

# IERG4180 Network Software Design and Programming

## Project 3 Report

Name: Wang ZiFeng

SID: 1155194663

GitHub Repository: <https://github.com/Catalpa1maple/IERG4180-Project>

Requirement: C++11 and ws2\_32.lib (for windows)

### Feature:

- (a) Constructed a class Threadpool to handle monitoring, growing and shrinking the pool in Threadspool.h within Project 3

```
maplewong@Maples-Mac Project_3 % ./NetProbeServer
Server is listening on port 4180
Elapsed 1s ThreadPool 1/8 TCP Clients 0 UDP Clients 1
Pool size decreased to 4
□
Pool size increased to 8
```

- (b) As below

(c)

```
maplewong@Maples-Mac Project_3 % ./NetProbeClient -response
Mode: RESPONSE
Stat: 500 ms
Remote Host: localhost
Remote Port: 4180
Protocol: UDP
Packet Size: 1000 bytes
Packet Rate: 1000 bytes/second
Packet Number: 1000000000
Send Buffer Size: 0 bytes
Receive Buffer Size: 0 bytes
Presistent: No
Elapsed 9.41s Replies 94 Min 0.01ms Max 2.98ms Avg 0.20ms Jitter 7.23ms
```

The detail of (b) and (c) feature will be discussed later in exp part.

- (d) Implement a pre-socket TCP congestion control with <netinet/tcp.h> library  
As shown code snippet below.

```
if(!tcpcc.empty()){
    if(setsockopt(TCP_Trk_Socket, IPPROTO_TCP, TCP_CONGESTION, tcpcc.c_str(), tcpcc.length()) < 0) {
        cerr << "Failed to set TCP congestion control algorithm: " << strerror(errno) << endl;
    }
}
```

## Experiment

(1)

	Throughput	loss	CPU
Linux	~200000Mbps	~0%(very few)	~89%
Windows	~150000Mbps	~5%	~14%

Compared with Project\_2 which concurrency server without threadpool while the project 3 program is way more efficient therefore obviously the throughput increase around 20%.

(2)

\Client proto	1	2	5	10	30
TCP	~200000Mbps	~200000Mbps	~200000Mbps	~180000Mbps	~160000Mbps
UDP	~300000Mbps	~300000Mbps	~300000Mbps	~280000Mbps	~270000Mbps

(3)

	Throughput (1 Client)	20 Clients
Non-persistent	~200000Mbps	~160000Mbps
Persistent	~200000Mbps	~190000Mbps

As Non-persistent mode costs extra time on setting up connection therefore by accumulating and gradually Persistent one will handle more requests.

(4)

Cubic: Has higher throughput and performs better under low-loss network

BBR: Recovers faster from packet loss event and maintains higher throughput in lossy network

In conclude TCP BBR is better in lossy network (random packet loss rate > 0) as it recovers soon and sustainability.