Object: To Write some kind of Protocol? That will help determine what should be listening, what should be talking, and how to record all the data that is being heard.

In more generic terms, the protocol is intended to provide a framework upon which any proxy-proxy communication can be designed against.

The sequence of events for initial client to ‘server’ connection is as follows:

1.. Connect the client to the host

2.. Verify the connection via a predefined password (or via specific IP permission)

3.. Check the time difference between the 2 servers

4.. Update each client on connection

5.. Sending the data

6.. Requesting the data from the other party

7.. Acknowledging the request

8.. Send the Data

9.. Receiving the data

10. Parsing the data requested

11. Choosing if that data is to be added or discarded

One Central server to me seems like it's also going to be something that will be impossible to completely automate. The selection of the server is something that will have to be done by a person I would think. Assuming there is 1 Central server however I think we could say that it would be possible for the connections to be Semi-Automatic.

I have thought about this for a bit and come up with this: Connecting and synchronizing will not be automatically started by the proxy, for numerous reasons. I *do* believe that we will need to designate a ‘server’ for a lack of better terminology; the server will be the source that the other proxies use as a basis for updating, and it will be responsible for synchronization and serialization.

Connecting:

User will have Datasharing turned on or off, and they will also have it set to client or server.

Client:

When the user loads DROP and finally enters into the game then I think it would be best for DROP to Automatically attempt a connection to the last known IP of the Data server. If connection can not be made or if the last known IP is invalid (ie never used) then the client should Request the IP of the Data server via SS. The Server should reply the ip address via Hail. If no reply is given for a server then the broadcast will wait for a given time (60 seconds?) and then give a subsequent request. This should happen 5 times and if no response is given then the request move to a longer wait period (30 min?). If a disconnection happens between the Client and the Server then the Initial connection process should repeat.

I believe this is a good strategy, although I think that the actual start of the requesting for server ip should be manual. I think that the basic setup should as such: one person starts up and puts it in server mode, then everyone else starts theirs in client mode; when each client is started, it requests for a server and the server responds with it’s ip and port. The clients then connect, and the server manages how it communicates.

Server

Once DROP makes it into the game it will send out a request for a server, Similar to the client. The reason for this will be to make sure that there are no active servers running. Only 1 server should be running as it will be the unit PULLING most of the data.

After a given time (15 sec?) If no response has been given that there is a server running then the server will then be allowed to cut on. When the server first comes active it will broadcast it's launch over SS and begin to wait for connection request info to be broadcast.

I think this should be manually started

If during the 30 second wait a server Sends a reply, Or a message stating that a server has loaded then the server will stop trying to load and it will move into a client mode.

Once connected, Rather than using IP permission I think that we could use a password permission system. I only think this is a good idea because It is possible fore IP's to change, and dialup users of course would not be allowed to reconnect to the system until there is a new permission set on one of the connections. The password could be something as simple as the corps SS. Using a game recorded variable it would stop the user from needing to input the password into an option box. I think it would be nice to also have the option of setting a perm password in a box and setting the connection to use that password every time. The Server itself will have the same/similar options. There will be an option to set the data sharing module into either Server or Client mode.

This is probably a good idea

After connection has been made, and permission has been granted (via password?) then the time will need to be placed in sync. To do this I think the simple option is going to be using something similar to MIRC..

Send the word PING in plain text to the host, and the host will return the world PONG. With the PONG reply there will also be included a timestamp of the host in the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time.

The server will make adjustments for this time difference, ie, if the host is 840 seconds ahead then the server will change all incoming timestamps to remove ( 840 + ( .5 \* ping )). Also when the server sends data it will make this same adjustment of adding the ( 840 + ( .5 \* ping )). This should mean that no time changes need to be set in any server, and the data will still be in sync.

I think this is an acceptable way of syncing time

Once this is done then the data can start to be exchanged.

Before data can be sent it will need to be requested. The server should have virtually all data updated from all people. The only data that should not be sent from client to server is SubSpace information UNLESS requested, I only say that cause it will stop all users from sending the same data over and over to the server.

I agree that data normally will be on a per request basis, however initial connection will probably have to be an exception.

Data sent to the server will have the format of ..

(ESC)(ESC)(Type of values being

sent)":"Sector":"Timestamp(ESC)(Value)(Tab)(Value)(Tab) .. repeated ..

(ESC)(TAB)(ESC)

IE, you'll never send more than 1 sectors data in one line (I am assuming here that DROP will also be a step ahead of TWX/Swath/Tw-Attac and timestamp everything, including last time fig was there, and dscans.

The "Type of values being" is the chart to tell DROP how to parse the info

So I was thinking like

0000 0000 0000 0000

Compressing down to 8 chars.. Good or no?

Sector

0000

View Timestamp

Dscan Timestamp

Navhaz

Fig amount

0000

Fig Owner

Fig Type

Anom

Armids

0000

Armid Owner

Limpets

Port amount:amt:amt

Port%:%:%

0000

# of warps

Port Timestamp

Density

So something like 0001 0000 0000 0000

Would ask for the fig amount of each sector to be sent when seen?..

I think that a message structure is definitely preferable to a state machine. I think the message structure should be flexible in size, and use a checksum to verify message integrity. This is so that any type of message can be sent to/from a database. I see no reason why we can’t use doc’s message structure as a portion of the message object. Keep in mind that I want to use this communication protocol for *any* type of communication. This means proxy-proxy chats as well, and so it will require an interrupt messaging architecture as well as a request methodology

So this is basically the type of pattern I think might work for sending the data.. but for the INITIAL database update I think it should send all of the timestamps to the user at once and then the user only returns the relevant information (with of course the header showing what's updated).. then after the initial update is done then users can request only specific data.. ie someone can send a request for 0000 0000 0011 0000 and only new ports will be sent.. nothing else..

Well shrug

That's my idea lol.. Tell me what you want and I'll try to do it.. I can probably put this in some kind o fscript if you want but I doubt it'll be anything like c#

I think doc has a good start on *how* it should happen – ie the general way the proxies should expect one another to behave. I think we need to delve a bit further into a messaging structure, as well as figure out more explicitly how clients and servers are going to update themselves. There could be a real synchronization issue if we don’t think thoroughly about this.