RC23349 (W0409-154) September 24, 2004
Computer Science

IBM Research Report

A Taxonomy of Internet Instant Messaging and Chat Protocols

Raymond B. Jennings III, Erich M. Nahum, David P. Olshefski,
Debanjan Saha, Zon-Yin Shae, Chris Waters
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 703
Yorktown Heights, NY 10598

A Taxonomy of Internet Instant Messaging and Chat Protocols

Raymond B. Jennings III, Erich M. Nahum, David P. Olshefski,
Debanjan Saha, Zon-Yin Shae, Chris Waters

IBM TJ Watson Research Center
Hawthorne NY, 10532

ABSTRACT

Network chat and Instant Messaging 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 the gap by providing an overview of the system architectures, protocol specifications and available features of several network chat protocols. We present a survey of several popular systems: AOL Instant Messenger, Yahoo! Messenger, Microsoft Messenger, and Internet Relay Chat. We describe common features across these systems and highlight distinctions between them. Where possible, we discuss advantages and disadvantages of different approaches, particularly with respect to security.

INTRODUCTION

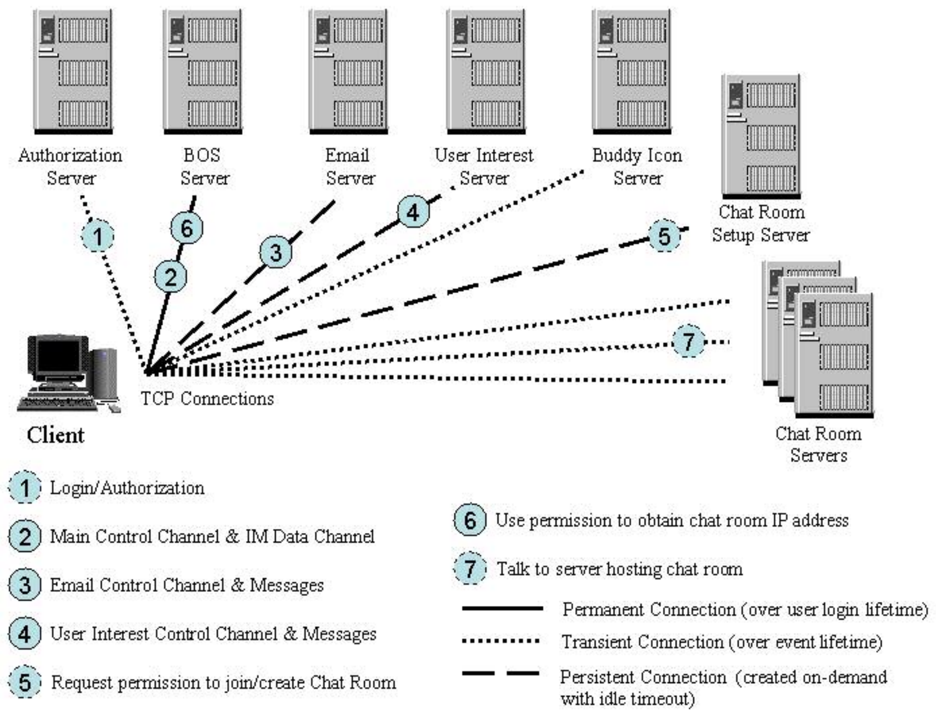
Internet Chat and Instant Messaging have seen enormous growth over the last several years. There are on the order of 100 million Internet chat users, where a user is defined as a unique name on one of the major chat networks – AOL Instant Messenger (AIM), Microsoft Messenger (MSN) or Yahoo! Messenger (YMSG) [1]. 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 chat services to lock out users of other systems, in an attempt to keep their customers private [2]. However, enough information is available to determine the broad characteristics of these systems.

We present an overview of chat protocols as exemplified by four popular systems: AIM, MSN, YMSG, and Internet Relay Chat (IRC). According to instantmessagingworld.com as of March 2004, AIM has estimated to have over 44 million users, and YMSN over 19 million [3]. MSN claims over 100 million users as of June 2003 [16]. IRC has over 1.3 million users across over 700 networks [15].

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 chat protocols allow for authenticating with a central server, engaging in private chats and conversing in public chat rooms. In addition, some chat systems allow for file transfers, Web cam usage, using privacy controls, maintaining buddy lists, voice chats and other options. We discuss these topics in more detail in the coming sections. We analyze the most recent chat clients available, specifically, AOL Instant Messenger v. 5.5.3595, MSN Messenger v. 6.2.0137, and Yahoo! Messenger v. 6,0,0,1710. However, all of the major chat protocols have undergone significant revisions over the years, and changes to the protocols occur on a regular basis.

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

SYSTEM ARCHITECTURE



Authorization Server
BOS Server
Email Server
User Interest Server
Buddy Icon Server
Chat Room Setup Server
Chat Room Servers
Client
TCP Connections
1 Login/Authorization
2 Main Control Channel & IM Data Channel
3 Email Control Channel & Messages
4 User Interest Control Channel & Messages
5 Request permission to join/create Chat Room
6 Use permission to obtain chat room IP address
7 Talk to server hosting chat room
Permanent Connection (over user login lifetime)
Transient Connection (over event lifetime)
Persistent Connection (created on-demand with idle timeout)

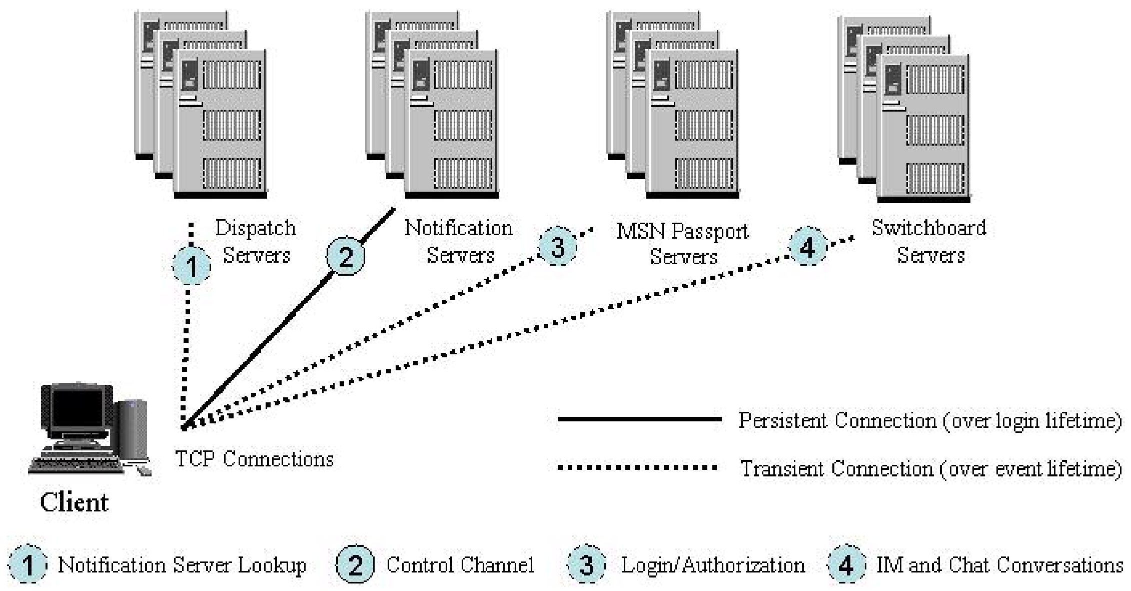
Figure 1: AIM System Architecture

We start by presenting an overview of the possible system architectures. Chat providers typically host a set of servers that customers log in to and exchange messages with. A fundamental issue faced by chat service providers, and thus designers of the protocols, is how the systems will scale with large numbers of users. Ideally, each organization 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. Of course, scaling can also occur by using multiple IP addresses for each server using dynamic DNS [5], and the three commercial systems all appear to do this.

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 and IRC take 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.

While IRC is also a symmetric architecture, it is unique from the other systems in that it is a system distributed across the wide-area. Servers are connected using spanning tree that describes the topology of the network. Thus, while a user could potentially use any server in the network, for the best response time and interactive experience, contacting a server as close as possible is preferred.



Dispatch Servers
Notification Servers
MSN Passport Servers
Switchboard Servers
1
2
3
4
TCP Connections
Client
Persistent Connection (over login lifetime)
Transient Connection (over event lifetime)
1 Notification Server Lookup
2 Control Channel
3 Login/Authorization
4 IM and Chat Conversations

Figure 2: MSN System Architecture

One of the well-known limitations of IRC is that it does not scale well with the number of servers. The way that this is handled is that there are, in fact, at least 4 different major IRC networks. While how these networks came about is a complex history, the net effect is that scale is reduced by having disjoint networks.

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 Figure 1. In AIM, after the client logs in with the main authentication server, the client is directed to a Basic Oscar Services (BOS) server. The client opens a single TCP connection to the BOS server, 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 and the user interest server. 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 chat room setup server to obtain permission and then presents that credential to the BOS server, which then points the client to a particular chat room server. 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.

The MSN system architecture is shown in Figure 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 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 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

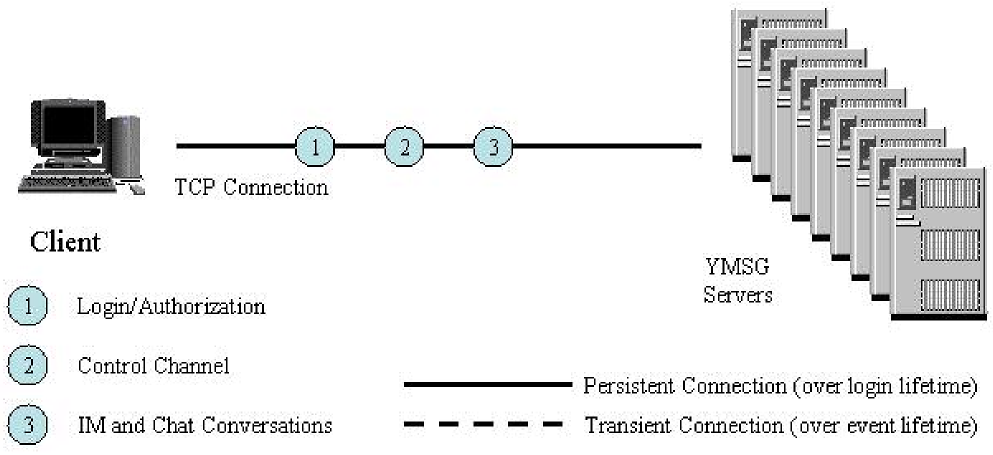

TCP Connection
Client
1 Login/Authorization
2 Control Channel
3 IM and Chat Conversations
YMSG Servers
Persistent Connection (over login lifetime)
Transient Connection (over event lifetime)

Figure 3: YMSG System Architecture

presence of users in the system, and points the client to individual switchboard servers when a new chat or IM is created. The switchboard server is used both for chats and IMs 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 setup between two users where additional users can be invited to the chat room (as long as voice is not being used). The TCP connection to the switchboard is open for the lifetime of the chat or IM to the other client. The switchboard server also handles invitations for file transfers and NetMeetings. 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 Figure 3. The same connection is used for all chat and instant messages. IRC also only uses a single connection, and is illustrated in Figure 4. IRC does not have a notion of a chat room. Instead, a similar mechanism called a channel is used, which is more like a multicast channel.

Many corporate environments employ firewalls to screen unwanted traffic, with a common default to allow HTTP traffic. Because of this, many chat systems allow tunneling over HTTP as a way around these firewalls. Interestingly, the three commercial chat 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 chat protocol is effectively encapsulated on top of HTTP, with commands and responses being multiplexed on top of the HTTP connection. AIM uses 2 HTTP connections to speak with the network; 1 for submitting requests asynchronously, and the other that blocks waiting for the responses. YMSG use 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.

IRC, on the other hand, does not specify any tunneling over HTTP; the RFC's only discuss straight IRC over TCP. However, this does not preclude tunneling IRC over HTTP using a tool such as httptunnel; of course, a second machine would be required to terminate the tunnel and gateway the IRC traffic directly onto an IRC network.

LOGON AND AUTHENTICATION

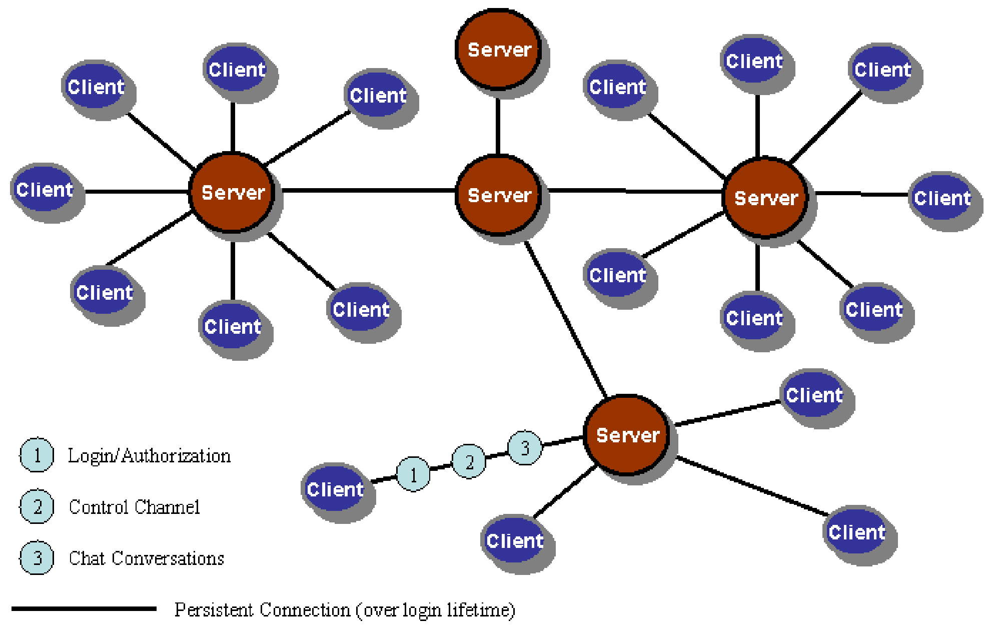


Figure 4: IRC Architecture

The first thing users do is log on to a chat network, in order to authenticate themselves to the system. Again, several approaches are possible here, with clear implications for security. Some chat systems do not go through the full authentication process that is done in other contexts (e.g., SSL), 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, once the name and password are decided, login authentication is typically done by exchanging hashes of the shared secret, combined with nonces and challenges provided by the peers. 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 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, 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, leading to several possible attacks, such as impersonation and man-in-the-middle [11, 12].

IRC is the least secure of these systems in that the protocol specifies that a user name and password are sent in the clear, much like the original telnet protocol. While not secure to an eavesdropper, the RFC [4] notes that the security has been considered sufficient in most cases for the use of the network. While extra mechanisms are used in practice, they are not codified in the standard.

CHAT DATA TRANSFER

One of the key issues in any chat or IM protocol is how protocol headers 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; a 16-bit value is smaller to express than the text-equivalent 16,384. 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 2-level binary structure, called FLAP and SNAC packets, illustrated in Figures 4 and 5 respectively. FLAP packets have fixed-length headers and variable-length data; SNAC packets are a sub-type of FLAP packets than include several additional fixed-length fields and then a variable data component.

Command Start (1 bytes)	Channel ID (1 byte)	Sequence Number (2 bytes)	Data Length (2 bytes)	Data Field (variable)

Figure 4: AIM FLAP Packet Format

Family Type (2 bytes)	Sub-Type (2 bytes)	Flags (2 bytes)	Request ID (2 bytes)	SNAC Data (variable)

Figure 5: AIM SNAC Packet Format

YMSG, in contrast, has a single-level structure of six fixed-length fields followed by variable-length data, as shown in Figure 6. 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 appears to favor a custom variable-length encoding that may be more space-efficient in how much space on the wire it takes; YMSG has a more regular structure that appears to be more simply decoded.

"YMSG" (4 bytes)	Protocol Version (4 bytes)	Data Length (2 bytes)
Service (2 bytes)	Status (4 bytes)	Session ID (4 bytes)
Data (variable length)		

Figure 6: YMSG 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

Unlike AIM and YMSG, MSN and IRC headers are text based. MSN headers take the form of <command, transactionID, parameterList, \r\n>, where command is a 3-letter encoding, transactionID is an integer number, and parameterlist depends on the command. Figure 7 shows an example of some MSN messages.

IRC has a similar approach, where messages take a straight <command, options, \r\n> format. The number of options is variable based on the command type.

One potential problem to chat 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 chat 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 even eventually disconnect the client. YMSG has a static limit of three IM's a second, which appear to be enforced by the client. This implies that rate limiting could be circumvented by third-party clients (such as gaim or xchat) that do not enforce the limit. Even IRC has a rate limiting mechanism specified by the protocol as 1 message every 2 seconds. MSN, on the other hand, does not appear to have any rate limiting control.

Another way that chat systems minimize the load on their networks is by getting rid of idle clients. 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 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 client which must then respond with an MD5 hash of the challenge and the client ID. IRC does not appear to have any application-level timeouts; it thus relies solely on TCP-level heartbeats.

OTHER FEATURES

A usability feature that some chat systems provide is meta-messages that indicate that the other user in a chat session is typing. This allows 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 chat protocol. AIM, YMSG and MSN have such message types. AIM even has three granularities: typing, not typing, and typed but erased. IRC does not have this feature.

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 stores the messages on persistent state and then delivers them to the recipient when that person logs on.

A popular feature provided by many chat systems is voice chat, allowing users to talk in a full-duplex, interactive fashion. Again, several approaches are possible. AIM uses a peer-to-peer (P2P) approach where the initiator talks directly to the recipient, after coordinating through the system. Two clients thus talk directly over UDP, without using a chat room, using a proprietary voice protocol which samples every 180 milliseconds. AIM allows only 2 participants in a voice chat. YMSG also offers a voice service, but all traffic is routed through a centralized voice chat server. Clients first contact a setup server "vc.yahoo.com" using a new connection on a different port number. The setup server redirects the client to the voice chat hosting server. Invitations, accepts, and rejects are sent over the standard yahoo message connection. Voice data is routed through the voice chat server, and is sampled every 60 milliseconds. YMSG voice appears to be a proprietary format, since it has a UDP/RTP header but with an unassigned type 22 voice codec. Voice communication can include conferencing, i.e., more that 2 participants. MSN offers both P2P voice chats and conferencing through a chat room. MSN uses SIP, with UDP/RTP for voice, with a voice codec type of G723. The IRC protocol does not offer voice chats, although a non-standard method could be offered by the chat client.

Peer-to-peer text communication is also offered by some systems using direct TCP connections between clients. These are sometimes called "side chats." AIM and YMSG have this feature, but MSN does not. IRC does not provide this feature, although users can approximate it through a PRIVMSG command to a single user, which is not assigned to a channel, but it is still routed through the server.

An interesting feature offered by AIM is the ability to engage in secure communications by encrypting the chat session. Clients can obtain public keys and corresponding certificates to verify them from AOL. 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, MSN and IRC do not appear to have any similar capability.

ADMINISTRATIVE FUNCTIONS

Most chat 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 to 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, these 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.

FUTURE DIRECTIONS

A recent activity within the IETF is working to make chat and instant messaging interoperable through open standards. Two competing standards are being developed: one based on SIP [6] and one based on XMPP [9].

SIP defines two mechanisms: the session model and the pager model. The pager model is appropriate when a user wishes to send a small number of short instant messages to a single (or small number of) recipients. The session model is intended for extended conversations such as chat groups. Both models are transported under SIP. Currently, there is no standard HTTP tunneling method specified.

The pager model is currently more defined, specified as SIMPLE (Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions) [6]. The pager metaphor is similar to that of a two-way pager or SMS enabled handset; there is no explicit association between messages. IM payloads are carried inside the SIP packet via a new MESSAGE method. Since SIP can be encapsulated over UDP, there is potential for traffic congestion within the SIP infrastructure when a SIP message contains IM payloads. This congestion could also interfere with SIP call signaling traffic. To help address this issue, the IM payload is limited at 1300 bytes in SIMPLE.

In the session model, there is an explicit conversation with a clear beginning and end. In the SIP environment, an IM session would be a media session initiated with an INVITE transaction and terminated with a BYE transaction. The IM payload is not carried in the SIP message, but in the media session established by SIP message. In the pager model, since the IM session is not sent within the same SIP infrastructure as the SIP message, there is no limitation on the size of the payload. The message format conforms to the CPIM messaging format.

SIMPLE messages may send bodies of type message/CPIM. Since the message/CPIM format is expected to be supported by other instant message protocols, endpoints using different IM protocols, but otherwise supporting message/CPIM body types, should be able to exchange messages without modification of the content by a gateway or other intermediary. This helps to enable end-to-end security and interoperability between endpoints that use different IM protocols.

In normal usage, most SIP requests are used to setup and modify communication sessions, not communicate data directly, which happens in the media sessions. The MESSAGE method in SIP changes this assumption, implying that MESSAGE requests have a greater need for security than most other SIP requests. The SIMPLE specification thus requires user agents implement end-to-end authentication, message integrity, and message confidentiality mechanisms, as specified by the S/MIME RFC.

To prevent the replay of old SIP requests, all signed MESSAGE requests and responses must contain a Date header field covered by the message signature. Any message with a date older than several minutes in the past, or which is more than several minutes in the future, will not be delivered to the user.

XMPP, the Extensible Messaging and Presence Protocol, is the rival to SIP as an open-standards based protocol for presence and instant messaging. While XMPP provides a generalized, extensible framework for exchanging XML data, it is intended mainly for the purpose of building instant messaging and presence applications that meet the requirements of RFC 2779 [7]. The basic syntax and semantics were developed originally within the Jabber open-source community [8] in 1999. IETF chartered the XMPP working group in 2002 with adapting the Jabber protocol to be suitable as an IETF instant messaging (IM) and presence technology. XMPP is thus more fully developed and deployed, with current estimates of over 200,000 registered users in the Jabber system [8].

While XMPP is not bound to a specific architecture, it is currently deployed in a client-server manner, with two TCP connections between client and server. The deployed architecture is thus similar to IRC 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 TLS as a method for securing the stream from tampering and eavesdropping.

	AIM	YMSG	MSN	IRC
Binary Based Protocol	Y	Y	N	N
ASCII Based Protocol	N	N	Y	Y
HTTP Tunneling support	Y	Y	Y	N
Supports P2P Connections	Y	Y	N	N
Rate Limiting support	Y	Y	N	Y
Instant Messages	Y	Y	Y	Y
Private Chat Rooms	Y	Y	Y	Y
Public Chat Rooms	Y	Y	Y	Y
User created public chat rooms	N	Y	Y	Y
Voice chat	Y	Y	Y	N
File Transfers	Y	Y	Y	N
Persistent Server Storage	Y	Y	Y	N

Table 1: Chat Protocol Comparison

SUMMARY

Little is known about the technical aspects of commercial internet chat and instant messaging protocols, due to the closed proprietary nature of these systems. We presented a taxonomy of the most common systems, namely AOL Instant Messenger (AIM), Yahoo Messenger (YMSG), and MSN Messenger (MSN), and compared them with the open standardized Internet Relay Chat (IRC). An overview of the protocols may be found in Table 1. Out of all the systems, AIM appears to support the most features and thus is the most complex network chat 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 chat and IM using IETF standardized protocols such as SIMPLE and XMPP. It seems clear that chat and IM protocols are here to stay, and will continue to evolve over time.

REFERENCES

[1] "Big Three Slug It Out for Consumer Control," May 2004, http://www.instantmessagingworld.com/public/article.php/3355251

[2] "Yahoo! Protocol Change Blocks Third Party Clients," June 2004, http://www.instantmessagingplanet.com/public/article.php/3373211

[3] "IMPlanet Roundup: The News," October 2003, http://www.instantmessagingworld.com/wireless/article.php/3096091

[6] B. Campbell, J. Rosenberg,, H. Schulzrinne, C. Huitema, D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. IETF RFC 3428, December 2002.

[7] M. Day, S. Aggarwal, J.Vincent. Instant Messaging / Presence Protocol Requirements. IETF RFC 2779, February 2000.

[8] Jabber Software Foundation. http://www.jabber.org

[9] IETF Extensible Messaging and Presence Protocol Working Group. http://www.ietf.org/html.charters/xmpp-charter.html

[11] D. Kormann and A. Rubin. Risks of the Passport Single Signon Protocol. Computer Networks, V. 33, pp. 51—58, 2000.

[12] A. Pashalidis and C. Mitchell. A Taxonomy of Single Sign-on Systems, 8th Australasian Conference on Information Security and Privacy, Wollongong, Australia, July 2003.

[13] A. Gelhausen. Summary of IRC networks. http://irc.netsplit.de/networks/

[14] Blake Irving. MSN Messenger 6 Scores Big With IMers. http://www.microsoft.com/presspass/features/2003/jun03/06-17MSNIM.asp

AUTHOR INFORMATION

The authors work at the IBM T.J. Watson Research Center, Yorktown Heights, New York. Raymond B. Jennings III is an advisory engineer and works in the area of network system software and enterprise networking. Erich Nahum is a research staff member and works in the areas of networked server performance, workload characterization and generation, TCP, HTTP, and security. David Olshefski is an advisory programmer and works in the area of network system software and enterprise networking. Debanjan Saha is a research staff member and has authored numerous papers on various topics of networking 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. Zon-Yin Shae is a senior engineer and works in the areas of multimedia networking, SIP/VoIP converged networks, multimedia traffic and data analysis. Christopher J. Waters performs research in several areas including network security and analysis of communications metadata.