FLUTE-RDIST specifications

Vincent Roca, Christoph Neumann

FLUTE-RDIST Specifications

INRIA Rhône-Alpes, Planète reserach team, France
vincent.roca|christoph.neumann@inrialpes.fr
http://www.inrialpes.fr/planete/roca/mcl

May 10, 2005

1 FLUTE-RDIST protocol

1.1 Principles

The goal of flute-rdist is to mirror a set of files located on a server to a set of clients. Therefore the server runs the flute-rdist server tool, and the clients the rdist client tool. The set of mirrored files is not necessary the same for each client.

The flute-rdist server uses the file delivery protocol FLUTE to distribute the files to the clients. A set of control messages are used to coordinate the distribution of the files and guarantees that all files have been received successfully. More precisely the control messages are used to (1) announce ongoing multicast sessions to the clients, (2) to inform the server on the reception and synchronisation status of each client, (3) to enable new clients to join an flute-rdist mirror session and (4) to perform point to point file repair if required.

The basic concept of rdist is depicted in figure 1.

At any moment in time the file-rdist server knows the list off all the clients that are interested in a given set of files. To each file that needs to be mirrored is associted an md5sum. To know if a client is synchronized, the md5sum of the two files (one on server side the other on client side) has to be compared. If the the two values are not equal the client has an not up-to-date version of the file, and assiocated actions have to be taken (see later in this document).

1.2 Control messages

This section described the basic control messages. All control messages use HTTP [2].

Session anouncement server → clients

The session announcment message is sent from the server to the client, either in multicast or unicast. It describes an ongoing FLUTE delivery session using the sdp protocol (as described in [4]). The sdp file describing the session is

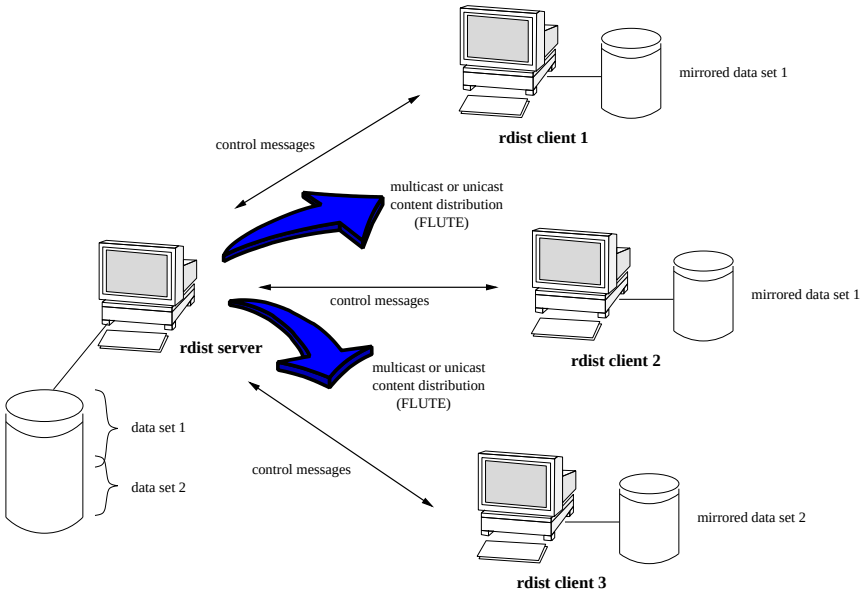

Figure 1: Basic concept.

contained in the body part of the message. For an example sdp file for a session see section 2.1.

The session announcment messages are delivered using HTTP POST.

The client responds with an HTTP response (200 OK or client error status). Error should be returned if client is unable to join session (e.g. because multicast traffic is not supported, ...).

Report request server → clients.

The report request message is sent from the server to the clients, either in multicast or unicast. It triggers a reception and synchronization report on client side. The scope (i.e. the list of concerned files) of the triggered reception and synchronization report message can be defined within the report request.

The report request messages are delivered using HTTP GET.

The client responds with an HTTP response including the Reception Report in the data part.

Reception and Synchronization Report Notify clients → server.

The reception and synchronization report notify message is sent from the clients to the server in unicast. It informs the server on the reception state of a set of files and about its reception capabilties (support of multicast or not).

The Reception and Synchronization Report Notify messages are delivered using HTTP POST.

The server responds with an HTTP response (200 OK or redirection 302 Found-Redirection with sdp file in data part).

Join request clients → server.

The join request message is sent from a client to the server in unicast. By sending this message the clients ask the server to get a mirrored copy of the data for which new clients are allowed (indicated with a "*" in the rdsit configuration file).

The join request messages are delivered using HTTP GET.

The server responds with an HTTP response

Repair Request (optional) clients → server.

This message is optional. Repair messages are sent from the server to the client in unicast. For more details read [1] section 9.3, section 9.3.4, section 9.3.5 and Tdoc S4-AHP156.

1.3 Normal state: Actions for file synchronization

The server needs to now the state of each file on each receiver. The server may store the state of each file on each receiver locally, or regularly do a pooling. We will use the latter solution in the following section:

1. Pooling The server regularly (e.g. each hour and only if files have been updated, or immediatly after a file has been updated) starts pooling. Therefore the server sends a Report Request to all clients. Each client respond to that message.
   (a) Report request: server → all clients, scope: all files
   (b) Report request response: client → server
   Each clients responds to the message using HTTP response and indicating state (200 OK, error etc.). The clients includes a recepetion and synchronization report in the data part of the message.
2. Decision taking According to the reception reports received the server has to take decisions. The server has to decide on how to optimally schedule the file transmissions, that means (1) to decide which mechanism to use for file distribution (either FLUTE multicast, FLUTE unicast, or repair messages) and to whom it should be send, (2) to decide which files to send in grouped sessions.
3. Starting sessions
   (a) Session announcement: server → clients
   (b) Session announcement response: client → server
4. optionally: report request

5. Reception Report Notify At the end of a data transmission session, a client signals to the server the files he received succesfully and those he could not receive. This done by sending a reception report notify message to the server.

   (a) Reception Report Notify: client → server

   (b) Reception Report Notify response: server → client

1.4 Actions taken according to occured events

A file has been updated on server side three options:

1. do nothing and wait for pooling
2. immediatly send session announcement
3. immediatly do pooling with scope "the updated file"

A file has been deleted on server side

A new client wants to join a session

A client has lost a file

2 Body formats

2.1 SDP

Here is an example sdp file:

```
v=0
o=user123 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24
s=File delivery session example
i=More information
t=2873397496 2873404696
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=flute-tsi:1
a=flute-ch:1
a=content-desc:http://www.example.com/flute-sessions/session001
m=application 12345 FLUTE/UDP 0
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
```

2.2 Recpetion and Syncronization Report

The reception report is represented using the following XML-Schema:
Here is an example XML file:

```
<?xml version="1.0" encoding="UTF8?>
<receptionReport>
    <fileURI status="received" md5sum="ff9d42a15e8239750247faaed43a6ca8">file1.3.mp3</fileUR
    <fileURI status="not received">file2.3.mp3</fileURI>
    <fileURI status="received" md5sum="b510ded05129c81bbb48f56c55536dc5">file3.3.mp3</fileUR
</receptionReport>
<multicastable\>
```

2.3 Report request scope

The request scope is represented using the following XML-Schema:
Here is an example XML file:

```
<?xml version="1.0" encoding="UTF8?>
<ReportRequestScope>
    <fileURI>file1.3.mp3</fileURI>
    <fileURI>file2.3.mp3</fileURI>
    <fileURI>file3.3.mp3</fileURI>
</ReportRequestScope>
```

Here is a second example XML file:

```
<?xml version="1.0" encoding="UTF8?>
<ReportRequestScope>
<allFiles\>
</ReportRequestScope>
```

3 The flute-rdist configuration file

The configuration file has been largely inspired by the configuration file used by `rdist` [3].

3.1 Minimal set of features

The configuration file contains a sequence of entries that specify the files to be copied, the destination hosts, and what operations to perform to do the updating. Each entry has one of the following formats.

```
<variable name> '=' <name list>
<source list> '->' <destination list> <command list>
```

The first format is used for defining variables. The second format is used for distributing files to other hosts. It is used to copy out of date files and/or directories. The source list specifies a list of files and/or directories on the local host which are to be used as the master copy for distribution. The destination list is the list of hosts to which these files are to be copied.

Comments begin with `#` and end with a newline.

Variables to be expanded begin with `'$'` followed by one character or a name enclosed in curly braces (see the examples at the end).

The source and destination lists have the following format:

```
<name>
```

or

```
'(' <zero or more names separated by white-space> ')'
```

3.2 Optional features

The shell meta-characters `'['`, `']'`, `'{'`, `'}'`, `'*'`, and `'?'` are recognized and expanded (on the local host only) in the same way as csh(1). They can be escaped with a backslash. The `'~'` character is also expanded in the same way as csh but is expanded separately on the local and destination hosts.

The command list consists of zero or more commands of the following format:

```
'except'      <name list>   ';'
'except_pat'  <pattern list>';'
```

The `except` command is used to update all of the files in the source list except for the files listed in name list. This is usually used to copy everything in a directory except certain files.

The `except_pat` command is like the except command except that pattern list is a list of regular expressions (see ed(1) for details). If one of the patterns matches some string within a file name, that file will be ignored. Note that since `'\'` is a quote character, it must be doubled to become part of the regular expression. Variables are expanded in pattern list but not shell file pattern matching characters. To include a `'$'` , it must be escaped with `'\'`.

3.3 Example

An example configuration file:

```
HOSTS = ( * goofy 194.199.24.108 )

FILES = ( /bin /lib /usr/bin /usr/games
          /usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
          /usr/lib /usr/man/man? /usr/ucb /usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
```

```
          sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )

${FILES} -> ${HOSTS}
          except /usr/lib/${EXLIB} ;

IMAGEN = (ips dviimp catdvi)

/usr/local/${IMAGEN} -> planetcast
```

References

[1] *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Setvice; Protocols and Codecs (Release 6)*, Nov. 2004. 3GPP TS 26.346 v1.5.0.

[2] R. Fielding, J.Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, June 1999. Request for Comments 2616.

[3] MagniComp. *rdist - remote file distribution client program.* URL: http://www.magnicomp.com/rdist/.

[4] H. Mehta, R. Walsh, I. Curcio, J. Peltotalo, and S. Peltotalo. *SDP Descriptors for FLUTE*, May 2005. Work in Progress: <draft-ietf-rmt-flute-sdp-02.txt>.