Network Working Group
Internet-Draft
Expire in six months

Ross Finlayson
Live Networks

1998.01.16

The Multicast Attribute Framing Protocol

<draft-finlayson-mafp-02.txt>

1. Status of this Memo

2. Abstract

The Internet has recently seen the emergence of applications that involve the ongoing transmission, or ''pushing'', of structured data from a server to one or more client nodes. Most current applications send this data using unicast communications - usually over TCP connections. However, similar applications can also be implemented using multicast-based protocols. Multicast not only improves the scalability of this particular class of application, but also makes possible an additional class of application in which the participants can act as peers - sending data as well as receiving.

This Internet Draft describes the ''Multicast Attribute Framing Protocol'' (MAFP) - a generic, attribute-based data representation, intended for a wide variety of multicast-based applications. It is currently being used to implement the ''multikit'' generic multicast session browser (http://www.lvn.com/multikit). This draft describes an early version of MAFP that is likely to undergo changes in the future. However, it is being described now, in the hope that it will promote open discussion of this and similar protocols - ideally leading to the adoption of an open, interoperable standard for this class of application.

3. Introduction

MAFP is used to describe one or more objects that are being transmitted or 'announced' over the Internet, typically to a multicast address. (Other transmission mechanisms are possible, however, including unicast (UDP or

TCP), or even email.) These announced objects are called "programs" (by analogy with TV programs, not computer programs).

Each program is announced to a specific "directory". That is, an announcement in MAFP cspecifies both the program being announced, and a directory to which it is being announced. More than one program can - and typically will - be announced to the same directory (in separate announcements). If the announcements are being made using multicast, then the directory will be associated with a multicast address.

A program is described primarily as a set of "attributes" (name-value pairs). MAFP does not define any semantics for these attributes; the names and values can be treated merely as opaque strings. (Future documents may define attribute profiles for specific applications.)

In addition to attributes, some program descriptions may also contain IP address, port, and TTL ("time to live") parameters. Recipient(s) may use these parameters for additional communication (usually in a manner defined by the attributes). These parameters are distinguished from attributes; this allows firewalls or proxies to perform appropriate checking and/or translation, without having to understand anything about the attributes themselves.

MAFP was inspired in part by SDP [2] - a format designed specifically for announcing multimedia sessions (primarily, over the MBone). MAFP, however, extends the SDP model in the following ways:

- directories are 'first-class' objects, and can be 'announced' within other directories, just like any other session
- sessions can have arbitrary attributes (not just the fixed set of attributes that were defined for SDP)
- SDP's session bundling mechanism is generalized: Any arbitrary collection of sessions can be combined and announced as a single 'bundle'

MAFP describes the layout (within a packet) of a "program announcement". MAFP is independent of the underlying transport protocol, which may have only "best efforts" datagram semantics (e.g., UDP), or may provide some degree of reliable delivery. The only requirement is that a program announcement either be delivered entirely, or not at all. (Future versions of MAFP, however, may also allow announcements to be fragmented in ways that might be exploited by underlying transport protocols [3].)

Section 4 describes the syntax of a MAFP "program announcement". In section 5 we present examples of MAFP announcements. We close with an outline of possible future directions for this protocol.

4. The Syntax of a MAFP Packet

4.1 Basic Lexical Structure and Syntax

The lexical structure of the current version of MAFP is based upon that of the Tcl scripting language [4]. Future versions of MAFP will be more language-independent.

A MAFP program announcement is a character string, possibly terminated by a NUL character (i.e., value zero) or a LF (ASCII 0x0a). It consists of a "list" in the Tcl sense - i.e., a sequence of string "elements", each separated by white-space. Tcl's quoting rules apply to elements - in particular, elements that contain white space are enclosed in braces {}.

A MAFP program announcement is given by the following BNF grammar :

```
<PROGRAM ANNOUNCEMENT> ::= <version> <command> <incarnation> \
        <directory id> <PROGRAM DESCRIPTION>

<PROGRAM DESCRIPTION> ::= \
        <program id> <parent id> <expiration date> <PROGRAM SPECIFIER>

<PROGRAM SPECIFIER> ::= \
        "general" <ATTRIBUTE PAIRS>
|       "channel" <GROUP EID> <ATTRIBUTE PAIRS>
|       "bundle" <MEMBER DESCRIPTIONS> <ATTRIBUTE PAIRS>

<MEMBER DESCRIPTIONS> ::= \
        <empty>
|       <PROGRAM DESCRIPTION> "|"
|       <MEMBER DESCRIPTIONS> <PROGRAM DESCRIPTION> "|"

<ATTRIBUTE PAIRS> ::= \
        <empty>
|       <ATTRIBUTE NAME> <ATTRIBUTE VALUE> <ATTRIBUTE PAIRS>

<ATTRIBUTE NAME> ::= <string>
<ATTRIBUTE VALUE> ::= <string>

<GROUP EID> ::= <IP ADDRESS> <PORT> <SCOPE>
<IP ADDRESS> ::= x.y.z.w (i.e., a 'dotted quad' IPv4 address)
<PORT> ::= \
        a string representing a decimal integer in the range 0 through 65535
<SCOPE> ::= <TTL> <encryption key>
<TTL> ::= \
        a string representing a decimal integer in the range 0 through 255
```

The terminal symbols "general", "channel", "bundle", and "|" appear exactly as shown (without the quote symbols).

The remaining (nonobvious) nonterminal symbols - <version>, <command>, <incarnation>, <directory id>, <program id>, <parent id>, <expiration date>, and <encryption key> - are described below

4.2. <version>

This field indicates the version of MAFP that is being used. The current version of MAFP is 3, represented by the character '3' (ASCII 0x33). To allow for possible future versions of MAFP that may use a raw binary encoding, this must be the first character in the packet, and must be immediately followed by a space (ASCII 0x20).

4.3. <command>

The <command> describes the nature of the announcement, and thus the action to be taken by each receiver. Three commands are currently defined: "d", "p", and "x".

Both the "p" and "d" commands are handled as follows:

- If the receiver has no current record of the announced program, then it should create such a record, unless it also has no current record of the 'parent' program (see below).
- If the receiver has a current record of the announced program, then it should update its existing record with the information from the announcement. Any new attributes defined in the announcement are added to the existing record, but any attributes in the existing record that are not redefined in the announcement should remain.

When handling a "d" command, the receiver updates its local copy of the announced program, but does *not* re-announce it to anyone else. The "p" command, on the other hand, indicates that each recipient, in addition to updating its local state, should also participate in announcing this program to others. (The "d" command is analogous to someone holding up ("displaying") a sign: the sign is visible only while the person is present. The "p" command, on the other hand, is analogous to someone "posting" a sign on a notice board: the sign remains visible even after the person has left.)

The "x" command indicates that the program is to be *deleted* from the specified directory. Like the "p" command, each receiver should also participate in announcing this deletion to others. If the receiver does not already have a record of this program, then he should add a 'dummy' record in its place, indicating that the program has been deleted. This allows the receiver to ignore any 'old' announcement for the same program that it may see later. (See the description of <incarnation> below.)

4.4. <incarnation>

The <incarnation> field is a non-negative integer (string) that indicates the 'version' of the announcement. A sender should increase the value of this field if it changes the announcement in some way - e.g., when an attribute changes, or when a "d" or "p" announcement is changed to an "x" (i.e., deletion) announcement, or vice versa. Receivers must ignore any announcement whose <incarnation> is less than that which it already has on-record for the program.

MAFP does not define the contents of this field - only that this it is a non-negative integer that is increased (i.e., added to) whenever a program's announcement changes. If more than one node can make changes to a program, then MAFP does not specify how these nodes coordinate their respective <incarnation> fields, except for specifying that "the highest value wins". One possible approach, however, is to simply use timestamps (under the assumption that the nodes' clocks are at least roughly synchronized).

4.5. Program Ids

<directory id>, <program id>, and <parent id> are each "program ids": unique identifiers that denote, respectively:

- the directory to which the announcement is being made,
- the program that is being announced, and
- the parent of the program that's being announced, or "{}" (the empty string) if the program has no parent.

The "parent" mechanism is used to implement a simple attribute inheritance hierarchy. Any attributes of the parent that are not redefined in an announcement of the child are inherited by the child.

At present, MAFP does not define the structure of program ids; they can merely be considered opaque strings. Specific applications that use MAFP may impose some structure on program ids - e.g., to ensure global uniqueness. Future versions of MAFP might also impose some structure - e.g., to embed a public key representing the program's 'owner'.

The receiver should ignore an announcement if the <directory id> does not correspond to the transmission mechanism (e.g., a multicast group address) on which the announcement is being made.

As a special case, a <program id> of "SELF" (without the quote symbols) indicates that this description is for the directory itself, rather than for a program within the directory. In this case, <command> must be "d" or "p" only - not "x", and the <PROGRAM SPECIFIER> fields must denote a "channel" (see below).

This (optional) self-description mechanism allows a receiver to learn the complete attributes of a directory, given only the directory's IP address and port.

4.6. <expiration date>

This field specifies the time at which the announced program will expire (and can thus be removed from the receiver's records). It is an integer in "Unix time format": the number of seconds since 1/1/1970 UTC.

Note that, because of attribute inheritance, a program's expiration date should not exceed that of its parent. If an announcement is received that attempts to set a program's expiration date to a later time than that of its parent, then the parent's expiration date should be used instead.

4.7. <encryption key>

In the future this field will be used to specify an encryption key for a "channel" (see below). Currently, the only value defined for this field is the string "nokey", meaning: no encryption is performed.

4.8. Channels and Bundles

MAFP recognizes three distinct kinds of program: channels, bundles, and 'general' (everything else).

A channel has an associated IP address, port number, and TTL. (A directory is itself a channel.)

Bundles are used to implement a simple grouping mechanism. An announcement of a bundle includes each of its members.

5. Examples

The first example below illustrates the announcement of a program - specifically a bundle "prog2" with two members: (i) a channel "prog4" (note the multicast address, port, and TTL), and (ii) a 'general' program (i.e., neither a channel nor a bundle) "prog6". (The announcement is being made to the directory "prog1".)

Each of the three programs (the bundle and its two members) has an attribute named "anAttributeName"; the channel has an additional attribute named "anotherAttributeName". Note the use of braces to 'quote' multi-word attribute values.

The line breaks in this example are for clarification only. Newline characters must *not* appear in an actual "on-the-wire" announcement.

```
2 d 0 prog1
    prog2 prog3 857203200 bundle
        prog4 prog5 857203200 channel
            238.236.141.215 50470 127 nokey
            anAttributeName {This is an attribute of the channel} |
        prog6 prog7 857203200 general
            anAttributeName {This is an attribute of the 'general' program}
            anotherAttributeName foobar |
        anAttributeName {This is an attribute of the 'bundle'}
```

The second example - taken from an actual use of MAFP - illustrates the announcement of an earthquake report. (Once again, the line breaks are for clarification only.)

```
2 p 880038262 P:206.86.37.13:78801286
        P:206.86.37.1:879941367:world R::TEMPLATE-AtomicNonChannel
            880200567 general
                info {SAMOA ISLANDS REGION}
                startTime 879941367
                magnitude 5.4Ms
                longitude 175.79W
                latitude 14.04S
                depth 33.0
```

6. Possible Future Directions

(In no particular order)

- Make the syntax more programming language independent, and (ideally) leverage off an existing attribute description format - possibly XML [5]
- Use NTP time format (instead of Unix)?
- Define attributes as (name, type, value) triples, rather than (name, value) pairs?
- Define a way to delete (rather than just redefine) attributes
- Define a compression scheme, to efficiently support low-bandwidth links
- Allow program announcements to be digitally signed, for integrity
- Try to be as consistent as possible with SAP [6], and/or any reliable multicast protocols that emerge from the IRTF's "reliable multicast" working group [7].
- I18N
- Support IPv6 addresses as well as IPv4

6. References

[1] "multikit" - a distributed, multicast-based directory browser
        http://www.lvn.com/multikit/
[2] M Handley, V. Jacobson. SDP: Session Description Protocol
        draft-ietf-mmusic-sdp-04.txt
[3] David D. Clark, David L. Tennenhouse. Architectural Considerations for
        New Generation Protocols. SIGCOMM 1990.
        ("Application Layer Framing")
[4] Tcl/Tk
        http://sunscript.sun.com/
[5] World Wide Web Consortium. Extensible Markup Language
        http://www.w3.org/TR/WD-xml
[6] M Handley. SAP: Session Announcement Protocol
        draft-ietf-mmusic-sap-00.txt
[7] IRTF "reliable multicast" working group
        http://www.east.isi.edu/RMRG/