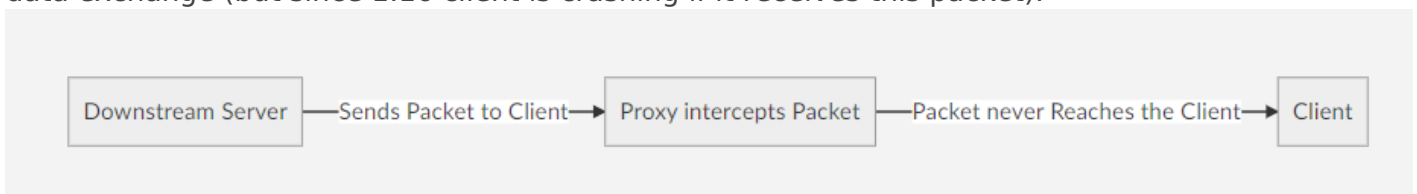# Proxy Communication

Its common that bigger networks requires some synchronization and communication between downstream servers and proxy. There are different ways how communication can be implemented.

Plugin Messages

This type of communication is not supported in WDPE. But understanding how this method works may be useful.

The downstream server sends a packet thought client connection to the proxy. Proxy will then intercept the packet and handle it as it likes, parsing the payload to get request data or whatever the downstream server sent. In specific situation proxy could send packet to client as an additional data exchange (but since 1.16 client is crashing if it receives this packet).

More about plugin messages can be found on Tobias's gist.

Custom socket communication

Creating extra connection between downstreams and proxy can be done thought TCP/UDP sockets. Usually this is the most effective way of data synchronization.

If you are looking for socket solution we recommend to check StarGate, a project developed by one of the WaterdogPE developers, which allows exactly such communication. You can get support with this project on our Discord server.

There are more architectures how could communication be done:

Proxy-Server

This is the simpler implementation which is usually good enough for all networks. In this architecture proxy acts as server for downstream socket clients - manages connections, forwards data between clients. All clients are connected to specific proxy.

Pros:

- Easy data exchange between proxy and downstream.
- Handled data can be easily proceed by proxy itself.

Cons:

- Proxy acts as the master. If proxy went down downstream clients will be disconnected.
- Implementation of custom clients which would be part of the backend system may be harder.
- Using multiple proxy instances would mean multiple open downstream client connections.

Backend-Server

Networks which are using more proxies or needs synchronization with backend system would probably prefer this architecture. Custom application running in backend acts as server. Proxy and downstream servers are in this clients to the application. This application (server) handles received data from downstream client, process it and if needed sends to destination proxy.

Pros:

- Ability to communicate with multiple proxies using one downstream client.
- Ability to communicate between proxies.
- Proxy can be terminated without disconnecting any of downstream clients.

Cons:

- Maintenance of whole stack may be harder.
- Communication between downstream and proxy may be bit slower.

Not recommended solutions

Due to lack of knowledge many users tend to use "poor" solutions which will do their job. Periodically scanning database table, querying results isn't the correct way how to exchange data between downstreams and proxy. We do not recommend any of this methods:

- Using Minecraft query methods to get information from downstreams.
- Any response system based on SQL or database queries.
- Shared file databases (and SQLite) between proxy and downstreams

If you really want to use any of this non-recommended methods consider using Redis instead of file, database or sockets solution.

Revision #7

Created Sun, Nov 29, 2020 3:23 PM by Alemiz

Updated Sun, Oct 30, 2022 7:52 AM by Alemiz