A Study of Internet Instant Messaging and Chat Protocols

Raymond B. Jennings III, Erich M. Nahum, David P. Olshefski, Debanjan Saha, Zon-Yin Shae, and Chris Waters, IBM T.J. Watson Research Center

Abstract

Instant messaging (IM) and network chat communication have seen an enormous rise in popularity over the last several years. However, since many of these systems are proprietary, little has been described about the network technology behind them. This analysis helps bridge this gap by providing an overview of the available features, functions, system architectures, and protocol specifications of the three most popular network IM protocols: AOL Instant Messenger, Yahoo! Messenger, and Microsoft Messenger. We describe common features across these systems and highlight distinctions between them. Where possible, we discuss the advantages and disadvantages of different technical approaches used in these systems to support different features and functions. We also briefly discuss ongoing efforts to standardize IM and chat-based protocols in IETF and other standards bodies.

Instant messaging (IM) and Internet chat communication have seen enormous growth over the last several years. IM is the private network communication between two users, whereas a chat session is the network communication between two or more users. Chat sessions can either be private, where each user is invited to join the session, or public, where anyone can join the session. There are on the order of 100 million Internet IM users, where a user is defined as a unique name on one of the major public IM networks — AOL Instant Messenger (AIM), Microsoft Messenger (MSN), or Yahoo! Messenger (YMSG). To date, little has been documented about the network protocols used by these systems. The protocols are not standardized, many of them are proprietary, and they are even seen as a control point in this business by the companies involved. This is demonstrated by the repeated attempts of the IM services to lock out users of other systems, in an attempt to keep their customers private. However, enough information is available to determine the broad characteristics of these systems. We have also used packet tracing of IM traffic in order to glean further details into these protocols and systems.

In this article we present an overview of IM protocols as exemplified by the three popular systems: AIM, MSN, and YMSG. While each has been designed and implemented separately, the overall group exhibits similar characteristics with respect to network and system architecture. For example, all of the IM protocols allow authenticating with a central server, engaging in private messages, and conversing in public chat rooms. In addition, some IM systems allow file transfers, Web cam usage, using privacy controls, maintaining buddy lists, voice chat sessions, and other options. We discuss these topics in more detail in the sections to follow. We analyze the most recent IM clients available. However, all of the major IM protocols have undergone significant revisions over the years, and changes to the protocols occur on a regular basis.

As with all networked applications, IM and chat protocols have a large potential design space. This survey helps expose some of the dimensions available to a protocol designer and how existing IM systems chose to decide them. Where possible, we describe advantages and disadvantages of each design choice, especially when the choice affects security.

Features and Functions

Most IM systems, including the three that we analyze herein, use a client-server architecture. IM providers typically host a set of servers that customers log in to and exchange messages with. A fundamental issue faced by IM service providers, and thus designers of the protocols, is how the systems will scale with large numbers of users. Ideally, each provider desires to have millions of customers logged on to their systems at each time. This in turn requires that organizations have a system architecture that can scale with the number of users. Two approaches are available here: *symmetric* and *asymmetric*. In a symmetric architecture, each server performs identical functions, such that a client need not distinguish which server it contacts to engage in an activity with. In an asymmetric approach, each server is dedicated to a particular activity such as logging in, discovering other users on the network, maintaining a chat room, or forwarding an instant message.

The client-server architecture allows IM service providers to keep some degree of control over their users. On the positive side, it helps overcome some of the technical issues associated with traversing the firewalls that the clients are often behind. On the negative side, since both control and data paths go through the central servers, scaling the service to millions of users is difficult. The scalability issue is particularly difficult for voice chat sessions. As IM services are beginning to support voice-chat communications, peer-to-peer data paths are being used.

AIM uses a client-server architecture for normal operations but uses a peer-to-peer approach for voice-chat sessions where the initiator talks directly to the recipient after coordinating through the system. Two clients thus communicate

0890-8044/06/$20.00 © 2006 IEEE

directly, without using a chat room, using a proprietary voice protocol. YMSG also uses a client-server architecture for normal operations as well as voice-chat service. YMSG voice traffic is routed through a centralized voice-chat server. Clients first contact a setup server "vc.yahoo.com" which then redirects the client to the voice-chat hosting server. One benefit of the YMSG centralized voice server approach is that it can support multiple users within the same voice-chat session and each user can specify their own voice specification with the central voice server based on their network speed. MSN uses a client-server architecture for normal operations and peer-to-peer for voice-chat communication. MSN voice-chat sessions are also limited between two users.

All three services provide a range of administrative and management functions. Most IM systems have mechanisms for maintaining lists of friends (and even enemies). These are typically called "buddy lists," "allow lists," and "block lists." These lists are maintained as persistent state on the server, which the clients synchronize with when they log in. The lists are used for several purposes. Buddy lists identify people that a user wishes to monitor the presence of (for example, to be notified when they log in). Block lists identify people that a user wishes to be isolated from, so that the user is not bothered or harassed by those people. Block lists are a form of blacklisting; some systems have the complementary feature of a whitelist called allow lists, which specify that only people on the list may communicate with the user. AIM, YMSG, and MSN all have buddy lists and block lists. AIM and MSN also have allow lists. MSN even has "reverse forward lists," which informs you of those users that have you on their forward (allow) lists. AIM has an additional feature that specifies a granularity of blocking, called a warning. Warnings are sent in response to received messages that the client finds unpleasant or inappropriate. Recipients of warning messages are penalized by having their sending rate lowered. Warning levels degrade slowly over time.

A usability feature that some IM systems provide is meta-messages that indicate that the other user in an IM session is typing. This improves interactivity, allowing the user to realize that the other party is in the process of composing a message and potentially hold off on their own typing. The "typing" messages are consequently a message type in the IM protocol. AIM, YMSG, and MSN have such message types. AIM even has three granularities: typing, not typing, and typed but erased. One option YMSG provides that the others do not is the ability to send IM's to users that are not currently logged on to the system. The system saves the messages on persistent storage and then delivers them to the recipient when that person logs on to the service.

An interesting feature offered by AIM is the ability to engage in secure communications by encrypting the IM session. Clients can obtain public keys from AOL, as well as the corresponding certificates to verify them. Secure instant messages are done using SSL and the two peer public keys. Secure chat rooms are created using a shared 256-bit AES secret key chosen by the chat room creator; invitations to the chat room include the secret key. YMSG and MSN do not have any similar capability. Peer-to-peer text communication is also offered by some systems using direct TCP connections between clients, sometimes called "side chats." AIM and YMSG have this feature, but MSN does not.

System Architecture

All three commercial systems use server clusters for scalability. AIM and MSN take the asymmetric approach. AIM defines several types of servers: login, BOS (basic OSCAR services), icon, user search, chat room setup, and chat room hosting. MSN defines three types: dispatch, notification, and switchboard. We describe how these servers are used in more detail below.

In contrast, YMSG takes the symmetric approach. Clients need only contact one type of server and then route all kinds of activities though that particular server. For example, YMSG connects to a random server in the cs##.msg.dcn.yahoo.com domain, where ## is a two-digit decimal number. All subsequent communication is routed through that server.

Session Distribution

We now examine in detail how the different systems distribute sessions across the servers in response to different actions.

The AIM system architecture is depicted in Fig. 1. In AIM, after the client logs in with the main authentication server (step 1 in Fig. 1), the client is directed to a BOS server. The client opens a single TCP connection to the BOS server (step 2), which is effectively the control channel. Most subsequent communication occurs over this connection, such as basic instant messages. Persistent connections are also made to the email server (step 3) and the user interest server (step 4). New services (checking email status, looking up a user, etc.) require sending a service request to the BOS server, which replies with a new IP address and TCP port number to contact for that particular service. A new connection is then made to that server. The exception is when a user wishes to join or create a chat room session. In this case, the client first contacts the BOS server to get access to the chat room setup server (step 5), which grants permission to a chat room. The credentials from the chat room setup server are then presented to the BOS server (step 6), which then points the client to a particular chat room server (step 7). Each chat room session is maintained using a separate TCP connection. The connection to the chat room setup server persists until several minutes after all chat room sessions are ended. The BOS server can force a client to switch to another BOS server through a migration message.

In 1998, AOL purchased Mirabilis Ltd., the creator of the ICQ instant-messaging software, and converted the AIM network to use a version of the ICQ OSCAR protocol. OSCAR, which stands for Open System for Communication in Real-time, is somewhat misleading, since AOL has never published the specifications of the protocol. There are some differences between features supported by ICQ and AIM but overall the underlying protocol is the same.

The MSN system architecture is shown in Fig. 2. MSN also has an asymmetric architecture, but with only three types of servers: dispatch, notification, and switchboard. A client initially contacts the well-known dispatch server (step 1 in Fig. 2) if it does not know of any notification servers. The dispatch server then redirects the client to a notification server. The client then opens a single connection to the notification server (step 2) and maintains this connection as long as the client is logged into the system. This is the control channel in the MSN architecture. The notification server maintains the presence of users in the system, and points the client to individual switchboard servers when a new instant message or chat session is created (step 4); step 3 will be discussed in the next subsection. The switchboard server is used both for chat sessions and instant messages to other clients; this differs from the other services in that MSN treats instant messages and private chat rooms identically. Instant messages are actually chat rooms set up between two users where additional users can be invited to the chat room. The TCP connection to the switchboard is open for the lifetime of the chat or IM commu-

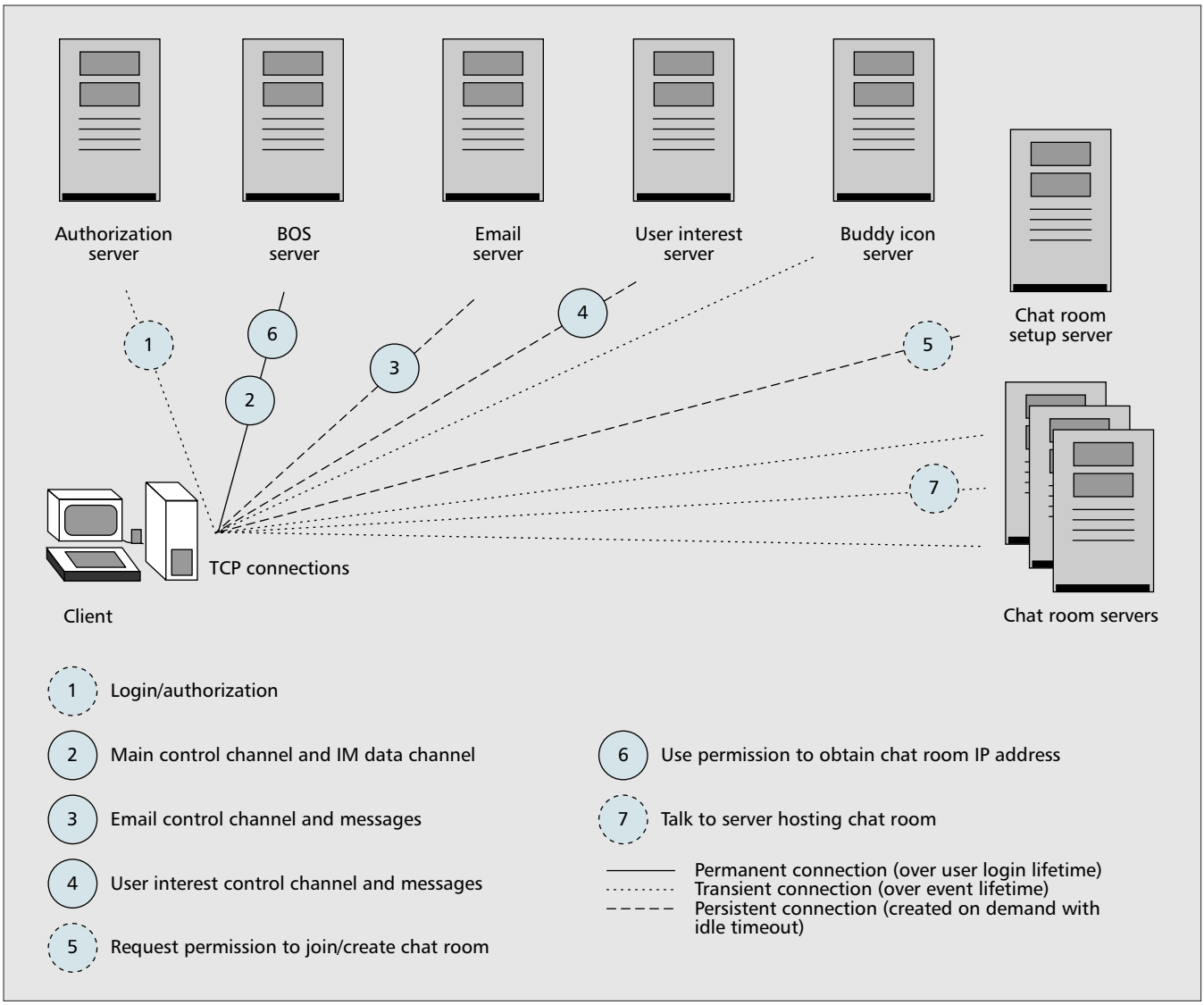


Figure 1. *AIM system architecture.*

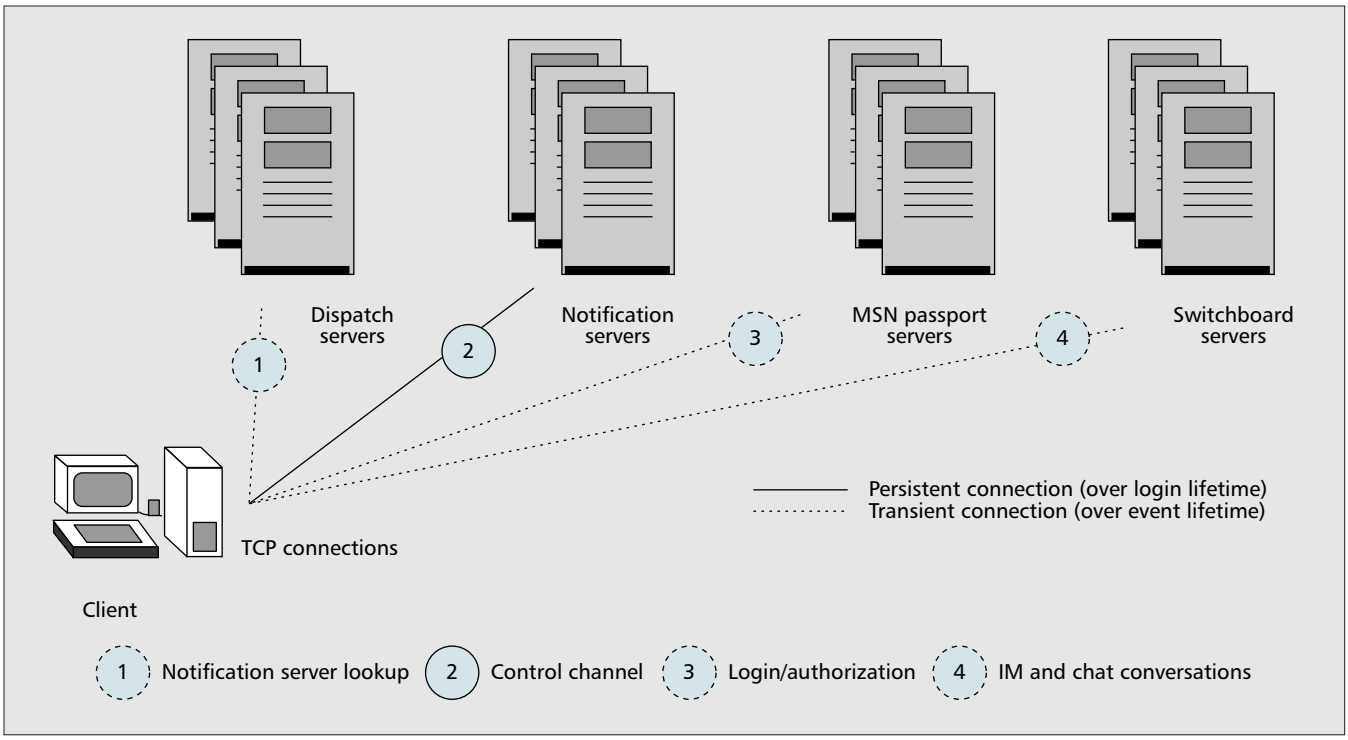
nication to the other client. The switchboard server also handles invitations for file transfers, video, and voice. While MSN does not have an explicit migration mechanism, the notification server can close the client connection, forcing the client to start over.

YMSG, on the other hand, is very simple due to its symmetric architecture, and is shown in Fig. 3. The same connection is used for all instant messages and chat sessions.

Many corporate environments employ firewalls to screen unwanted traffic, with a common default to allow HTTP traffic. Because of this, many IM systems allow tunneling over HTTP as a way around these firewalls. Interestingly, the three commercial IM systems all use the same symmetric architecture when tunneled over HTTP; namely, the client only interacts with a single HTTP front-end server. The native IM protocol is effectively encapsulated on top of HTTP, with commands and responses being multiplexed over HTTP connections. AIM uses two HTTP connections; one for submitting requests asynchronously, and the other that blocks waiting for the responses. YMSG uses a single synchronous connection, such that each request blocks until a response is received from the network. MSN also uses a single connection, but submits requests asynchronously and either receives a response or polls for a response depending upon the type of request.

User Authentication

The first thing users do when they log on to an IM network is authenticate themselves to the system. Again, several approaches are possible here, with clear implications for security. Some IM systems do not go through the full authentication process that is done in other contexts (e.g., SSL/TLS [1]), since both the user and the system share a secret key known only to the two of them: the user's name and password. While the initial system sign-up is typically done using HTTP secured by SSL/TLS, once the name and password are decided, login authentication is typically done by exchanging hashes of the shared secret. In this way, the password is never transmitted in the clear over the network, although the user name is. Both AIM and YMSG work this way. The advantage to this approach is that expensive crypto operations are avoided, such as RSA public key or AES shared key encryption. Instead, relatively cheaper authentication algorithms based on MD5 and/or SHA are used. The disadvantage is that confidentiality is not provided; observers can monitor the packet exchanges and determine who has logged in, even if they cannot determine the password. Since the hash algorithms are well known, and the challenge and hash result are sent in the clear, the systems are vulnerable to dictionary attacks. Users must therefore use passwords that are difficult



■ Figure 2. *MSN system architecture.*

to crack. In addition, performing the exchange in the clear could lead to connection hijacking; for example, AIM uses the cookie returned by the logon server as a credential sent in the clear to the BOS server. This credential must be used within 30 seconds or the connection will be terminated by the BOS server. This suggests that there is a window of opportunity where an adversary could monitor the conversation, capture the cookie, and use it to impersonate the victim to the BOS server.

MSN uses the Microsoft Passport system. After a client identifies itself to the MSN notification server, it is redirected to the Passport login server (step 3 in Fig. 2), where authentication is performed over SSL. The login server then supplies the client with several encrypted cookies that serve as credentials to the MSN notification servers. While the internal crypto algorithms are not publicly documented, the encrypted cookies are sent in the clear. Thus an attacker could attempt to use the cookies for impersonation and man-in-the-middle attacks [2, 3].

Data Transfer

One of the key issues in any IM or chat protocol is how protocol headers and payloads are encoded. The representation of this data can take two forms. Historically, many network protocols have used a binary representation of data in network byte order; examples include TCP and IP. Application-layer protocols such as HTTP and SMTP have tended to use a text-based approach. The main advantage to the binary representation is that it makes most efficient use of space on the network. The advantage of the text-based approaches is that the representation is closer to the way humans view information, and thus debugging is easier.

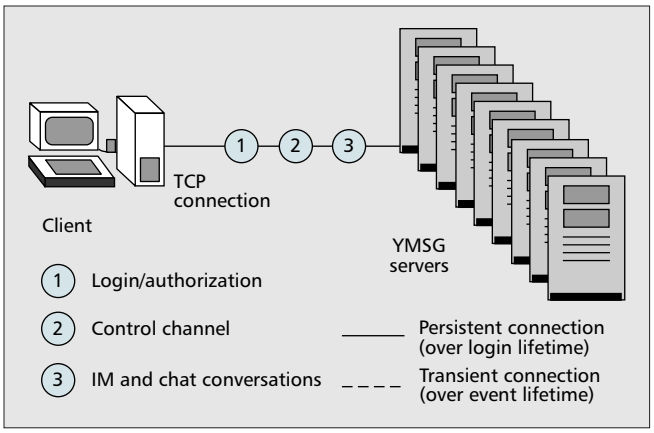
AIM and YMSG both use binary representation for their headers. AIM uses a two-level binary structure, called FLAP and SNAC packets, illustrated in Fig. 4. FLAP packets have fixed-length headers and variable-length data; SNAC packets are a subtype of FLAP packets that include several additional fixed-length fields followed by a variable data component.

YMSG, in contrast, has a single-level structure of fixed-length fields followed by variable-length data, as shown in Fig. 5. The data field is a sequence of key-value pairs, where keys are represented as a variable-length ASCII number.

AIM and YMSG have different methods of encoding header information. AIM favors a variable-length encoding that is more efficient in how much space on the wire it takes; YMSG has a more regular structure that is simpler to parse and decode.

Unlike AIM and YMSG, MSN headers are text based, as shown in Fig. 6. MSN headers take the form of <command, transactionID, parameterList, \r\n>, where command is a three-letter encoding, transactionID is an integer number, and parameterList depends on the command. Figure 7 shows an example of some MSN messages during the login phase, where different protocol and operating versions are specified and the client is transferred to a notification server. VER indicates what native protocol versions are supported by the client. CVR indicates Locale ID, OS type, OS version, platform architecture, client type, client version, and fixed string of "MSMSGS" followed by the passport ID. XFR is sent by the server indicating the IP address and port of the new notification server NS followed by a 0 and the old IP address and port.

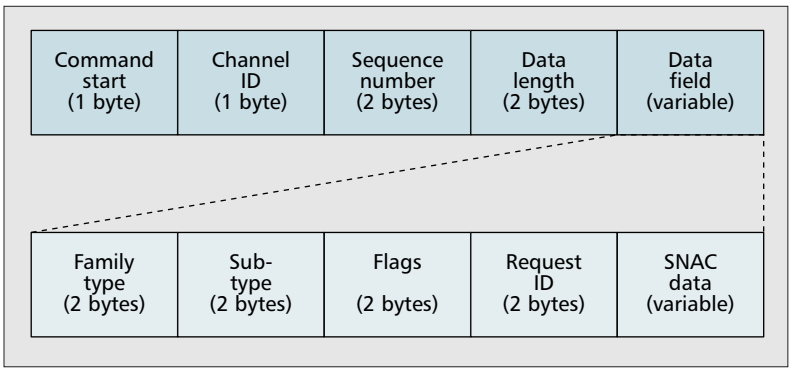
One potential problem for IM service providers are users that send data at excessive rates, flooding the network with useless traffic and inconveniencing other users. While TCP provides some protection against this through congestion control, some IM providers have apparently decided that this is not sufficient. Thus, several systems provide some kind of rate control to prevent SPAM or denial of service within their networks. AIM has a relatively complex algorithm that has different rate limits based on the message type. Rates are based on a time window (in seconds). If the client exceeds the rate, the user will be warned, and if the bad behavior persists, the server will start dropping messages and will even eventually disconnect the client. YMSG has a static limit of three IMs per second, which is enforced by the client. This implies that rate limiting could be circumvented by third-party clients (such as



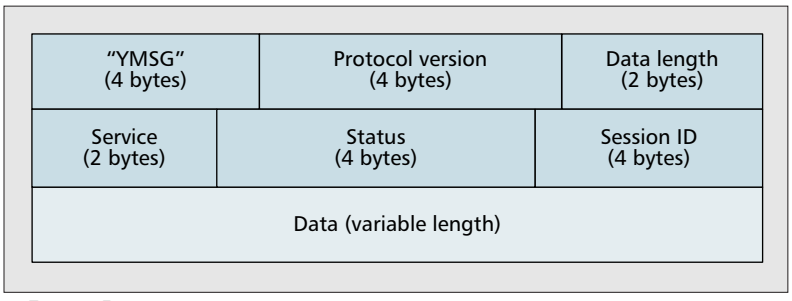
■ Figure 3. *YMSG system architecture.*

gaim or xchat) that do not enforce the limit. MSN, on the other hand, does not have any rate-limiting control.

Another way that IM systems minimize the load on their networks is by getting rid of idle clients. Idle clients cause load on the systems by consuming memory (such as connection state) and even CPU cycles (through timer management). Thus, each system maintains a keep-alive heartbeat message; if the client does not provide a heartbeat or response to a query, the connection may be terminated. In the case of AIM, the client must send a keep-alive every minute to the server. YSMG has two types of heartbeat requests, a primary and a secondary, that the server generates and the client must respond to. It is not immediately clear why two types of session timeouts are used. Typical values are 60 minutes for the primary and 13 minutes for the secondary. MSN has both client and server heartbeats. When the client pings, the server responds with how long the client should wait until the next ping. When the server pings, it is a challenge to the clientå, which must then respond with an MD5 hash of the challenge and the client ID.



■ Figure 4. *AIM FLAP and SNAC packet formats.*



■ Figure 5. *YMSG packet format.*

Future Directions

Recently, the IETF has embarked on an effort to standardize IM and chat protocols. Two competing standards are being developed: one based on SIMPLE [4] and a second one based on XMPP [5].

SIMPLE is an extension to the Session Initiation Protocol (SIP) [6] thst adds instant messaging and presence. SIP is a text-based control-plane protocol for establishing multimedia sessions such as Voice over IP. SIP can be transmitted over UDP, TCP, or SSL/TLS. The SIP/SIMPLE working group defines two models for messaging and chat sessions: the pager model and the session model. The pager model is appropriate when a user wishes to send a small number of short messages. The session model is intended for extended conversations, such as chat groups.

The SIP/SIMPLE pager metaphor is similar to that of a two-way pager or SMS enabled handset; there is no notion of a session with an explicit start and end, nor any explicit association between messages. IM payloads are carried inside the SIP packet via a new MESSAGE method. SIMPLE thus uses the SIP routing infrastructure to deliver messages to endpoints. Since SIP is designed and used primarily for transporting control messages, there is potential for traffic congestion within the SIP infrastructure when SIP messages contain IM payloads. To help address this issue, the IM payload is limited at 1300 bytes in SIMPLE which allows it to be carried in a 1500 byte Ethernet packet.

In the SIMPLE session model, there is an explicit conversation with a clear beginning and end. The IM payload is not carried in the SIP message, but in the media session established by SIP, and transported using the Message Session Relay Protocol (MSRP) [7]. The session model is much more complex than the pager mode, but has a number of benefits [8]. In MSRP, since the IM message payloads are not transported using the SIP signaling infrastructure, the messages sent can be very large. MSRP can establish both IM and voice sessions simultaneously; it also allows endpoints to create local IM servers and invite peers to enter a chat/voice session at that server. These features make it much easier to integrate IM with voice applications and are consequently very desirable among wireless carriers and network providers (Table 1).

XMPP, the Extensible Messaging and Presence Protocol, is an alternative to SIMPLE. The basic syntax and semantics of XMPP were developed originally within the Jabber open-source community [9]. While XMPP provides a generalized, extensible framework for exchanging XML data, it is intended mainly for the purpose of building IM and presence applications. IETF chartered the XMPP working group in 2002 with the objective of adapting the Jabber protocol to be suitable as an IETF instant-messaging and presence technology. XMPP is thus more fully developed and deployed, with current estimates of more than 200,000 registered users in the Jabber system [9]. While XMPP is not bound to a specific architecture, it is currently being deployed in a client-server manner, similar to IRC [10] or network mail (SMTP). XMPP supports both instant messages and chat rooms, and relies on TCP for congestion control. XMPP allows, but does not require, the use of SSL/TLS as a method for securing the stream from tampering and eavesdropping.

XMPP has enjoyed support from the Jabber open-source community. SIMPLE, on the other hand, has gained industry support (e.g., Microsoft Messenger, IBM Sametime). Notably, SIMPLE and XMPP can benefit from each other and seem to be moving in that direction. The Jabber community has indicated the desire to use SIP to set up messaging sessions. The SIMPLE working group is making use of XML for message transport (e.g., XCAP to allow users modify IM and presence information accessibility policy).

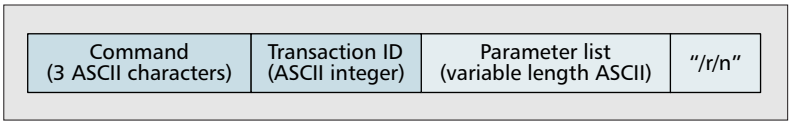


Figure 6. *MSN packet format.*

```
VER 29 MSNP10 MSNP9 CVR0\r\n

CVR 30 0x0409 winnt 5.0 i386 MSNMSGR 6.2.0137 MSMSGS
erichnahum@hotmail.com\r\n

XFR 31 NS 207.46.106.126:1863 0 207.46.104.20:1863\r\n
```

Figure 7. *MSN message examples.*

Summary

Little is known about the technical aspects of commercial Internet IM and chat protocols, due to the closed proprietary nature of these systems. We have presented a taxonomy of different feature and functions supported by most common systems, namely, AOL Instant Messenger (AIM), Yahoo Messenger (YMSG), and MSN Messenger (MSN). We have also examined the system architectures and protocols that power these systems. Out of all the systems, AIM supports the most features and thus is the most complex network IM protocol. This may be a result of the fact that AIM has the largest user base of the three systems. We also briefly discussed possible future approaches to IM and chat communication using IETF standardized protocols such as SIMPLE and XMPP. It seems clear that IM and Internet chat are here to stay, and will continue to evolve over time.

	AIM	YMSG	MSN
Binary-based protocol	Y	Y	N
ASCII-based protocol	N	N	Y
Supports P2P connections	Y	Y	N
Rate-limiting support	Y	Y	N
User-created public chat rooms	N	Y	Y

Table 1. *IM protocol comparison.*

References

[1] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," IETF RFC 2246, Jan. 1999.
[2] D. Kormann and A. Rubin, "Risks of the Passport Single Signon Protocol," *Comp. Networks*, vol. 33, 2000, pp. 51–58.
[3] A. Pashalidis and C. Mitchell, "A Taxonomy of Single Sign-on Systems," *8th Australasian Conf. Info. Sec. and Privacy*, Wollongong, Australia, July 2003.
[4] B. Campbell *et al.*, "Session Initiation Protocol (SIP) Extension for Instant Messaging," IETF RFC 3428, Dec. 2002.
[5] P. Saint-Andre, Ed., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence," IETF RFC 3921, Oct. 2004.
[6] J. Rosenberg *et al.*, "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.
[7] B. Campbell, R. Mahy, and C. Jennings, "The Message Session Relay Protocol," draft-ietf-simple-message-sessions-11.txt, July 2005.
[8] R. Mahy, "Benefits and Motivation for Session Mode Instant Messaging," draft-mahy-simple-why-session-mode-01.txt, Feb. 2005.
[9] Jabber Software Foundation, http://www.jabber.org
[10] J. Oikarinen and D. Reed, "Internet Relay Chat Protocol," IETF RFC 1459, May 1993.

Additional Reading

[1] P. Vixie *et al.*, "Dynamic Updates in the Domain Name System," IETF RFC 2136, Apr. 1997.
[2] M. Day, S. Aggarwal, and J. Vincent, "Instant Messaging/Presence Protocol Requirements," IETF RFC 2779, Feb. 2000.
[3] C. Dewes, A. Wichmann, and A. Feldmann, "An Analysis of Internet Chat Systems," *ACM SIGCOMM Internet Measurement Conf.*, Miami Beach, FL, Oct. 2003.

Biographies

RAYMOND B. JENNINGS III (raymondj@us.ibm.com) received his B.S. in electrical engineering from Western New England College and his M.S. in computer engineering from Manhattan College. He is an advisory engineer at the IBM T. J. Watson Research Center, Yorktown Heights, New York. He works in the area of network system software and enterprise networking. He is also a Ph.D. candidate in the Department of Computer Science at Polytechnic University.

ERICH NAHUM (nahum@us.ibm.com) received his B.A. in computer science from the University of Wisconsin-Madison, and his M.S. and Ph.D. degrees in computer science from the University of Massachusetts at Amherst. He is a research staff member at the IBM T. J. Watson Research Center. He works in the areas of networked server performance, workload characterization and generation, TCP, HTTP, and security.

DAVID OLSHEFSKI (olshef@us.ibm.com) received his B.S. in computer science from the State University of New York at Albany and his M.S. in computer science from Rensselaer at Hartford. He is an advisory programmer at the IBM T. J. Watson Research Center. He works in the area of network system software and enterprise networking. He is also a Ph.D. candidate in the Department of Computer Science at Columbia University.

DEBANJAN SAHA (dsaha@us.ibm.com) received a B.Tech. degree from the Indian Institute of Technology, and M.S. and Ph.D. degrees from the University of Maryland at College Park, all in computer science. He is with the Security and Networking Department of the IBM T. J. Watson Research Center. He has authored numerous papers and is a co-recipient of IEEE Communications Society's 2004 Fred W. Ellersick prize paper award and 2003 William R. Bennett prize paper award. He is one of the first implementers of MPLS, a primary author of the GMPLS standard in IETF, and a co-author of a book on that subject.

ZON-YIN SHAE (zshae@us.ibm.com) received his B.A. and M.S. degrees in electronic engineering from National Chiao-Tung University, Taiwan, and his Ph.D. in electrical engineering from the University of Pennsylvania at Philadelphia. He is a senior engineer at the IBM T. J. Watson Research Center. He works in the areas of multimedia networking, SIP/VoIP converged networks, multimedia traffic, and data analysis.

CHRISTOPHER J. WATERS received a B.A. in mathematics from the University of Utah. He works as a mathematician for the Department of Defense. He performs research in several areas, including network security and analysis of communications metadata. He has taught network security classes for government employees. He is collaborating on research projects at IBM T. J. Watson Research Center.