

UDP Multithreaded Client/Server Design Document

Note: The code and (in-line) documentation for this project can also be found at the following GitHub link: <https://github.com/ariblack17/simple-server>.

Description

The Server: The server opens a serverSocket on a non-reserved port (13000), then waits for wildcard queries from clients. Each client is handled on a separate thread (up to 5 threads, supporting up to 5 clients in parallel), and each client is able to send multiple queries to the server for multiple responses.

The wildcard query supported in this server follows the same format as in the TCP server.

The Client: The client opens a clientSocket selected by the operating system, sends a wildcard query request to the server, then receives and unpacks a response from the server. This process of sending and receiving messages repeats until the client terminates communications.

Application Protocol: The protocol for this UDP server is the same as in the TCP server, except the Connection-type header in client requests has been renamed to Request-type header.

Trade Offs

The main trade off for the UDP multithreaded server matches the *Program Structure* trade off listed in the TCP document. Developing an object-oriented design in favor of the selected design would have resulted in a different (but practically functionally-identical) set of code.

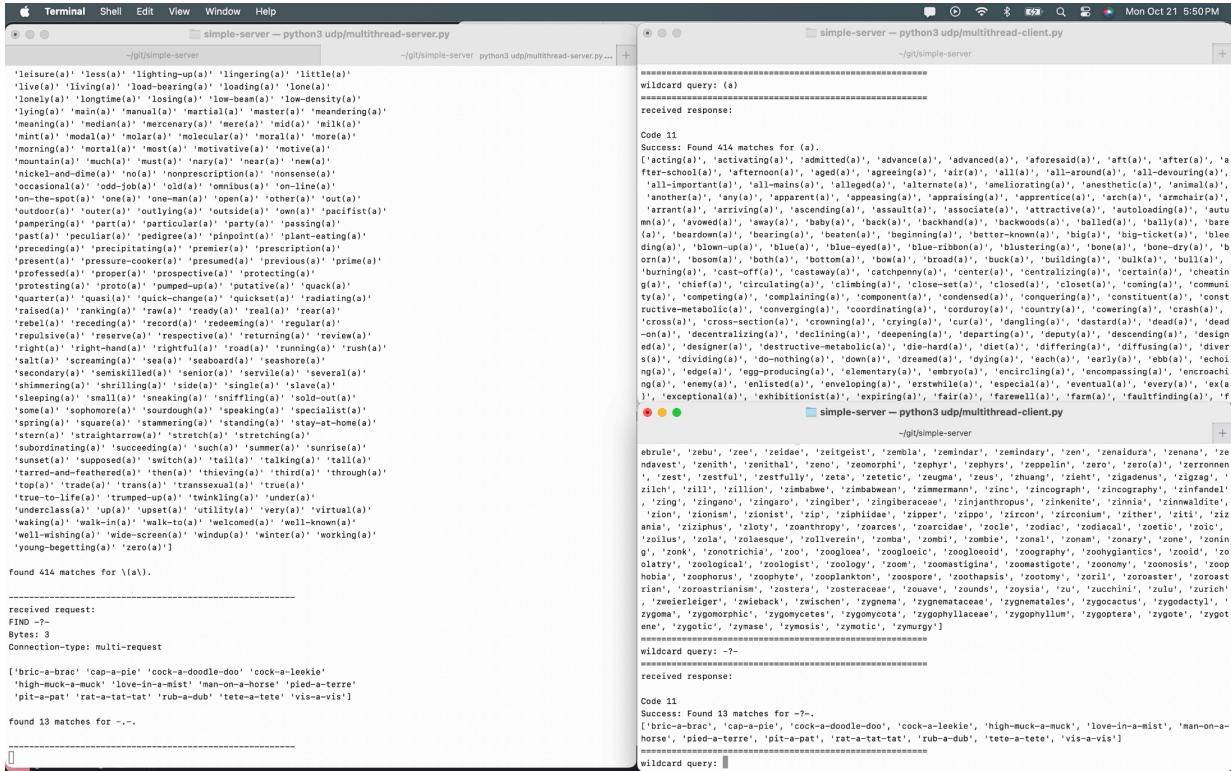
Extensions

The main extensions for this UDP multithreaded server match the *Improved Structure and Optimization* and *Generating Better “UI” for the Client* extensions listed in the TCP document.

Identifying Source/Destination Addresses: A known requirement for UDP implementation in client/server design is the accession of packet source and destinations in order to correctly route application data to processes. While this is already taken care of

(a) | -?-

Test server response for a query including special characters (parentheses and dashes).



The screenshot shows two terminal windows side-by-side. The left window is titled 'simple-server -- python3 udp/multithread-server.py' and the right window is titled 'simple-server -- python3 udp/multithread-client.py'. Both windows have a title bar with icons and a status bar at the bottom.

Code 11
Success: Found 14 matches for (a).
[...]
wildcard query: (a)
=====
received response:

Code 12
Success: Found 414 matches for (a).
[...]
wildcard query: (a)
=====
received response:

Code 13
Success: Found 13 matches for -?-.
[...]
wildcard query: -?-
=====
received response:

In the left terminal, the user has run a command to find matches for '(a)' and another to find matches for '-?-'. The results are displayed in large blocks of text. In the right terminal, the user has run a wildcard query for '(a)' and a command to receive the response from the server, which is also shown in a large block of text. The server's responses are identical to the ones in the left terminal.

quit

Test server response for client quit, or termination of communications.

```
simple-server -- python3 udp/multithread-server.py > /dev/null
simple-server -- zsh

~/git/simple-server ~/git/simple-server python3 udp/multithread-server.py > /dev/null
simple-server -- zsh

-tarred-and-feathered() 'then(a)' 'thieving(a)' 'third(a)' 'through(a)'
'topia)' 'trad(a)' 'trans(a)' 'transsexual(a)' 'true(a)'
'true-to-life(a)' 'trumped-up(a)' 'twinkling(a)' 'under(a)'
'underclass(a)' 'up(a)' 'upset(a)' 'utility(a)' 'very(a)' 'virtual(a)'
>waking(a)' 'walk-in(a)' 'walk-to(a)' 'welcomed(a)' 'well-known(a)'
'well-wishing(a)' 'wide-screen(a)' 'windup(a)' 'winter(a)' 'working(a)'
'young-begetting(a)' 'zero(a)'

found 414 matches for \(\a\).

-----
received request:
FIND ~?
Bytes: 3
Connection-type: multi-request

['bric-a-brac' 'cap-a-pie' 'cock-a-doodle-doo' 'cock-a-leekie'
'high-muck-a-muck' 'love-in-a-mist' 'man-on-a-horse' 'pied-a-terre'
'pit-a-pat' 'rat-a-tat-tat' 'rub-a-dub' 'tete-a-tete' 'vis-a-vis']

found 13 matches for ~-.

-----
received request:
FIND a??
Bytes: 4
Connection-type: multi-request

['abate' 'abatement' 'abating' ... 'zealot' 'zealotry' 'zooplankton']

found 1546 matches for a..t.

-----
received request:
FIND hello
Bytes: 5
Connection-type: multi-request

['hello' 'othello' 'othellos' 'phellodendron' 'seychellois']

found 5 matches for hello.

-----
received request:
FIND quit
Bytes: 4
Connection-type: multi-request

-----
received request:
FIND quit
Bytes: 4
Connection-type: multi-request

-----
```

```
simple-server -- zsh
simple-server -- zsh

hobis', 'zoophorus', 'zoophyte', 'zooplankton', 'zooptosis', 'zootomy', 'zoril', 'zoroaster', 'zorastrian', 'zoroastrianism', 'zostera', 'zosteraceae', 'zouave', 'zounds', 'zoyzia', 'zu', 'zucchini', 'zulu', 'zurich', 'zweierleiger', 'zweiback', 'zwischen', 'zygema', 'zygennatales', 'zygocactus', 'zygodactyl', 'zygoma', 'zygomorphic', 'zygomycetes', 'zygophyllaceae', 'zygophyllum', 'zygoptera', 'zygote', 'zygotene', 'zygotinic', 'zymase', 'zymotic', 'zymurgy'

-----
wildcard query: ~-
=====
ariblack@owlis: simple-server %
```

```
simple-server -- zsh
simple-server -- zsh

Code 11
Success: Found 13 matches for ~-
(['bric-a-brac', 'cap-a-pie', 'cock-a-doodle-doo', 'cock-a-leekie', 'high-muck-a-muck', 'love-in-a-mist', 'man-on-a-horse', 'pied-a-terre', 'pit-a-pat', 'rat-a-tat-tat', 'rub-a-dub', 'tete-a-tete', 'vis-a-vis'])

-----
wildcard query: hello
=====
ariblack@owlis: simple-server %
```

```
simple-server -- zsh
simple-server -- zsh

Code 11
Success: Found 5 matches for hello.
(['hello', 'othello', 'othellos', 'phellodendron', 'seychellois'])

-----
wildcard query: quit
=====
ariblack@owlis: simple-server %
```

00

Test server response for a query that has no matches.

```
Terminal Shell Edit View Window Help
~/git/simple-server -- python3 udp/multithread-server.py
ariblack@owls simple-server % python3 udp/multithread-server.py
=====
server started successfully!
=====
received request:
PING _*
Bytes: 2
Connection-type: multi-request
[]
found 0 matches for *_.

-----
simple-server -- python3 udp/multithread-client.py
ariblack@owls simple-server % python3 udp/multithread-client.py
=====
wildcard query: _*
=====
received response:
=====
Code 0
No matches found
[]
=====
wildcard query: _*



simple-server -- python3 udp/multithread-client.py
ariblack@owls simple-server % python3 udp/multithread-client.py
=====
wildcard query: _*



Last login: Mon Oct 21 14:56:22 on ttys016
ariblack@owls simple-server % python3 udp/multithread-client.py
=====
wildcard query: _*
```

Known Issues: The major known issue with the UDP client/server programs is similar to the final issue listed in the TCP design document – sending non-ASCII characters may cause an error in the server (but ?, -, and () are okay).

Resources

The code used in the *dispatcher()* method in *udp/multithread-server.py* is heavily influenced by some sample code accessible at the following link:

<https://pythontic.com/socketserver/threadingudpserver/introduction>. Specifically, the code structure for iterating through each thread, appending the thread to a threads list, starting the threads process, then joining all threads immediately afterward was adopted from the listed source.

The remainder of the code (and all documentation) throughout the project was developed independently.