e-mail. Whether e-mail can or should be used to integrate all EMSs seems an open question. Although e-mail provides the proper means to transfer messages, we argue that there are inherent problems with integration through e-mail. For example IM has synchronous properties, and USENET News uses flooding to gain high accessibility. It is not obvious how e-mail could be used to integrate these two systems.

We argue that pervasive electronic messaging requires rethinking the messaging model such that existing systems can be integrated transparently into a unifying framework. In other words, a transparent integration of EMSs or a *Unified Messaging System* (UMS) is needed. A pervasive messaging system should provide unified messaging *anytime* and *anywhere*. Equally important is that the UMS gives users control over what they receive. We strongly believe that the UMS should put the *recipients in control*, not the senders.

In this paper, we describe a UMS model that meets these requirements and indicate how it can be implemented efficiently on a worldwide scale. We make the following two contributions. First, we provide a taxonomy that allows to compare very different EMSs. Second, we provide a model that unifies existing EMSs and that can be used as the basis for integrating these systems. We demonstrated the feasibility of our model by writing a simple proof-of-concept implementation.

This paper is organized as follows. In Section 2 we present a taxonomy that allows us to compare existing EMSs and explain what role e-mail has with respect to pervasive messaging. In Section 3 we describe our model and in Section 4 we briefly describe our proof-of-concept implementation, followed in Section 5, by concrete examples that run on our current implementation. We conclude in Section 6.

2 Pervasive Messaging

Pervasive messaging is ubiquitous, baseline communication at anytime under what ever circumstance [7]. A user does not want to be bothered with the incompatibilities between Yahoo Messenger and other EMSs like AOL Instant Messenger (AIM), ICQ, or MSN Messenger. A user just wants the message delivered. Most users know that those IM systems do not have a connecting path to the fax or USENET News system. They have to be aware of these matters. In a truly pervasive messaging system, a user does

not have to consider which system is being used by the recipient, or how two EMSs are connected.

Pervasive messaging should (1) transparently offer everything that is provided by the current EMSs and (2) be accessible anytime, anywhere. Since mobile phones and pagers integrate rapidly with Personal Digital Assistants (PDAs), we feel confident that, in the near future, there will be messaging devices allowing anytime, anywhere access, thus solving the second requirement. This paper is about solving the first requirement: how to integrate the current EMSs. The current situation is illustrated in Figure 1 which shows two interfaces (a PC and a GSM phone) and a small selection of current EMS protocols. Many messaging sys-

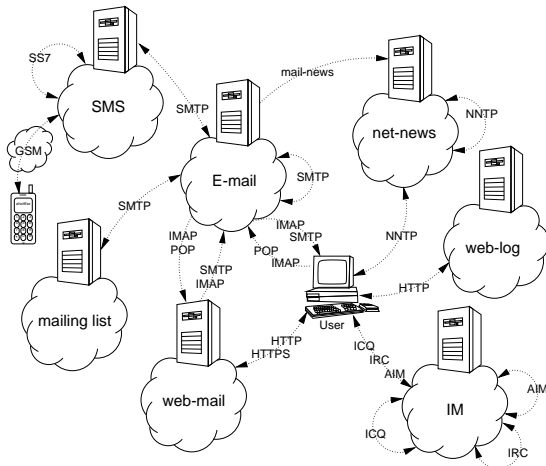


Figure 1. A few interfaces and EMS protocols.

tems are connected to the biggest and best known EMS: Internet e-mail. It might be stated that e-mail is the de facto nexus, connecting many EMSs. Having a nexus simplifies the path towards integrating EMSs. Internet e-mail is probably the best choice for this central role, but it is questionable whether it will carry us all the way into pervasive messaging. We will show that e-mail cannot sustain that connecting role in the future. In fact, we believe that non of the current EMSs can take that central role. To pinpoint where e-mail and other EMSs fall short, we have to analyze and categorize them.

Despite considerable debate, there is no consensus how to compare or categorize EMSs. It is not obvious how to compare the fax system to USENET News. There is not even a common set of topics or keywords for papers on EMSs. Therefore, we first introduce a taxonomy for EMSs.

2.1 A Taxonomy for EMSs

Our taxonomy is organized along the four most important aspects of EMSs from a *user's perspective*, as opposed to a technical or design perspective. With this taxonomy,

any EMS can be scaled with respect to four independent dimensions. Figure 2 shows the four dimensions and their values. An EMS can have one of three values in the time

dimension	values
time	immediate, impermanent, permanent
direction	simplex, duplex
audience	world, group
address	single, list, all

Figure 2. The EMS taxonomy.

dimension: *immediate*, meaning that all messages are short lived or available only once during a short period; *impermanent*, meaning that all messages are available pending their expiration or revocation by some set of rules; *permanent*, meaning that all messages are available indefinitely unless a message is explicitly revoked by an authorized user.

An EMS can have one of two values in the direction dimension: *simplex*, meaning that a write only storage or channel is used for message delivery, a reply has to be directed towards another storage or channel; *duplex*, meaning that one store or channel is used for both reading and writing.

The audience of an EMS is the set of users that *can* receive a message through this system. In the audience dimension an EMS can have two values: *world*, standing for every user that has the hardware, software, and connectivity to use the system; *group*, standing for a true subset of all users. In a grouped EMS, users cannot post messages to a user outside their audience, even though this outsider is ready for any message and uses the same system. Restriction of audience (grouping) can be the result of restrictions related to the infrastructure or implementation. The system can also limit the audience as a service, security measure, or due to politics.

In the address dimension an EMS can have three values: *single*, if the system allows only one recipient per message; *list*, if the system allows for addressing more than one explicitly addressed recipient; *all*, if the system allows for some form of broadcasting.

The four dimensions are truly independent, although not all of the 36 combinations are equally useful.

Note that it is easy to confuse audience and address: both are subsets of recipients. The audience comes with the system and users have no direct influence on it. The address is something the user determines (per message) and the system has no influence on. The intersection of audience and address is the set of recipients that will receive the message.

In Figure 3 we show three EMSs and their classification according to this taxonomy.

	e-mail	USENET News	fax
time	permanent	impermanent	immediate
direction	simplex	duplex	simplex
audience	world	group	world
address	list	all	single

Figure 3. Classification of three current EMSs.

2.2 E-mail Will Not Cut It

From the position e-mail has in this taxonomy, (permanent, simplex, world, list) it becomes clear where the shortfalls are. The Internet e-mail system lacks duplex communication and it is hard to address large groups of users [8]. In fact, e-mail covers only 12 out of 36 possible combinations.

The key to unification does indeed not lie with e-mail, nor any other current EMS. To see why, imagine a user who is about to post a message. This user has to determine (subconsciously) the value for each dimension. This allows her to select a proper EMS for posting the message in question. Ergo, if we want to free the user from having to select the proper EMS, we need an EMS that can handle all combinations. We conclude that we need a *Unified Messaging System* (UMS). This UMS must be able to handle all 36 variants of messaging. In the next section, we describe such a UMS.

3 The Unified Messaging System

Basically, the UMS is a middleware layer that manages the distribution of message objects. On top of the UMS middleware layer sits the application layer, providing a (graphical) user interface or proxy for lightweight pervasive devices. Several name space layers can operate next to the UMS layer. To offer compatibility with current EMSs during the diffusion phase, the application layer could use one or more legacy EMS layers (see Figure 4).

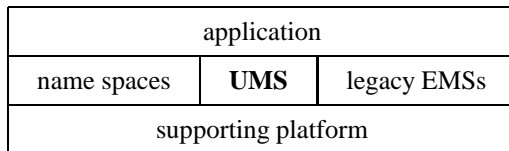


Figure 4. The position of the UMS layer.

3.1 Definitions and Goals

In the UMS model each message is *targeted* because it is directed towards a specific user or a group of users, a message is *immutable* because it cannot be changed after it has

been sent and it is usually *short*. These properties lead us to use the term *Targeted Immutable Short Message* (TISM). We also use the name *target* to denote the destination of a TISM. Throughout this section, the word *user* can be read as *the application program on behalf of the user*.

The UMS model does not include methods for presence, secrecy, authentication, non repudiation, and integrity control, because these can be done *end-to-end* (i.e., at the application level) [3].

The UMS model is designed with the following goals in mind:

1. *Large-scale messaging*: handling hundreds of billions of TISMs per day between billions of users.
2. *Independence of trusted sites*: allowing (a combination of) a client/server or a peer-to-peer communication model [2].
3. *Prevention of spam*: preventing unsolicited messages, without restricting the freedom of speech.
4. *Orthogonality of dimensions*: deciding on time, direction, audience, and address independently.

3.2 TISMs and Targets

It is no surprise that the main objects in the UMS model are: *TISM*, containing a short text (i.e., a subject) and a long text (i.e., the message body); and *target*, which contains a short text (i.e., a name or description) and a set of TISMs.

3.3 Protection and Identification

A target protects each TISM with public key encryption and a message digest [4]. In our model each target is associated with a unique *post-key/read-key* pair. To post a TISM, the proper post-key is needed. Likewise, to read a TISM, the proper read-key is needed. Without a read-key, it is sufficiently hard to reconstruct a TISM, even if a post-key and a copy of the encrypted TISM are available. Without a post-key, it is very hard to spoof a TISM even if the read-key is available. The UMS will generate a new post-key/read-key pair for every new target.

In the UMS a target is identified by a systemwide unique binary string. We define a *target-ID* as this unique binary string. A (post-key, read-key, target-ID) tuple is denoted as *post/read-tuple*. Likewise we use *read-tuple* and *post-tuple*. When the UMS creates a target for a user, the user is returned a post/read-tuple, from which a separate post-tuple and read-tuple can be created. Typically, a user might create a target and distribute its read-tuple to others, enabling them to get the encoded TISMs from the target (using the target-ID) and to decode those TISMs (with the read-key). This is similar to a Web-log EMS. Had the user distributed the post-tuple, an e-mail like EMS would have resulted.

The UMS user has a number of ways to distribute (key, target-ID) tuples. A user could: pass on a tuple wrapped in a TISM; distribute a tuple through the World Wide Web; or store a tuple in a specially designed local name space system. Other lookup models are also feasible. Note that not being bound to any particular lookup mechanism or address space is one of the strengths of the UMS.

3.4 Taking Control

To utilize the fine-grained control the UMS offers, the user needs a separate target for each different communication partner or group. This may sound complex, especially to users that manage all their Internet e-mail from one, so called, in-box. However, most e-mail users already have many sub-mailboxes. Likewise, most IM systems allow users to create any channel/room they want to. As another example, every USENET News user can create a new `alt.*` group at will (like the actually existing `alt.-swedish.chef.bork.bork.bork`). Creating a new box or channel, in one of these legacy EMSs, is limited by the ability to create a new entry in the accompanying name space. For example, finding a meaningful name for a new `alt.*` USENET News group that does not already exist, is hard, as is the case for IM channels/rooms.

In the UMS system, the target-ID is not bound to any name so users can easily create *thousands* of targets if need be. Note that this will necessitate support from the application layer to hide complexity.

3.5 Mimicking Legacy Messaging System

Let us now give a coarse description of UMS-based applications that mimic the functionality of the following legacy EMSs:

1. *Internet e-mail*: being large and well known.
2. *USENET News*: targeting groups of users.
3. *Instant messaging*: featuring a real-time component.
4. *Web logging*: featuring subtle rules for posting.

3.5.1 Internet E-mail Imitation

The e-mail system is a (permanent, simplex, world, list) messaging system.

A message-management program (let's call it u-mail) would distribute a post-tuple of a newly created target (let's call it mailbox target). U-mail would further display selected TISMs from the mailbox target. U-mail would allow new TISMs to be posted to any target it holds a post-tuple for.

To be backward compatible with the Internet e-mail system, the u-mail program could also feature legacy protocols like SMTP, POP3, IMAP4.

3.5.2 USENET News Imitation

The USENET News system is a (impermanent, duplex, group, all) messaging system.

A message-management program (let's call it u-news) would allow users to create a new target and distribute its post/read-tuple. The u-news program would list the short text of the TISMs of the subscribed targets. Users can then select the TISMs they want to read. U-news would further allow users to read from and post to those subscribed targets. A fair amount of backward compatibility could be realized here too. In Section 5 we will show a distribution of (key, target-ID) tuples allowing newsgroup moderating.

3.5.3 Instant Messaging Imitation

Most IM systems (like AIM, ICQ, and IRC) are (immediate, duplex, group, all) messaging systems [1].

An IM interface program (let's call it u-talk) would allow users to create a target and distribute its post/read-tuple. U-talk would allow users to select a target. The u-talk program would supply the selected target with a call-back function, that the target would call upon the arrival of new TISMs. The u-talk program would display a split screen, showing all the new TISMs in the top half and allow the user to type in lines of text in the bottom half. The u-talk program would post each line the user types, prefixed with a user alias, as a new TISM. Again backward compatibility could be introduced.

3.5.4 Web Logging Imitation

A Web log is a combination of simplex and duplex communication. In its purest form a blogger (i.e., a user who runs the Web log) appends messages to a page on the WWW. The blogger can append messages and other users can only read the messages. However, users can react to a web log message by posting a follow-up message. The Web log EMS is thus a (impermanent, simplex/duplex, world, all) message system.

A Web-log-management program (let's call it u-blog) would allow a user to create a new target (let's call it blog target) and distribute its read-tuple. U-blog would further allow selecting a blog target and reading TISMs from the selected blog target. U-blog would allow a blogger (i.e., a user in possession of a post/read-tuple) to post a new TISM to a blog target. Appended to this TISM is a post/read-tuple of a newly created target (let's call it follow-up target). U-blog would allow reading from, and posting to, any follow-up target. Obvious extensions, like having multiple bloggers or moderated follow-ups can be realized along similar lines, using still more targets and careful management of the (key, target-ID) tuples.

4 Proof-of-Concept Implementation

We have written a proof-of-concept implementation. This implementation uses a simplistic peer-to-peer protocol to find and retrieve objects.

There are many details like expiring, replication, and encryption, but space limitations prevent us from describing these. For the details we refer the reader to the full implementation, available from our web site. However, there is one aspect that we will just mention, without discussing the implementation details.

Every Target and TISM has a *home location*. This is the place of origin (i.e. the UMS-server on the users network). It is also the only place that *has* to keep this Target or TISM until it expires. There are two important observations to make about this: first, unlike in most EMSs, the *poster* has to supply the (storage, process and bandwidth) resources; second, there is at least one known place where the document is available (until it expires). This illustrates how the UMS puts the recipient in control, contrary to most existing EMSs.

5 Examples

As we have shown, the UMS can be used to mimic existing EMSs, but the model has much more to offer. Proper distribution of (key, target-ID) tuples allows the UMS to implement almost any EMS thinkable. Some might find it surprising that with only three tuples, some novel and complex forms of messaging can be facilitated. In the figures below these tuples are signified by arrows with the labels *post*, *read* and *post/read*. A thicker arrow signifies ownership. The application layer program is depicted as a stick-person and the target is depicted as an amoeba-like blob.

If there would be one target for which all users had a *post/read*-tuple, it would be a form of say-all, hear-all messaging (see Figure 5). It would be hard to deny access to individual users. Note that in the UMS model, it is feasible to have a high-volume target like this without a huge central server, because TISMs are stored with (the home location of) the poster.

This model can be extended. If one user (let's call her moderator) created a new target (i.e., a moderated target) and distributed a *read*-tuple to a number of other users, a form of moderated messaging would result (see Figure 6). The moderator would cross post a selection from the unmoderated target into the moderated target. The beauty of this scheme is that any user can start doing this at any time. By the way, even if the moderated target would contain many TISMs, the required resources for the moderator would be modest for there is no need to copy all the TISMs from the unmoderated target, only the meta information would have to be stored.

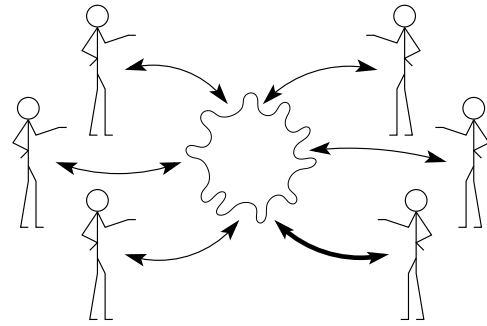


Figure 5. Say-All, Hear-All Messaging.

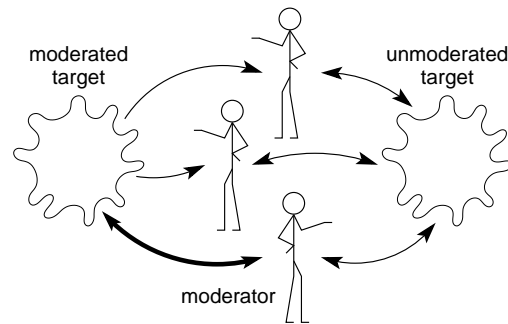


Figure 6. Moderated Messaging.

Sometimes a moderator needs the ability to deny access to an individual user (let's call him BIFF[5]). An EMS can be constructed such that input from BIFF can be made invisible. First the moderator creates a moderated target and distributes the *read*-tuple, just as in the previous example. Then, the moderator solicits from each user, the *read*-tuple for a new target. From these so called input targets TISMs can be cross posted into the moderated target. The result is shown in Figure 7. This way a moderator can oust BIFF simply by ignoring BIFF's input target. Reverse all the arrows in Figure 7 and an output-moderated EMS would result, allowing the moderator to select no, or a number of, appropriate TISMs for each individual user. A combination of both would lead to a highly configurable input-/output-moderated EMS.

Input-/output-moderation is, in fact, a combination of manual selection and manual forwarding. The forwarding part could be done automatically by a computer (i.e., a server). Automatic forwarding is known as a mailing list. A mailing list is a form of controlled reading. If there is controlled reading, there must be controlled posting. This control is actually missing from the classical Internet e-mail mailing list EMS. Any user that has the mailing list's address can post messages to it, even if the user is not on the list.

A fully controlled mailing-list EMS is depicted in Fig-

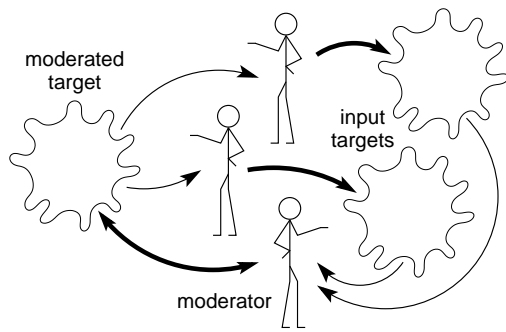


Figure 7. Input Select Moderation.

Figure 8. By controlling the automatic forwarding A, B, C, and D, it becomes possible to control which individual user can post and/or read.

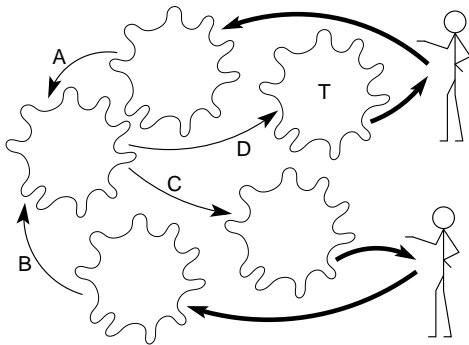


Figure 8. Fully Controlled Mailing List.

It can be hard to get rid of an Internet e-mail mailing lists subscription, but it is always simple to unsubscribe from an UMS mailing list, because in this system, a user does not depend on another person to stop the forwarding. The user that is reading from target T can stop reading from it at any time, and because target T collects TISMs *only* from this specific mailing list, no other TISMs are lost.

This is a general feature of the UMS model. Users can be very selective, since each communication platform is supported by an ad hoc created EMS. Users will have many targets to receive TISMs from: their spouse, their boss, their mother, their company's mailing list, their hobby club, their government, and so on, each of which can be ignored independently without further ado. As mentioned above, the poster is initially responsible for resources, not the receiver.

We have shown, with these few examples, that careful creation of targets and distributing their (key, target-ID) tuples, can lead to the creation of sophisticated EMSs.

6 Conclusions and Future Research

In our research, we have looked at the history, current status, and possible futures of Electronic Messaging Systems (EMSs). A number of our conclusions are presented in this paper. We have presented a taxonomy of EMSs and we have shown that pervasive messaging depends on integrating most, if not all, current EMSs. However, if EMSs are to be integrated, simply interconnecting them through Internet e-mail will not do. Rethinking the messaging model is necessary. The Unified Messaging System (UMS) is proposed as a solution. The UMS allows to easily create novel and sophisticated, ad hoc EMSs that allow a fine-grained control far beyond the capability of current EMSs.

We have created a simple proof-of-concept implementation in Java that can be downloaded from the WWW (see <http://www.cs.vu.nl/~jms/ums/>). It is inspired by results from the Globe wide-area distributed system that has been developed in our research group [6]. The next step in our research is working out the detailed design issues of a real life implementation.

References

- [1] M. Day, J. Rosenberg, and H. Sugano. A Model for Presence and Instant Messaging. *RFC 2778*, February 2000.
- [2] Dejan S. Milojevic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyn, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, Hewlett Packard Laboratories, Palo Alto, CA, March 2002.
- [3] J.H. Saltzer, D.P. Reed, and D.D. Clark. End-to-End Arguments in System Design. *ACM Trans. Comp. Syst.*, 2(4):277–288, November 1984.
- [4] B. Schneier. *Applied Cryptography*, page 435–441. John Wiley, New York, 2nd edition, 1996.
- [5] R. Sexton. The story of BIFF. <http://www.vrx.net/usenet/history/biff/>
- [6] M. van Steen, P. Homburg, and A.S. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, 7(1):70–78, January 1999.
- [7] M. Weiser. The Computer for the 21st Century. *Scientific American*, pages 67–83, September 1991.
- [8] A. Westine and J. Postel. Problems with the Maintenance of Large Mailing Lists. *RFC 1211*, March 1991.