Undersökning av Minecraft i Wireshark

Hugo Mattson, hu5174ma-s
Felix Forsström, fe2117fo-s

***Abstract*— Denna undersökning tittar på nätverkstrafiken för spelet Minecraft. Wireshark används för att ta reda på hur infrastrukturen med servrar ser ut och vilka protokoll som används för transport av data. Analys kring hur många paket det är som skickas vid uppkoppling till officiella respektive privata, lokala spelservrar utförs. Undersökningen visade att det skickas fler paket mot officiella spelservrar och att det primära protokollet förvånansvärt nog var TCP. Nätverksinfrastrukturen tillhandahålls av servrar på större molntjänster och att känslig data är väl krypterad med TLS.**

I. Introduktion

Mängden nätverkstrafik som är relaterad till onlinespel ökar konstant och förutspås stå för 4% av den global IP-trafiken år 2022[1]. Den ökande mängden människor med tillgång till onlinespel, samt den ökande prestandan hos spelen, ställer stora krav på nättrafiken. Spelutvecklare måste kunna leverera stabila, snabba och säkra spelupplevelser till spelare med en mängd olika förutsättningar för uppkoppling och näthastigheter. Detta kräver optimerad nätkod som utnyttjar de protokoll, servrar och säkerhetsrutiner som är bäst lämpade för situationen.

Denna rapport studerar spelet Minecraft. Relevant är hur mycket- och vilken data spelet sänder. Samt vilka protokoll som utnyttjas i vilka situationer. Datan som samlas in kommer presenteras i en analys med fokus på datamängd, protokoll och säkerhet.

II. Bakgrund

Minecraft utvecklas av det svenska spelutvecklingsföretaget Mojang och är ett spel baserat på utforskande och kreativitet. Stilen är medvetet pixelerat och kantigt med världar som byggs upp av kubikmeterstora block med spelaren och diverse varelser som invånare. Dessa världar genereras efterhand som de utforskas vilket gör deras storlek i princip oändlig.

Det finns inget bestämt mål med Minecraft utan din upplevelse är endast begränsad av din egen fantasi. Denna frihet har varit ett koncept som genom åren lockat många användare i alla ålderskategorier. Resultatet syns i att det sedan lanseringen 2011 har sålts i över 180 miljoner exemplar, vilket gör det till världens mest sålda digitala spel[2]. Minecraft går alldeles utmärkt att spela ensamt men för denna rapport är det flerspelarläget som är aktuellt.

För den som vill spela med andra finns tre sätt att gå till väga: En värld skapad för ensamt spel kan öppnas upp så din klient agerar server och andra på ditt lokala nätverk kan ansluta, så kallat LAN. Man kan även ansluta till dedikerade servrar över internet som hanteras av privatpersoner/privata organisationer. Slutligen kan man spela på så kallade Realms vilket är servrar tillhandahållna av Microsoft som sedan 2014 äger Mojang.

A. Frågeställning

Rapporten syfte är att undersöka hur Minecraft hanterar de krav som ställs på det på grund av dess unika spelidé. Alltså hur det klarar av de många olika situationer som kan uppstå i de många olika sätten att spela spelet. Frågeställningarna som ska besvaras är därmed följande:

- Hur skiljer sig mängden paket som skickas mellan server och klient i olika situationer?
- Vilka protokoll används?
- Vilka servrar används och hur är de konstruerade?
- Vilken säkerhet appliceras på paketen?

III. Metodbeskrivning

För att utföra mätningarna användes en dator körandes Linux operativsystemet Manjaro med kernelversion 4.19.85-1-MANJARO och Minecraft version 1.14.4.

Mätningar gjordes på en dator uppkopplad via Wi-Fi till Lunds Universitets nätverk *eduroam* vars ISP (internet service provider) är Lunds Universitet. Lunds Universitet ligger sedan i sig inom universitetsnätverkskoncernen SUNET som agerar slutgiltig ISP. Eduroams router utnyttjar NAT (Network Adress Translation) för att konvertera de publika IP-adresserna som Lunds universitet tillhandahåller till privata IP-adresser.

För att isolera datan mellan Minecraftklienten och servern stängdes först bakgrundsprocesser ned i största möjliga mån. I Wireshark skapades sedan ett filter för att urskilja de paket som var relaterade till Minecraft. Filtret skapades genom att låta datorn stå inaktiv med Wireshark lyssnande på all data som skickades via Wi-Fi i 10 minuter. De IP-adresser som identifierades där, skilt från den egna IP-adressen, lades till i filtret. När inga mer paket identifierades startades Minecraft. De paket som då identifierades ansågs tillhöra spelets kommunikation.

IV. Experiment

Totalt utfördes tre tester. Det första testet undersökte serveruppkopplingar vid menynavigation och utfördes av en spelare. De två sista testerna undersökte nätverkstrafik vid interaktion inuti spelet. De utfördes båda på de två serverkonfigurationerna LAN respektive Realm med två aktiva spelare för varje test. Speciellt för LAN var att den ena spelaren agerade både klient och server.

Det första testet inleddes med en start av klientlaunchern vilket är öppningsmenyn för inloggning. Efter inloggning

startades spelet och spelmenyn öppnades. I spelmenyn valdes de olika menyvalen: Singleplayer, Multiplayer, Minecraft Realms och Options med ett jämt mellanrum på ca 20 sekunder per val. Då alla val var besökta avslutades mätningen.

Vid det andra testet skapades nättrafik genom att de båda spelarna samtidigt genererade ny värld. Mätningen utfördes av en spelare och gick till på följande sätt för LAN: Först anslöt ena spelaren till världen som laddades in och öppnade upp den för LAN-nätverk, sedan startades en ny mätning i Wireshark och efter ca 10 sekunder anslöt den andra spelaren till servern som klient. Efter 30 sekunder från bådas anslutning började var spelare gå med sin karaktär i direkt motsatta riktningar för att börja generera ny värld. Då fyra minuter totalt hade passerat avslutades mätningen. För Realm utfördes mätningen på samma sätt med skillnaden att ingen spelare agerade server utan båda spelare anslöt direkt.

Det tredje testet gick ut på att undersöka hur spelet hanterade hög belastning, specifikt då stora mängder TNT detonerades. TNT har förmågan att förstöra block i en 4 blocks radie. Om andra TNT-block finns inom radien aktiveras dessa, vilket skapar en kedjereaktion av explosioner. Alla explosioner beräknas individuellt för varje block TNT och blir fort beräkningsintensivt. Testet startades inuti spelet med båda spelare redan anslutna. Först genererades en kub bestående av 23^3st TNT-block, sedan startades en mätning i Wireshark. Efter 30 sekunders inaktivitet i spelvärlden antändes det första blocket TNT. Mätningen avslutades efter att hela kedjereaktionen var färdig och världen samt servern återhämtat sig.

A. Validitet och begränsningar

De två största felkällorna är troligen antalet undersökningar och filtreringen av IP-adresser.

Vad gäller antalet undersökningar gjordes det endast en undersökning under menytestet och en för vardera konfiguration av de andra testerna. Resultaten skulle kunna variera vid flera undersökningar. Detta på grund av olika faktorer, exempelvis: begränsad bandbredd på grund av annan trafik på nätet, olika bakgrundsprocesser på datorn som inte hanterats etc. vilket inte tagits hänsyn till. Något som talar för validiteten av de två sista testerna är att de var lika i utförande vilket gav två mätningar på vardera serverkonfiguration. Dessa kan inte jämföras för att göra något uttalande om den specifika datamängd som skickas men kan ge en övergripande bild av de protokoll som utnyttjades.

Relaterat till detta är faktumet att filtreringen av IP-adresser inte är garanterat perfekt. Det kan finnas applikationer som skickar data som inte relaterar till Minecraft som inte identifierades i filtreringen. Här kan vi dock se att mängden data som skickas till servrar dedikerade till Minecraft överträffar det fåtal paket som kunde skickats av en annan applikation i sådan mån att det inte borde ge någon märkbar skillnad i resultaten.

Vad som talar för validiteten hos experimenten är att de är gjorda så konsekvent som möjligt mellan testerna och mellan de olika serverkonfigurationerna, vilket ger jämförbara resultat.

V. Resultat

IP	Protokoll	Plats	Företag	Roll
143.204.236.67	TLSv1.2	Danmark	Amazon	Autentisering
143.204.239.175	TLSv1.2	Danmark	Amazon	Session
3.225.96.24	TLSv1.2	USA	Amazon	Realmlista
172.65.212.227	TCP	USA	Amazon	Okänd
172.65.254.166	TCP	USA	Amazon	Okänd
51.124.10.13	TCP	Holland	Microsoft	Realm

TABLE I
IP adresser man ansluter till.

I tabell I ovan syns de IP-adresser som är del av Minecrafts infrastruktur. Server för autentisering verifierar inloggningsförsök, sessions-servern har information om spelares karaktärer(ID, namn och utseende)[3] och realmlistan berättar vilka Realms du har åtkomst till och vilken IP de har. Protokoll spalten indikerar vilket protokoll som servern utnyttjade. TLSv1.2 är den vanligaste versionen av det krypterade nätprotokollet TLS (Transport Layer Security) som bygger på toppen av TCP[4]. Det utnyttjades av de servrar som hanterar verifiering och annan känslig data. Själva spelservern i Holland utnyttjade endast TCP.

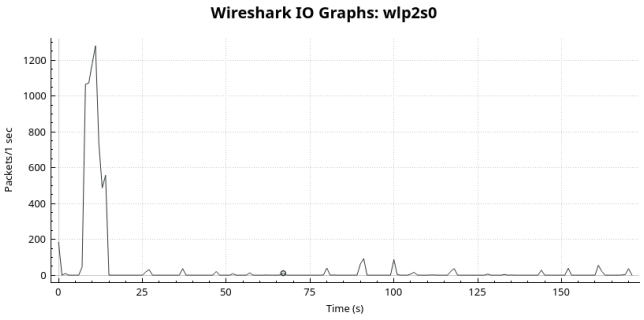

Fig. 1. Menynavigering

Generellt kan ses i figur 1 att 90% av datan skickades i två spikar av data inom de första 15 sekunderna då spelet initierades. Den första ökningen kom då launchern startades och den andra då spelklienten startades. Efter dessa skickades betydligt mindre data överlag. Datan som sedan skickades kom i spikar som motsvarar de menyval som gjordes i spelklienten.

Tre olika protokoll användes vid navigering av menyn: TLSv1.2 stod för 60.6% av paketen. Icke krypterad TCP stod för 39.2% vilket gjorde att TCP och TLSv1.2 totalt stod för 99.8% av alla skickade paket. De sista 0.2% bestod av DNS-meddelanden. I total datamängd stod TLSv1.2 för nästan 100% då TCP-paketen i princip bara var ACK på 66 bytes.

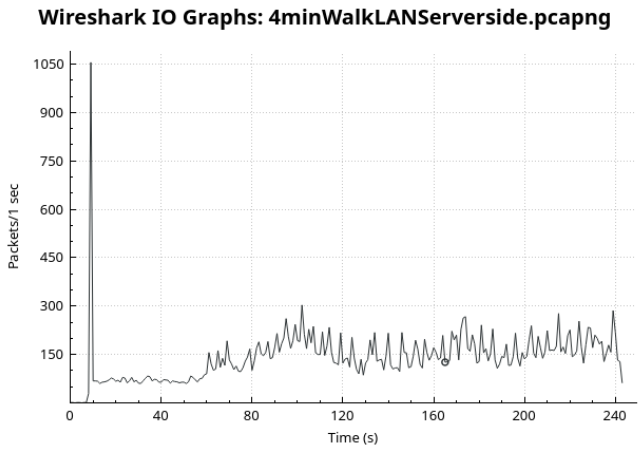

Fig. 2. Världsgenerering på LAN-server

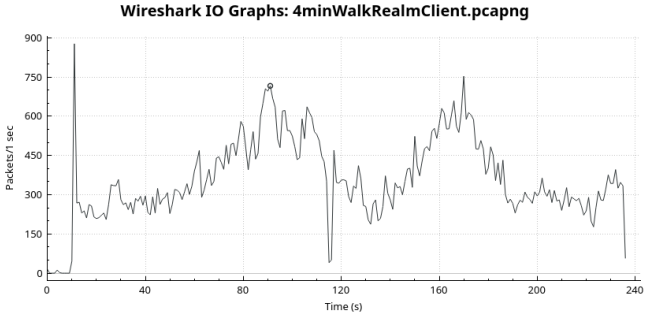

Fig. 3. Världsgenerering på Realm

Då spelare ansluter till servern syns en tydlig spik i datatrafik i figur 2. Efter denna skickas väldigt få paket/s till dess att de båda spelarna börjar röra på sig och trafiken ökar. I början av mätningen skickas det några TLS-paket till session-servern i tabell I och några UDP-paket med okänt syfte. För dessa två protokoll var det sammanlagt ca 20 paket som skickades och resten av de 34000 paketen i mätningen står TCP för.

För mätning mot Realm syns i figur 3 liknande resultat som mot LAN-server. Det skickas TLS-paket till session-servern i början men ingen trafik med UDP plockas upp denna gång. Den stora spiken är återigen att spelaren ansluter till servern. Snart därefter följer ökningen i datatrafik när spelarna börjar röra på sig. Det är TCP som står för alla paket som totalt denna gång är 85000 stycken.

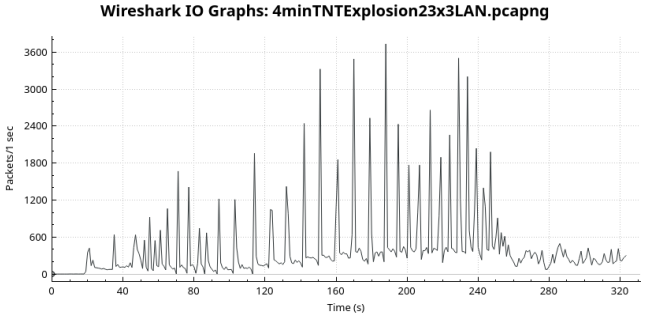

Fig. 4. Explosioner på LAN-server

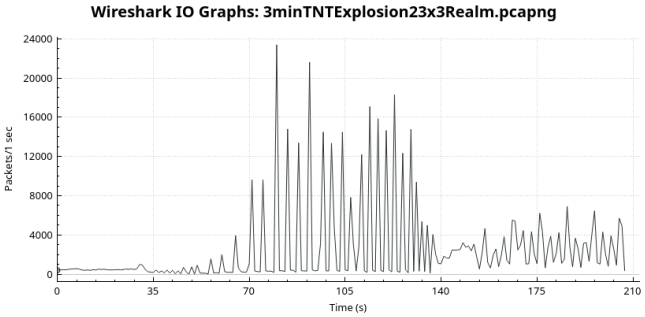

Fig. 5. Explosioner på Realm

För de båda mätningarna då TNT detonerades syns tydliga likheter mellan figur 4 och 5. Mätningen börjar lugnt innan explosionerna satts igång. Alla de höga spikarna är den information som skickas under tiden som explosionerna pågår och därefter minskar dataöverföringen markant.

För de båda mätningarna var TCP det enda protokoll som användes, däremot skilde sig mängden paket. Totalt skickades 131000 och 511000 paket till LAN respektive Realm.

VI. Diskussion

Spelet som helhet visar i de olika testerna på att data alltid skickades via en central server, se tabell I, som sedan hanterar kommunikationen med andra spelare. Detta även i fallet med LAN då den ena spelaren agerar server. Vilket verkar rimligt då det inte finns något annat sätt att hantera en värld som spelare ska kunna komma åt när som helst och modifiera vilket innebär att världens senaste tillstånd måste finnas tillgängligt när en ny spelare ansluter. En central server ger även ökad säkerhet i form av att känslig data såsom IP-adresser inte exponeras direkt mot andra användare. Detta skulle kunna användas för t.ex. denial-of-service (DoS) [5] attacker.

För att kunna tillhandahålla dessa centrala servrar utnyttjar Minecraft de två olika molntjänsterna Microsoft Azure (azure) och Amazon Web Services (aws), se tabell I. Anledningen är högst sannolikt att de utnyttjas för deras skalbarhet. I och med att Minecraft erbjuder Realms är det både smidigt och kostnadseffektivt att bara hyra så mycket serverprestanda som krävs för att tillhandahålla den aktiva mängden Realms.

Skillnaden mellan vilka servrar som utnyttjades i de olika testerna är intressant. I menynavigationstestet kommunicerade klienten endast mot aws servrar medans den i de aktiva speltesterna endast kommunicerade mot azure servrar. Det är något märkligt att Microsoft som äger både azure och Minecraft väljer att hosta delar av sin funktionalitet på aws. Det går inte att dra någon säker slutsats om varför det är fallet men det syns tydligt i menynavigationstestet att all funktionalitet som inte är aktivt spelande i en Realm tillhandahålls av aws. Medan all data som tillhandahålls av realmservern vid aktivt spelande går dock via azure, se figur I. En spekulation skulle alltså vara att autentisering och liknande funktionalitet har sedan tidigare legat på aws och att den inte migrerats över till azure.

Vad gäller de olika protokollen som utnyttjas så syns det tydligt att aws servrar andvänder TLSv1.2 för att kryptera all sin kommunikation som inte är simpla ACK meddelanden. Detta är rimligt då de hanterar autentisering och känslig data som måste ha en god nivå av säkerhet.

När det kommer till själva spelandet som hanteras på azure så används i princip bara TCP. Detta skiljer sig från andra flerspelarspel och realtidsapplikationer som oftast brukar använda UDP då det är snabbt och man oftast inte bryr sig om paketförluster. I Minecraft är det dock viktigt att information om ändringar utförda på block kommer i rätt ordning för att spelvärlden ska fungera korrekt. Detta ledde till att valet föll på att använda TCP istället, något som skaparen av Minecraft diskuterar på sin blogg [6][7]. Då speldatan inte heller behöver krypteras är TLS inte nödvändigt.

Vid kommunikation mot Realm skickades det konsekvent mer data än mot LAN-server. Det kan bero på att Mojang valt att dessa servrar ska skicka fler mindre paket för att minska mängden data som måste skickas om ifall ett paket försvinner. Annars kan det helt enkelt vara så att Realm-servern är mycket snabbare och därmed hinner skicka paket oftare.

Hur paketen skickades var också annorlunda beroende på situationen. I mätningen för världgenerering, figur 2 och 3, skickades data konsekvent efter hand som mer värld upptäcktes. När det kom till TNT-testet, figur 4 och 5, skickades istället datan i stora lass med jämna mellanrum. Vilket tyder på att det finns optimeringar i spelet för att hantera olika situationer. Det ska dock nämnas att under TNT-testet var det i stort sätt omöjligt att interagera med spelet på grund av den höga belastningen.

VII. Slutsatser

Minecraft utnyttjar en skalbar molnlösning i sin nätverkskonfiguration för att hantera varierande mängder trafik, vilket är mycket bra för en ökad mängd spelare och tillåter spelande med god anslutning.

I nätverkstrafiken för Minecraft skickas det hela tiden många paket, särskilt ifall man väljer att spela på en Realm. Men det finns optimeringar för olika situationer. De skulle dock behöva förbättras för en bättre spelupplevelse, speciellt för folk med datorer med sämre prestanda.

Till vår förvåning var det inte UDP utan TCP som användes som transportprotokoll för speldata. Utöver TCP är det bara ett till protokoll som används, TLSv1.2 som används för att kryptera känslig data, säkerheten för Minecraft anses därmed vara god.

References

[1] Cisco. (2019, 12) Cisco visual networking index: Forecast and trends, 2017–2022 white paper. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] Wikipedia. (2019, 12) Best-selling video games. [Online]. Available: https://en.wikipedia.org/wiki/List_of_best-selling_video_games

[3] MojangWiki. (2019, 12) Mojang API. [Online]. Available: https://wiki.vg/Mojang_API

[4] Wikipedia. (2019, 12) Transport layer security. [Online]. Available: https://en.wikipedia.org/wiki/Transport_Layer_Security

[5] ——. (2019, 12) Denial-of-service. [Online]. Available: https://en.wikipedia.org/wiki/Denial-of-service_attack

[6] The Word of Notch. (2019, 12) All the boring parts are done. [Online]. Available: https://notch.tumblr.com/post/773786850/all-the-boring-parts-are-done

[7] ——. (2019, 12) The hottest monday like ever. [Online]. Available: https://notch.tumblr.com/post/802279517/the-hottest-monday-like-ever