A Flexible Middleware Layer for User-to-User Messaging

Jan-Mark S. Wams and Maarten van Steen

Vrije Universiteit Amsterdam
Department of Computer Science
{jms,steen}@cs.vu.nl
http://www.cs.vu.nl/~{jms,steen}

Abstract. There is growing trend to unify user-to-user messaging systems to allow message exchange, independent of time, place, protocol, and end-user device. Building gateways to interconnect existing messaging systems seems an obvious approach to unification. In this paper we argue that unification should take place at the level of the underlying messaging models. Such a unification results in one messaging model that has maximum adaptability, allowing one system to deliver the same messaging services that all currently existing messaging systems deliver, as well as hitherto impossible mixes of those services.
We present a novel unified messaging model that supports maximum adaptability. Our approach supports the same services that all current messaging models support, including those of e-mail, fax, SMS, ICQ, i-mail, USENET News, AIM, blog, MMS, and voicemail.
To substantiate the claim that such a unified model can be implemented efficiently on a worldwide scale, we present the design of an accompanying highly adaptable and scalable messaging middleware system.

1 Introduction

User-to-user messaging services continue to increase in popularity. Billions of messages are daily relayed through messaging systems like e-mail, fax, SMS, ICQ, i-mail, USENET News, AIM, blog, MMS, voicemail and so on. For many people it is hard to imagine life without these services.

Technological change as well as change in expected service cause new features to be added to existing messaging systems and totally new systems to emerge. However, the unstructured way in which messaging systems have been constructed and changed so far has caused much unnecessary overhead, reinventing of wheels, and running into dead-ends that should have come as no surprise. Besides these development problems, there is an ever-growing incompatibility between all these systems that seemingly offer a very similar service. It would already be a huge step forward if the choice for the sending system would be independent from the choice of the receiving system. Users would then, for example, be able to use a cell-phone to send a photo to a bulletin board, and another user would be able to use a laptop in a café to look at it, while yet another user would receive the photo using a fax machine.

J.-B. Stefani, I. Demeure, and D. Hagimont (Eds.): DAIS 2003, LNCS 2893, pp. 297–309, 2003.
© IFIP International Federation for Information Processing 2003

Most existing messaging systems are built directly on top of one communication platform like the Internet, GPRS, or POTS. In this paper we introduce a middleware layer for user-to-user messaging systems that provides maximum adaptability. Maximum adaptability is needed to deal with the multitude of communication technologies and messaging services.

Adapting a messaging service to changes is usually realized by adapting (often extending) its **underlying system**. For example, user demand for off-line and multi-point access to Internet e-mail has lead to extensions known as POP and IMAP. Likewise, user demand for an electronic bulletin board system prompted the development of mailing-list servers on top of Internet e-mail. However, due to fundamental difficulties with addressing an ever-changing population [7], it turned out to be impractical to build a full-fledged bulletin board system like USENET News on top of Internet e-mail. Researching messaging systems has lead us to conclude that the associated messaging **model** sometimes lacks enough adaptability to make adapting the system practical. To assess the adaptability of a messaging service both the model as well as the system that implements it, need to be analyzed. In Section 3 we present a taxonomy, that allows analysis of messaging models.

Princeton University's WordNet defines "adaptability" as "the ability to change or be changed to fit changed circumstances." In the context of a taxonomy, "change" can be interpreted as a change of position. Hence we define:

> ***A system has "maximum adaptability" within a given taxonomy, if the system can easily move or be moved to any position within that taxonomy.***

Using our taxonomy to categorize (the models of) messaging systems, we show that non of the popular large-scale messaging systems has maximum adaptability; most such messaging systems lack an adaptable model . In Section 4 we introduce a unified messaging model that does have maximum adaptability and, therefore, could be used as a basis to unify all major existing messaging systems, including e-mail, fax, SMS, ICQ, i-mail, USENET News, AIM, blog, MMS, and voicemail. In Section 5, we focus on the main contribution of this paper, the design of a middleware layer for messaging with maximum adaptability. We dubbed this system **Unified Messaging System** or UMS for short. Our design demonstrates the feasibility of a large-scale UMS that supports maximum adaptability, and which is capable of providing the same services as existing messaging systems. To the best of our knowledge such a UMS does not yet exist. In Section 6 we give some example scenarios of the use of the UMS middleware layer. We conclude in Section 7.

2 Related Work

Work on "Unified Messaging" shows that some define unified messaging as "the ability to allow a user to receive faxes, voice-mails and e-mails in one personal

mailbox." [8]. Since most user-to-user messaging is either Internet based or telephony based, unified messaging is sometimes defined as; "Internet-based integration of telephony and data services spanning diverse access networks" [6]. Neither of these approaches to unification comes close to ours.

There are also commercial offerings of unified messaging, basically these are centralized places where users can collect all there messages, usually limited to e-mail, voicemail, faxes, and phone-text messages. These kind of unifications are designed and/or implemented with gateways, interconnecting various systems, leading to complex addressing and a common-denominator service. Also these forms of unification are usually asymmetric in how they handle the sending and receiving of messages. Receiving messages from different messaging systems is straightforward: have one gateway/forwarder per connected messaging system. However, sending is not always (if at all) possible in a uniform way: differences between the way that messages can be delivered have to be dealt with by the user, often even if the recipient uses the same integrated messaging service. Our unified messaging is different in that we strive for one middleware layer, connecting many communication networks to user interfaces on many platforms.

Though the user-interfaces will, most likely, have to be redesigned and replaced, this approach has some enormous benefits. The number of interfaces reduces, combinations of messaging services become possible, it is much easier to add a new user interface or new communication network, and, most importantly, there is symmetry between sending and receiving.

3 A Taxonomy for Messaging Models

In this section we introduce a taxonomy for messaging models. Issues like throughput, latency, and portability are implementation dependent, and, although important, are ignored in our taxonomy. The taxonomy does also not take into account presence, secrecy, authentication, non repudiation, and integrity, because these can (and should) be done on an end-to-end basis [2]. We introduced our taxonomy in a previous paper [5], so we will just list the four dimensions and their values in Fig. 1.

dimension	values (from min to max)
time	immediate, impermanent, permanent
direction	simplex, duplex
audience	group, world
address	single, list, all

Fig. 1. The messaging system taxonomy.

Given this taxonomy, models of existing messaging systems can be classified and researched for adaptability. Fig. 2 shows the classification of eight messaging systems. Many interesting observations can been made when using the taxonomy to

system	time	direction	audience	address
e-mail	permanent	simplex	world	list
voicemail	permanent	simplex	world	single
news	impermanent	duplex	group	all
mailing-list	permanent	duplex	group	all
fax	immediate	simplex	world	single
IM	immediate	duplex	group	all
SMS	permanent	simplex	world	single
blog	permanent	duplex	world	all

Fig. 2. Classification of some existing messaging systems.

compare messaging models. For example, the fax messaging system and the SMS messaging system both are (immediate, simplex, world, single) systems, revealing that they share an underlying model. Due to the different output devices and infrastructure, their systems are, however, very different and incompatible. Interestingly but not surprisingly the SMS system is meeting exactly the same user demands for service extension that the fax system (invented over 100 years earlier) has met. Users will want support for sending a single message to multiple recipients, automatic forwarding to other recipients or locations, non-repudiation, authentication, and so on. Note that voice-mail systems are also (immediate, simplex, world, single) and was also confronted with similar user demands. As a side effect of our research, we have come to conjecture that messaging systems that have coinciding positions in the taxonomy, usually have coinciding development paths. Another interesting observation is that when a system is built on top of another system, its classification clearly reveals what property (if any) has been downgraded in favor of the upgrading of some other property. For example, the mailing-list messaging system is built on top of the e-mail system, downgrading audience in favor of upgrading its addressing capabilities.

Most messaging systems do not have the maximum value in all dimensions, in other words, they do not possess maximum adaptability. There is one messaging system in Fig. 2 that does have the maximum value in all dimensions: the blog system. The blog system (also known as weblog or what's-new-page) is a messaging system in which one person (or a small number of persons) collects noteworthy messages. Potentially every person on the Internet can read these messages and can append addenda (a message and its addenda are often called a "thread"). This type of messaging is more subtle than the other messaging systems with respect to controlling who can post what message. More precisely, most implementations of the blog-type messaging system put some restrictions on its duplex character (i.e., not every participant can start a thread). Some blogs have many readers (like slashdot.org), many blogs have a few dedicated readers. Since a blog system is (permanent, duplex, world, all), it might possess maximum adaptability. Unfortunately, it is stuck at the extreme of the taxon-

omy. Bloging is relaying a message to anyone who wants to hear it, therefor, it has to be adapted to enable more private forms of communication. We have come to the conclusion that non of the messaging systems we know of can be easily forged into a system with maximum adaptability.

4 The Unified Messaging Model

In this section we introduce a Unified Messaging Model (UMM) as underlying model for the messaging system we will define in Section 5. The UMM has to support four major objectives. First, the UMM has to support large-scale messaging: billions of users jointly exchanging billions of messages per day. Second, the UMM should not be dependent on trust and should allow for a distributed peer-to-peer implementation. Third, the UMM has to hinder SPAM by putting the recipient in control. Fourth, the UMM has to posses maximum adaptability: it should offer everything any existing messaging system has to offer. Moreover, it should allow users (or user agent software) to dynamically create new types of messaging systems on demand, mixing and matching messaging service properties at will.

We have kept our model as simple as possible. It has two main entities. The first entity is called **target**, and stands for a collection of user-to-user messages. A target is a generic in-box, news-group, channel, paper-role, tape and so on. The second entity is called **TISM**, short for **targeted immutable short message**. A TISM is a generic e-mail, news article, remark, fax, voice-mail-message and so on. The acronym TISM is used to reflect the choices we have made for our UMM. TISMs are targeted because they are directed towards targets—not users. They are immutable by design: that which has been sent, can no longer be modified. TISMs are said to be short, thereby focusing the UMM on real-life user-to-user messaging. The focus is thus not on large messages of, say, millions of bytes. To give the UMM maximum adaptability, four (dimensional) constraints are put upon targets. First, targets should allow for immediate, impermanent, and permanent storage. Second, targets should support an access control mechanism for posting TISMs into targets to allow simplex and duplex messaging on demand. Third, targets should support an access control mechanism for reading TISMs from targets, and to allow both group and world access to targets. Fourth, access to targets should allow individual (single), selected (list), and total (all) access to TISMs. If these objectives can be (orthogonally) materialized into an implementation, the result would be a messaging system with maximum adaptability. We will introduce such a messaging system in the next section.

5 Design of the Unified Messaging System

In this section we introduce the **Unified Messaging System (UMS)**. We start with a description of the implementation of targets and TISMs. The conceptual target and TISM from the UMM is implemented as a distributed shared object. Such an object not only encapsulates state and the implementation of operations on that state, but also encapsulates a distribution policy that prescribes

how the state is distributed, replicated, and migrated across different locations. Distributed shared objects were first introduced in Globe [3]. For our UMS, we adopt the concept of distributed shared objects but provide a specific, more efficient, implementation for targets and TISMs.

In the following, we distinguish between referring to a target by its name, say T, and referring explicitly to its realization as a distributed shared object. In the latter case, we will talk about T's **instance**. A distributed shared object can be thought of as a collection of local objects, where each local object is hosted by a single site. Correspondingly, we will refer to a **local instance** of a target T. Likewise, we make a distinction between a TISM m, its instance, and a local instance of m.

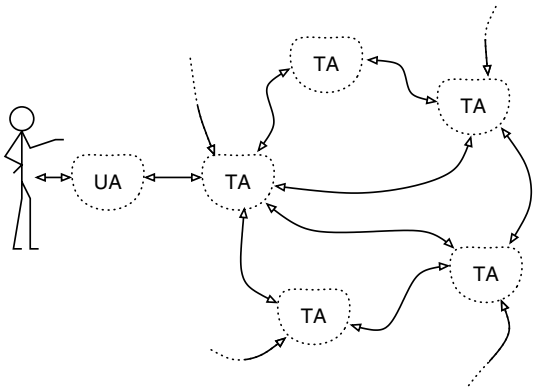

Fig. 3. Conceptual view.

5.1 Target Agents and User Agents

As with the design of our unified messaging model, we keep the design of the UMS as simple as possible. Apart from the actual user, there are only two types of communicating parties, as shown in Fig. 3. The first type is a **target agent** (**TA**). Its primary function is to store and forward (the state of) targets and TISMs, in addition to carrying out the associated distribution policies. As such, a TA is designed to be continuously operating, that is, on line. The second type is a **user agent** (**UA**), which is responsible for generating and managing keys that are need for security, as well providing the user access to facilities for sending and receiving TISMs. The UA is (part of) an application, and is thus not part of the middleware layer. Therefore, we only describe its lower-level part that is responsible for communication with the TAs. We do not go into the specifics of the higher-level parts like the GUI, or the representation of targets or collections.

Whenever a UA wants to create a target or TISM, it needs to contact a TA that is willing to operate as the **home** of that target or TISM. The home TA will store a local instance of every target or TISM created by such a UA, thereby providing minimal access guarantees to those objects. Access is provided until a

specified expiration time, after which the local instance of the target or TISM can be permanently removed.

5.2 Key Management

The UA associates every target T with a (private,public) key pair (K_T^-, K_T^+), where K_T^- denotes a private key and K_T^+ a public key. Any information on target T that leaves a UA, is encrypted with K_T^-. This specifically includes TISMs and information on TISMs that are contained in target T. Every TISM has an associated target that contains it. A TISM has no associated key pair, but instead is encrypted with the private key of its associated target. End-to-end secure communication between UAs proceeds as follows. Whenever a UA encrypts a TISM for storage or transport by a TA, the TISM m is first encrypted with it own unique (random) symmetric key (SK), which, in turn, is encrypted with the key K_T^- that is associated with the target that is containing this TISM. This type of two-staged encryption is referred to as a hybrid protocol [1].

Not only does this scheme save time, because symmetric encryption and decryption is usually much faster than its asymmetric counterpart, but also cross posting will be (computationally) much cheaper. Cross posting is done by reading a TISM, m, from a target, T_1, and than posting it into an other target, T_2. The two targets will have different private keys, but that is of no consequence to the encrypted version of m, which is encrypted with the symmetric key SK. Only SK has to be decrypted and re-encrypted. In general SK will be much smaller than m, therefor decrypting/encrypting this way, takes less time. Cross posting this way will deliver similar gains in communication and storage.

5.3 The Content List

In the UMM a target is an unordered collection of TISMs. In the UMS a target is represented as a list of TISMs ordered by creation date, as shown in Fig. 4-a. We decided on the ordering, because it will make the "old" part of the list more static and thus easier to compress. Imagine a real-time messaging system sending a list of 1 million TISMs over the network every second for comparison to a replica list. Performance wise, a better alternative would be to sent the hash of a list of 1 million TISMs every second for comparison to the hash of the replica list. For efficient comparison and updating of lists, we introduce an elaboration on this simple hashing scheme, dubbed a **contentList**. A contentList is a list of descending dates with between each two dates either a list of TISMs or a hash of a list of TISMs and the number of TISMs in the hash, see Fig. 4. A contentList of a particular TISM list can range in size from a few bytes (see Fig. 4-b) to just over the size of that TISM list (see Fig. 4-e).

5.4 Information Flow

As mentioned, the UAs and TAs exchange information. The principal operation is simple. Each target and TISM is uniquely identified by the combination of a

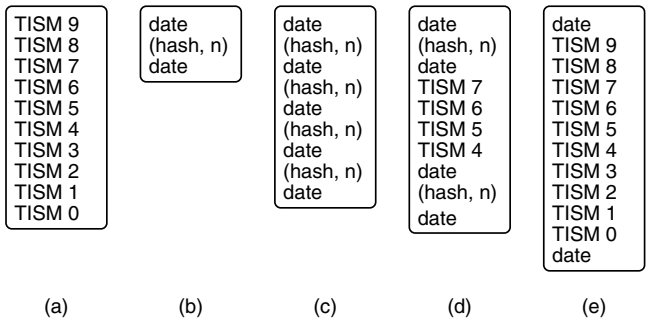

Fig. 4. An ordered list of TISMs in five different formats, (a) plain TISM list, (b) minimal contentList, (c) hash only contentList, (d) mixed contentList (e) TISM only contentList.

home TA and an ID unique relatively to that TA. If a UA needs (information on) a target or TISM it simply contacts the respective home TA. Requests and replies are transmitted in an asynchronous and connection less fashion (notably UDP). As shown in Fig. 5-a, incoming information is processed by the TA using a simple scheme of **decode** and **execute** functions. The decode function is responsible for analyzing the type of input information. For example, we make a distinction between requests for information on a target, and requests concerning a specific TISM. Depending on this type, the decode function passes the request to a specific execute function. The execute function processes the request, possibly storing data in, or fetching data from a local store. A reply is subsequently put in an output queue for transmission to the requester.

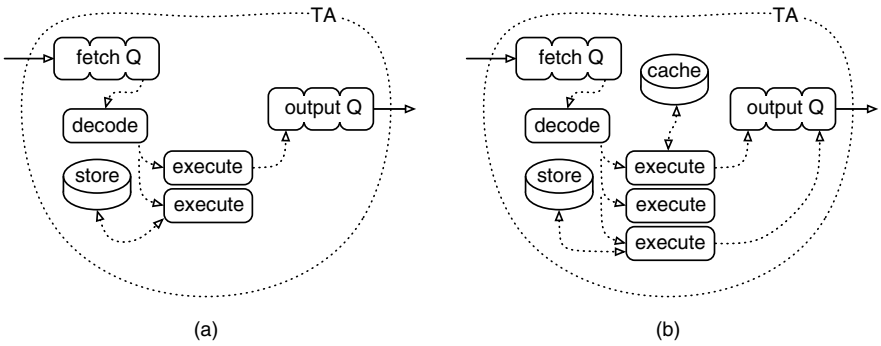

Fig. 5. Information flow within a TA, (a) basic scheme, (b) TA with replication.

With this simplistic scheme, UAs will have to pull any information directly from a home TA. With just this pull-on-demand scheme our messaging system would be fully functional. Obvious drawbacks are its lack of scalability and performance, and also its lack of robustness. These drawbacks are all caused by

the fact that we are relying on a single home TA (per target or TISM). For example, every time a UA would want to know whether a target contained new TISMs, it would have to access the home TA of that target.

This basic scheme can be easily improved. For example, a pushing scheme could automatically deliver the information to a **close TA**: a TA on the same LAN as the UA, or a TA at the ISP. Other adaptations that improve the overall performance of the system can easily be thought of. In the context of developing distributed Web services, we have observed that differentiating distribution schemes is important for achieving performance [4]. For our messaging system, we believe such a differentiation is also important. What we are therefore seeking is an organization by which we can easily associate a specific **distribution** or **replication policy** separately with each target or TISM. Our basic scheme is already capable of supporting this flexibility as the decode function can distinguish different types of input information from which it can derive the appropriate execute function that should be executed.

We considered three ways of differentiating policies. The first option was to allow each target or TISM to run arbitrary (e.g., Java) code. The second option was to invent some replication scripting language and have a target or TISM carry a replication script. The third option was simply to assign identifiers to a fixed number of policies and let each target and TISM specify its prefered policy through this identifier. We chose the third option for both targets and TISMs because it has a low overhead per message. It is, however, reasonable flexible because policies can be added as updates to our messaging system in a backwards compatible fashion.

These adaptations lead to a slightly modified version of the internals of a TA, as shown in Fig. 5-b. First, we make a distinction between the storage and a cache. The storage is used for targets and TISMs for which the TA acts as the home. These objects can be removed only after the specified expiration time. Moreover, requests to store information on a target, or to store a TISM are executed only when they come from an authenticated and authorized UA.

The cache is used to **voluntarily** store information. The TA stays in full control of the cache: it can decide any time what to store or remove. The responsibility for having information on a target or TISM available lies completely with the home TA, which uses its storage for that purpose. There are two ways information finds its way into the cache of a TA. First, a request that has been forwarded by a TA might result in a reply from a peer. This information will then be relayed to the requesting UA and might be stored in the cache. Second, a replication of a target or TISM might be pushed to a TA because a peer executes a replication strategy. The TA could, for example, decide to cache the replica because it is associated with a popular target.

With this in mind, it makes sense for a UA to try to get a target or TISM from a close TA. This close TA will do a cache lookup, and if the lookup fails, it can forward the request to one of its peers. However, to protect against malicious and erroneous requests, only a **signed** request from a **known** UA can result in the forwarding of the request. Basically this means that a TA will only look in

its own cache and storage to satisfy a request. If the request is signed by a known UA, the TA will forward it (unsigned) to a peer, notably the home TA for the referenced target or TISM, but the peer, in turn, will not forward the request any further (for the peer received the request from a TA, not an authenticated UA). Consequently, requests will not be forwarded ad infinitum in search for a non existing target or TISM. It may seem at first that this forwarding restriction can render some information unaccessible. However, targets and TISMs can never become unreachable because replication only improves performance and robustness: targets and TISMs remain available at their respective home TA.

The information flow for Fig. 5-b is similar to that of Fig. 5-a, however, the input flow for Fig. 5-b contains replicas of targets and TISMs that have to be dealt with. Information is fetched and decoded. In the case of a replica the identified replication policy is executed and will, if appropriate, result in the insertion of one or more replicas to the output queue. The replication policy does any of three things; update the cache, queue a reply to a UA, and queue one or more replicas to peer TAs. No general limit on forwarding of replicas is enforced. Due to the fact that replication policies need to be explicitly referenced by an identifier, limitations on replication are safely enforcible. In fact there are no rules for what replication strategy code can do. Typical behavior would include, checking the cache for related objects, storing (local instances of) objects in the cache, forwarding many objects to many peers, updating objects in the store, and so on.

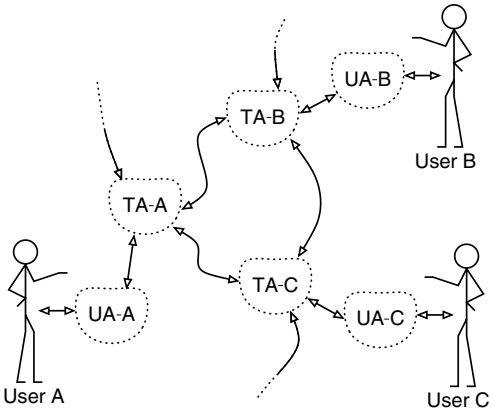

Fig. 6. Three UMS users A, B, and C.

6 Examples

In this section we will demonstrate the behavior of the middleware layer for some archetypical user-to-user communication patterns between three users A, B, and C. Let us assume each of the three users has their own UA and home TA, as seen in Fig. 6. For simplicity reasons, let us assume each UA contacts the middleware layer only through its home TA.

	TA-A		TA-B		TA-C		comments
day	store	cache	store	cache	store	cache	
1	T						target T is created
2	T		m_1				user B posts TISM m_1
3	T	m_1	m_1				user A reads m_1
4	T	m_1	m_1		m_2		user C posts TISM m_2
5	T	m_1 m_2	m_1		m_2		user A reads m_2
6	T	m_2	m_1		m_2		TA-A de-caches m_1
7	T		m_1		m_2		TA-A de-caches m_2

Fig. 7. Cache and store utilization with e-mail like behavior.

	TA-A		TA-B		TA-C		comments
day	store	cache	store	cache	store	cache	
1			T_1				target T_1 is created
2	m		T_1				user A posts TISM m
3	m		T_1	m			user B reads m
4	m		T_2 T_1	m			user B creates target T_2
5	m		T_2 T_1	m			user B forwards m
6	m		T_2 T_1	m		m	user C reads m
7	m		T_2 T_1	m		T_2 m	user C lists T_2

Fig. 8. Cache and store utilization with forwarding.

In Fig. 7 the storage and cache are depicted of the three TAs in a typical e-mail-like exchange. On day 1, user A instructs UA-A to create a new target T. TA-A stores the target and returns a unique ID to UA-A. User A sends users B and C the (address of TA-A, unique ID, public key) tuple in an other TISM (not shown). On day 2, user B has UA-B post a TISM m_1 to user A. TISM m_1 is put in the storage of TA-B, the changes to T are forwarded to TA-A due to the (e-mail mimicking) replication strategy of T. On day 3, user A requests the TISM m_1 from TA-A, and TA-A fetches it from TA-B. TA-A stores TISM m_1 in the cache (due to the replication strategy) and also forwards it to UA-A. On day 4, user C decides to post a TISM m_2 and user A reads m_2 on day 5. At days 6 and 7, m_1 and m_2 are discarded from TA-A's cache. At the end of day seven, T, m_1 and m_2 are not cached in any TA. This will be the typical situation with low-usage targets and TISMs. Note that the UA-A probably has a cache too and that this cache will hold on to T, m_1, and m_2 much longer.

In Fig. 8 we consider some other common behavior: forwarding a TISM to another user. For this, UA-B will create a new target, in which TISM m is cross posted. User A sends a TISM m to user B, and user B forwards TISM m to user C. On day 1, user B creates a target T_1 and forwards the relevant information to user A (not shown). On day 2, user A posts a TISM m into target T. On day 3,

user B reads TISM m and decides that user C might be interested in TISM m. User B requests its UA-B to privately forward m to user C. UA-B creates a new target T_2, cross posts TISM m in target, and then forwards the (home TA of m, unique ID of m) and (home TA of T_2, unique ID or T_2, private key of T_2) tuples to UA-C in another TISM (not shown). On day 6, user C reads m. On day 7, user C decides to list the content of T_2, and finds only m is in T_2. Note that if user A were to post a new TISM into target T_1, this new TISM would not automatically show up in target T_2, as would be the case if user B had shared T_1 with user C. More sophisticated examples can be given, and practical usage of the UMS middleware layer probably is much more elaborated than these examples. It is the task of the UA to transform simple wishes from the user into the usage and creation of targets and TISMs. Typically a user would ask the UA to give some other user access to a set of coherent messages, analogous to the "newsgroup thread" or chain of "e-mail followups."

7 Conclusions

In the paper, we have introduced the design of a truly unified user-to-user messaging system in the form of a generic middleware layer for messaging. Our paper illustrates the feasibility of developing a such a middleware layer to allow efficient integration of existing messaging services. Compared to a messaging service offered by a collection of existing messaging systems and connecting gateways, our approach does not suffer from common denominator restrictions and frees the users from having to deal with differences between the individual underlying messaging systems. Our approach and its accompanying design should be able to support very large communities of users. It is flexible enough to simultaneously support a variety of replication schemes, a feature which has shown to be important when performance is an issue. In fact, we allow differentiation not only on a per-target basis, but can even support different schemes at the level of TISMs. To validate our approach, we are currently developing a prototype implementation.

References

1. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Fifth Printing, August 2001.
2. J.H. Saltzer, D.P. Reed, and D.D. Clark. End-to-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.
3. M. van Steen, P. Homburg, and A.S. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, 7(1):70–78, January 1999.
4. G. Pierre, M. van Steen and A.S. Tanenbaum. Dynamically Selecting Optimal Distribution Strategies for Web Documents", *IEEE Transactions on Computers*, pages 637–651, June 2002.
5. J.M.S. Wams and M. van Steen. Pervasive Messaging. *In IEEE Conference on Pervasive Computing*, pages 499–504, March 2003.

6. H.J. Wang, B. Raman, C. Chuah, R. Biswas, R. Gummadi, B. Hohlt, X. Hong, E. Kiciman, Z. Mao, J.S. Shih, L. Subramanian, B.Y. Zhao, A.D. Joseph, and R.H. Katz. ICEBERG: An Internet-core Network Architecture for Integrated Communications. *IEEE Personal Communications*, pages 10–19, August 2000.
7. A. Westine and J. Postel. Problems with the Maintenance of Large Mailing Lists. *RFC 1211*, March 1991.
8. C.K. Yeo, S.C. Hui, I.Y. Soon, and G. Manik. Unified Messaging : A System for the Internet. *International Journal on Computers, Internet, and Management*, September 2000.