Testarea și evaluarea performanței protocoalelor TCP, UDP și QUIC pentru transferul de date

Introducere

Acest raport prezintă analiza și comparația dintre protocoalele TCP, UDP și QUIC în transferul de date între un client și un server. Sunt urmărite și evaluate timpul de transmisie a datelor, rata de pierdere a datelor și impactul mărimii mesajelor asupra performanței generale.

Acest experiment a fost realizat utilizând implementări pentru client și server scrise în Python și testate într-un mediu WSL Ubuntu.

Mod de utilizare

Pentru implementarea TCP și UDP sunt utilizate script-urile **client.py** și **server.py** ce pot fi apelate cu următorii parametri variabili:
client.py:

-protocol: protocolul folosit pentru transmiterea datelor poate fi: "tcp" sau "udp"

-mod de transmisie: tcp folosește automat modul streaming("st"), pentru UDP modul poate fi specificat: streaming("st") sau stop-and-wait("sw")

-adresa IP a serverului: ex: "127.0.0.1"

-dimensiunea blocului de date: un număr între 1 și 65536 ce reprezintă numărul de octeți ce vor fi trimiși într-un mesaj

-dimensiunea totală a datelor trimise: poate fi "500MB" sau "1GB"

Exemplu pentru client.py:
python3 client.py udp st 127.0.0.1 1024 1GB

server.py:

-protocol: protocolul folosit pentru transmiterea datelor poate fi: "tcp" sau "udp"

-mod de transmisie: tcp folosește automat modul streaming("st"), pentru UDP modul poate fi specificat: streaming("st") sau stop-and-wait("sw")

Exemplu pentru server.py:
python3 server.py udp st

Pentru implementarea QUIC sunt utilizate script-urile **client_quic.py** și **server_quic.py** ce pot fi apelate cu următorii parametri variabili:

client_quic.py:

-mod de transmisie: streaming("st") sau stop-and-wait("sw")

-adresa IP a serverului: ex: "127.0.0.1"

-dimensiunea blocului de date: un număr între 1 și 65536 ce reprezintă numărul de octeți ce vor fi trimiși într-un mesaj

-dimensiunea totală a datelor trimise: poate fi "500MB" sau "1GB"

Exemplu pentru client_quic.py:
python3 client_quic.py udp st 127.0.0.1 1024 1GB

Pentru utilizarea server-ului va fi necesară generarea cheii de criptare a stream-urilor:
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes -subj "/CN=localhost"

server_quic.py:

-mod de transmisie: streaming("st") sau stop-and-wait("sw")

-certificatul creat:

-cheia privată generată:

Pentru comunicarea dintre server și client este utilizat automat portul 5001.

Analiza rezultatelor

Protocol	Marime mesaj	Marime totala	Timp	Data Loss
TCP Streaming	1	500MB	529.16 ~ 9 min	0
TCP Streaming	1024	500MB	0.60	0
TCP Streaming	10000	500MB	0.14	0
TCP Streaming	65535	500MB	0.07	0
TCP Streaming	1	1GB	1157.76 ~ 19 min	0
TCP Streaming	1024	1GB	1.29	0
TCP Streaming	10000	1GB	0.23	0
TCP Streaming	65535	1GB	0.15	0
UDP Streaming	1	500MB	916.22 ~ 15 min	0
UDP Stop and Wait	1	500MB	-	-
UDP Streaming	1024	500MB	1.04	0.0154%
UDP Stop and Wait	1024	500MB	30.82	0
UDP Streaming	10000	500MB	0.13	2.74%
UDP Stop and Wait	10000	500MB	3.74	0
UDP Streaming	65507	500MB	0.05	0.36%
UDP Stop and Wait	65507	500MB	0.63	0
UDP Streaming	1	1GB	-	-

UDP Stop and Wait	1	1GB	-	-
UDP Streaming	1024	1GB	2.20	0
UDP Stop and Wait	1024	1GB	69.12	0
UDP Streaming	10000	1GB	0.29	0.7%
UDP Stop and Wait	10000	1GB	5.30	0
UDP Streaming	65507	1GB	0.10	0.8%
UDP Stop and Wait	65507	1GB	1.30	0
QUIC Streaming	1	500MB	-	-
QUIC Stop and Wait	1	500MB	-	-
QUIC Streaming	1024	500MB	23.63	23.73%
QUIC Stop and Wait	1024	500MB	152.18 ~ 2.5 min	0
QUIC Streaming	10000	500MB	0.97	91.81%
QUIC Stop and Wait	10000	500MB	46.43	0
QUIC Streaming	65535	500MB	1.56	96.93%
QUIC Stop and Wait	65535	500MB	32.11	0
QUIC Streaming	1	1GB	-	-
QUIC Stop and Wait	1	1GB	-	-
QUIC Streaming	1024	1GB	55.81	0.35%
QUIC Stop and Wait	1024	1GB	325.17	0
QUIC	10000	1GB	7.70	88.55%

Streaming				
QUIC Stop and Wait	10000	1GB	99.04	0
QUIC Streaming	65535	1GB	2.92	97.14%
QUIC Stop and Wait	65535	1GB	70.35	0

În general, mărimea mesajelor trimise joacă un rol important în influențarea timpului de transmitere a datelor. Rezultatele înregistrate arată o scădere a timpului de transmitere o dată cu creșterea dimensiunii mesajelor indiferent de protocolul utilizat, acest lucru apare din cauza numărului ridicat de operații de trimitere.

Din punctul de vedere al pierderii de date în timpul trimiterii protocoalele analizate și tipurile de transmitere produc rezultate variate. TCP garantează livrarea completă a datelor în toate scenariile, indiferent de mărimea mesajelor. Toți octeții trimiși ajung la destinație datorită mecanismelor de retransmisie și corectare a erorilor.

UDP, în modul streaming, oferă transmisia foarte rapidă a datelor însă pot apărea mici pierderi de date ce se amplifică o dată cu creșterea mărimii mesajelor trimise. În modul stop and wait, UDP asigură transmiterea integrală a datelor, fară pierderi, însă acest tip de transmisie crește semnificativ timpul de transmisie demonstrând un compromis între viteză și fiabilitate.

În modul stop and wait QUIC asigură livrarea completă a datelor, dar, la fel ca în cazul protocolului UDP fiabilitatea vine cu costul unui timp de transmitere ridicat.

În modul streaming, însă, QUIC arată pierderi semnificative pentru mesaje mari – de exemplu, doar aproximativ 76% din datele trimise sunt primite pentru mesaje de 1024 octeți la 500MB și chiar mai puțin pentru mesaje de 10.000 sau 65.535 octeți. Aceasta indică probleme potențiale în modul de gestionare internă a fluxului sau în modul în care sunt procesate datele la nivelul implementării. O problemă identificată este trimiterea rapidă a datelor de către client spre server, fără ca acesta să aibă suficient timp să le proceseze

în totalitate. O potențială soluție este întreruperea clientului între terminarea trimiterii datelor și terminarea conexiunii, timp de 60s.

Concluzii finale

În concluzie, rezultatele experimentale evidențiază un compromis clar între viteză și fiabilitate:

- TCP garantează livrarea completă, dar poate fi extrem de ineficient cu mesaje mici.
- UDP streaming oferă viteză maximă, dar cu riscuri minore de pierdere a datelor, în timp ce stop and wait asigură 100% livrare, la un cost de latență mai mare.
- QUIC, deși are potențialul de a combina avantajele ambelor, prin gestionarea mai multor stream-uri pe o singură conexiune, mecanisme complexe de control al fluxului și congestionării și latență scazută la realizarea conexiunii prin zero-round-trip-time, necesită optimizări suplimentare pentru a obține performanțe excelente în regim de streaming.