Distributed Replicated Files Protocol Specification

Sining Ma
sma87@stanford.edu

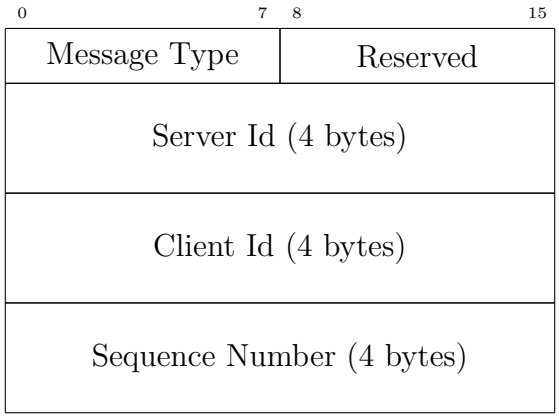
May 11, 2014

1 Protocol Specification

This section describes all message packets syntax, semantics and timing. Messages bytes are in network byte order.

1.1 Common Header

All packets start with the same header structure.



Message Type Each type of packet in this homework which is assigned

Reserved This field is reserved and should always be zeros.

Server Id This field is for all messages that sent by servers. Server Id of clients sent messages is not .

Client Id This field is for all messages that sent by clients. Client Id of servers sent messages is not .

Sequence Number Each outgoing packet contains a monotonically increasing sequence number. This number wraps back to zero when overflows.

1.2 Init (Message Type 0xC0)

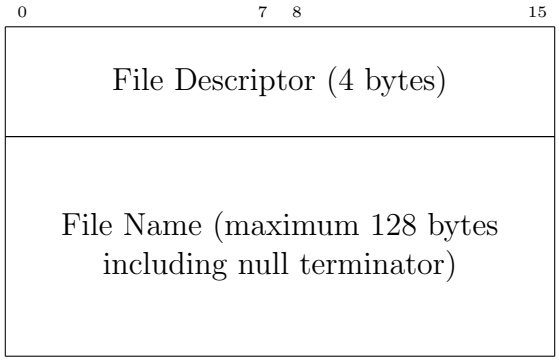
The client sends Init messages when InitReplFs() is called to detect the servers.

1.3 InitAck (Message Type 0xC1)

Servers send InitAck messages when receive Init message to inform the client the server existence.

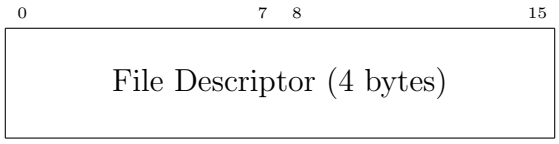
1.4 OpenFile (Message Type 0xC2)

The client sends OpenFile messages when OpenFile() is called to open a new file on local client file system and the servers.



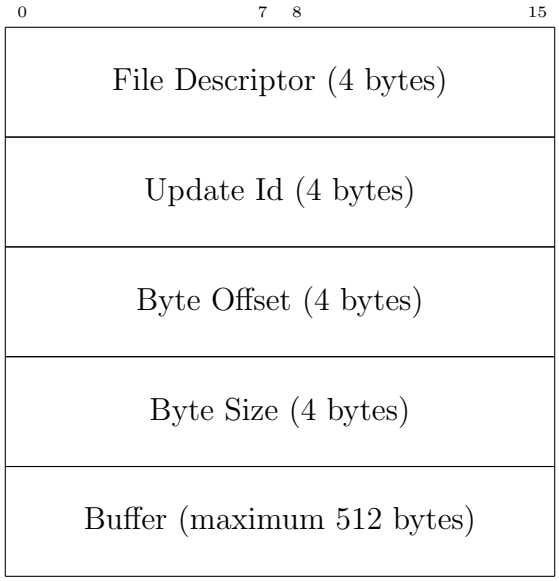
1.5 OpenFileAck (Message Type 0xC3)

Server send OpenFileAck messages to acknowledge OpenFile message with file descriptor generated by the server. If the server is unavailable or error happens when open file, -1 returns.



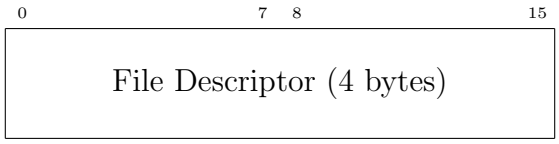
1.6 WriteBlock (Message Type 0xC4)

The client sends WriteBlock message when WriteBlock() is called to stage a contiguous chunk of data. No Ack message will be sent from servers. Before one commit, the client can do multiple WriteBlock call to update the file. Update Id is used to mark update sequence before commit call.



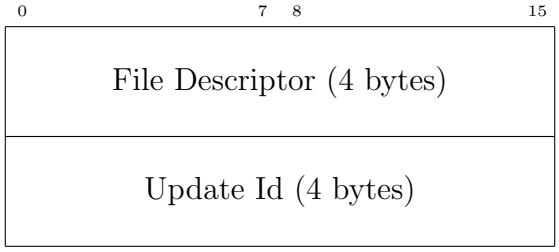
1.7 Vote (Message Type 0xC5)

Two phase commit is used to ensure that all updates in WriteBlock are received on the server side. The client sends Vote message when in commit request phase to check if all the servers receive all updates and are ready to do final commit.



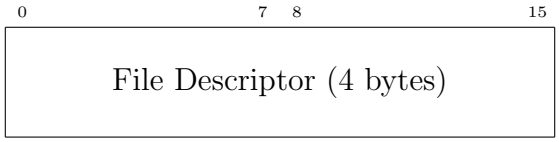
1.8 VoteYes (Message Type 0xC6)

Servers send VoteYes message with an update Id which indicates the latest update Id that the server has received successfully from the client. When the client receives this message, the client checks if update Id in the message is identical to the latest update Id that is required from the client side for this commit. If update id does not match, message retransmission happens.



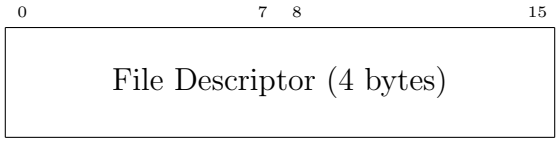
1.9 VoteNo (Message Type 0xC7)

Servers send VoteNo message if any error happens on the server side. The client receives this message and is supposed to send abort message to tell servers rollback all the changes.



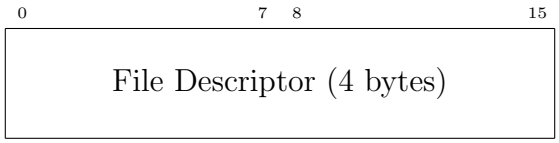
1.10 Commit (Message Type 0xC8)

The client sends Commit message to all servers to complete and commit all updates to the file.



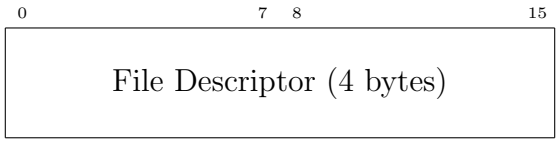
1.11 CommitAck (Message Type 0xC9)

Servers send CommitAck message to the client to tell the client to complete the transaction.



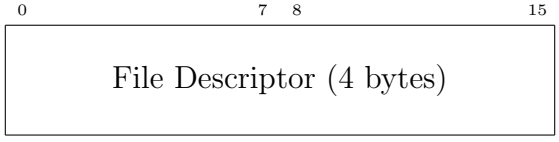
1.12 Abort (Message Type 0xCA)

The client sends Abort message to all the servers when receive VoteNo message. All the servers undo the transaction which rollbacks file update.



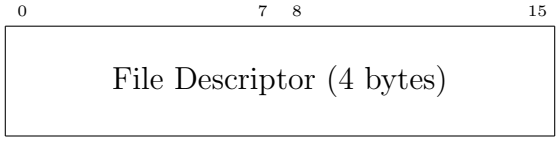
1.13 AbortAck (Message Type 0xCB)

Servers sends AbortAck message to the client. This message informs the client file rollback on servers is done.



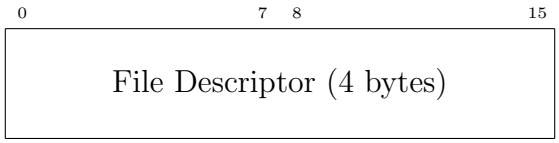
1.14 Close (Message Type 0xCC)

The client sends Close message to the servers to close the opened file.



1.15 CloseAck (Message Type 0xCD)

Servers send CloseAck message to indicate the file is closed.



2 Protocol Details

2.1 Client and Server Id generation

Client and server Ids are generated randomly with a 32-bit integer with a good seed for number generator. The probability of conflict is 5.32101e-20. This is negligible. In real world setting, each client and server is assigned an unique identifier, e.g. UUID.

2.2 Protocol Operation Flow

- InitReplFs(), the client sends init message to the multicast group to detect if there are enough servers count in init timeout time. If responding servers count is smaller than numServers, the client returns error. Otherwise, if there are more servers, the client selects the first numServers count, and only accepts acknowledge messages from these selected servers.

- OpenFile(), the client sends OpenFile message to servers to open a file and waits for OpenFileAck message.

- OpenFileAck message is sent from the servers. If file descriptor is -1, this means that OpenFile fails and the client returns error. If selected servers response count is less than init phase servers count, the client returns error.

- WriteBlock() sends file update message to the servers without acknowledge. Client remembers total update count before one commit. Servers update his latest update Id only when receives a valid WriteBlock updates. For example,The client sends update 1, 2, 3, 4 and 5 to the servers. If the server receives update 1, 2 and 3, but 4 is lost, latest update Id is 3. Then even if update 5 is received, this update is ignored and latest update Id is still 3, because 4 is lost. But Client update Id is 5.

- Commit(), the client sends vote message to all servers, and waits for vote response. The servers receive vote message and reply with the latest update Id or vote no message if error happens.

- The client receives many vote yes message in vote timeout time. If all servers response vote yes message with update id match client update Id, client sends commit message to all servers. The second case is that the client receives vote yes message with update Id smaller than client update Id. The client waits till vote timeout. The client checks all

servers vote yes messages and finds the smallest update Id. The client starts retransmit file updates from the smallest update Id. On the server side, in one commit, any update Id smaller than server latest update Id is ignored. Servers only update file when an update id bigger than latest is received.

- The client receives a vote no message from one server. Client sends abort message to all servers to rollback. The client returns error.
- Commit or Abort Ack message is received or timeout. This indicates one commit transaction is done.
- The client calls abort() instead of commit(). The client sends abort messages.
- The client sends close message to close the file. If no Close Ack message receives, the client returns error.

2.3 Protocol Timing

2.3.1 InitReplFs

The client sends Init messages to servers every 200ms to multicast group, this init phase timeout is 2s. The client only accepts servers that response in init phase timeout.

2.3.2 OpenFile

The client sends OpenFile messages to servers every 200ms to multicast group, this OpenFile phase timeout is 4s.

2.3.3 Vote

The client sends Vote messages to servers every 200ms to multicast group, this Vote phase timeout is 4s. If any server has no response in 4s, the client treats this case as response vote no message, and sends abort message.

2.3.4 Commit and Abort

The client sends Commit or Abort messages to servers every 200ms to multicast group, this Commit or Abort timeout is 4s. If any server has no response in 4s, the client returns error. If any server receives file update but no commit or abort message in this 4 seconds, the server rollback the file.

2.3.5 Close

The client sends Close messages to servers every 200ms to multicast group, this Close phase timeout is 2s.