Transportprotokoll för multiplayerspel: Jämförelse mellan UDP & TCP

Maximilian Schön & Emanuel Eriksson

***Abstract*— En implementation av Pong har används som testmiljö för att analysera hur olika typer av transportprotokoll över olika typer av medium kan påverka prestanda i spel. Enligt insamlad testdata visar sig UDP vara överlägsen i alla fall förutom det mest stabila, direktkopplad länk med en tvinnad parkabel, där UDP fortfarande är bäst men med mindre marginal.**

I. Introduktion

Idag är "online multiplayer"-spel ett mycket populärt sätt att fördriva tiden. Men i takt med att fler och fler väljer att spela online har kraven på nätverksprestanda ökat fort, då många moderna spel kräver att hundratals spelare ska kunna kommunicera med en spelserver samtidigt. Mängden data som skickas har också ökat tillsammans med spelens komplexitet. Spelare har även mycket höga förväntningar på de produkter de investerar både pengar och tid i, och därför är det viktigt att spelens nätverksprestanda inte halkar efter. Ett enkelt sätt att sammanfatta dessa förväntningar är att uppkopplingen emellan spelet och spelaren inte ska drabba själva spelupplevelsen.

En term som ofta kastas runt i diskussioner av detta slag är överföringsprotokoll, mer specifikt valet mellan TCP eller UDP. Eftersom överföringsprotokoll på många sätt kan ses som grunden för nätverkskommunikation är det viktigt att använda rätt typ av protokoll, som kan leverera data till många spelare på ett pålitligt sätt. Det är även viktigt att känna till skillnaden mellan dessa protokoll, då de skiljer sig i grunden. Valet beror mycket på vilket typ av spel som ska använda protokollet.

Denna rapport ämnar undersöka hur valet av överförings-protokoll påverkar upplevelsen av ett online multiplayerspel. Men istället för att utgå ifrån ett existerande spel, där det kan vara utmanande att urskilja vilken skickad data som är vilken, används istället ett egenskrivet multiplayer spel, Pong, tillsammans med snifferprogrammet Wireshark. Detta beslut skapar möjligheten att exakt veta hur data skickas i koden, och det blir även lättare att isolera data i Wireshark.

II. Teori

A. TCP, Transmission Control Protocol

IP, Internet Protocol, är ett *best effort*-protokoll utan någon förbindelse i benämningen att det inte garanterar att någon data kommer fram, att det är rätt data som kommer fram eller att den kommer fram i rätt ordning. Dessutom skickar IP data utan att känna till någonting om mottagaren. För att förbättra dessa brister har protokollet TCP skapats. TCP är ett förbindelseorienterat protokoll på Lager 4 (Transportlagret) som garanterar att felfria paket kommer fram till mottagaren i rätt ordning.

TCP skapar en förbindelse mellan två applikationer genom en så kallad handskakningsprecedur. I slutet av denna process kan båda datorerna skicka och ta emot på TCP-förbindelsen (som egentligen är två parallella förbindelser i respektive riktning). Detta innebär att servern kan prata med klienten och klienten kan prata med servern och vara säker på att all data som skickas kommer fram i korrekt ordning. Detta sker genom att mottagaren av ett paket skickar tillbaka ett ACK-paket, som berättar för sändaren att paketet tagits emot. Om mottagaren får paket 2 innan den får paket 1, så sparar den undan paket 2 tills den fått paket 1 och skickar ut en ack för paket 3 (för att förmedla att nästa paket enheten vill ha är paket 3) när paket 1 tagits emot. [1]

B. UDP, User Datagram Protocol

UDP är precis som IP ett förbindelsefritt *best effort*-protokoll som inte garanterar någonting om datan som skickas. Eftersom att UDP är ett förbindelsefritt protokoll finns där även begränsningar på storleken av paketen. Detta innebär att ifall det ska skickas större datafiler, så måste dessa delas upp i mindre paket för att kunna skickas med UDP-paket. [1]

C. TCP/UDP och multiplayerspel

I en artikel på sin blogg diskuterar spelutvecklaren Christoffer Lernö för och nackdelarna med TCP/UDP, och även spelindustrins syn på dessa. Lernö skriver:

> *In theory, the advantages of TCP are things like:*
>
> - *Straightforward persistent connections*
> - *Reliable messaging*
> - *Arbitrarily sized packets*
>
> *...However, the most damning property of TCP is the congestion control. Basically TCP interprets packet loss as a result of limited bandwidth, and throttles packet sends.*

Lernö sammanfattar till sist sin syn på TCP/UDP genom att förklara att TCP är bäst ämnat för spel där någon form av fördröjning är accepterat, och UDP för spel där fördröjning är oacceptabelt. [2]

D. Sockets i Java

Socket-klassen i Javabiblioteket används för att representera en ändpunkt i en förbindelse. En socket kan därav både skicka data och ta emot data. [3]

E. ServerSockets i Java

ServerSockets är Javas implemenation av en TCP-länk. För att en TCP-förbindelse ska startas måste mottagaren (servern) acceptera förbindelsen från sändaren (klienten). [4]

F. DatagramSockets i Java

DatagramSockets är Javas implementation av en förbindelsefri socket. Servern väntar endast på packet och hanterar dem om de kommer fram. [5]

III. Metod

A. Multiplayer Pong

Idén med att analysera ett program med känd källkod var grundad i tidigare erfarenheter och kunskaper om hur program kommunicerar över nätverk. Misstankar fanns om att stora kommersiella program och spel skickar mängder data som inte skulle vara relevant till analysen, som dessutom skulle vara krypterad. Färdiga spel tillåter även inte användaren att byta mellan det i huvudsak använda protokollet, och en direkt jämförelse mellan två spel med olika överföringsprotokoll skulle inte vara mycket värd, då spelens implementation skulle skilja sig helt. Ett program där det är enkelt kunde byta mellan protokollen samt enkelt isolera och analysera data behövdes, så ett egetskrivet program var en optimal testmiljö.

Pong räknas av många bland de första datorspelen. Två spelare styr var en paddel, på vardera sida av en spelbräda. En boll rör sig på brädan och studsar från den övre och undre kanterna. Spelet går ut på att skydda sin egna vertikala kant från bollen med sin paddel. I takt med att bollen hålls studsande ökar bollens hastighet för att försvåra spelet. I vår multiplayer-implementation ansluter sig två spelare till en spelserver, och skickar förändringen på sin paddel till servern, som i sin tur skickar bollens och motspelarens position. Två nästan identiska versioner av spelet skapades, som endast skilde sig i vilket överföringsprotokoll de använde, TCP eller UDP. Båda versioner hade ett eget server och klientprogram. Eftersom spelet frågar efter ett paket varje gång det målar om brädan är måttet "paket i sekunden" direkt kopplat till "frames per second". Detta ger ett bra mått på spelets prestanda under olika nätverksförhållanden. För att begränsa antalet paket skickade per sekund pausade vi klienten i 16,7 millisekunder mellan varje paketförfrågan, vilket resulterade i att spelet teoretiskt kunde uppnå 60 paket i sekunden, motsvarande 60 "frames per second".

B. Simulerade nätverksförhållanden

I syfte att simulera olika nätverksförhållanden som multiplayerspel kan spelas på delades analysen upp i tre typsituationer: Wired Ethernet (Peer-to-Peer), WiFi (WLAN), och över Internet. I Wired Ethernet anslöts klienten direkt till serverdatorn med en tvinnad parkabel, och i WiFi anslöts de trådlöst i ett lokalt nätverk. I ett testexempel användes även en anslutning över internet, genom att öppna porten som socketen använde. För att simulera realistiska avstånd låg servern i Malmö och klienten i Lund. Över Internet testades tvinnad kabel till ändpunkterna samt WiFi till ändpunkterna.

C. Insamling och analys av data

Metodiken för insamling och analys av data skilde sig inte åt emellan de olika simulerade nätverksförhållandena. Först startades servern upp, sedan anslöts en klient. Wireshark startades med rätt "capture filter", och sedan användes analysverktyget "IO Graphs" med filter för skickade och mottagna paket. Eftersom koden i Pong skickade och mottog paket i praktiken samtidigt överensstämmer graferna med varandra. De resulterande graferna sparades och analyserades.

Samtidigt studerades även spelupplevelsen. Motiveringen till att denna subjektiva faktor togs med är att rapportens utgångspunkt – om valet av nätverksprotokoll påverkar spelupplevelsen – är en subjektiv observation. Tillsammans med den objektiva analysen av paket med Wireshark producerades nog med data som grund för en vetenskaplig diskussion.

IV. Resultat

Den insamlade informationen består av IO-grafer som visar hur många paket som tas emot och skickas av servern per sekund. Denna data är direkt kopplad till prestandan eftersom servern bestämmer hur bollen förflyttar sig och hur paddlarna rör sig. Om det uppkommer fördröjning på klientsidan märks det även genom ett lägre paketantal på servern, eftersom att det är klienten som talar om när den har fått ett paket och vill ha nästa.

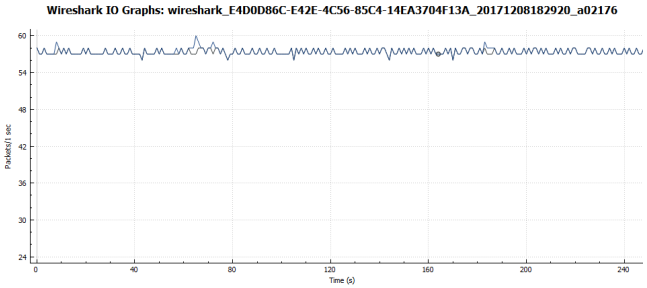

Fig. 1. TCP över tvinnad parkabel

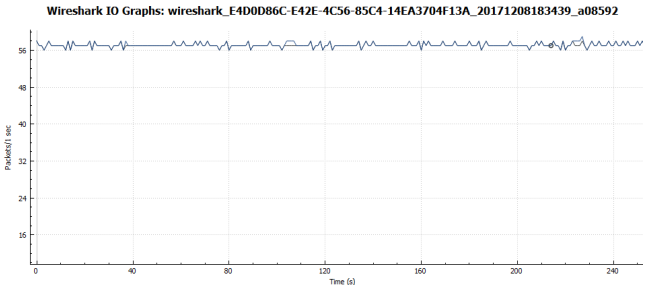

Fig. 2. UDP över tvinnad parkabel

Med tvinnad parkabel som överföringsmedium i ett eget LAN är det inte så stor skillnad emellan antalet paket per sekund. TCP snittade ungefär 56 paket per sekund, medan UDP låg på ungefär 57. UDP presterade lite bättre, vilket kan ses genom att jämföra figur 1 och 2. Båda transportprotokollen är dessutom väldigt stabila, de varierar endast i stabilitet med upp till $\pm$ 2 paket per sekund. Det var ingen märkbar skillnad på spelupplevelsen mellan TCP och UDP.

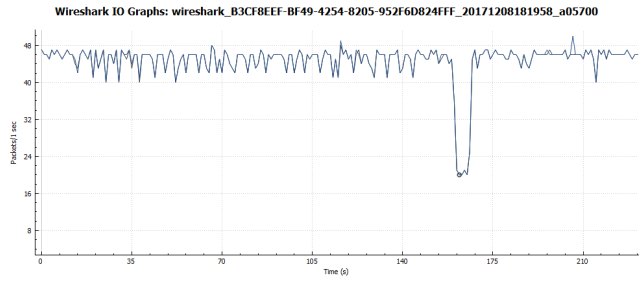

Fig. 3. TCP över WiFi

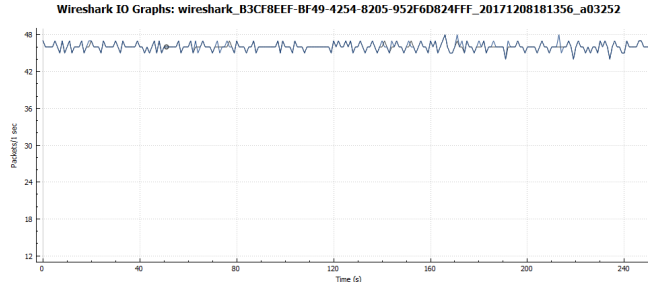

Fig. 4. UDP över WiFi

Med WiFi som överföringsmedium i ett eget LAN kan vi se en viss instabilitet i paket-per-sekund-antalet, se figur 3 och 4. UDP respektive TCP ligger på cirka 46 respektive 44 paket per sekund. UDP är fortfarande väldigt stabilt och varierar i paket-per-sekund-antal med upp till $\pm$ 2 paket per sekund medan TCP varierar med upp till $\pm$ 4 paket per sekund (eller cirka 25 vid dal). Spelupplevelsen är fortfarande i ett spelbart skick, förutom vid dal.

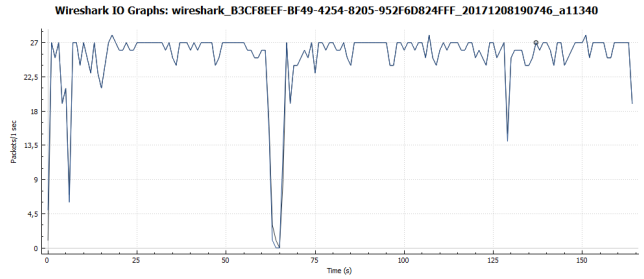

Fig. 5. TCP över Internet, wifi till ändpunkterna

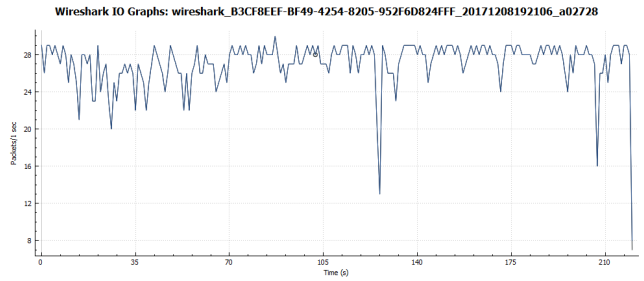

Fig. 6. UDP över Internet, wifi till ändpunkterna

Med WiFi som överföringsmedium men över Internet gav det sämsta resultatet av alla, se figur 5 och 6. TCP var otroligt ojämnt, med flera stora dalar, och grafen var extremt instabil, med varians mellan 23 och 27 paket per sekund. Det gick att spela, men bollen försvann då och då, och upplevelsen var överlag medioker. UDP presterade lite bättre, men inte mycket. UDP varierade med upp till 25$\pm$ 4 paket per sekund, med enstaka dal.

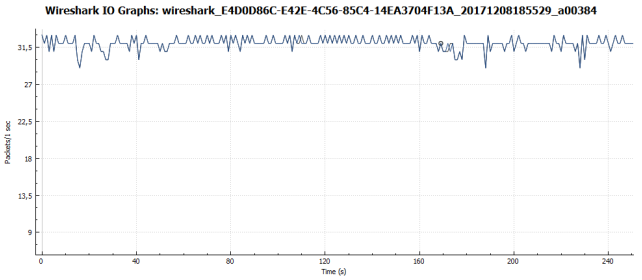

Fig. 7. UDP över Internet, tvinnad parkabel till ändpunkterna

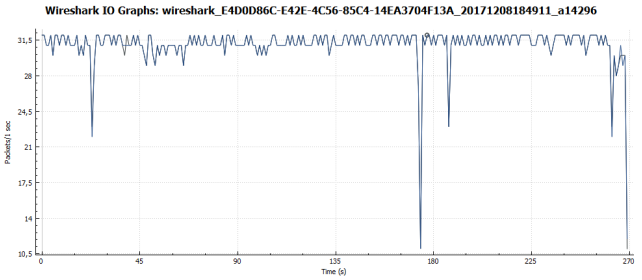

Fig. 8. TCP över Internet, tvinnad parkabel till ändpunkterna

Med sladd som överföringsmedium men över Internet kan man tydligt se skillnaden mellan TCP och UDP, se figur 7 och 8. UDP hamnade på ungefär 33 paket per sekund, som resulterade i mindre paket per sekund än de andra anslutningarna, men var fullt spelbart. Dessutom var UDP mycket stabilt, jämfört med TCP som fluktuerade lite mer. TCP led dessvärre av flera dalar där antalet paket per sekund minskade drastiskt, ibland ner till 11 paket per sekund. Detta resulterade i en mycket ojämn spelupplevelse, där bollen försvann kort och dök på ett annat ställe strax efter.

V. Analys

A. Validitetshot

När datorerna inte kopplades upp till ett eget LAN så användes ett hemnätverk med fler enheter än testdatorerna. Även om detta är en realistisk miljö innebär det att det kan finnas inkonsekvens beroende på de andra nätverksenheternas nätverksanvändning.

Implementationen av spelet är inte optimalt på grund av saknad programmeringserfarenhet när det gäller spel. Detta leder till dåliga lösningar, användningen av Java SWING (som inte är optimalt för spel) och att “pausa” klienten. Den upplevda mängden fördröjning blir därav mycket högre än vad den egentligen behöver vara. Men kan också förtydliga hur nätverksprotokollet påverkar spelupplevelsen.

B. Diskussion

I alla datainsamlingar har UDP visat sig mer stabil och med ett högre paketantal än sin TCP-motsvarighet, detta beror troligtvis på grund av att TCP måste skicka ett ACK innan nästa paket skickas. Högst och mest stabilt antal paket per sekund av alla var UDP över tvinnad parkabel. Det lägsta och minst stabila paket per sekund var TCP över Internet när ändpunkterna använde sig av WiFi som överföringsmedium. Även UDP över samma överföringsmedium är instabilt, men har kortare intervall av lägre antal paket per sekund.

I alla datainsamlingar med TCP, förutom över tvinnad parkabel, syns det att TCP tenderar att kraftigt minska och sedan återhämta sig. Detta beror på TCPs "congestion control". Att det inte syns lika tydligt när det gäller UDP beror på att vid "packet loss" så behöver inte UDP skicka om några paket och kan därför fortsätta att skicka paket.

Den resulterande spelupplevelsen var direkt kopplad till antalet paket i sekunden, och uppkopplingens kvalitet. En stabilare uppkoppling gav ett högre paketantal - notera det lägre paketantalet som uppstod tidvis med WiFi som överföringsmedium i figur 1, 2 och 4. Dessa dalar resulterade direkt i att bollen glappade och blev tillfälligt osynlig, ett mindre problem när bollen rörde sig långsamt, men mycket störande för spelets gång när bollen rörde sig fort. Snittantalet paket i sekunden påverkade också spelets kvalitet. Figur 6, UDP över tvinnad parkabel, gav bäst resultat med ungefär 58 paket i sekunden, mycket nära maxantalet. Både bollen och paddlarna rörde sig då mycket jämnt och flytande, medan figur 2 visar den lägsta paket per sekund-antalet, TCP över Internet, WiFi till ändpunkterna. Där uppnåddes endast 27 paket i sekunden, som det gick att spela med, men var en jämförelsevis mycket värre spelupplevelse.

En annan aspekt som bör diskuteras är om analysen av Pong egentligen representerar verkligheten. Moderna spel är överlag på en annan komplexitetsnivå, så det kanske inte är en rättvis representation att analysera Pong. Dock är det många faktorer som spelar in på nätverksprestanda, och en ökad spelkomplexitet behöver inte nödvändigtvis betyda att analysen förlorar sin relevans. Andra faktorer skalar upp samtidigt som spelkomplexiteten, till exempel bättre servers och spelimplementationer. Det är därför svårt att dra slutsatser, varken negativa eller positiva, om hur väl Pong-spelet kan skalas upp. Däremot kan vi dra slutsatsen att protokollet fungerar likadant i mer komplexa miljöer (spel).

VI. Slutsats

Det är tydligt att UDP är ett bättre transportprotokoll än TCP för att skicka realtidsdata. Samma resultat återkom oavsett överföringsmedium. När överföringsmediet tvinnad parkabel används spelar inte transportprotokollet lika stor roll, men även i den miljön visade UDP på ett högre antal paket per sekund och därav bättre resultat. En direkt koppling emellan paket per sekund och spelupplevelsen är etablerad, vilket också innebär att spelupplevelsen också alltid blev bättre med transportprotokollet UDP. Anledningen till att UDP visar bättre resultat i alla mätningar är på grund av TCPs congestion control. Congestion control är en egenskap som kan behövas i vissa miljöer, men realtidsspel antas inte vara en sådan.

References

[1] Maria Kihl, Jens A. Andersson
Datakommunikation och nätverk. 181-195, 2015.

[2] Christoffer Lernö
Game servers: UDP vs TCP https://1024monkeys.wordpress.com/2014/04/01/game-servers-udp-vs-tcp, hämtad 18/12-17

[3] Oracle
Lesson: All About Sockets https://docs.oracle.com/javase/tutorial/networking/sockets hämtad 18/12-17

[4] David Reilly, Michael Reilly
Java™ Network Programming and Distributed Computing http://www.informit.com/articles/article.aspx?p=27633&seqNum=7, hämtad 18/12-17

[5] JavaPoint
Java DatagramSocket class https://www.javatpoint.com/DatagramSocket-and-DatagramPacket, hämtad 18/12-17