Лабораторийн ажил №4

TCP, UDP протокол

МТЭС, МКУТ, Компьютерийн ухаан

Б.Барсболд, 22B1NUM4397

Ажлын зорилго

Энэхүү ажлаар TCP/IP -ын тээвэрлэлтийн давхарга (*Transport layer*) -ын Transmission Control Protocol (TCP), User Datagram Protocol (UDP) протоколуудын ажиллах зарчмыг судлах болно.

Онолын судалгаа

1. Холболтод суурьласан бус өгөгдөл дамжуулал: UDP (User Datagram Protocol)

UDP нь RFC 768 стандартаар тодорхойлогддог бөгөөд дамжууллын түвшиний протоколоос маш бага давуу талтай бөгөөд multiplexing/demultiplexing функц болон маш хөнгөн алдаа шалгалтаас бусдаар IP-д өөр юу ч нэмдэггүй. Энэ нь UDP-р өгөгдөл дамжуулхыг сонговол шууд IP ашиглаж байгаагаас бараг ялгаагүй. UDP нь аппликейшн процессоос мессежийг аван хүлээн авах порт, илгээж буй портуудыг хавсаргадаг бөгөөд энэ нь multiplexing болон demultiplexing сервисд хэрэглэгддэг. Мөн эдгээрээс өөр 2 жижиг талбар нэмэн сүлжээний давхрагад дамжуулдаг. Сүлжээний давхрага үүнийг IP датаграм болгон битүүмжлээд хамгийн боломжтой хүргэлт хийх оролдлогоор хүлээн авагч руу илгээнэ. UDP хүлээн авах портыг өгөгдлыг зөв аппликейшн процесс-д дамжуулхад ашиглана. UDP-г холболтод суурьлаагүй гэж нэрлэдэг шалтгаан нь өгөгдөл дамжуулал эхэлхийн өмнө ямар handshake үйлдэл хийгдэдгүй.

DNS нь аппликейшн давхрагын протокол-д UDP ашигладаг жишээнүүдийн нэг юм. DNS аппликейшн хүсэлт явуулах үедээ UDP хүсэлтийн мессежийг байгуулан UDP-д дамжуулдаг. Сүлжээний нөгөө талд байгаа UDP-тэй ямар ч handshake хийхгүйгээр зөвхөн толгой хэсгийн талбаруудыг нэмэн сүлжээний давхрагад дамжуулна. Сүлжээний давхрага UDP сегментийн IP датаграмд битүүмжлэн датаграмыг name server руу илгээнэ. Тэгээд DNS аппликейшн хүсэлтийн хариу мессеж ирхийг хүлээнэ. Хэрэв хариу ирэхгүй бол сүлжээний түвшинд хүсэлт эсвэл хариулт гээгдсэн гэж үзээд дахин хүсэлт илгээдэг.

Доорх шалтгаануудын улмаас бид найдвартай TCP-с илүү UDP-г сонгодог:

- Аппликейшн түвшинд, илүү сайн ямар өгөгдөл удирдах хэзээ илгээгдхийг удирдах. UDP ашиглаж байгаа үед аппликейшн өгөгдлийг UDP-д дамжуулмагц UDP сегмент болгон сүлжээний давхрага руу дамжуулдаг. Нөгөө талаас TCP нь илгээгч болон хүлээн авагч хостуудын хоорондох нэг буюу хэд хэдэн холбоосууд хэт түгжрэлд орох үед TCP илгээгчийг саатуулдаг түгжрэлийг хянах механизмтай. TCP нь хэр удаан найдвартай хүргэхээс үл хамааран сегментийг хүлээн авсныг хүлээн авагч мэдэгдэх хүртэл сегментийг дахин илгээдэг. Бодит хугацаанд ажилдаг аппликейшнууд хоцрогдол багатай өгөгдлийн дамжуулал чухал байдаг учир TCP модел ийм төрлийн аппликейшнуудад тохиромжгүй юм.
- Холболт үүсгэлт шаардахгүй. 2.4-д дурдсанаар TCP холболт эхлэх үед three-way handshake хийдэг. Харин UDP ямар ч өмнөтгөл болон албан хүсэлт байхгүйгээр шууд л дамжуулалт эхлүүлэнэ. Тиймээс UDP холболт эхлүүлхэд ямар ч цаг зардаггүй. Энэ магадгүй яагаад DNS TCP биш UDP ашигладаг гол шалтгаан юм.
- Холболтын төлөв хадгалдаггүй. TCP төгсгөлийн системүүд холболтын төлөвүүдийг хадгалдаг. Энэ нь жишээ нь send болон receive buffer-ууд, түгжирлийг удирдах параметерүүд, дарааллын болон хүлээн авалтын параметерүүд юм. Нөгөө талаас UDP-д эдгээр параметерүүд ямар ч хамаагүй бөгөөд алийг нь ч хадгалдаггүй. Тиймээс UDP серверүүд TCP серверүүдээс олон клейнтуудтай харицаж чаддаг.
- Пакетуудын толгойн хүндрэл бага. TCP сегмент 20 байт толгой хэсэгтэй байдаг бол UDP сегмент зөвхөн 8 байт.

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Name translation	DNS	Typically UDP

Зураг 1 Түгээмэл интернет аппликейшнууд болон тэдгээрийн ашигладаг протоколууд

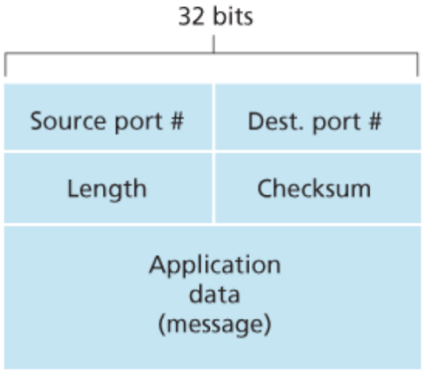
Хэдийгээр өнөөдөр ихэвчлэн ашиглагддаг мультимедиа аппликейшнууд UDP дээр суурилсан нь маргаан дагуулдаг. Дээр дурдсанчлан UDP нь түгжрэлийн хяналтгүй байдаг. Гэхдээ сүлжээг маш бага ашигтай өгөгдөлийн тээвэрлэлт хийдэг түгжрэлд оруулахгүйн тулд түгжрэлийг хянах шаардлагатай. Хэрэв хүн бүр ямар ч түгжрэлийн хяналтыг ашиглахгүйгээр өндөр битийн хурдтай

видеог цацаж эхэлбэл router-үүд дээр маш их пакет халих болно. Үүнээс шалтгаалан маш цөөхөн UDP пакетууд илгээгчээс хүлээн авагч хүртлэх замыг амжилттай даван туулах болно. Түүгээр ч зогсохгүй, хяналтгүй UDP илгээгчээс үүдэлтэй өндөр loss нь TCP илгээгчийг (сүлжээнд хэт их түгжрэл үүсэх нь тэдний илгээлтийн хурдыг бууруулдаг) хурдаа эрс бууруулахад хүргэдэг. Тиймээс, UDP-д түгжрэлийг хянахгүй байх нь UDP илгээгч болон хүлээн авагчийн хооронд өндөр алдагдал, TCP дамжуулал удаашрах зэрэг ноцтой асуудал үүсгэж болзошгүй юм [Флойд 1999]. Олон судлаачид бүх илгээгч, түүний дотор UDP илгээгч түгжрэлийн хяналтыг хэрэгжүүлэх шинэ механизмыг санал болгосон [Mahdavi 1997; Флойд 2000; Колер 2006: RFC 4340].

UDP ашиглах үед програм найдвартай өгөгдөл дамжуулах боломжтой. Найдвартай байдлыг програмд тусгайлан тооцон хэрэгжүүлсэн тохиолдолд үүнийг хийж болно. Жишээ нь acknowledgment, дахин дамжуулах механизмуудыг нэмэх замаар. Google-ийн Chrome вэб хөтөч дээр ашигласан QUIC протокол [Iyengar 2015] нь UDP дээр суурилсан аппликейшн түвшний протоколд найдвартай байдлыг хэрэгжүүлдэг. Гэхдээ энэ нь програм хөгжүүлэгчийг удаан хугацааны турш алдааг хянан завгүй байлгах энгийн ажил юм. Гэсэн хэдий ч, найдвартай байдлыг шууд аппликешнд бий болгосноор программ нэг сумаар хоёр туулай буудах боломжтой. Өөрөөр хэлбэл, хэрэглээний процессууд нь TCP-ийн түгжрэлийг хянах механизмаас тогтоосон дамжуулах хурдны хязгаарлалтад өртөхгүйгээр найдвартай харилцаж чадна.

1.1. UDP-ийн сегментийн бүтэц

UDP сегментийн бүтцийг RFC 768-д тодорхойлсон байдаг. Аппликейшний өгөгдөл нь UDP сегментийн өгөгдлийн талбарт байршдаг. Жишээлбэл, DNS-ийн хувьд өгөгдлийн талбар нь асуулгын мессеж эсвэл хариултын мессежийг агуулна. Audio streaming програмын хувьд аудионий хэсгүүд нь өгөгдлийн талбарт байрладаг. UDP толгой хэсэг нь тус бүр нь хоёр байтаас бүрдэх дөрвөн талбартай. Өмнөх хэсэгт дурьдсанчлан портын дугаарууд нь хүлээн авагчийн систем дээр ажиллаж байгаа програмын өгөгдлийг зөв процесс руу дамжуулах (өөрөөр хэлбэл, мултиплексийн функцийг гүйцэтгэх) боломжийг олгодог. Length талбар нь UDP сегмент дэх байтуудын тоог зааж өгдөг. Өгөгдлийн талбарын хэмжээ өөр өөр байж болох тул уртын тодорхой утга шаардлагатай. Checksum нь хүлээн авагч хост сегментэд алдаа үүссэн эсэхийг шалгахад ашигладаг. Checksum талбарыг UDP сегментэд нэмэлтээр байдаг талбарт тооцдог.



Зураг 2 UDP сегментийн бүтэц

1.2. UDP Checksum

UDP Checksum нь алдаа илрүүлэхэд ашиглагддаг. Өөрөөр хэлбэл, checksum нь UDP сегмент дэх битүүд нь илгээгчээс хүлээн авагч руу дамжуулагдах үед өөрчлөгдсөн эсэхийг тодорхойлоход ашиглагддаг (жишээлбэл, кабле дахь шуугиан эсвэл чиглүүлэгчид хадгалагдаж байх үед). Илгээгч тал дахь UDP нь сегмент дэх бүх 16 битийн үгсийн нийлбэрийг хоёртын тоололд гүйцэтгэдэг бөгөөд нийлбэрийг тооцоолж байх үед тохиолдох аливаа халилтийг эхлэл төгсгөл нь холбоотой гэж үзэн нэмдэг. Энэ үр дүнг UDP сегментийн checksum талбарт оруулна. RFC 1071-ээс тооцооллыг үр ашигтай хэрэгжүүлэх, бодит өгөгдөл дээр ажиллах талаарх дэлгэрэнгүй мэдээллийг авч болно [Stone 1998; Чулуу 2000].

2. Холболтот суурьласан дамжуулал[1]: TCP (Transmission Control Protocol)

TCP нь интернетийн дамжууллын давхарга[2]-ын холболтод суурилсан, найдвартай дамжууллын протокол юм. Мөн TCP нь маш олон үндсэн зарчмууд дээр тулгуурладагаас дурдвал алдаа илрүүлэх, дахин дамжуулах, cumulative acknowledgment[3], хугацаа хэмжих, дараалал зэрэг багтах юм. TCP нь дараах стандартуудын дагуу тодорхойлогддог: RFC 793, RFC 1122, RFC 1323, RFC 2018 болон RFC 2581.

2.1. TCP холболт

TCP холболт нь холболтод суурилсан учир хоёр аппликейшн хооронд өгөгдөл дамжуулхаас өмнө бие биетэйгээ холболтоо эхлүүлэх буюу “handshake” хийх шаардлагатай. Өөрөөр хэлбэл тэд

[1] Connection-Oriented Transport

[2] Transport layer

[3] Сүлжээний ачааллыг хэмнэхийн тулд илгээгчийн олон пакетуудад ганцхан ширхэг бүгдийг нь хүлээн авсан гэх acknowledgment (ACK) мессеж явуулах процесс

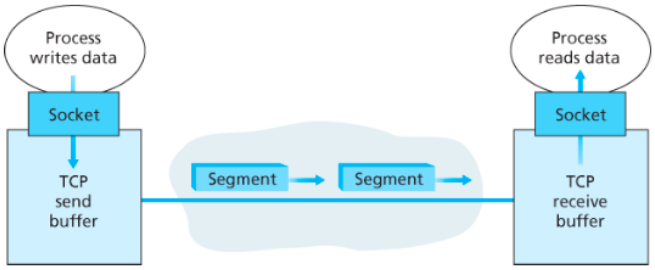
дараагийн өгөгдөл дамжуулах параметрүүдийг тогтоохын тулд зарим урьдчилсан сегментүүдийг бие биедээ илгээх ёстой. TCP холболтын эхлүүлэх процессийн нэг хэсэг болгон холболтын хоёр тал TCP холболттой холбоотой хэд хэдэн TCP төлөвийн хувьсагчдыг үүсгэнэ.

Мөн TCP холболт нь эхлэлээсээ төгсгөл хүртэл TDM эсвэл FDM шиг цахилгаан хэлхээнд суурилсан холболт биш юм. Харин TCP холболт нь логик бөгөөд TCP-ийн харилцах хоёр төхөөрөмж адил төлөвүүд хадгалдаг. TCP протокол нь завсрын дамжуулагч сүлжээны элементүүдэд (router, link-layer switch) биш эцсийн төхөөрөмжүүдэд ашиглагддаг тул төлөвүүд дундын төхөөрөмжүүд дээр хадаглагдахгүй. Яг үнэндээ дундын чиглүүлэгч төхөөрөмжүүд холболтыг TCP эсэх талаар мэддэггүй. Дамжиж буй датаг мэдэж чадах ч холболтыг мэдэж чадахгүй.

TCP холболт нь full-duplex буюу холболтын 2 төгсгөлд байгаа төхөөрөмжүүд хоёул дата илгээх болон дата хүлээн авах чадвартай гэж хэлж болно. Жишээ татвал нэг хост дээр Process A, өөр нэг хост дээр Process B ажиллаад хоорондоо TCP хоболт үүсгэсэн гэж үзвэл өгөгдөл A-с B руу болон B-с A руу нэгэн зэрэг явж чадна. Түүнчлэн TCP нь point-to-point буюу ганц илгээгч ганц хүлээн авагтай байна. Энэ нь нэг илгээгч өгөгдөл илгээх үед нэгээс олон хэрэглэгч тэр өгөгдлийг хүлээн авч болохгүй гэсэн үг.

TCP холболтод холболт үүсгэж буй процессийг клейнт буюу хэрэглэгч процесс харин нөгөө процессийг сервер процесс гэж нэрлэдэг. TCP холболт үүсэхэд клейнт эхний тусгай TCP сегмент илгээдэг. Дараа нь сервер хоёр дахь тусгай TCP сегментээр хариулдаг. Үүний дараа клейнт гурав дахь тусгай TCP сегмент илгээдэг. Эхний хоёр сегмент ямар ч payload-гүй байдаг бөгөөд энэ нь аппликейшн түвшиний өгөгдөлгүй гэсэн үг. Харин гурав дахь сегмент payload өгөгдөлтэй байж болно. Энэ холболт эхлүүлэх процесс нь **three-way handshake** гэж нэрлэгддэг

Нэгэнт TCP холболт үүссэн бол хоёр аппликейшн хоорондоо өгөгдөл дамжуулах боломжтой болно. Нэг процессийг нь клейнт нөгөө процессийг нь сервер гэж үзвэл клейнт процесс өгөгдлийн урсгалыг socket (процессийн хаалга гэж ойлгож болно)-р дамжуулна. Зураг 3-д харуулсанаар илгээгч талын өгөгдлийг three-way handshake-ийн үед үүсэх TCP холболтын **send buffer**-д оруулна. TCP үе үе send buffer-с өгөгдлийг хэсэгчлэн авч network давхарга руу дамжуулна. RFC 793-д тодорхойлсоноор TCP send buffer-д байгаа өгөгдлийг өөрийн хамгийн тохиромжтой байдлаар илгээх ёстой гэж заасан байдаг. Мөн нэг удаа send buffer-с авах дата нь **maximum segment size (MSS)**-р тодорхойлогдоно. MSS нь ихэнхдээ анх тус машины сүлжээгээр илгээж чадах хамгийн том link-layer-ийн фреймээр тодорхойлогддог бөгөөд энэ нь **maximum transmission unit, MTU** гэж нэрлэгддэг.



Зураг 3 TCP send and receive buffer

TCP клейнтийнн өгөгдлийн хэсэг бүрийг TCP header холбодог. Ингэснээр **TCP segment**-ийг үүсгэж байгаа юм. Сегментүүд сүлжээний давхрага (network layer) руу дамжуулагддаг бөгөөд энэ давхрагад бүгд тусдаа IP datagram[4] дотор битүүмжлэгддэг. Тэгснээр IP datagram сүлжээгээр илгээгддэг. Нөгөө талд TCP сегментийг хүлээж авах үед, сегментийн дата TCP холболтын receive buffer-д ордог бөгөөд аппликейшн өгөгдлийн урсгалын энэ buffer-с уншиж авдаг. Холболтын хоёр тал хоёул өөрийн гэсэн send buffer болон receive buffer-тай байдаг.

2.2. TCP сегментийн бүтэц

TCP сегмент нь толгой талбарууд болон өгөгдлийн талбараас бүрдэнэ. Өгөгдлийн талбар нь аппликейшн-ний өгөдлийн хэсгээс бүрдэх бөгөөд түүний хэмжээ нь дээр дурдсанаар MSS-ээр хязгаарлагдана. TCP зураг веб хуудас гэх мэт том хэмжээний өгөгдөл дамжуулах үед бүгд MSS-ийн хэмжээтэй жижиг хэсгүүдэд хуваагдана (сүүлийн хэсгээс бусад нь, сүүлийн хэсэг нь MSS-с бага байж болно). Гэсэн ч интерактив аппликейшнууд ихэнхдээ MSS-с жижиг өгөгдлийг дамжуулдаг. Жишээ нь Telnet шиг remote login аппликейшнуудын дамжуулж буй TCP сегментийн өгөгдлийн талбар ихэвчлэн нэг байт байдаг. TCP-гийн header хэсэг ихэвчлэх 20 байт (UDP-ийн header-с 12 байтаар том) байдаг учраас Telnet-ийн илгээж буй сегмент 21 байтийн урттай байж болно.

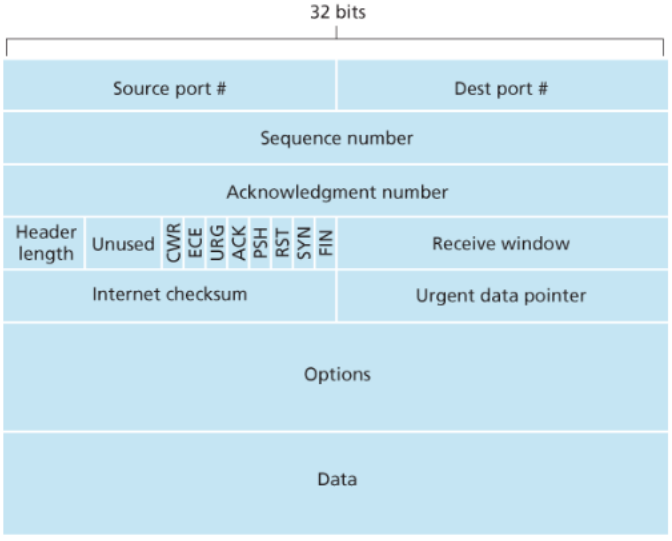
Зураг 4-д TCP сегментийн бүтцийн харуулав. UDP-тэй адил header нь sounce болон destination портын дугааруудтай. Энэ нь multiplexing болон demultiplexing хийж өгөгдлийг дээд давхрагын аппликейшн-ээс авах болон өгөхөд хэрэглэгддэг. Мөн UDP-д байсан шиг checksum талбартай. Үүнээс гадна TCP сегментийн header нь дараах талбаруудтай:

- 32 бит дарааллын дугаарын талбар болон 32 бит хүлээн авалтын[5] дугаарын талбартай. Эдгээр нь TCP илгээгч болон хүлээн авагч талууд найдвартай өгөгдлийн дамжуулалт хийхэд хэрэглэгдэнэ.

[4] Internet Protocol-р дамжуулагдах хамгийн жижиг нэгж

[5] Acknowledgment

- 16 бит receive window талбар нь урсгалыг удирдхад хэрэглэгдэнэ.
- 4 бит header-ийн урт талбар нь толгой хэсгийн уртыг хадгалана. TCP протоколын толгой хэсэгийн урт нь тогтмол биш хувьсан байдаг. Энэ нь нэмэлт option хэсэгтэй байдгаас шалтгаалдаг (Ихэнх тохиолдолд options талбар хоосон байдаг тул түгээмэл TCP header-г 20 байт байдаг).
- Хувьсах хэмжээтэй options талбар (заавал биш). Энэ талбар илгээгч болон хүлээн авагч нар MSS-г харилцан тохиролцох үед ашиглагддаг.
- 6 бит flag талбар. ACK талбар нь acknowledgment хэсэгт агуулагдаж буй утга нь үнэн зөв гэдгийн баталгааг илэрхийлдэг. Амжилттай хүлээн авсан сегмент acknowledgment талбарыг агуулдаг. RST, SYN болон FIN битүүд нь холболтын эхлэлын тохиргоо болон холболтыг дуусгахад ашиглагддаг. CWR болон ECE битүүд сүлжээний түгжрэлийн тухай тодорхой мэдэгдэл хийхэд ашиглагдана. PSH бит нь хүлээн авагч тус сегментийг шууд дээд давхрага руу дамжуулхыг мэдэгдэнэ. URG бит нь илгээгч талын дээд давхрагаас энэ сегментийн өгөгдөл чухал гэж мэдэгсэнийг илэрхийлнэ. Энэ чухал өгөгдлийн төгсгөлийн байршилийг urgent data pointer гэх 16 бит талбар хадгалдаг. TCP холболтын хүлээн авагч тал дээд давхрагадаа хэрэв чухал өгөгдөл байгаа бол төгсгөлийн байршилийг мэдэгдэх хэрэгтэй. (Практикт PSH, URG болон urgent data pointer-ууд ашиглагддаггүй бөгөөд эдгээр талбаруудыг TCP-ын бүрэн бүтэн байдалд зориулагдсан.)



Зураг 4 TCP сегментийн бүтэц

TCP холболтын найдвартай өгөгдөл дамжуулах чадварын хамгийн чухал 2 талбар бол Sequence number болон Acknowledgment number юм.

Зураг 5 Файл өгөгдлийг TCP сегментүүдэд хуваах

TCP нь өгөгдлийг бүтэцлэгдсэн биш ч дараалсан байтийн цуваа гэж авч үздэг. TCP sequence number-г дамжуулагдсан сегментеер биш харин дамжуулагдсан байтаар тоолдог. Тэгхээр сегментийн sequence number нь түүнд тээвэрлэгдэж буй өгөгдлийн эхний байтийн байрлалийг агуулдаг.

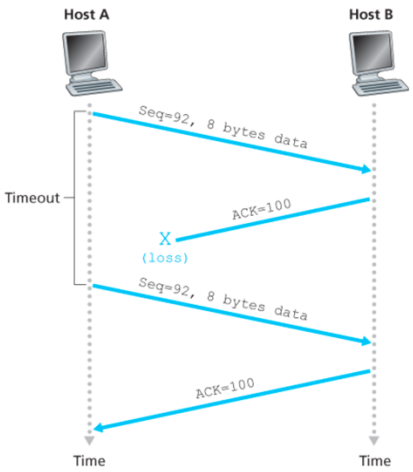
TCP acknowledgment number-ийн тухай ярихын өмнө TCP нь full-duplex буюу сервер хост болон клейнт хостууд аль аль нь өгөгдөл илгээх болон хүлээн авах чадвартай гэдгийг санах хэрэгтэй. Full-duplex холболт үүсгэхийн давуу тал нь TCP холболт үүсгэсэн А болон В хостууд байгаа бөгөөд В хостоос А хост руу өгөгдөл дамжуулагдаж байгаа гэж үзье. Энэ үед А хост В хост руу өгөгдөл илгээж байх үедээ өгөгдөл хүлээн авах чадвартай гэсэн үг. Хост А-с В-д ирж буй TCP сегмент бүр өөрийн sequence number-г агуулж байгаа бөгөөд энэ нь Хост В хост А-с үүний дараа хүлээн авах сегментийг байтийн дугаарын хадгалж байгаа. Үүнийг хост В хост А руу илгээж буй сегментийн acknowledgment number-д хийн явуулдаг.

2.3. Найдвартай өгөгдөл дамжуулалт

Интернетийн сүлжээний давхрага[6] буюу IP сервис нь найдвартай бус юм. IP нь датаграмыг эцсийн цэгт хүргэх, дарааллын дагуу хүргэх болон датаграмуудын агуулж буй өгөгдлүүдийн холбоо хамаралтай байдалд баталгаа өгдөггүй. Энэ нь IP сервисээр дамжиж буй датаграмууд дараалал нь алдагдаж, битүүдийн алдаа үүсэж түүнчлэх эцсийн цэгтээ хүргэгдэхгүй ч байж болно. Дамжууллын давхрагын сегментүүд энэ давхрагаар дамждаг тул, TCP сегментүүдэд ч бас энэ найдваргүй байдал нь нөлөөлдөг.

TCP нь IP сервис дээр тулгуурлан найдвартай өгөгдөл дамжуулалт хийдэг. TCP нь өөрийн receive buffer-д байгаа өгөгдлийн ургалын бүрэн бүтэн байдал, давхардалгүй байдал, эвдрэлгүй байдал болон дарааллын дагуу байхыг анхаардаг.

[6] Network layer



Зураг 6 TCP өгөгдлийн дахин дамжуулал

2.4. TCP холболтын удирдлага

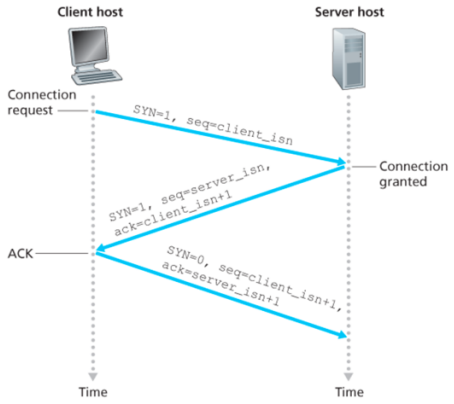
Нэг хост дээр ажиллаж буй процесс (клейнт) өөр нэг хост дээр ажиллаж байгаа процесс (сервер)-тэй холболт үүсгэх гэж байна гэж бодъё. Энэ үед клейнт процесс клейнт TCP-д сервер дээр ажиллаж буй процесс-той холболт үүсгэх гэж буйгаа мэдэгдэнэ. Дараа нь клейнт TCP дараах байдлаар серверийн TCP-тэй холболт үүсгэнэ:

- 1-р алхам. Клейнт талийн TCP эхлээд сервер талийн TCP руу тусгай TCP сегмент илгээнэ. Энэ сегмент нь аппликейшн давхрагын өгөгдөл агуулаагүй ч толгой хэсгийн flag битүүдийн нэг нь буюу SYN бит 1 гэсэн сегмент байна. Тийм учраас энэ сегментийг SYN гэж нэрлэдэг. Дээрээс нь клейнт тал санамсаргүйгээр sequence number талбарт (client_ins) эхний утгыг сонгодог. Энэ сегмент нь IP датаграм-д битүүмжлэгдэн сервер руу илгээгддэг. Sequence number санамсаргүй байдлаар сонгогдож байгаа нь зарим аюулгүй байдлийн шалтгаанаас болдог.
- 2-р алхам. Нэгэнт TCP SYN сегментийг агуулсан датаграм сервер талд ирэх үед (энэ датаграм заавал ирдэг гэж үзвэл) сервер тэр датаграм-с TCP SYN сегментийг задлан авч TCP buffer-г нөөцөлж холболттой холбоотой хувьсагчуудыг үүсгэнэ. Үүний дараа холболт зөвшөөрөгдсөн гэх сегментийг сервер TCP руу илгээнэ. Энэ сегмент мөн адил ямар ч аппликейшн давхрагын өгөгдөл агуулахгүй хэдий ч 3-н чухал мэдээллийг толгой хэсэгт агуулдаг. Эхнийх нь SYN бит 1 гэж тохируулагдсан байх бөгөөд хоёрт нь acknowledgment талбар нь client_ins+1 гэсэн утга хадгална. Сүүлийх нь сервер өөрийн эхлэл sequence number (server_ins) тоог сонгон sequence number талбарт хийн клейнт тал руу явуулдаг. Энэ нь сервер тал клейнт талд “Би амжилттай чиний холболт эхлүүлэх SYN пакетийг client_ins sequence number-тайгаар хүлээн авлаа. Миний эхлэл sequence number бол server_ins шүү” гэж хэлж

байгаатай адил. Энэ холболт зөвшөөрөгдсөн гэсэн сегментийг SYNACK сегмент гэж нэрлэдэг.

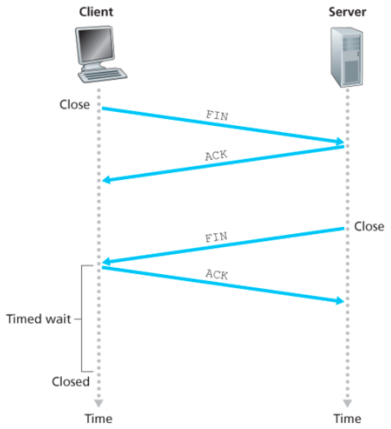
- SYNACK сегментийг хүлээн авсан клейнт тал мөн адил buffer-г нөөцлөн авч холболттой холбоотой хувьсагчуудыг үүсгэдэг. Үүний дараа клейнт сервер руу дахин нэг сегментийг илгээх ба энэ сегмент серверээс ирсэн сегментийг буцаан явуулдаг ба нэг өөрчлөгдөх зүйл нь толгой хэсэгт байрлах acknowledgment талбарыг server_ins+1 болгоно. Гэхдээ холболт эхэлсэн гэж үзэх тул SYN битийг 0 болгон явуулна. Three-way handshake-ийн энэ үед сегмент клейнт-с сервер руу дамжуулах өгөгдөлийг агуулсан байж болно.

Ингэж three-way handshake дууссанаар клейнт болон сервер хоорондоо өгөгдөл агуулсан сегментүүдийг дамжуулж чадна. Үүнээс хойш дамжуулагдах бүх сегментийг SYN бит нь 0 байна.



Зураг 7 Three-way handshake-ийн бүдүүвч зураг

TCP холболтод оролцож буй аль ч тал холболтийг дуусгаж болно. Холболт дуусах үед хоёр талд нөөцлөгдсөн хувьсагчууд болон buffer-ууд чөлөөлөгдөнө. Зураг 8-д харуулсанаар клейнт тал холболтыг дуусгахаар шийдсэн үед тусгай TCP сегмент сервер руу илгээх бөгөөд энэ сегмент нь толгой хэсгийн FIN бит 1 болж тохируулагдсан байна. Сервер энэ сегментийг хүлээн авсны дараагаар клейнт руу acknowledgment сегментээр хариулах бөгөөд дараагаар нь сервер өөрийн холболт дуусгах сегментийг клейнт тал руу илгээнэ. Харин хариуд нь клейнт acknowledgment сегмент илгээнэ. Ингэснээр хоёр тал нөөцүүдээ чөлөөлөн холболт дууссаныг илтгэнэ.

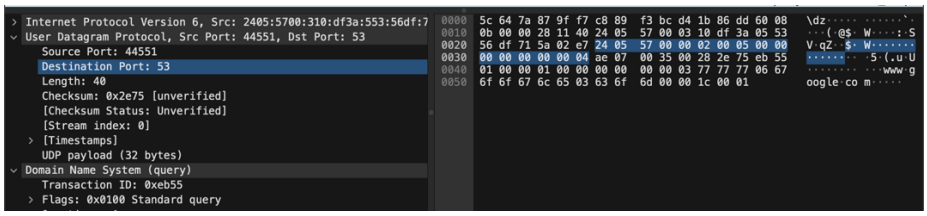


Зураг 8 TCP холболт дуусгах процесс

Даалгавар

A. User Datagram Protocol

1. Өөрийн цуглуулсан өгөгдлийн урсгалаас дурын нэг UDP пакетыг сонгоно. Энэ пакетын толгой мэдээлэл (header) хэсэгт ямар талбарууд байгааг тайлбарлана уу



Зураг 9 Өгөгдлийн урсгалд бариж авсан DNS хүсэлт

Дээрх үр дүнд UDP-ын толгой хэсэгт Source Port нь 44551 байна. Энэ нь илгээгч процессийн ашиглаж буй порт юм. Харин Destination Port нь 53 байна. Энэ нь хүлээн авах процессийн портийг илэрхийлж байна. Харин энэ сегментийн уртийг Length талбарт хадгалсан байна. Checksum буюу алдааг илэрүүлэх талбарт 0x2e75 гэсэн 16-ын тоололд буй тоо байна.

2. Wireshark дээр уг пакетын packet content хэсгийг ажиглаж, UDP пакетын толгойн мэдээллийн урт хэдэн байт болохыг тэмдэглэнэ.

Дээрх үр дүнд content хэсэг 32 байт байгаа бөгөөд үүнээс үзвэл UDP-ын тогой хэсэг 8 байт байна.

3. UDP пакетын толгойн хэсэгт Length талбар юуны уртыг илэрхийлж байгаа вэ? Өөрийн цуглуулсан UDP пакет дээрээ баталж, тайлбарлан тайланд оруулна.

Тус талбар нь энэ сегментийн нийт уртыг илэрхийлэх бөгөөд UDP payload хэсэгт WireShark програм 32 бит өгөгдөл дамжуулагдаж ирсэн гэж харуулж байна. Үүн дээр толгой хэсгийн 8 байт байгаа бөгөөд нийт урт 40 байт байна.

4. UDP payload хэсэгт хамгийн ихдээ хэд байтын өгөгдлийг дамжуулах вэ?

   Онолын хувьд payload хэсгийн хязгаар нь 65527 байт байна.

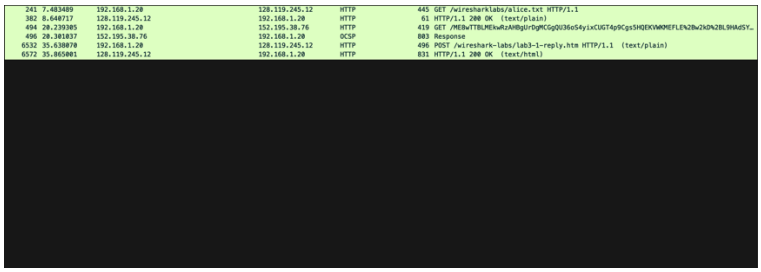
5. Илгээгчийн портын дугаар хамгийн ихдээ хэд байх боломжтой вэ?

Зураг 2 UDP сегментийн бүтэц-т харуулсанаар илгээгчийн портын дугаарыг агуулах талбар нь 16 бит бөгөөд үүнд $2^{16} - 1$ гэсэн тоо хадгалагдаж чадах бөгөөд энэ нь 65535 гэсэн тоо байна.

6. UDP протоколын дугаар (type) сүлжээний давхаргад хэд байна вэ? Сүлжээний давхаргын толгой мэдээллийг ажиглаж хариулна уу

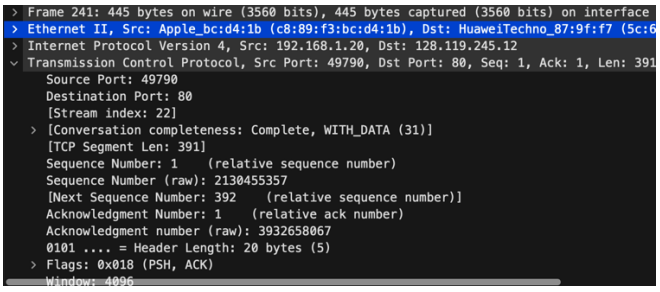
   Дээрх үр дүнгээс харвал 17 дугаартай байна.

B. Transmission Control Protocol



Зураг 10 TCP туршилт 1-ийн үр дүн

1. Илгээгч, хүлээн авагч төхөөрөмжүүдийн IP хаяг, TCP портын дугаар ямар байна вэ? Аль пакетаас уг мэдээллийг авч байгааг тайланд тусгаарай.



Зураг 11 Эхний HTTP GET хүсэлтийн дэлгэрэнгүй

Дээрх үр дүнгээс харвал Source Port буюу миний машин дээр ажиллаж буй аппликейшн буюу веб хөтөчийн порт нь 49790 байна. Харин Destination Port буюу веб серверийн порт нь 80 байна.

2. Илгээгч компьютер болон серверийн хооронд TCP холболт үүсгэж эхлэхэд ашиглагдаж байгаа TCP SYN сегментийн дарааллын дугаар (sequence number) ямар байна вэ? Уг портын дугаарыг дараа дараагийн дамжуулалд хэрхэн ашиглаж байгааг ажиглан тайланд тусгана уу.

Дээрх үр дүнгээс харвал эхний SYN сегментийн sequence number нь 2130455356 байна. Харин серверээс ирж буй SYN/ACK сегментийн acknowledgment number нь 2130455357 байна. Энэ нь эхний SYN сегментийн sequence number-г 1-ээр нэмэгдүүлсэн тоо байна. Харин sequence number нь 3932658066 байна.

3. Серверээс клиент рүү илгээгдсэн SYN ACK сегментийн дарааллын дугаар ямар байна? SYN ACK сегментийн acknowledgement талбар ямар утгатай байна вэ? Сегментийн аль утга дээр үндэслэн SYN ACK сегмент таньж байгаа вэ?

Серверээс ирж буй SYN/ACK сегментийн acknowledgment number нь 2130455357 байна. Энэ нь эхний SYN сегментийн sequence number-г 1-ээр нэмэгдүүлсэн тоо байна. Энэ сегментийн толгой хэсгийн flag талбаруудын SYN болон ACK битүүд 1 гэж тэмдэглэгдсэн байгаа учир энэ сегментийг SYN ACK гэж таниж болж байна.

4. HTTP POST мессэжийг агуулж байгаа TCP сегментийн дарааллын дугаар ямар байна вэ?

Энэ сегментийн sequence number нь 3659874654 байна.

5. TCP холболтын эхний сегментээр HTTP POST агуулж байгаа TCP сегментийг гэжүзвэлэхний 6 сегментүүдийн дарааллын дугаар ямар байна вэ? Дээрх 6 сегмент тус бүрийнилгээсэн болон acknowledgement хүлээн авсан хоорондын хугацаа болох RTT (RoundTripTime) нь сегмент тус бүр ямар байна вэ? ACK хүлээж авсны дараах EstimatedRTT утга ямар байх вэ?

Дүгнэлт

Энэ лабораторийн хүрээнд бид TCP болон UDP-ийн талаар судлан WireShark программаар тус протоколуудаар дамжиж буй пакетуудыг бариж аван шинжиллээ.