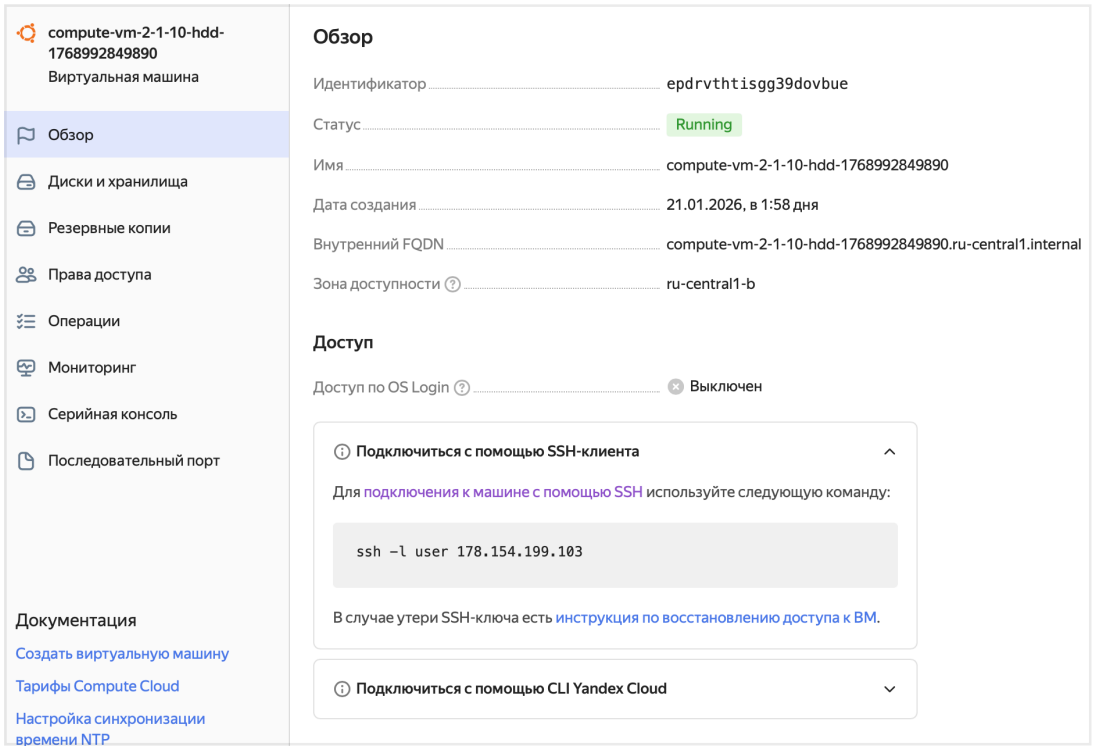
Computer Networks Task1

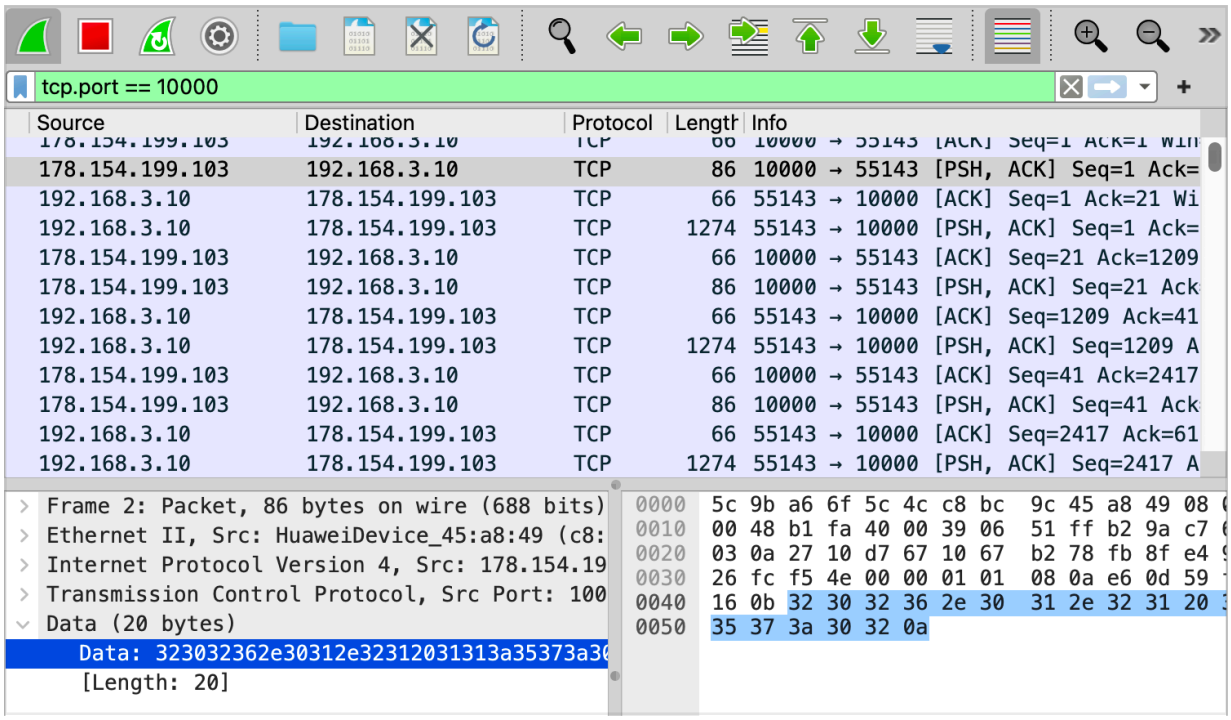
Frolov Ivan
Note: the file was automatically converted to a PDF. Sorry if the formatting is not perfect

Here are some screenshots on running a VM
Running a Yandex VM

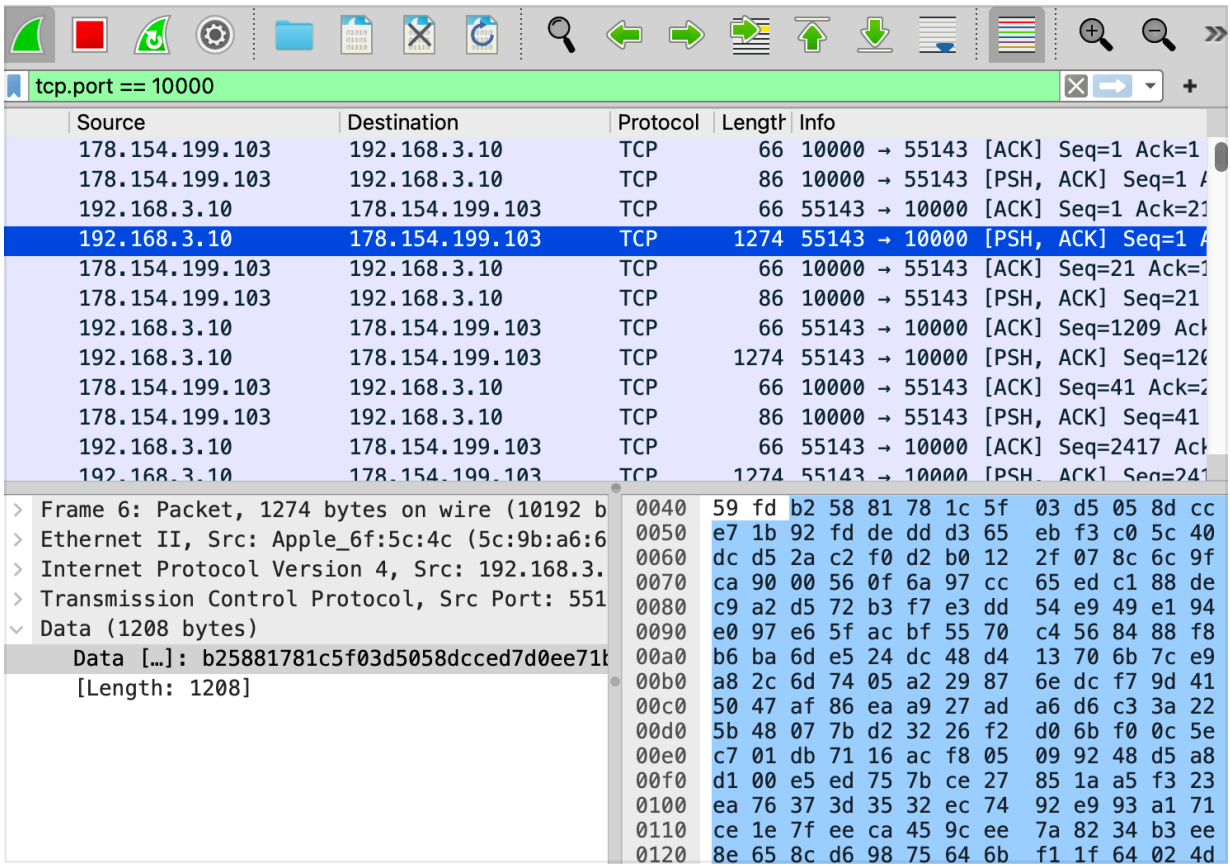


Server IP: 178.154.199.103
Client IP (started locally): 192.168.3.10

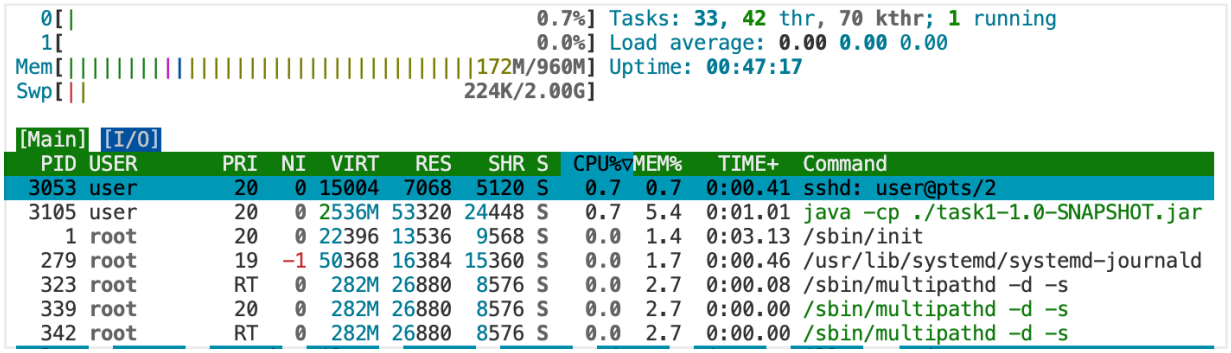
Server sends to a client



Client sends to a server



The process is not very heavy (using htop)



Note: at first my server was breaking
That's what I mean:
It started reading bytes from client normally
Then the messages started arriving very very fast
And finally the server just locked and didn't receive anything
Just hanging like this

The way I discovered to fix it was to send to the server not only the bytes array

with data but also the size of that buffer

```
ByteBuffer bb = ByteBuffer.allocate(4);
                    bb.putInt(dataLength);
                    out.write(bb.array());
                    out.write(dataToSend);
```

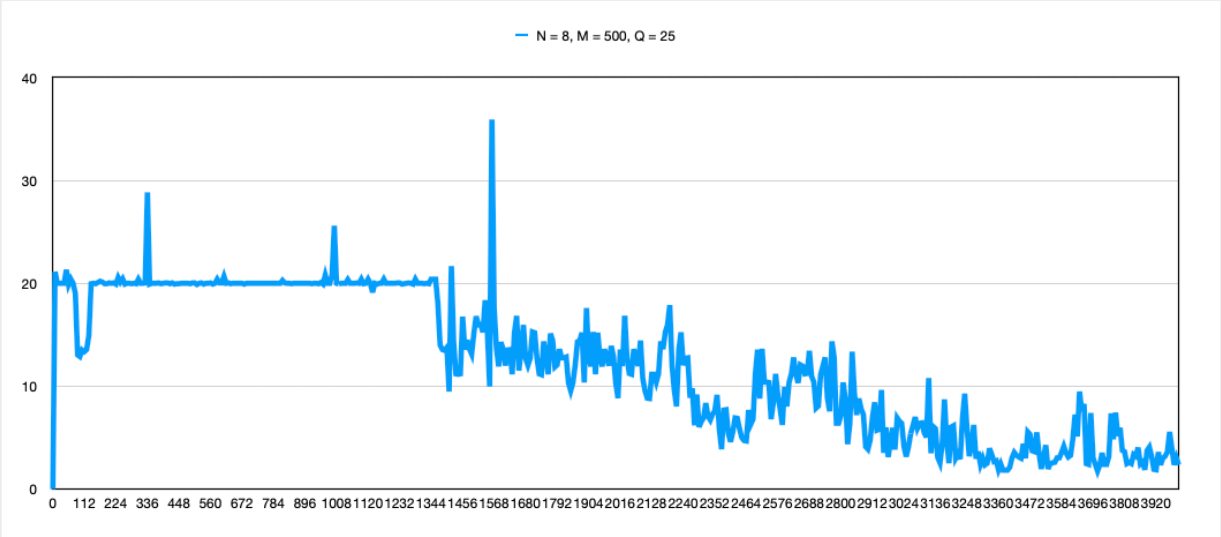
And then the server will read exactly the size of the buffer

That was crazy to fix so hopefully it works correctly

Note: I'll be using N = 500 or less
N = 5000 is incredibly slow
I just runs forever

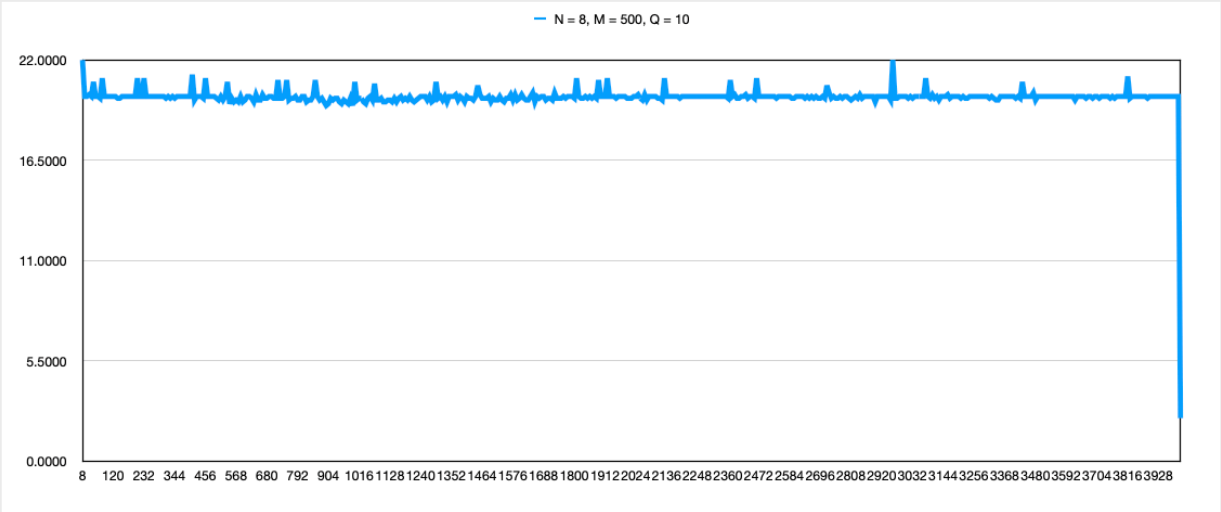
Local test
No sure it it is relevant
But the time depends on the array size almost linearly

N=8, M=500, Q=25



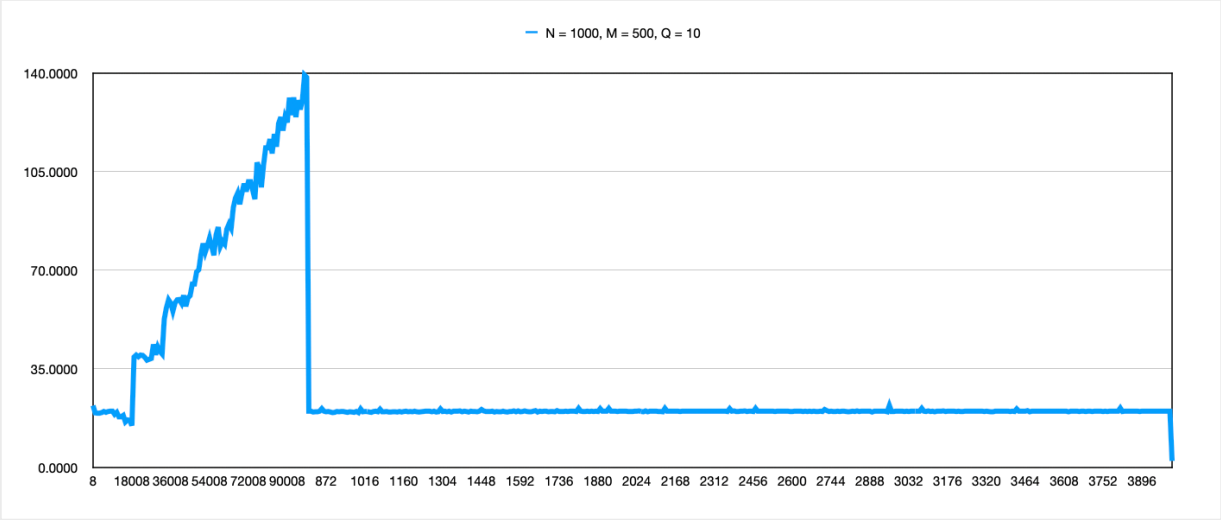
Now let's do the VM

N = 8, M = 500, Q = 10



Our first test
With low N the speed pretty much stays the same
The increases in array sizes have almost no influence on speed

N = 1000, M = 500 , Q = 10

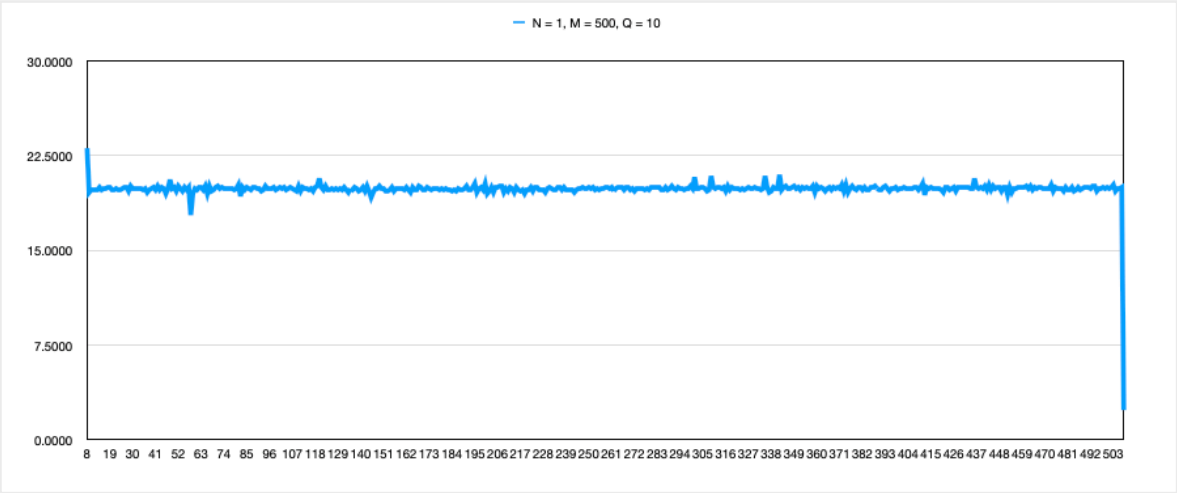


When using N = 1000 it got much slower
So using M = 100 iterations here

We can see that the time really increases as the array size grows
And then it hits a threshold and drops, staying low
Honestly that’s quite interesting

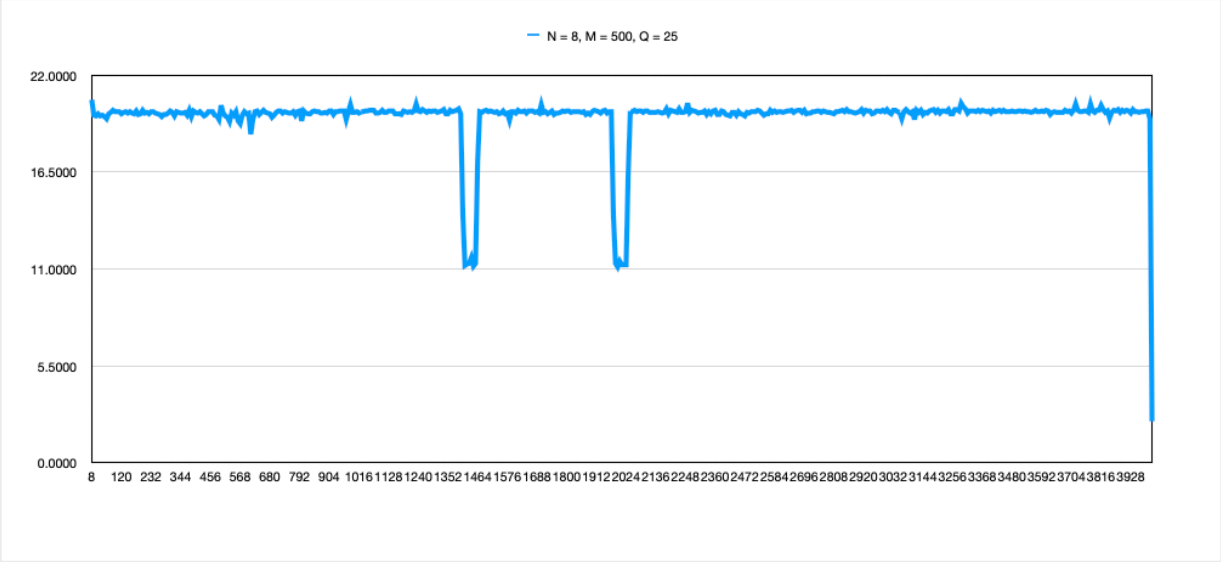
I’ve read it is related to header overhead
So as the array size gets large enough we can send them as full packets which is more efficient

N=1, M=500, Q=10



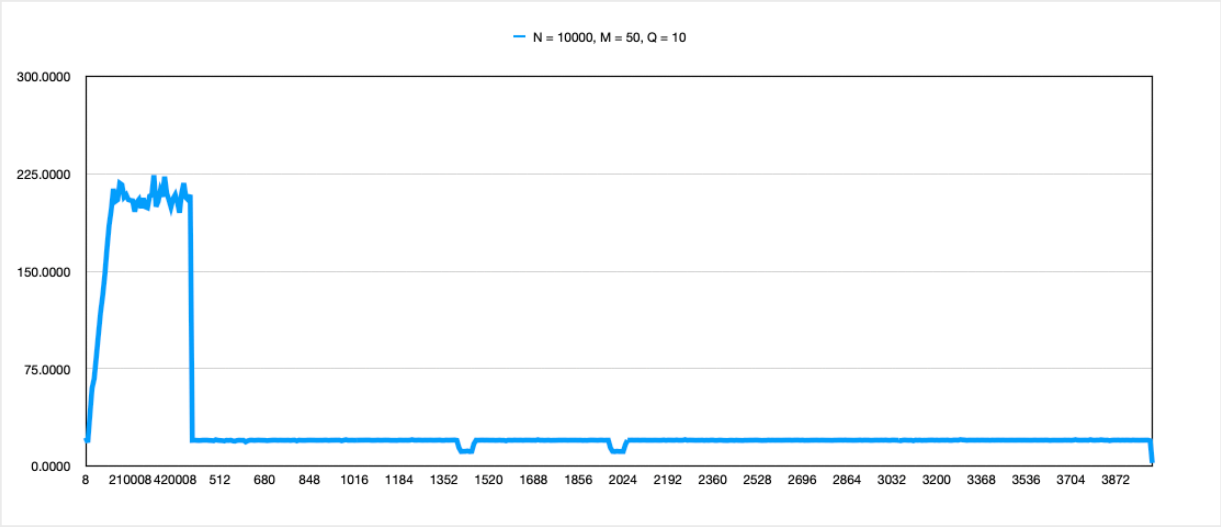
Here we will see if making N = 1 (small sizes) improves the speed
It is actually pretty much the same as N = 8

N=8, M=500, Q=25



Here we will see what difference increasing Q makes
It is similar to our first VM graph - stable time across the array sizes
And we see two drops - probably some Java / Network optimizations

N=10,000, M=50, Q=10



I was really interested to see if sending super big array sizes influences the speed
Making N=10,000
We are running just for about 50 iterations so that I don't wait for a week)
Looking at the logs in CLI, it get's much slower (at least feels like that)

Looking at the graph, we see similar picture to N=1000
Yet it seems like at first the speed is not optimized at all is consistently very high - about 225,000
And then it gets optimized to the level of N=1000
Maybe it is because the longer the connection runs the more optimized it gets
So that's cool