AdiIRC - Bug #4184

Issues/observations related to DCC

11/02/2018 02:27 AM - Paul Janson

Status:	Assigned	**Start date:**	11/01/2018
Priority:	Normal	**Due date:**	
Assignee:	Per Amundsen	**% Done:**	0%
Category:	Scripting	**Estimated time:**	0.00 hour
Target version:	3.3		
Operative System:	Windows 7	**Regression:**	No

Description

making sure there's no download file and create dummy file:
//remove downloads\test2.txt | btrunc test2.txt 0 | bwrite test2.txt 4095 1 t | echo -a size $file(test2.txt).size

Create an event to monitor what's happening with the DCC:

```
CTCP :*::echo ~~s ctcp from $nick : $1~~
```

Bug1: The menu says "ON CTCP" even though there's no 'ON'. Ditto with "ON RAW".

Send to self:
//dcc send $me test2.txt

DCC Send of test2.txt to self complete (00:00:00 24.69KB/Sec)
DCC Get of test2.txt from self complete (00:00:00 249.99KB/Sec)

Bug2: DCC send/resume should trigger the CTCP event, but did not.
Bug3: I'm having trouble replicating the cause, but I ended up with a test1.txt which didn't allow writing to it nor could Windows Explorer delete it. Once this happened, of course /bwrite fails to write to it, and /btrunc can't change the filesize either. However there is no error message, and there's nothing that halts the next next command from executing. There is an error when /write /remove or /rename try to mess with it.

change dummy size smaller:
//bwrite -c test2.txt 2047 1 33 | echo -a size $file(test2.txt).size

fix: add -c switch to the TODO list, filesize still 4096

Other related issues with handling of DCC protocol.

1. If incoming file is same filesize as the file you already have, should it really do a DCC RESUME so it can ask for the last 8kb of the file? Best case scenario is that you receive data identical to the last 8192 bytes of the existing image.jpg of the same filesize, and the file is not changed other than updating the timestamp. The alternative is that the incoming file is coincidentally the same filename/filesize as a file you already have but a different image, so a valid image gets the final 8kb replaced with the content of a different image.

To avoid having the sender think you were rejecting the file, or had problems accepting files, instead of resuming an identical filesize, it could 'lie' to the sender and say it received the DCC ok but really does nothing. Like it does when the incoming file is larger than the existing file.

2. If the file in the download folder has filesize 0-8192, it requests a dcc-resume at offset zero. Would it be reasonable for the client to just do a normal ACCEPT, then if the transfer begins successfully, just replace the content beginning at the start of the file. It's even doing a dcc-resume at offset zero if the existing file is size zero. My prior ISP caused a problem where sometimes the port changed somewhere outside the client, which meant normal DCC worked fine, but which caused all dcc-resume to fail because the port was unknown. In the case of size 8192 or smaller, this seems wasted handshaking when the successful DCC does the same thing as if the DCC overwrites the existing filename entirely.

I know DCC's not supposed to truncate the file unless the transfer actually begins, otherwise someone could delete your small file or else snip 8kb from the tail end just by starting a DCC then failing.

3. I'm assuming the reason for resuming the last 8192 of a file was in case the last packet of the transfer was corrupted, to either to make sure to repeat the last full packet when packet size was 4096, or because 8192 used to be the max valid packetsize. However now that packet sizes of up to 64k are allowed, should this 8192 also be configurable too? Either allowing it to be zero to not repeat

anything, or to have it be 100% or 200% of whatever your configured dcc-get packetsize is? i.e. if I change my packet size to 65536, resuming at 8192 below the existing filesize doesn't make sense, since that's retrieving only 1/8th of a packet. Related to #1 above, I'm not sure if packets being corrupted would happen when a transfer reports to both sides that it was successful at 100% of filesize.

History

#1 - 11/02/2018 03:21 AM - Per Amundsen

- Category set to Scripting

- Status changed from New to Assigned

- Assignee set to Per Amundsen

> Bug1: The menu says "ON CTCP" even though there's no 'ON'. Ditto with "ON RAW".

If you are talking about the events wiki, this is deliberate, even though "ON" is not used, I think most people thinks "ON RAW" and "ON CTCP" in their head.

> Bug2: DCC send/resume should trigger the CTCP event, but did not.

I will look into that.

> Bug3: I'm having trouble replicating the cause, but I ended up with a test1.txt which didn't allow writing to it nor could Windows Explorer delete it. Once this happened, of course /bwrite fails to write to it, and /btrunc can't change the filesize either. However there is no error message, and there's nothing that halts the next next command from executing. There is an error when /write /remove or /rename try to mess with it.

While transferring, the file is locked it can not be written to by any command or other programs, it's possible that something went wrong and AdiIRC didn't properly close the file when the transfer is aborted/finished, if you figure out how to replicate, let me know.

> //bwrite -c test2.txt 2047 1 33 | echo -a size $file(test2.txt).size
>
> fix: add -c switch to the TODO list, filesize still 4096

Was not aware of this, I will add the -c switch for next beta.

> 1. If incoming file is same filesize as the file you already have, should it really do a DCC RESUME so it can ask for the last 8kb of the file? Best case scenario is that you receive data identical to the last 8192 bytes of the existing image.jpg of the same filesize, and the file is not changed other than updating the timestamp. The alternative is that the incoming file is coincidentally the same filename/filesize as a file you already have but a different image, so a valid image gets the final 8kb replaced with the content of a different image.
>
> To avoid having the sender think you were rejecting the file, or had problems accepting files, instead of resuming an > identical filesize, it could 'lie' to the sender and say it received the DCC ok but really does nothing. Like it does > when the incoming file is larger than the existing file.

The reason why both AdiIRC and I think mIRC does as well, is for compatibility with buggy/older clients and specifically xdcc bots.

I don't remember all the details anymore, but there are two different cases, one where a bot/client disconnects before the last X bytes are sent, it will trigger a resend or something like that. The other case is where a bot initiates a transfer, if no data is sent, the bot/client thinks you don't have the full file and will not send any other files until the issue is resolved, in this case AdiIRC could do a dummy transfer, but I prefer to keep it simple since networking is not my strong suite.

> 2. If the file in the download folder has filesize 0-8192, it requests a dcc-resume at offset zero. Would it be reasonable for the client to just do a normal ACCEPT, then if the transfer begins successfully, just replace the content beginning at the start of the file. It's even doing a dcc-resume at offset zero if the existing file is size zero. My prior ISP caused a problem where sometimes the port changed somewhere outside the client, which meant normal DCC worked fine, but which caused all dcc-resume to fail because the port was unknown. In the case of size 8192 or smaller, this seems wasted handshaking when the successful DCC does the same thing as if the DCC overwrites the existing filename entirely.

8192 is configurable from 512 to 65536, your point makes sense, but I would prefer not to mess with DCC, it took me a very long time to make it work reliably. I will write it down though and look into it.

> I know DCC's not supposed to truncate the file unless the transfer actually begins, otherwise someone could delete your small file or else snip 8kb from the tail end just by starting a DCC then failing.
>
> 3. I'm assuming the reason for resuming the last 8192 of a file was in case the last packet of the transfer was corrupted, to either to make sure to repeat the last full packet when packet size was 4096, or because 8192 used to be the max valid packetsize. However now that packet sizes of up to 64k are allowed, should this 8192 also be configurable too? Either allowing it to be zero to not repeat anything, or to have it be 100% or 200% of whatever your configured dcc-get packetsize is? i.e. if I change my packet size to 65536, resuming at 8192 below the existing filesize

doesn't make sense, since that's retrieving only 1/8th of a packet. Related to #1 above, I'm not sure if packets being corrupted would happen when a transfer reports to both sides that it was successful at 100% of filesize.

8192 was derived from testing xdcc transfers in mIRC, it's possible that this could work using a different number such as the configured packet size, I will add a new hidden option for next beta, something like /setoption DCC DccResumeLastPacket=N .. where N is either percentage or 0 as you said.