Basic UDP Bridge v1.0.0

System Nodes

Server-tunnel:

This is the UDP bridge as such, it is executed like next example:
`$node ./tunnel/app.js -s 0.0.0.0:5000 -c 0.0.0.0:4000 -t 500 -e 100`

- Where:
  -s: means server-tunnel endpoint to listening to **s**treamer, default value if omit `0.0.0.0:5000`.

  -c: means server-tunnel endpoint to listening to **c**lient, default value if omit `0.0.0.0:4000`.

  -t: means **t**ime to expire cache data in miliseconds, default value if omit `500 ms`.

  -e: means interval time to s**e**nd all cache available to clients connected, default if omit `100 ms`.

Client:

This is the client which will get the cache data from server, execute as:
`$node ./client/app.js -s 127.0.0.1:4000 -k 30 -i 1234`.

- Where:
  -s: means **s**erver-tunnel endpoint which is listening to the client, especial attention to port parameter, it has to be the same as the tunnel has for this client, default if omit `127.0.0.1:4000`.

  -k: means time interval in miliseconds to send **k**eepalive message to the tunnel, default if omit `1000 ms`.

  -i: means a unique **i**d amount all client connected to the server-tunnel, it is a way to identify the client no matter if he changed its endpoint through NAT/PAT, it could be any string.

Streamer:

This is a mock streamer, used to test but can be used in production if you want, example:
`$node ./streamer/app.js -s 127.0.0.1:5000 -t 100`

- Where:

  -s: means **s**erver-tunnel endpoint which is listening to the streamer, especial attention to port parameter, it has to be the same as the tunnel has for the streamer, default if omit `127.0.0.1:5000`.

  -t: means **t**ime interval to send mocked cache data to server, defautl if omit `100 ms`.

Quick localhost running:

As can be noted default values match with each others, so only execute following commands in differents terminal and enjoy:

```
$node ./tunnel/app.js
$node ./client/app.js
$node ./streamer/app.js
```

Notes:

- Feel free to modify, use and improve this first approach.
- Currently the client is sending a json `{id: [-id parameter], data: "k"}` as keepalive message and the mock streamer is sending the string `"Hello"` as cahce info.
- There is no a real concept of connection or server-client protocol in UDP-style sockets, in this application the clients connected will be concidered who have sent a keepalive before, it'll be identified inside the server-tunnel by the id set in scripts parameters, if there are ids repeated the last keepalive will be got as current connected client for that id, be sure set diferents id parameters for diferents clients.
- All line in the project preceded by a `"//DEBUG"` comment can be deleted or commented.