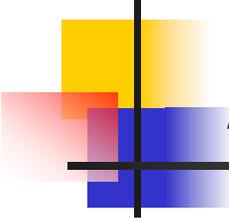


GridFTP Working Group Meeting

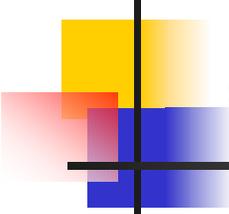
Igor Mandrichenko
GGF9, Chicago, Illinois
October 6, 2003

<http://www-isd.fnal.gov/gridftp-wg>
<http://forge.gridforum.org/projects/gridftp-wg>



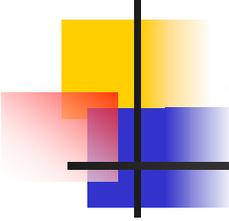
Agenda

- Where are we
- GridFTP v2.0 Proposals
 - EOF in Stream mode
 - GET/PUT
 - eXtended Block mode
- Moving to Grid Forge
- Miscellaneous



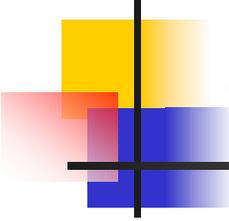
GGF8 results

- GridFTP v1.0 document is up for public comments
- During GGF8:
 - Charter was approved
 - Goal – incremental improvement of GridFTP protocol -> GridFTP v2.0
 - List of points of improvements is produced



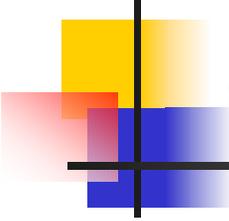
Overview of proposals: EOF

- EOF in Stream mode
 - Help server to distinguish between end of successful Stream mode upload and premature client termination
 - New command – EOF with “commit” semantics
- URL: <http://www-isd.fnal.gov/gridftp-wg/eof.htm>



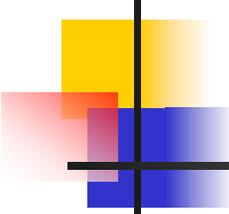
Overview of proposals: GET/PUT

- GET/PUT
 - Combine RETR/STOR with PASV/PORT into single command so that the server can come up with right data socket address based on file path and perhaps other information
- URL: <http://www-isd.fnal.gov/gridftp-wg/getput/getput.htm>



Overview: eXtended Block mode

- Modification of Extended Block mode
- Free of the “uni-directional” requirement:
 - *Data must flow in the same direction as data connection: sender always establishes data connection*
- Allows parallel transfers in both directions in NAT/firewall environment
- URL:
<http://www-isd.fnal.gov/gridftp-wg/eblock/EBlockProposal.htm>



EOF in Stream mode

- What happens now when client is terminated prematurely:

Client

STOR file

(sending data)

(client is terminated)

(control socket is closed)

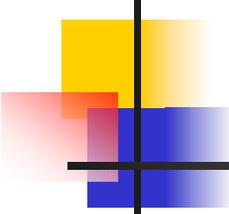
Server

120 Opening data connection

(receiving data)

(end-of-file on data socket =
end of data transmission)

226 Data transfer complete



EOF Command

- Using EOF command to signal end of successful transmission

Client

STOR file

(sending data)

(client closes data channel)

EOF

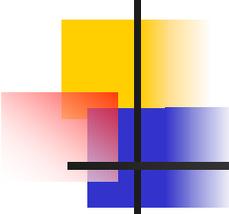
Server

220 Opening data connection

(receiving data)

(end-of-file on data socket =
end of data transmission)

226 Data transfer complete



EOF Command

■ Premature client termination

Client

STOR file

(sending data)

(client terminates)

(control connection is closed)

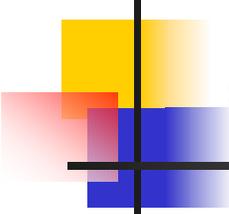
Server

220 Opening data connection

(receiving data)

(end-of-file on data socket =
end of data transmission)

(no EOF received: server knows
that the transfer was incomplete)



GET/PUT

- RFC959 FTP in passive mode:

Client

Server

PASV

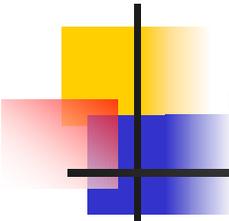
220 OK (12.23.34.45.56.67)

STOR /path/file

120 Opening data connection

...

- When server receives PASV, it has to reply with data socket address without knowing what file is about to be transferred
- Not always possible with distributed servers, etc.



GET/PUT

■ General protocol:

Client

GET <parameter>=<value>;...

Server

1xx ...

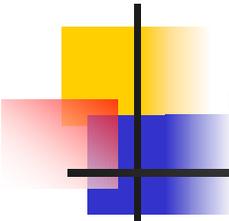
1xx ... PORT=(a.b.c.d.e.f)

...

2xx Data transfer complete

■ Parameters:

- Connection mode (active/passive)
- Transfer mode (S,B,E,X,...)
- Max number of data channels
- Other transfer parameters



GET/PUT Examples

■ Passive download

Client

Server

GET connect=pasv;path=/path/file;mode=s

120 OK, port is (12.23.34.45.56.67)

...

220 Data transfer complete

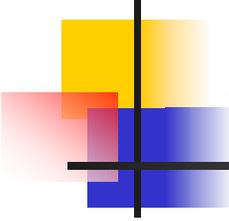
■ Active upload

PUT connect=acvt;address=(43.32.21.78.54.43);path=/path/file

120 OK opening data connection

...

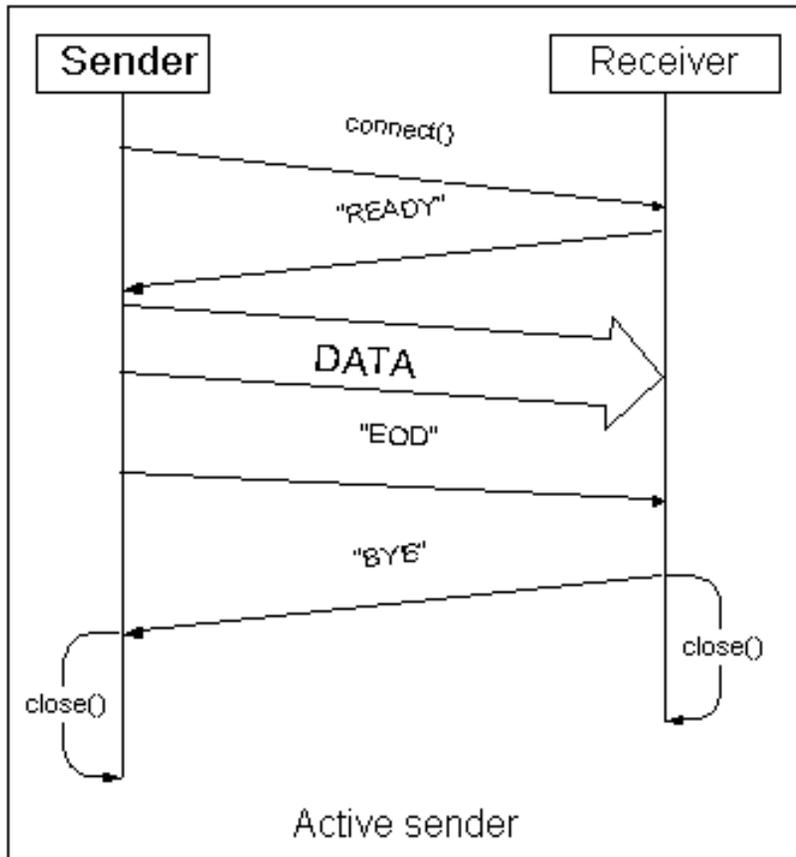
220 Data transfer complete



eXtended Block mode

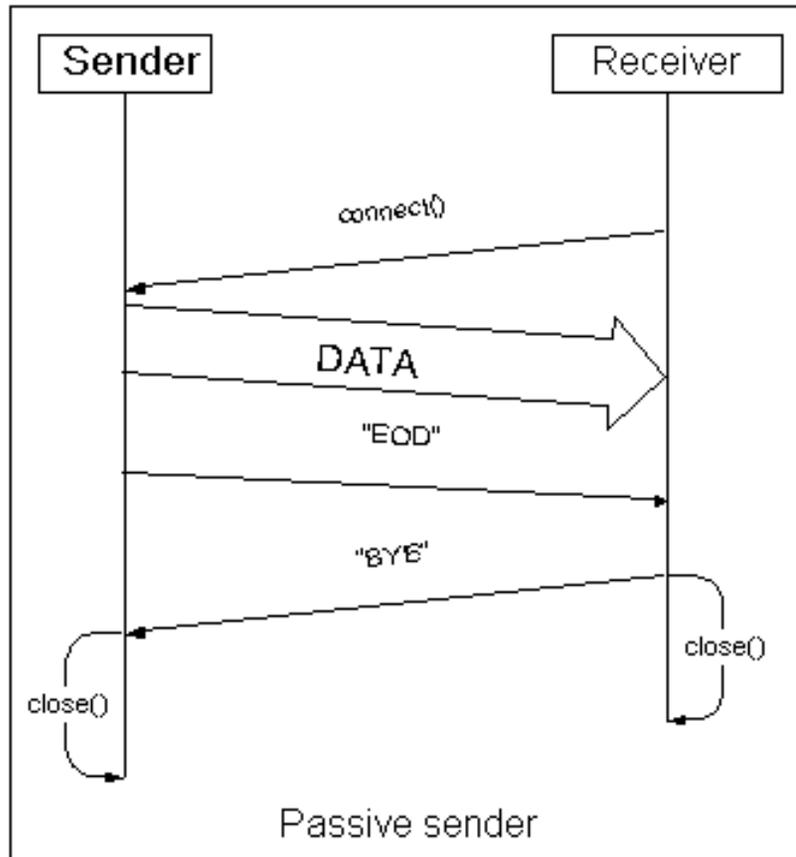
- Modification of Extended Block mode
- Free of the uni-directional transfers requirement
- Idea: replace EODC with reliable and explicit data channel establishment/termination
- Benefits:
 - No data is sent before data channel opening is confirmed by both peers
 - Data sender is assured no data is lost

X mode: how it works ?



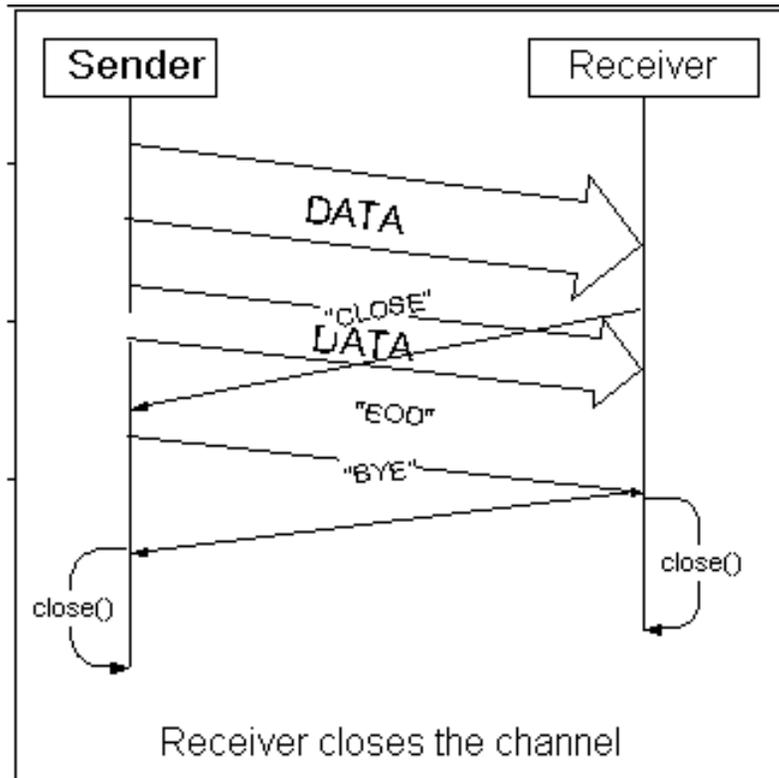
- Send some control information on data channel in the opposite direction to the data flow:
- "READY" – to confirm data channel establishment
- "BYE" – to confirm receipt of EOD and data channel termination

Passive Sender in X mode

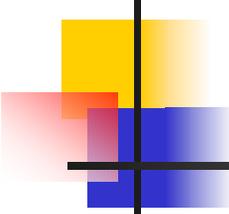


- No need to send "READY" because receiver initiates data connection

Traffic control by receiver

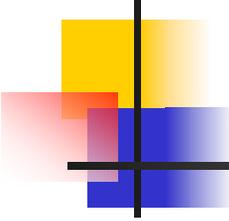


- Receiver requests termination of a data channel using "CLOSE" message



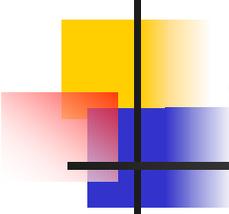
EOF in X mode

- Sender sends block with EOF bit set on one or more data channels
- EOF is sent only after all open channels are confirmed with "READY"
- After EOF is sent/received:
 - No new data channels will be initiated/accepted
 - Existing data channels will exist until closed with EOD/BYE



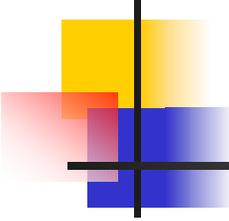
Summary of X mode

- Explicit and reliable data channel initiation/termination
- No race condition in passive sender/active receiver mode
- Result: bi-directional parallel transfers in presence of NAT/firewall
- Dynamic data channel opening/closing
 - Initiator can open/close data channels
 - Acceptor can close data channel



Other proposals for GridFTP v2.0

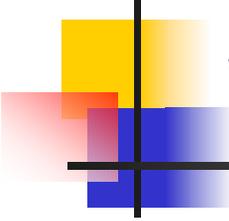
- Structured directory listing, “stat” functionality
 - Adopt IETF draft
- IPV6 support
 - Adopt RFC 2428



Issues without concrete proposals

- Control of server feedback
 - Performance markers, keep alive noise, etc.
- Data integrity verification
 - CRC over whole file, each block ...?
- Packed transfers
 - Tar them up and send as one file
- Flexible striping control algorithms

- These issues need volunteers !



What is next ?

- Get proposals for other items
- By GGF10 produce final draft for GridFTP 2.0 document
- Create working prototypes
- Present results