CrossTalk: Scalably Interconnecting Instant Messaging Networks

Marti Motoyama and George Varghese
University of California, San Diego
9500 Gilman Drive
La Jolla, California 92093
{mmotoyam,varghese}@cs.ucsd.edu

ABSTRACT

We consider the problem of interconnecting a simple type of social network: Instant Messaging services. Today, users are members of various IM communities such as AOL, Yahoo, and MSN. Users often want to engage in conversations that span multiple IM communities, since their friends may use competing IM clients. While client-side solutions exist in the form of Trillian and Pidgin, they require multiple logins and offer a subset of the features present in official IM clients. We propose a different solution based on translating gateways that only requires a single login and allows users to keep their existing IM clients. We claim that such interconnection empowers users and encourages the development of third-party applications. We propose using an overlay of *bypass gateways* that avoids many of the scalability limitations of standard gateways. We argue that smaller IM networks have the right incentives to use these gateways, and larger networks cannot easily obstruct bypass gateways. Deploying these gateways into a system we call CrossTalk can ultimately aid in protocol standardization. We describe the architecture of bypass gateways and the implementation challenges faced in interconnecting MSN, AOL, Jabber and Yahoo for IM. We briefly discuss to extensions to other domains such as interconnecting SIP and Skype.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Applications; C.2.4 [**Distributed Systems**]: Distributed applications; D.2.12 [**Interoperability**]: Interface definition languages

General Terms

Design, Standardization

Keywords

Instant Messaging, Interconnection, XMPP, DHT

WOSN'09, August 17, 2009, Barcelona, Spain.

1. INTRODUCTION

> "[LinkedIn] users have been loyal because there is a switching cost. For someone to go and duplicate what they're doing, and even if they came up with a superior platform tomorrow, they wouldn't necessarily switch because they've built their business network. In many ways, the connections can be worth more than the platform." — Jonathan Yarmis, AMR Research [10]

A vast number of socially-based applications have emerged in recent years, providing individuals with numerous information sharing tools. These applications enable new interaction modes (e.g., IM, VoIP), allow content sharing (e.g., file sharing), and facilitate the formation of online communities (e.g., social networks).

However, the value of these social applications is largely derived from the quantity and quality of their user populations. One negative aspect associated with the "network effect" can be observed in such applications as chat and social profiling sites: users will often choose to utilize a service not based upon its quality, but because the provider has achieved the highest popularity in their social circles. In other words, "the connection can be worth more than the platform." As a result of the network effect, users become locked in to a particular vendor's product.

One solution to the vendor lock-in problem is standardization, where a protocol is created that will allow multiple parties to intercommunicate. For example, TCP/IP was developed at a time when many competing vendors were introducing proprietary protocols. The problem with standardization, however, is that the market leaders lack a vested interest in migrating to the standard. This is because standardization forces market leaders to compete on the qualities of their services and not on the sizes of their user populations.

In this paper, we describe CrossTalk, a system that mitigates the problem of vendor lock-in by enabling communication across closed networks using what we call *bypass gateways*. The cost of switching to a new network N with innovative features is reduced because adopters of N can still communicate with their old networks. Besides early adopters, *all* users benefit by gaining the ability to communicate across network boundaries using the client of their choice.

Developers also benefit from our system by gaining a larger audience for their applications. For example, a number of innovative third-party services have already been deployed

over *specific* IM communication dialects, though not much interest has been generated by these applications. These add-ons perform such functions as controlling home devices, tracking the physical location of buddies, and playing FM radio. Each application, however, is limited to a specific closed network. We contend that interconnecting closed IM networks would encourage innovation among developers, since their applications could reach *all* IM users.

Although we focus on IM, we believe that this study provides insight into interconnecting other types of social networks. We implemented bypass gateways for IM, allowing transparent inter-communication between Yahoo, MSN, AIM, and any XMPP-compatible chat network. The use of our gateways and a federated name space (AOL user Alice is represented uniquely as Alice@AOL) can provide many of the benefits that result from the adoption of a uniform standard. However, we suggest that our gateways can further catalyze the movement to a standard.

2. RELATED WORK

We review the disadvantages of two well-known solutions to interconnecting IMs, which motivate our bypass gateways.

2.1 Client Consolidation

In *client consolidation*, the user runs a single application that speaks to every service network via a uniform interface. Examples in the IM world include Trillian, Pidgin [13], and Adium, while examples in the social networking domain [16] include 8hands and Socialstream. However, client consolidation is only moderately popular, possibly due to three fundamental disadvantages:

- Feature subtraction: Existing providers do not expose their internal software interfaces, forcing developers of client consolidation software to reverse engineer each protocol. This generally results in missing or poorly implemented features, since reverse engineering takes substantial effort and specific protocol details remain cryptic. For example, file transfer from Pidgin to AOL frequently fails [1], and video chat performs poorly in Trillian [11].
- Multiple identities: Client consolidation requires that users join as many communities as possible to facilitate connectivity. Wikipedia lists 16 "common" IM protocols in the U.S. but the worldwide list is larger, including country favorites such as QQ in China.
- Software and network overhead: While adding additional software is cheap for a PC, the storage and power consumption that results from running 16 IM protocols on one device can be a problem for thin clients, such as cell phones.

Among these three disadvantages, feature subtraction is most problematic. For example, AOL users that switch to client consolidation software must weigh possibly reduced functionality when communicating with users in their "base networks" (where the majority of their buddies exist) against reaching other communities (where fewer buddies reside). Finally, we note that client consolidation encourages the status quo (multiple closed networks) and does not promote the uniform adoption of an open protocol.

Jabber gateways [7] and Karaka [17] are variants of the client consolidation approach, though much of the work is outsourced to a gateway. Both require multiple IDs; more fundamentally, these solutions only work if the IM protocol supports the notion of a gateway, which appears to only be true for XMPP today. There does not seem to be an incentive for larger IM providers like MSN to introduce these types of gateways.

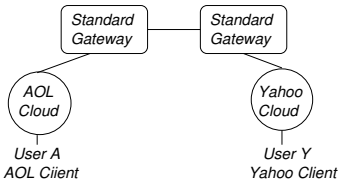

Figure 1: A standard gateway is a client on each network that it connects to and translates between the protocols of each network. Such gateways do not scale because of limits (such as number of buddies) imposed on users by networks.

2.2 Standard Gateways

The classical way to achieve interconnectivity is to use what we call *standard translating gateways*, which are typically used to interconnect networks with different Layer 3 protocols [5]. Such gateways date back to Cerf's original proposal [3] to interconnect the ARPANET and the ARPA Packet Radio Network.

Figure 1 demonstrates how standard gateways function in the context of IM. In Figure 1, such a gateway (commonly known as a "bot") is used to interconnect AOL user A with Yahoo user Y. The gateway on the left poses as a client on the AOL network. A contacts the left gateway when he or she wishes to send a message to Y. The message is then relayed to the gateway on the right, where the message is translated into Yahoo's IM format and sent to user Y.

A commercial example of a standard translating gateway for IM is GTalk2VoIP [6], which allows users to send IMs between MSN and Google Talk. However, users must first add a service "bot" to their contact lists, then adhere to awkward messaging semantics to speak with buddies across networks. For example, to send IMs from MSN to Google Talk, a MSN user must message the service bot in the following way: **IM gtalk:user@domain.com *message***. A more fundamental problem is that standard translating gateways suffer from severe scalability issues.

The scalability problem occurs because the standard gateway is a *client* in every network that requires bridging. Thus, the gateway is subject to the restrictions imposed by the network on its user population. For example, most IM networks limit the number of buddies allowed for a given client (e.g., $<$ 512 for MSN) and only permit communication between two users who are buddies. Even if the network allows the gateway to communicate with an arbitrary number of clients, there is still the issue of how many users a given client can communicate with *concurrently*. For example, a Skype client can only participate in one phone call at a time.

More concretely, suppose we wish to interconnect 100 million AOL clients with the Yahoo network. If a standard gateway wants to relay presence updates (i.e., notification a buddy has come online) between users in the AOL and Yahoo networks, the gateway must become buddies with all users in both networks. But if a single gateway can only have 512 buddies, the gateway must be replicated 200,000 times.

In general, if there are N clients on a network, and there is a restriction that each gateway can only communicate with B clients, then there must be N/B gateways to cover all clients. For Skype, $B = 1$. Such massive replication would also require obtaining N/B accounts on each network. Since the problem arises because a gateway is a client in each network, the natural solution is for the gateways to "bypass" the actual IM networks to achieve interconnectivity.

Thus, we introduce *bypass gateways*, where the gateway appears as a proxy to the user's base IM client while forming an out-of-band network that enables inter-network communication. We propose our gateway technique in Section 3 and provide details regarding a prototype implementation that translates between AOL, MSN, Yahoo, and XMPP.

3. CROSSTALK

We discuss the CrossTalk system components in Section 3.1, the implementation in 3.2, the evaluation in 3.3, sample third party applications in 3.4, and a deployment path in 3.5.

3.1 CrossTalk Components

Figure 2 illustrates the components of the CrossTalk architecture using bypass gateways. First, we employ a simple naming convention to identify users from different IM networks. For instance, if AOL user A wishes to make Yahoo user Y a buddy, A specifies $Y@yahoo$ in A's list of contacts. The system has three components:

- ***Clients:*** Users continue to instant message through *unmodified* client software; thus, we overcome feature subtraction, since CrossTalk users can still access the features present in their official clients while interacting with buddies from the *same* network. However, to make use of our service, each participant must modify a few parameters in his or her client software to point to our gateways (the boxes labeled G in Figure 2). [1]
- ***Gateways:*** The bypass gateways interpose between the original client and servers; however, they also communicate in an out-of-band fashion using a gateway-to-gateway network shown in Figure 2. We assume that each participating institution will run a bypass gateway to serve its user population.
- ***Gateway Network and DHT***: The gateway-to-gateway network is a logical network that consists of machines organized into a Distributed Hash Table or DHT (Figure 2). The DHT is used to store both hard and soft state for users. For example, the DHT trivializes routing between gateways by storing the mapping of user name M (e.g., $M@MSN$) to the IP address of M's bypass gateway.

We now elaborate on the CrossTalk implementation.

3.2 CrossTalk Implementation Details

Our bypass gateway is currently capable of translating instant messages between Jabber/XMPP, AOL, MSN, and Yahoo; market studies [4] show these four to be the leading IM providers. The gateway was coded in Python, as many libraries are available for parsing messages from these protocols. We emulate the gateway network of Figure 2 using OpenDHT [14].

The bypass gateway intercepts, translates, reroutes and passes through IM messages. Each bypass gateway has a corresponding manager that maintains state for each client. For IM, the manager maintains the client's roster and bypass gateway location. The bypass gateways perform three major functions, as depicted in Figure 2.

- ***Translation:*** To avoid the problem of translating between every pair of IM protocols, each bypass gateway translates between a provider-specific format and a canonical format. We chose the IETF-standard XMPP [15] as our canonical format. For example, in Figure 2, IM messages sent by Yahoo user Y to AOL user A will be translated to XMPP by Y's gateway and sent to A's gateway. A's gateway then translates the message to OSCAR (AOL's protocol) before forwarding the message to user A.
- ***Bifurcating Presence Information:*** The gateway must relay the presence information for users to their base networks while updating the user's status in the gateway-to-gateway network. Thus, in Figure 2, when MSN User M comes online, M's gateway will proxy the presence messages (as usual) to the MSN Network. However, M's gateway will *also* (Figure 2) advertise M's presence on the gateway-to-gateway network by updating the state of M in the DHT.
- ***Merging Buddy and Presence Information:*** Each gateway must *merge* the roster of local buddies from a user's base network with the roster of foreign buddies stored in the DHT. For example in Figure 2, suppose AOL User A has an AOL buddy B, and two foreign buddies M (from MSN) and Y (from Yahoo). A will receive a roster containing B from the AOL server, and A's gateway will query the DHT to find that M and Y are A's foreign buddies. A's gateway will then merge B, M, and Y into an updated roster before passing the roster to A.

The bypass gateway performs a core set of operations on all traffic between a client and its chat network. When a client connects to our system, it first establishes a SOCKS4 connection to the bypass gateway, where it specifies the address of the chat server. We chose SOCKS4 over HTTP because a SOCKS4 proxy does not add a level of encapsulation (i.e., HTTP headers) to every message. While a fixed proxy gateway is not fault-tolerant, this problem is shared by other proxies, and can be fixed using a load balancer.

The gateway scans traffic for markers denoting a particular protocol. For example, AIM sends "*" in all its packet headers. Once the client protocol is identified, the gateway parses packets received from both the client and the server. First, the gateway searches for the client's identity. The gateway then waits until the server verifies the user's identity before performing any other action. This is important, because traffic between foreign buddies bypasses the server network. Without this check, a client could spoof another user and attempt to send messages to a foreign buddy.

Once the user's identity has been confirmed, the gateway informs the manager of the user's online status. The manager then inserts two key-value pairs consisting of (UserId +

[1] In our implementation we use a SOCKS4 proxy that requires 2 parameters: one to specify the proxy host name, and one to specify the TCP port number.

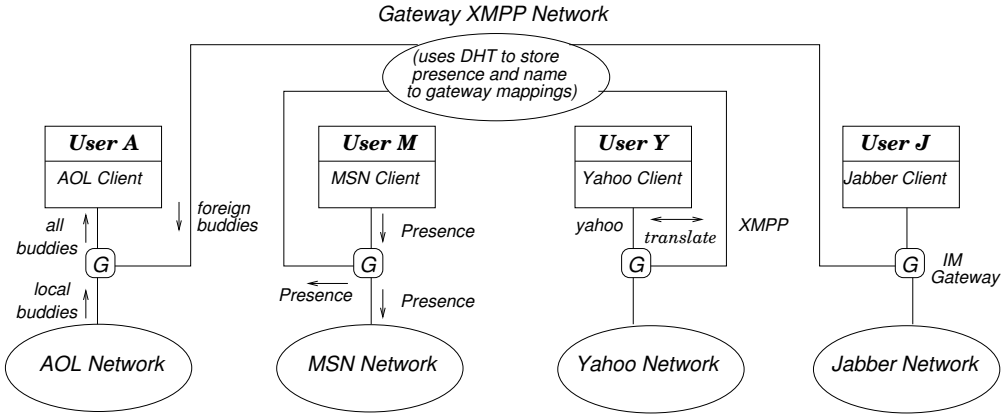

Figure 2: In the IM solution, clients must be behind a bypass gateway denoted by G to speak to foreign buddies who must also be behind such a gateway. The gateways perform three functions: translation, buddy merging, and exporting presence to the DHT.

GatewayToken, GatewayAddress) and (UserId + StatusToken, Online). The manager then retrieves the user's roster, queries the DHT for the location/presence information of the user's buddies, and relays the user's status to the managers of his/her buddies. The manager sends the roster and presence information back to the responsible gateway. The same process occurs when a user logs out.

Next, the gateway intercepts the roster returned from the chat server to perform the buddy merging function alluded to earlier. The gateway merges the roster of foreign buddies returned from the manager with the local buddies sent from the server. Finally, the gateway scans all transmitted instant messages to intercept IMs destined for other IM networks. The gateway also translates and sends inter-IM presence updates.

With this infrastructure in place, let us consider AOL User A who wishes to communicate with Yahoo user Y for the first time. The following steps will take place (see Figure 2):

1. *Initial Contact:* A's gateway intercepts a buddy addition request from A for $Y@yahoo$. The gateway then looks up $Y@Yahoo$ in the DHT to find the IP address of Y's gateway.
2. *Ask for Permission:* A's gateway sends the permission request to Y's gateway. If user Y is online, Y accepts or denies the buddy addition request, and the response is communicated back via the gateways to A. If user Y is not online, information about the pending buddy addition is stored in the DHT. The request will be processed when Y comes back online and its gateway checks the DHT for such pending requests.[2]
3. *Send IM:* Assuming that Y accepts, A adds Y to its buddy list. Now, assume that A sends an IM to Y. The IM is intercepted by A's gateway as usual.
4. *Translate to XMPP:* A's gateway translates from OSCAR to XMPP and sends the XMPP message to Y's gateway.
5. *Translate from XMPP:* Y's gateway translates from XMPP to Yahoo.

[2]While pending buddy addition appears to be a source of a DoS attack, note that we do pending buddy additions only for validated users, and the number of buddies is limited to a few hundred.

		PROTOCOL			
TIME (ms)		AIM	JAB	MSN	YAH
NETWORK RTT	avg	96.0	78.0	54.5	75.0
	std	0.5	20.8	4.6	3.4
	max	98.3	170.1	63.3	80.5
	min	95.4	60.9	47.8	68.5
MESSAGE RTT	avg	101.6	86.6	283.5	82.5
	std	14.4	24.6	41.4	7.6
	max	237.1	192.3	412.6	138.1
	min	93.1	63.0	207.9	72.0
ESTIMATED SPT		5.6	8.6	229.1	7.5

Table 1: Latencies observed in real IM networks.

6. *Receive IM:* User Y receives an IM from $A@AOL$. The IM looks exactly like a standard IM except for the modified user name.

3.3 CrossTalk Evaluation

To evaluate our system, we compare the overhead imposed by our bypass gateways against the messaging delays we observe on the various chat network servers. First, we estimate the amount of time messages spend inside the chat servers for clients *not* utilizing our bypass gateways. Over the course of 24 hours, we logged a sending client S and a receiving client R into each of the AOL, Jabber (specifically, jabber.org), MSN, and Yahoo IM networks. S pinged the chat server to estimate the network RTT, then sent an IM message containing only an embedded timestamp to R. R subsequently parsed the timestamp and computed the total time spent receiving the message, which we call the message RTT. This entire process was repeated 100 times for each hour of the experiment, and Table 1 shows our results. The Estimated Server Processing Time (SPT) row represents the difference between the average network RTT to the IM server and the average message RTT computed by R. Thus, the SPT measure is independent of network delay.

We then tested the CrossTalk implementation on a 3.2 Ghz Pentium 4 with 1 GB of memory. We used the observations in [18], which describes the IM traffic characteristics for an enterprise of over *4000 individuals*, to generate a load for our gateway. The load consisted of: 520 clients spread across 13 VMs, each sending one 150 character-long message per second to 20 other users. Recall that each bypass gateway will serve a particular organization; therefore, subjecting our prototype to a load representing 4000 people is

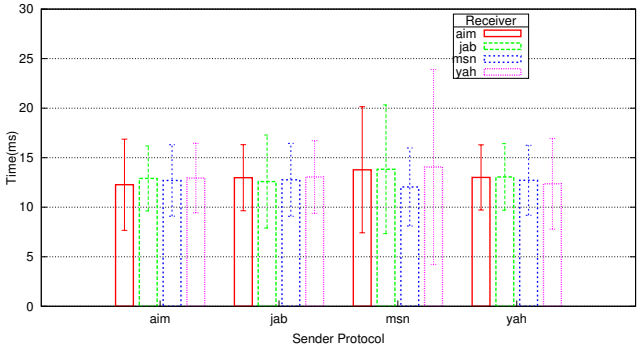

Figure 3: The average delay caused by our gateway for pairs of clients sending various types of IM traffic through our gateways.

reasonable. Note that one can achieve further scalability by adding more machines. Figure 3 shows the average delay introduced by our gateway between protocols. All protocol traffic took less than 15 ms to process, which is comparable to the amount of time messages spend inside the actual chat servers (as represented by the SPTs in Table 1). Note that Table 1 shows that an IM sent between two users in the MSN network (not behind bypass gateways) experiences an average delay of 229.1 ms, which is much higher than the overhead imposed by our prototype.

3.4 Third Party Applications Using CrossTalk

We return to one of our motivations for application interconnectivity: increasing the reach of third party applications. We created two example applications that demonstrate the benefits of interconnecting IM protocols. The first application provides IP geolocation, and the second exports Last.fm information. The applications were built on top of Yahoo Messenger 8 and AIM Lite.

When a user A wants to know user Y's location or Last.fm listening habits, A simply types "/location" or "/music" into a conversation window with Y. A sends an IM containing the command to Y through the gateway. If A and Y are using different IM protocols (e.g., AOL and Yahoo), the gateway will translate the IM sent by A. The plugin at Y will observe the command embedded in the IM message, then make an HTTP request to the online APIs for hostip.com or Last.fm. Lastly, the plugin will auto respond to A's request, sending back the information through an IM. The IM may be translated by the gateway before reaching A.

Without our gateway, AIM clients could only get location and music information from other AOL clients, and similarly for Yahoo clients. Thus, CrossTalk greatly extends the reach of the applications by allowing IMs to reach users in both networks, which may motivate developers to write new IM applications. Note also that we wrote separate plugins for each client. During the development of the plugins, we observed that AIM and Yahoo have similar API calls. Ideally, a developer would only have to write a single plugin for all IM networks, similar to the platform offered by the popular OpenSocial [12]. We describe our vision for a general architecture in Section 4.2 that allows this feature.

3.5 CrossTalk Deployment

We now sketch a possible deployment path for CrossTalk, discussing incentives and impediments for each stakeholder involved: large IM providers, small IM providers, enterprises, and end users.

Large Providers: Large providers may alter their protocols to break our translation efforts. Constant changes to protocols, however, affect users by forcing them to upgrade their clients, and disgruntled users can respond by switching IM clients. Second, the provider cannot distinguish bypass gateways from ordinary proxies; they cannot impair the former without hurting the latter. Lastly, unlike other solutions such as Pidgin, our approach *continues to play advertisements from the base networks*. We play advertisements from AOL even when an AOL user is conversing with a Yahoo user. Thus, our gateways do not cannibalize the revenue streams of large providers.

Small Providers: A "long tail" [2] of smaller IM providers exists across the world. Our software contains translation modules for 3 major IM protocols whose vendors would most likely not contribute to CrossTalk. However, smaller providers have a strong incentive to develop translators, since doing so will increase the reach of their IM services.

Enterprises: The use of a DHT for storing IM state allows organizations (e.g., universities, companies) to buy and maintain machines that serve as both gateways and DHT nodes. Our performance evaluation shows that a few inexpensive PCs suffice for IM. If performance becomes an issue (for more heavyweight traffic such as VoIP), a market could emerge for implementing bypass gateways in hardware.

End Users: If we can build critical mass via an initial community of users, then a "network effect" will set in. Again, users have many incentives for using our bypass gateways, which we addressed in Section 1.

4. FUTURE WORK

We now propose improvements to our system, and briefly discuss how our bypass gateways can lead to a general architecture.

4.1 Dealing with Encryption

The solution depicted in Figure 2 is infeasible for encrypted protocols. Skype, for example, encrypts all client-to-server communication, making the in-network interception and translation of protocol messages impossible. In this case, the gateway's only option is to intercept messages *before* they enter the network. In other words, we need to replace the idea of an in-network gateway *box* with a gateway *shim layer* that runs on the host machine. We envision inserting a shim layer between the user input and the actual base client, either through an SDK or by intercepting user interface commands (key presses, mouse clicks). Traffic sent to and from the base network is passed through unchanged. However, traffic sent from the client (e.g., Skype) to a user in a different network (e.g., SIP) is sent in a canonical format (e.g., SIP) through our gateway-to-gateway network to the recipient.

4.2 General Architecture

We now describe our vision for a general architecture (shown in Figure 4). At the top of Figure 4 are the clients. Individuals who use unmodified clients can utilize our bypass gateways or shim layers to achieve interconnectivity. However, we believe a better approach for the future is to migrate to a new abstract client layer. The new layer should ask for services using abstract calls (e.g., Send IM, Dial Call, Get File). As with IP, this allows clients to remain insulated

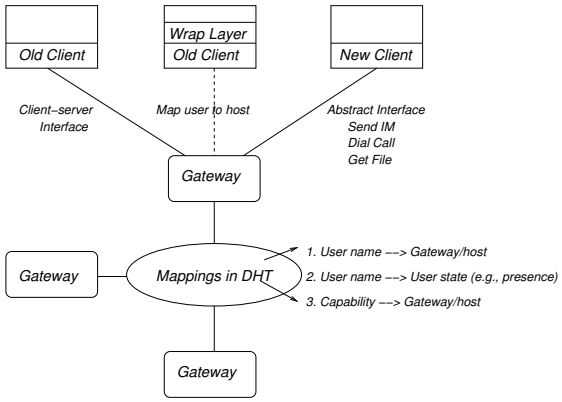

Figure 4: In the general architecture, new clients use a more abstract interface, and the DHT allows service composition by mapping capabilities to IP addresses.

from changes in technology, as say Skype is replaced by the next VoIP client du jour, and Facebook is supplanted by the next vogue in social networks. A modified client could also explicitly build in fault-tolerance to gateway failures.

The DHT stores mappings on behalf of all applications; recall that the IM solution needs to maintain mappings between usernames and the IP addresses of their bypass gateways. The DHT can also be leveraged to store any user state that needs to be shared beyond the responsible gateway (e.g., presence information in IM). A more interesting possibility that a DHT opens up is something akin to service composition (e.g., [8, 9]). Suppose we want flexibility in our choice of canonical protocols. Rather than require that all gateways translate to and from XMPP, we could allow a sequence of translations to occur. The DHT can be used store a mapping from capabilities (e.g., protocol translations, exits to foreign networks at a specified region) to IP addresses (gateways, or even hosts) that can perform a particular function. These capabilities can then be composed by finding a path through the resulting graph encoded in the DHT.

5. CONCLUSIONS

Interconnecting application level networks has three benefits. First, it mitigates the problem of vendor lock-in by allowing users to switch to new networks without losing access to their old networks. Second, even users who do not switch benefit by gaining access to a larger community of individuals. This is important for applications such as IM because the market is segmented into a number of popular providers. Third, developers of third-party applications benefit by gaining wider audiences for their add-ons.

Existing solutions to the problem of interconnecting application networks are fairly ad hoc, ranging from agreements between pairs of providers to gateway/bot solutions with scalability issues. Even the cleanest approach, client consolidation (e.g., Pidgin, Socialstream), presents several barriers to adoption for the average user. By contrast, our solution, CrossTalk, is general and scalable. Adopters can use whichever client software they prefer while avoiding the need to create multiple accounts in every network they wish to participate in. Because the bypass gateway network internally speaks a standard protocol, the solution encourages newer clients to use the standard protocol, thereby avoiding the need to write translators. Larger providers do not have incentives to break our service because we encourage the usage of their vanilla clients, which often generate revenue by showing ads. The move to standardization is facilitated when a large number of clients speak the standard solution.

While our paper has focused on IM, we believe that the ideas developed in this paper can be applied to more recent social networking applications. In particular, our analysis of the issues with client consolidation and standard gateways, the use of bypass gateways, and the need for a deployment path with the appropriate incentives, are all applicable to the general problem of interconnection. That said, we clearly recognize that the specific problem of interconnecting say MySpace and Facebook has a number of complex aspects that we leave for future work.

We believe that providing a standard substrate for IM, VoIP, and social networks to enable third-party innovation and user empowerment is a vision worth pursuing. It was in 1973 that ARPA initiated the Internet project; the results in terms of user empowerment and the variety of applications that run on top of TCP/IP are a matter of historical record. Nearly twenty five years later, in the face of application interconnection issues that each of us face every day, perhaps it is worth taking a trip back to the future.

6. REFERENCES

[1] gaim faq. `http://gaim.sourceforge.net/faq.php#q49`.

[2] C. Anderson. *The Long Tail: Why the Future of Business is Selling Less of More*. 2006.

[3] BTinternet. The arpanet, the internet and tcp/ip. `http://www.btinternet.com/~sandyloan/TMA04.htm`.

[4] comScore. `http://www.comscore.com/press/release.asp?press=800`.

[5] M. Fiuczynski, V. Lame, and B. Bershad. The design and implementation of an ipv6/ipv5 network address and protocol translator. In *Proceedings of USENIX Annual Technical Conference*, 1998.

[6] gtalk2voip. Gtalk-to-voip. `http://www.gtalk2voip.com`.

[7] Jabber. Jabber Gateways. `http://www.jabber.org/user/userguide/#usegateways`.

[8] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle. Ocala: An architecture for supporting legacy applications over overlays. In *Proceedings of USENIX/ACM NSDI*, 2006.

[9] K. Lakshminarayanan, I. Stoica, and K. Wehrle. Support for service composition in i3. In *Proceedings of Multimedia*, 2004.

[10] C. Lynch. `http://www.nytimes.com/external/idg/2008/10/27/27idg-Is-underdog-Lin.html`.

[11] A. M. McEvoy. Trillian Basic., 2005. `http://www.pcworld.com/article/id,123956/article.html`.

[12] OpenSocial. OpenSocial. `http://code.google.com/apis/opensocial/`.

[13] pidgin. `http://www.pidgin.im/`.

[14] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: a public dht service and its uses. In *Proceedings of ACM SIGCOMM*, 2005.

[15] P. Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. Technical report, IETF, 2004.

[16] S. Schroeder. `http://mashable.com/2007/07/17/social-network-aggregators/`.

[17] Vipadia. Karaka. `http://www.vipadia.com/products/karaka`.

[18] Z. Xiao, L. Guo, , and J. Tracey. Understanding instant messaging traffic characteristics. In *Proceedings of International Conference on Distributed Computing Systems*, 2007.