INTERNET-DRAFT   Greg Hudson
Expires: March 22, 1999   ghudson@mit.edu
                                                                    MIT

Instant Messaging / Presence Protocol Design Issues

draft-hudson-impp-issues-00.txt

1. Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Please send comments to the IMPP working group at impp@iastate.edu.

2. Abstract

This document describes design issues in the creation of the instant messaging and presence protocols in the IMPP working group. The goal is to objectively present arguments for and against the various options on each issue.

3. Terminology

The following terms are defined in [Model] and are used with those definitions in this document:

INSTANT INBOX
INSTANT MESSAGE
PRESENCE INFORMATION
PRESENTITY
SERVER
WATCHER

The following terms are defined in [Reqts] and are used with those definitions in this document:

DOMAIN

IDENTIFIER

4. Form of IDENTIFIERS

An IDENTIFIER uniquely determines a PRESENTITY or INSTANT INBOX and is intended for humans. It may never appear in its human-readable form in the wire protocols or in the message format, but it is nevertheless important that the protocols define what an IDENTIFIER is and means; it would not do to have different clients using different text strings as IDENTIFIERS.

The most obvious conception of an IDENTIFIER is a string containing two parts: a DNS domain part, which identifies the messaging or presence provider, and the username part, which gives the local name assigned to a PRESENTITY or INSTANT INBOX by the provider. There are, of course, other ways to identify a person than through their provider's DNS domain, but since there exists no Internet directory service with the reach of the DNS, choosing some other method would present an undesirable barrier to entry for users.

All that remains is how to represent these two components in a string. There are two obvious precedents to follow: email addresses and World Wide Web URLs. The email address form would be "username@domain", whereas the obvious URL form would be either "impp://domain/username", or "im://domain/username" for messaging and "presence://domain/username" for presence information. Naturally, choosing the email address form would not preclude the existence of a secondary URL form; along the lines of the "mailto" URL, we would expect to see at least "imto:username@domain" and possibly "presence:username@domain".

Two obvious advantages of the email address format are conciseness and the possibility of having a single contact address for email, messaging, and presence information. The obvious advantage of the URL format is that it makes the protocol explicit, and it also yields the possibility of having similar though not identical addresses for one's home page and for messaging and presence information.

5. Server Selection

Once a DNS domain is known for a PRESENTITY or INSTANT INBOX, there must be some process for determining what servers to contact for messaging or presence.

On the very simple end of the spectrum, we can do a simple A record lookup. This is how HTTP works, which is why the domain part of most URLs looks like "www.zone" instead of just "zone" (where "zone" is the actual administrative demarcation). In addition to resulting in longer domains, this scheme makes it difficult for multiple servers to handle the messaging and presence load for a domain. On the other hand, a simple hostname lookup is much easier in most current programming environments than lookups of other kinds of DNS records.

A SRV record lookup, as defined in [SRV], solves both of the above problems, providing both a level of indirection and a mechanism for selecting from a set of servers by priority and weight. If a SRV record does not exist for the domain, falling back to an A record lookup would lower the barriers to adoption by people who don't control their machines' DNS information or whose DNS servers do not support SRV.

6. Basic Transport

Whatever protocols become standardized will be implemented on some transport layer, whether it is one of the core Internet protocols such as TCP or UDP, or a higher-level protocol such as HTTP, SMTP, or LDAP. The messaging and presence protocols may use different transports, and it may be possible to implement either protocol on multiple transports, but the design of the protocols will depend in large part on the preferred transport layers.

UDP bears the advantage of being lightweight for certain operations. Especially on Unix, where in-kernel state is required for TCP connections, a UDP-based protocol might increase the amount of load a messaging server can handle. However, there are a number of arguments against UDP:

* Over the wide-area Internet, it is important that communications between two end-points behave politely, which requires keeping connection state.

* A simplistic UDP protocol can actually increase the number of packets transmitted over the wire, by not allowing multiple operations to be sent in a single packet.

* A UDP protocol lends itself to a "routing" type of protocol where the server keeps very little state about the client. But modern point-to-point security mechanisms may require several packet exchanges at the beginning of a session and require at least some state to be kept about the client during the session.

* The UDP-based protocol could not be used for large messages without reimplementing the TCP logic for window sizes and message fragmentation.

* UDP-based protocols do not cooperate as well with firewalls as TCP-based protocols do (assuming the TCP protocol uses only client-initiated connections).

Assuming UDP is rejected, TCP becomes the natural substrate for a new protocol. The "politeness" argument against a UDP protocol also applies to a TCP protocol with many short-lived connections, so it is important that the protocol allow multiple operations to take place within a single connection.

But it is also possible to use a higher-level protocol for transport. The arguments for using a higher-level protocol are found in the features present in that protocol which would apply to instant messaging or presence information. The main argument against using a higher-level protocol is complexity; a new, tailored protocol using only TCP will generally be simpler than a layered protocol, when viewed as a whole.

Note that using a higher-level transport protocol does not imply using its port number.

One common proposal is to use [RFC 821] (SMTP) as the transport layer for instant messaging. As a messaging protocol, SMTP is at least superficially a good fit, meaning it wouldn't impose much complexity beyond what is actually necessary to support messaging. Here are arguments against using SMTP:

* SMTP's wide deployment cannot be considered as an argument in its favor. Existing implementations of SMTP center around the assumption that incoming messages should be written synchronously to disk and delivered to their destination at leisure, which is not appropriate for instant messaging.

* As a simple protocol, SMTP doesn't actually lend much machinery to the problem. Reimplementing the machinery provided by SMTP would not require a great deal of work or involve a large number of pitfalls.

* SMTP has an arbitrary limit on line length, and it cannot transmit messages which do not end in a newline. Thus, some messages must be encoded purely because of protocol limitations.

* As originally specified, SMTP operates in lock-step with many round trips per message, and can only be used for 7-bit ASCII messages. The use of the [RFC 2197] (PIPELINING) and [RFC 1652] (8BITMIME) service extensions remedy these limitations, but a new protocol would not force implementors to understand these bits of history.

Another proposal is to use [RFC 2251] (LDAP) as the transport layer for presence information. As a directory protocol, LDAP would seem to be a good fit for presence information. Here are arguments against using LDAP:

* As originally specified, LDAP only allows a client to fetch directory information, not subscribe to changes in it.

Since most of the interesting part of presence lookup is in subscribing, not fetching, that is a serious defect. A proposed "persistent search" extension to LDAP (in an expired draft) could be applied to presence subscriptions, but as a much more general tool, it might be difficult to implement efficiently and might complicate the retrieval of watcher information.

* As an ASN.1-based protocol, LDAP is not as simple as a new protocol could be.

Another proposal is to use [RFC 2518] (WebDAV) as the transport layer for presence information. WebDAV is a set of HTTP extensions for distributed authoring. With presence attributes treated as HTTP documents, WebDAV would provide the machinery for getting, setting, or listing presence attributes. It would still be necessary to design extensions for subscription and notification of presence information and to set access controls. The argument against WebDAV comes from its complexity: the vast majority of the machinery of HTTP and the WebDAV extensions is inapplicable to presence information. Moreover, the parts of the presence protocol WebDAV handles are probably the least complicated parts.

7. Protocol Command Encoding

Depending on the choice of transport protocol, it may be necessary to choose how to encode protocol commands as bytes. There are many desirable properties for encodings for protocol commands, many of which contradict each other:

* Simplicity (ease of implementation)
* Compactness
* Readability (can see the data easily)
* Self-description (can understand the data easily)
* Extensibility where it might be required
* Generality (no arbitrary restrictions on content)

One option is to design our own encoding. Tools such as [RFC 2234] (ABNF) or bit-packing diagrams (as used in [RFC 791]) can be used to specify the encoding precisely. This option gives us the most flexibility to make tradeoffs different ways in different places. Arguments against designing our own encoding are:

* It is more work for us. (Although probably not more work than arguing about what kind of encoding is best.)

* It is more work for the implementor if the implementor would have otherwise been able to use a general tool to handle the encoding part of the protocol. (It might be less work if the implementor would have otherwise had to implement some complicated general encoding specification, however.)

- It carries more pitfalls--we could accidentally introduce cases where the same encoding could correspond to two pieces of data, or cases where certain kinds of data cannot be encoded.

Two likely proposals are [XML] or [Binary XML]. An XML encoding would be readable and self-describing, but it has some drawbacks:

- The full range of XML is a fairly complicated piece of machinery and is not trivial to decode.

- Although we can choose to ignore or disallow some of XML's features such as attributes and processor instructions, an implementor who chooses to use an existing XML library will generally be confronted with those features in the process of learning how to use the library.

- If we choose to disallow some of the more complicated features of XML to make it easier to write a custom decoder, designers of future extensions may be fooled into believing that the full range of XML is fair game.

- In text form, XML is gratuitously verbose; an element terminator does not need to include the name of the element for readability either by humans or machines. This oddity is due to XML's heritage in SGML, which allowed overlapping elements.

Binary XML is a more rigid format and would be somewhat easier to implement for someone lacking an appropriate XML library, but the encoding would not be readable or self-describing.

Another proposal is [SPADE]. A SPADE protocol would be readable, simple, and general, but not self-describing. Although it would not be a binary encoding per se, it would be fairly close to a binary encoding in spirit.

Another proposal is [ACP]. An ACP-based protocol would be readable, fairly simple, general, and self-describing. It would be similar in spirit to [RFC 2060] (IMAP), which is a text protocol.

8. Presence Information Format

We will need to choose a model and encoding for presence information.

One approach is to take a very minimal and restrictive attitude towards presence information, mandating that it consists only of a fixed set of simple fields, preferrably containing only information which varies frequently. This approach has a certain appeal and would make it easy to design a custom encoding for presence information, but it might leave too many potential adopters unsatisified.

If the minimal approach is not taken, presence information will tend to grow complicated and highly structured over time. The encoding of presence information will have to take this need for structure into account, so simple formats like a series of name-value pairs would be inadequate. Writing a decoder for a format which can handle arbitrary structured information starts to approach being a non-trivial problem, so it makes sense to put more weight on comforming to a format which has existing implementations.

Probably the most widely accepted such format is [XML]. The same arguments which apply against XML for protocol commands also apply against it here, of course. One new argument in favor of XML in this realm is the applicability of [XML-namespace] to the problem of multiple parties definining new presence fields. A piece of presence information using heavily-constrained XML and namespaces might look like:

```
        <?xml version="1.0"?>
        <pi:presence xmlns:pi='http://www.ietf.org/impp/schema'>
        <pi:status>online</pi:status>
        <pi:textloc>In my office</pi:textloc>
        <pi:phone><pi:type>work</pi:work>(333) 333-3333</pi:phone>
        </pi:presence>
```

Note that the URL "http://www.ietf.org/impp/schema" is merely a name; [XML-namespace] does not require or ensure that there is a document available at that URL describing the schema in any way. All of the field names are examples only.

Another possible format is the structure format used by [ACP]. This format is less verbose and would be less likely fool people into designing extensions using attributes and processor instructions, but is also less widely accepted and implemented. A piece of presence information using this format might look something like:

```
        ((STATUS "online")
         (TEXTLOC "In my office")
         (PHONE "(333) 333-3333" (TYPE "work")))
```

The field name examples are given in uppercase merely to offset them from the content. Note that ACP does not specify a way of associating values with names, just formats for atoms, strings, numbers, and parenthesized lists. Specifying how to create nested name-value pairs from those units would be a task for us, albeit a simple one. There is no namespace extension specified for use with ACP, although of course nothing would prevent us from adapting the ideas of [XML-namespace] to an ACP-based presence structure.

Another possible format is the Kerberos profile format, which is based on Windows INI files. A piece of presence information using this format might look like:

```
        [presence]
                status = online
                textloc = In my office
                phone = {
                        type = work
                        number = (333) 333-3333
                }
```

Note one subtle limitation of this format: a variable such as "phone" cannot have a text value in addition to sub-variables, so an existing variable such as "textloc" cannot be annotated with sub-variables.

Independent of the encoding, there is a question of whether we model presence information as a single document or as a collection of records which can be subscribed to individually. The finer-grain approach could result in both smaller notifications and fewer of them when only one component of presence information changse; the monolithic approach results in a simpler protocol. It is worth considering whether the bandwidth which will be used by presence notifications will be significant enough to warrant optimizations of this sort.

9. Instant Message Format

We will need to choose a format for instant messages. The most obvious option is to adopt [RFC 2045-2049] (MIME). As a message specification, MIME is a perfect fit for the problem at hand and provides a lot of machinery which would be difficult to recreate. The drawbacks of MIME come from its history in [RFC 822]; that is, MIME is somewhat more complicated than it could have been if it had no existing practice to be backward compatible with.

Since we have no existing practice to be compatible with and potentially no SMTP restrictions to be concerned about, we could also attempt to modify MIME, perhaps reusing the header specifications but encoding the headers or message body in a different way. This option might yield a message format which can be more easily implemented from scratch, or it might lead to a lot of confusion.

10. Security

So far there have been no concrete proposals for security, except for one sample implementation which uses OpenPGP. The problem breaks down into several parts:

* When a WATCHER requests PRESENCE INFORMATION, how can it verify that it is receiving correct data from the authorized SERVER and not something tampered with by a third party?

* When a WATCHER requests PRESENCE INFORMATION, how does it

authenticate to the SERVER? Likewise, when a PRESENTITY requests a change in its PRESENCE INFORMATION, how does it authenticate to the SERVER?

- How are INSTANT MESSAGES authenticated and encrypted?

Potential security proposals for each part could fall into one of three categories:

- Users authenticate and encrypt directly with other users ("end to end").

- Users authenticate and encrypt only with their local DOMAINS, and DOMAINS authenticate and encrypt to each other ("domain-based").

- Users authenticate and encrypt with local and foreign DOMAINS ("hybrid").

In the domain-based case, each part of the security problem breaks down into two sub-parts (user authenticating with local DOMAIN, DOMAINS authenticating with each other).

A domain-based security system has the advantage that local sites can take advantage of any existing security infrastructure they might have such as Kerberos. An end-to-end security system has the advantage that the compromise of a SERVER might not lead to the immediate compromise of user communications (although it could lead to the compromise of newly exchanged keys), and it would allow users to exchange keys through external channels if they do not wish to trust one or the other user's DOMAIN. But an end-to-end security system does not allow sites to reuse existing security infrastructure, and requires users to keep more state about other users.

A hybrid security system would seem to have all of the disadvantages listed above and none of the advantages.

Using a domain-based system would not, of course, prohibit users from also using an end-to-end system such as [RFC 2311, RFC 2312] (S/MIME) for instant messages.

Regardless of what type of security system is chosen, there are two essentially unsolveable problems:

- There is currently no security hierarchy with the reach of the Domain Name System.

- United States export controls prevent the development of a secure implementation which can be used outside of the United States.

These problems make it likely that many users will not be able to communicate securely.

[Possibly helpful machinery: DNSSEC, TLS, OpenPGP, SASL, GSSAPI/SPNEGO]

11. References

[Model]
M. Day, J. Rosenberg, H. Sagano. "A Model for Presence." Work in progress, draft-ietf-impp-model-03.txt.

[Reqts]
M. Day, S. Aggarwal, G. Mohr, J. Vincent. "Instant Message / Presence Protocol Requirements." Work in progress, draft-ietf-impp-reqts-03.txt.

[SRV]
A. Gulbrandsen. "A DNS RR for specifying the location of services (DNS SRV)." Work in progress, draft-ietf-dnsind-rfc2052bis-02.txt.

[SPADE]
G. Hudson. "Simple Protocol Application Data Encoding." Work in progress, draft-hudson-spade-03.txt.

[ACP]
R. Earthart. "Application Core Protocol." Work in progress, draft-earthart-acp-spec-00.txt.

[XML]
T. Bray, J. Paoli, C. M. Sperberg-McQueen. "Extensible Markup Language (XML) 1.0." W3C Recommendation REC-xml-19980210, February 1998.

[Binary XML]
"WAP Binary XML Content Format". http://www1.wapforum.org/what/technical/SPEC-WBXML-19990616.pdf

[XML-namespace]
T. Bray, D. Hollander, A. Layman. "Namespaces in XML." W3C Recommendation REC-names-19990114, January 1999.

[RFC 791]
J. Postel. "Internet Protocol." RFC 791, September 1981.

[RFC 821]
J. Postel. "Simple Mail Transfer Protocol." RFC 821, August 1982.

[RFC 822]
D. Crocker. "Standard for the format of ARPA Internet text message." RFC 822, August 1982.

[RFC 1652]

J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker. "SMTP Service Extension for 8bit-MIMEtransport." RFC 1652, July 1994.

[RFC 2045-2049]
N. Freed, N. Borenstein. "Multipurpose Internet Mail Extensions (MIME)." RFC 2045-2049, November 1996.

[RFC 2060]
M. Crispin. "Internet Message Access Protocol - Version 4rev1." RFC 2060, December 1996.

[RFC 2197]
N. Freed. "SMTP Service Extension for Command Pipelining." RFC 2197, September 1997.

[RFC 2234]
D. Crocker, Ed., P. Overell. "Augmented BNF for Syntax Specifications: ABNF." RFC 2234, November 1997.

[RFC 2251]
M. Wahl, T. Howes, S. Kille. "Lightweight Directory Access Protocol (v3)." RFC 2251, December 1997.

[RFC 2311]
S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka. "S/MIME Version 2 Message Specification." RFC 2311, March 1998.

[RFC 2312]
S. Dusse, P. Hoffman, B. Ramsdell, J. Weinstein. "S/MIME Version 2 Certificate Handling." RFC 2312, March 1998.

[RFC 2518]
E. Whitehead, A. Faizi, S. Carter, D. Jensen. "HTTP Extensions for Distributed Authoring." RFC 2518, February 1999.