Bevezetés a hálózatok világába

Szállítási réteg

A szállítási réteg feladata, hogy két alkalmazás között ideiglenes kommunikációs munkamenetet létesítsen, valamint adatokat kézbesítsen. Az alkalmazások által generált adatokat a forrásállomáson futó alkalmazásból a célállomáson futó alkalmazáshoz kell elküldeni, függetlenül a célállomás típusától, az adatátvitelhez használt közegtől, az adatok által megtett útvonaltól, az előforduló torlódástól, valamint a hálózat méretétől. Ahogy az ábrán is látható, a szállítási réteg egyfajta kapocs az alkalmazási réteg, valamint a hálózati átvitelért felelős alsóbb rétegek között.

A szállítási réteg olyan módot biztosít a hálózaton keresztül történő adatkézbesítéshez, amellyel a fogadó oldalon pontosan összeállíthatók az adatok. A szállítási réteg gondoskodik az adatok szegmentálásáról, valamint irányítja a szegmensek újbóli összeállítását adatfolyammá. A TCP/IP esetében a szegmentációs és újra-összeállítási folyamatok megvalósítását két roppant eltérő protokoll végzi: a TCP (Transmission Control Protocol) és az UDP (User Datagram Protocol).

A szállítási réteg protokolljai elsősorban az alábbiakért felelnek:

- A forrás- és célállomásokon futó alkalmazások közötti egyedi kommunikáció nyomon követése.
- Az adatok szegmentálása a jobb kezelhetőség céljából, valamint a szegmentált adatok ismételt összeállítása a rendeltetési helyen.
- A megfelelő alkalmazás azonosítása minden egyes kommunikációs folyamhoz.

Az egyedi párbeszédek nyomon követése

A szállítási rétegben a forrás- és a célalkalmazás között áramló minden egyes konkrét adathalmazt párbeszédnek nevezünk (lásd 1. ábra). Egy állomás számos alkalmazást futtathat, amelyek egyidejűleg kommunikálnak a hálózaton keresztül. Ezek mindegyike egy vagy több alkalmazással kommunikál, amelyek egy vagy több állomáson futnak. A szállítási réteg feladata, hogy fenntartsa és nyomon kövesse az ilyen többszörös párbeszédeket.

Az adatcsomagok szegmentálása és ismételt összeállítása

Az adatokat elő kell készíteni, hogy kezelhető darabokban lehessen átküldeni az átviteli közegen. A legtöbb hálózat korlátozza az egy csomag által szállítható adatmennyiséget. A szállítási réteg protokolljai olyan szolgáltatásokat nyújtanak, amelyekkel az alkalmazások adatai megfelelő méretű adatblokkokra szegmentálhatók. Ezek a szolgáltatások tartalmazzák az egyes adatszeleteken végrehajtandó beágyazást. Minden egyes adatblokkhoz – az ismételt összeállítást megkönnyítendő – hozzáadásra kerül egy fejléc. A fejléc segítségével nyomon követhető az adatfolyam útja.

A célban a szállítási rétegnek képesnek kell lennie az adatszeletek teljes adatfolyammá való visszaalakítására, amely használhatóvá teszi őket az alkalmazási réteg számára. A szállítási rétegben működő protokollok leírják, hogy a fejlécben szereplő információkat miként lehet felhasználni az adatszeletek - alkalmazási rétegnek átadandó - adatfolyammá történő ismételt összeállításához.

Az alkalmazások azonosítása

A hálózat minden egyes állomásán számos alkalmazás és szolgáltatás futhat. Az adatfolyamok megfelelő alkalmazásoknak történő átadásához a szállítási rétegnek azonosítania kell a célalkalmazást (lásd 3. ábra). Ennek érdekében a szállítási réteg mindegyik alkalmazáshoz egy azonosítót rendel. Ezt az azonosítót portszámnak nevezzük. Minden hálózati elérést igénylő szoftverfolyamathoz egy portszám van rendelve, amely egyedi az adott állomáson. A szállítási réteg a portok alapján azonosítja az alkalmazásokat, illetve szolgáltatásokat.

Párbeszéd multiplexelése

Bizonyos adattípusok (pl.: online videoközvetítés) teljes kommunikációs folyamként történő átküldése a hálózaton keresztül felemésztheti a rendelkezésre álló teljes sávszélességet, egyúttal megakadályozhat minden más egyidejű kommunikációt. Ez a hibajavítást és a sérült adatok újraküldését is megnehezíti.

Az ábrán látható, ahogy az adatok kisebb darabokra történő szegmentálása lehetővé teszi, hogy a különböző felhasználóktól származó, több különböző kommunikáció összefűzhető (multiplexelhető) legyen egyazon hálózaton. A szállítási rétegbeli protokollok által történő szegmentálás arra is módot kínál, hogy adatokat küldjünk, illetve fogadjunk, amikor több alkalmazást futtatunk egyidejűleg valamely számítógépen.

Szegmentáció nélkül mindössze egyetlen alkalmazás fogadhatna adatokat. Tekintsünk példaként egy online videoközvetítést, amely egyedüli kommunikációs folyamként teljes egészében felemésztené az átviteli közeget ahelyett, hogy megosztva használná. A videó megtekintésének ideje alatt nem fogadhatnánk e-maileket, nem cseveghetnénk azonnali üzenetküldőn, és weboldalakra sem látogathatnánk el.

Az egyes adatszegmensek azonosításához a szállítási réteg egy bináris adatokat tartalmazó fejléct ad minden szegmenshez. Ez a fejléc bitekből álló mezőket tartalmaz. A mezőkben szereplő értékek teszik lehetővé, hogy a szállítási rétegben működő protokollok különböző adatkommunikációs felügyeleti feladatokat lássanak el.

A szállítási réteg megbízhatósága

A szállítási réteg felel a párbeszédek megbízhatósági feltételeinek biztosításáért. A különböző alkalmazások eltérő megbízhatósági feltételeket támasztanak.

Az IP csak a struktúrával, a címzéssel és csomagok irányításával törődik. Az IP nem határozza meg, hogy miként történjen a csomagok szállítása és kézbesítése. A szállítási protokollok szabják meg, hogyan menjen végbe az üzenetek átvitele az állomások között. Ahogy az ábrán is látható, a TCP/IP két szállítási rétegbeli protokollt biztosít: a TCP-t (Transmission Control Protocol) és az UDP-t (User Datagram Protocol). A TCP/IP ezeket használja az állomások közötti kommunikáció biztosítására és az adatok átvitelére.

A TCP-t egy megbízható, teljes körű szállítási rétegbeli protokoll, amely garantálja az összes adat célba érkezését. Ezzel szemben az UDP egy rendkívül egyszerű szállítási rétegbeli protokoll, amely semmilyen megbízhatóságot nem kínál.

TCP

Ahogy az már korábban is szerepelt, a TCP-t megbízható szállítási protokollnak tekintjük. Ez azt jelenti, hogy az alkalmazások közötti megbízható átvitel eléréséhez a TCP nyugtázott kézbesítést használ. A TCP-átvitel sok hasonlóságot mutat a forrástól a célig nyomon követhető postai csomagküldéssel. Ha egy FedEx (csomagküldő szolgálat) rendelést több szállítmányra bontanak szét, akkor az ügyfél online ellenőrizheti a kiszállítások sorrendjét.

A TCP esetében a megbízhatóságot három alapművelet biztosítja:

- Az adatszegmensek nyomon követése.
- A megérkezett adatok nyugtázása.
- A nem nyugtázott adatok újraküldése.

A TCP szegmensnek nevezett kis részekre darabolja szét az üzenetet. A szegmensek sorszámot kapnak, majd az IP-folyamathoz kerülnek csomagokká alakítás céljából. A TCP figyelemmel kíséri azokat a szegmens sorszámokat, melyeket az adott alkalmazástól már elküldött a célállomásnak. Ha a küldő nem kap nyugtát egy bizonyos időn belül, akkor feltételezi, hogy a szegmens elveszett, ezért azt újraküldi. Így az egész üzenetnek csak az elveszett része kerül újraküldésre, nem maga a teljes üzenet. A fogadó állomás esetében a TCP felelős az üzenetszegmensek összeillesztéséért és az alkalmazáshoz történő továbbításáért. Például az FTP (File Transfer Protocol) és a HTTP (Hypertext Transfer Protocol) is olyan alkalmazások, amelyek a TCP használatával gondoskodnak az adatok kézbesítéséről.

A küldőtől a fogadó állomásig továbbított TCP-szegmensek megtekintéséhez kattintsunk a Lejátszás gombra!

Ezek a folyamatok többletterhelést jelentenek a hálózati erőforrásokra nézve, a nyugtázás, a nyomon követés és az újraküldés miatt. A megbízhatóság biztosításához több vezérlési adat továbbítása szükséges a küldő és a fogadó állomások között. A vezérlési információk a TCP-fejlécben találhatók.

UDP

Mivel a TCP megbízhatóságot szolgáló funkciói sokkal robusztusabb kommunikációt folytatnak az alkalmazások között, így az átvitel során többletterhelést és lehetséges késést vonhatnak maguk után. Kompromisszumot kell találni a megbízhatóság, valamint a hálózati erőforrásokra rótt teher között. A megbízhatóság érdekében okozott többletterhelés csökkentheti bizonyos alkalmazások használhatóságát, vagy károsan befolyásolhatja a működésüket. Ilyen esetekben jobb választás lehet az UDP szállítási protokoll használata.

Az UDP csupán alapfunkciókat biztosít az adatszegmensek megfelelő alkalmazások között történő szállítása során, így nagyon csekély többletterhelést okoz és adatellenőrzést sem végez. Az UDP egy „legjobb szándékú” (best-effort) szállítási protokollként ismert. Hálózatos környezetben a legjobb szándékú egyet jelent a megbízhatatlannal, mivel az adatok célba érkezésekor nincs semmiféle nyugtázás. Az UDP esetében nincs olyan szállítási rétegbeli folyamat, amely tájékoztatná a küldőt a sikeres kézbesítés tényéről.

Az UDP sokban hasonlít a normál, nem ajánlott postai levél kézbesítéséhez. A levél feladója ilyenkor nincs tisztában azzal, hogy tudja-e valaki fogadni az adott levelet, ugyanakkor a postahivatal sem felelős a levél nyomon követéséért vagy a feladó tájékoztatásáért, amennyiben a levél nem éri el végcélját.

A küldőtől a fogadó állomásig továbbított UDP-szegmensek megtekintéséhez kattintsunk a Lejátszás

gombra!

Megfelelő szállítási protokoll a megfelelő alkalmazáshoz

Mind a TCP, mind pedig az UDP alkalmazott szállítási protokollok. Az alkalmazás által támasztott feltételektől függően valamelyik, esetenként mindkét szállítási protokoll használható. A fejlesztőknek kell kiválasztani, hogy melyik protokolltípus felel meg az alkalmazások által támasztott követelményeknek.

Némelyik alkalmazás esetében a sikeres feldolgozáshoz a szegmenseknek szigorúan meghatározott sorrendben kell megérkezniük. Más alkalmazások esetében az összes adatnak hiánytalanul meg kell érkezni, mielőtt annak bármely részét fel lehetne használni. Az előbbi két esetben a TCP szállítási protokollt kell alkalmazni. Például az adatbázisok, a webböngészők, az e-mail kliensek és a hasonló alkalmazások megkívánják, hogy minden adat az eredeti sorrendben és hiánytalanul érkezzen meg. Bármely elveszett adat sérülést okoz a kommunikációban, amely így hiányos és feldolgozhatatlan lesz. Épp ezért az ilyen alkalmazásokat úgy tervezték, hogy a TCP protokollt használják. Az így felmerülő hálózati többletterhelés ezekhez az alkalmazásokhoz szükségesnek tartják.

Megint más esetekben az alkalmazás elvisel ugyan bizonyos mértékű adatvesztést a hálózati átvitel során, de elfogadhatatlannak tekint bármilyen késést. Az ilyen alkalmazások számára a kisebb mértékű hálózati többletterhelés miatt az UDP jobb választás. Az UDP-t olyan alkalmazások részesítik előnyben, mint a videó- és audiofolyam, valamint az IP alapú hangtovábbítás (VoIP). Ezek esetében a nyugtázás lelassítaná a kézbesítést és az újraküldés sem kívánatos.

Például, ha a videofolyam egy vagy két szegmense nem érkezik meg, az csupán pillanatnyi zavart okoz a közvetítésben. Ilyenkor torzulhat a megjelenített kép, de a legtöbb esetben a felhasználó észre sem veszi. A másik esetben viszont az online videoközvetítés képe szét is esne, ha a céleszköznek minden elveszett adattal foglalkozni kellene, és így az újraküldésre történő várakozás kését okozna. Ilyenkor célravezetőbb lehet, ha a beérkezett szegmensek alapján előállítjuk a lehető legjobb képet, és lemondunk a megbízhatóságról.

Az UDP-t használó alkalmazásokra egy másik példa az internetrádió. Ha az üzenet egy része a hálózaton megtett út során elveszik, az nem kerül újratovábbításra. Ha néhány csomag hiányzik, a hallgató esetleg egy kis fennakadást hallhat a hangnál. Ha a TCP-t használnánk és az elvesztett csomagok újraküldésre kerülnének, az adattovábbítás szünetelne annak érdekében, hogy megkapjuk őket és ez a hangkimaradás még észrevehetőbb volna.

A TCP (Transmission Control Protocol) bemutatása

A TCP leírását először az RFC 793 szabványtervezetben adták meg. Ahogy az ábrán is látható, az adatok szegmentálását és ismételt összeállítását támogató alapfunkciókon felül a TCP az alábbiakat is biztosítja:

- Összeköttetés alapú párbeszéd, munkamenetek létesítésével.
- Megbízható kézbesítés.
- Adatok sorrendben történő újraépítése.
- Adatfolyam-vezérlés.

Munkamenet létesítése

A TCP egy összeköttetés alapú (connection-oriented, kapcsolatorientált) protokoll. Az összeköttetés alapú protokoll még a forgalom megkezdése előtt egyezteti, majd létrehozza a forrás- és céleszközök közötti állandó kapcsolatot (más néven munkamenetet). A munkamenet létrehozása felkészíti az eszközöket az egymással történő kommunikációra. A munkamenet létrehozása során az eszközök egyeztetik az adott idő alatt továbbítható forgalom mennyiségét, valamint szorosan felügyelik a két fél közötti adatkommunikációt. A munkamenet csak az összes kommunikáció befejeződése után szüntethető meg.

Megbízható kézbesítés

A TCP által használt módszer biztosítja az adatok megbízható szállítását. Hálózati szempontból a megbízhatóság azt jelenti, hogy a forrás által küldött minden egyes adatszelet célba érkezik. Számos oka lehet annak, hogy a hálózaton keresztül átvitt adatszeletek megsérülnek vagy teljesen elvesznek. A TCP azzal képes garantálni, hogy az összes szelet célba érjen, hogy a forráseszközzel újraküldeti az elveszett vagy megsérült adatokat.

A sorrend megtartásával történő kézbesítés

Mivel a hálózatok számos, különböző átviteli sebességgel rendelkező útvonalat kínálnak, előfordulhat, hogy az adatok rossz sorrendben érkeznek meg. A TCP a szegmensek megszámozásával és sorba rendezésével képes garantálni, hogy azok a megfelelő sorrendben legyenek újra összeállítva.

Adatfolyam-vezérlés (Flow Control)

A hálózati állomások korlátozott erőforrásokkal (pl.: memória, sávszélesség) rendelkeznek. Ha a TCP értesül ezen erőforrások túlzott mértékű igénybevételéről, kérheti, hogy a küldő alkalmazás csökkentse az adatátvitel sebességét. Ezt a TCP a forrás által küldött adatmennyiség szabályozásával éri el. Az adatfolyam-vezérléssel megelőzhető az adatszegmensek elvesztése a hálózaton, így elkerülhető az újraküldés.

A TCP szerepe

Amint létrejön a TCP-kapcsolat, onnantól kezdve lehetséges a párbeszéd nyomon követése az adott munkameneten belül. Mivel a TCP képes az aktuális párbeszédek nyomon követésére, állapottartó protokollnak tekintjük. Az állapottartó protokoll olyan protokoll, amely nyomon követi a munkamenet minden változását. Például, ha TCP használatával történik az adatátvitel, a küldő arra számít, hogy a címzett majd nyugtázza az adatok fogadását. A TCP nyomon követi, hogy mely információk küldése, illetve nyugtázása történt meg. Ha az adatokat nem nyugtázták, a küldő feltételezi, hogy nem érkeztek meg, így újraküldi azokat. Az állapottartó kapcsolat a munkamenet létrehozásával indul, majd a munkamenet lezárásával ér véget.

Megjegyzés: Az állapotinformációk karbantartásához olyan erőforrásokra van szükség, amelyeket egy UDP-hez hasonló állapot nélküli protokoll nem igényel.

Ezen funkciók megvalósítása többletterhet jelent a TCP számára. Ahogy az ábrán is látható, minden egyes TCP-szegmens fejlécében 20 bájt szolgál az alkalmazási rétegbeli adatok beágyazása. Ez számottevően több, mint egy UDP-szegmens esetében, amely mindössze 8 bájt ilyen adatot tartalmaz. A többletteher az alábbiakat foglalja magában:

- Sorszám (Sequence number, 32 bit) - Az adatok ismételt összeállításához használják.

- Nyugta sorszám (Acknowledgement number, 32 bit) - Jelzi, hogy az adatok megérkeztek.
- Fejléc hossza (Header length, 4 bit) - „Adatkezdetként” is ismert, jelzi a TCP-szegmens fejlécének hosszát.
- Fenntartott (Reserved, 6 bit) - Ez a mező jövőbeli célokra van fenntartva.
- Vezérlőbitek (Control bits, 6 bit) - Olyan bitkódokat (más néven jelzőbiteket) tartalmaz, amelyek a TCP-szegmens célját és funkcióját jelzik.
- Ablakméret (Window size, 16 bit) - Jelzi az egyidejűleg fogadható szegmensek méretét.
- Ellenőrzőösszeg (Checksum, 16 bit) - A szegmens fejlécének és adattartalmának hibaellenőrzésére használják.
- Sürgős (Urgent, 16 bit) - Jelzi, ha az adatok sürgősek.

TCP-t használó alkalmazás például a webböngésző, az elektronikus levelezés vagy a fájlátvitel.

Az UDP (User Datagram Protocol) bemutatása

Az UDP-t legjobb szándékú protokollnak tekintjük, amelynek leírását az RFC 768 szabványtervezetben adták meg. Az UDP egy „könnyűsúlyú” (lightweight) szállítási protokoll, amely az adatok szegmentálását és ismételt összeállítását kínálja ugyanúgy, mint a TCP, leszámítva ez utóbbi megbízhatóságát és adatfolyam-vezérlési képességét. Az UDP olyan egyszerű protokoll, amelyet gyakran azzal jellemeznek, hogy mit nem tud a TCP-hez képest.

Ahogy az ábrán is látható, az alábbi tulajdonságok jellemzik az UDP-t:

- Összeköttetés-mentes (Connectionless) - Az UDP nem létesít kapcsolatot az állomások között az adatok küldését és fogadását megelőzően.
- Nem megbízható kézbesítés - Az UDP nem kínál olyan szolgáltatásokat, amelyekkel garantálható lenne az adatok megbízható szállítása. Nincsenek benne olyan folyamatok, amelyek adatvesztés és -sérülés esetén az újraküldést kérnének a feladótól.
- Az adatok helyreállítása nem sorrendben történik - Esetenként előfordul, hogy az adatok nem a küldési sorrendben érkeznek meg. Az UDP semmilyen módszerrel nem rendelkezik az adatok eredeti sorrendjének helyreállításához. Az adatokat egyszerűen érkezési sorrendben kézbesíti az adott alkalmazásnak.
- Nincs adatfolyam-vezérlés - Az UDP semmilyen módszerrel nem rendelkezik a forrás által küldött adatmennyiség vezérléséhez, amellyel elkerülhető lenne a céleszköz túlterhelése. A forrás elküldi az adatokat. Ha ez a fogadó állomást túlságosan igénybe veszi, akkor az nagy valószínűséggel eldobja az adatokat, amíg nem szabadul fel elegendő erőforrás. Az UDP - a TCP-vel ellentétben - nem rendelkezik semmilyen módszerrel az eldobott adatok automatikus újraküldéséhez.

Az UDP szerepe

Ahogy az ábrán is látható, az UDP bár nem rendelkezik a TCP-nél alkalmazott megbízhatóságot segítő és adatfolyam-vezérlési módszerekkel, az alacsony többletterhet jelentő adatkézbesítés ideális szállítási protokollá teszi olyan alkalmazások számára, amelyek képesek elviselni némi adatvesztést. Az UDP-adategységeit datagramnak nevezik, melyeket a „legjobb szándékkal” továbbít. UDP-t használó alkalmazás például a DNS, az online videoközvetítés, valamint az IP-alapú hangátvitel (VoIP).

Az élő videó és hang hálózaton keresztüli továbbításával szemben támasztott egyik legfontosabb feltétel a gyors adatáramlás. A video- és hangalkalmazások képesek elviselni az adatvesztést úgy, hogy annak kicsi vagy egyáltalán nem érzékelhető hatása legyen, így tökéletesen illeszkednek az UDP-protokollhoz.

Az UDP állapot nélküli protokoll, ami azt jelenti, hogy sem a kliens, sem pedig a szerver számára nem kötelező az adott munkamenet állapotának nyomon követése. Ahogy az ábrán is látható, az UDP nem foglalkozik a megbízhatóság és az adatfolyam-vezérlés kérdésével. Ha az adatok elvesznek vagy rossz sorrendben érkeznek, az UDP nem képes az adatok helyreállítására és sorbarendezésére. Amennyiben az UDP szállítási protokoll használata mellett mégis szükség van a megbízhatóságra, azt már az alkalmazásnak kell lekezelnie.

Az egyidejű kommunikációk szétválasztása

A szállítási réteg feladata, hogy képes legyen több, egyidejűleg zajló, különböző szállítási igénnyel rendelkező kommunikáció szétválasztására és kezelésére. Vegyük példaként egy felhasználót, aki a számítógépén keresztül kapcsolódik a hálózatra. Ez a felhasználó egyidejűleg küld és fogad e-maileket, valamint azonnali üzeneteket, közben weboldalakat nézeget és IP-alapú (VoIP) telefonhívásokat is bonyolít. A futó alkalmazások mindegyike adatokat küld és fogad a hálózaton keresztül egyazon időben, pedig eltérő megbízhatósági elvárásaik vannak. Továbbá, a telefonhívás adatai nem kerülnek át a webböngészőbe, és az azonnali üzenetek szövege sem jelenik meg egy e-mailben.

Megbízhatósági szempontból a felhasználók elvárják, hogy az e-mailek és a weboldalak hiánytalanul megérkezzenek, és teljes egészükben megjelenjenek, mivel csak így tekinthetők hasznos információnak. Az e-mailek és weboldalak betöltése során előforduló jelentéktelen késések általában elfogadhatók, ha a végeredmény egészben és hibátlanul jelenik meg. Ebben a példában a hálózat felügyeli a hiányzó információk újraküldését és pótlását, a végeredményt pedig egészen addig nem jeleníti meg, amíg meg nem érkezett minden, az összeállítás pedig nem hibátlan.

Ezzel szemben, az esetenként előforduló kisebb hangkimaradásokat egy telefonbeszélgetés során elfogadhatónak tekintjük. Még ha el is veszik néhány szótöredék, a szövegkörnyezetből ki tudjuk következtetni a hiányzó hangokat, vagy megkérhetjük a másik személyt, hogy ismételje meg, amit mondott. Inkább ezt részesítjük előnyben, minthogy nagyobb késéseknek legyünk kitéve, ami a hiányzó szegmensek hálózat általi kezeléséből és újraküldéséből ered. Ebben a példában a felhasználó, nem pedig a hálózat végzi a hiányzó információk újraküldését és pótlását.

Ahhoz, hogy a TCP és az UDP kezelje a változó feltételeket támasztó, egyidejűleg zajló párbeszédeket, mindkét szolgáltatásnak nyomon kell követni a különféle kommunikációt folytató alkalmazásokat (lásd ábra). Az alkalmazások adategységeinek megkülönböztetésére a TCP- és UDP-fejlécben találhatók olyan mezők, melyek azonosítják az alkalmazást. Ezek az egyedi azonosítók a portszámok.

TCP és UDP portcímzés

A forrás- és célport minden szegmens és datagram fejlécében szerepel. A forrásport egy szám, amely a helyi gépen futó, a kommunikációt kezdeményező alkalmazáshoz van rendelve. A célport pedig egy olyan, a kommunikációhoz tartozó szám, amely a távoli gépen futó célalkalmazáshoz van rendelve (lásd ábra).

Ha egy üzenet kézbesítésre kerül TCP vagy UDP segítségével, a protokollok és a kért szolgáltatások azonosítása egy portszámmal történik. A port egy azonosítószám minden egyes szegmensben, amely a párbeszédek és a kért célszolgáltatások nyomon követésére szolgál. Minden üzenet, melyet az

állomás elküld, tartalmaz egy forrás- és egy célportot.

Célport

A kliens elhelyez egy cél portszámot a szegmensben, hogy közölje a célszerverrel, milyen szolgáltatást kér. Például, a 80-as port a HTTP-t, vagyis a webszolgáltatást azonosítja. Amikor a kliens célportként a 80-as portot adja meg, az üzenetet fogadó szerver tudja, hogy webszolgáltatást kértek. Egy kiszolgáló egyidejűleg több szolgáltatást is kínálhat, például webszervert a 80-as porton, miközben FTP-kapcsolat létrehozását is engedélyezi a 21-es porton.

Forrásport

A forrás portszámot véletlenszerűen generálja a küldő a két eszköz közötti párbeszéd azonosítására. Ez egyidejűleg több párbeszédet tesz lehetővé. Például, egy eszköz számos HTTP-szolgáltatáskérést küldhet a webszervernek egyazon időben. Az elkülönített párbeszédek nyomon követése a forrásportokon alapszik.

A forrás- és célportok a szegmensben kerülnek elhelyezésre. A szegmensek ezt követően egy IP-csomagba ágyazódnak be. Az IP-csomag tartalmazza a forrás és a cél IP-címét. A forrás- és célállomás IP-címének, valamint portszámainak kombinációját socket-nek vagy szoftvercsatornának nevezzük. A socket használatos a szerver és a kliens által kért szolgáltatás azonosítására. Naponta állomások ezrei kommunikálnak ezernyi különböző szerverrel. Ezeket a kommunikációkat a socket azonosítja.

A szállítási réteg portszámai a hálózati réteg IP-címével kombinálva egyértelműen azonosítanak egy alkalmazást, mely egy meghatározott állomáson fut. A portszám és IP-cím kombinációt nevezzük socket-nek. Egy socket-pár, mely tartalmazza a forrás és a cél IP-címét, valamint portszámát, egyértelműen azonosítja a két állomás közötti párbeszédet.

Egy kliens socket, 1099-es portszámmal például a következőképpen nézhet ki: 192.168.1.5:1099

A webszerverhez tartozó socket pedig ilyen lehet: 192.168.1.7:80

Ezek együttesen egy socket-párt alkotnak: 192.168.1.5:1099, 192.168.1.7:80

A socket-ek létrehozásával a kommunikációs végpontok ismertté válnak, így így az adatok eljuthatnak az egyik állomás alkalmazásától egy másik állomás alkalmazásáig. A socket-ek teszik lehetővé, hogy a kliensállomáson futó folyamatokat, valamint hasonlóképpen egy szerverfolyamat kapcsolatait megkülönböztessük egymástól.

Egy klienskérés forrásportjának generálása véletlenszerűen történik. A generált portszám a kérést indító alkalmazás címének felel meg. A szállítási réteg nyomon követi a forrásportot és a kérést kezdeményező alkalmazást, így a válasz a megfelelő alkalmazáshoz érkezik vissza. A kérést indító alkalmazás portszáma szerepel majd a szervertől visszaérkező válasz célportjaként.

A portszámok kiosztását az IANA (Internet Assigned Numbers Authority) végzi. Az IANA egy szabványügyi testület, amely a különféle címzési szabványok engedélyezéséért felel.

A portszámoknak különböző típusai vannak (lásd 1. ábra):

- Jól ismert portok (0 és 1023 közötti számok) - Ezek a szolgáltatások és alkalmazások részére fenntartott portszámok. Olyan, gyakran használt alkalmazásokhoz tartoznak, mint a HTTP (webszerver), az IMAP/SMTP (e-mail szerver), valamint a Telnet. Azzal, hogy a szerveralkalmazásokhoz jól ismert portokat rendelünk, a kliens oldali alkalmazásokat úgy lehet

programozni, hogy a megadott porthoz kapcsolódva vegye igénybe a kapcsolódó szolgáltatást.

- Bejegyzett portok (1024 és 49151 közötti számok) - Ezek a felhasználói folyamatokhoz és alkalmazásokhoz rendelt portszámok. Főleg az ügyfél által telepített egyedi alkalmazások kapják, nem pedig olyan elterjedt programok, amelyekhez a jól ismert portszámok tartoznak. Ha ezek a portok nincsenek egy szerver erőforráshoz rendelve, akkor a kliens által dinamikusan kiválasztott forrásportként is használhatók.
- Dinamikus vagy privát portok (49151 és 65535 közötti számok) - Rövid életű portnak is nevezik, mivel általában akkor rendelik hozzá dinamikusan egy ügyfélalkalmazáshoz, amikor az kapcsolatot kezdeményez egy szolgáltatással. A dinamikus portot leggyakrabban a kliensalkalmazás azonosítására használják a kommunikáció során, míg az ügyfél a jól ismert port használatával kapcsolódik a szervertől igényelt szolgáltatáshoz. Szokatlan, ha a kliens a dinamikus tartományba eső célport használatával kapcsolódik egy szolgáltatáshoz (bár némelyik peer-to-peer fájlmegosztó programnál előfordul).

A 2. ábrán a TCP által használt néhány jól ismert és bejegyzett port látható. A 3. ábrán az UDP által használt néhány jól ismert és bejegyzett port látható.

TCP és UDP együttes használata

Némelyik alkalmazás TCP-t és UDP-t egyaránt használhat (lásd 4. ábra). Az UDP alacsony többletterhe teszi például lehetővé, hogy a DNS sok ügyfélkérést szolgáljon ki rendkívül rövid idő alatt. Ugyanakkor az is előfordulhat, hogy a kért információk megküldése a TCP megbízhatóságát igényli. A szolgáltatáshoz ebben az esetben az 53-as (jól ismert) portot használja mind a TCP, mind pedig az UDP.

Az alkalmazásokhoz rendelt portszámok aktuális listája megtalálható az IANA hivatalos weboldalán.

Néha szükséges tudni, hogy mely aktív TCP-kapcsolatok vannak nyitva, és melyek futnak egy hálózatba kötött állomáson. A netstat egy fontos hálózati segédprogram, mely ezen kapcsolatok ellenőrzésére használható. A netstat kilistázza a használt protokollokat, a helyi címeket és portszámokat, a külső címeket és portszámokat, valamint a kapcsolatok állapotát.

A tisztázatlan TCP-kapcsolatok komoly biztonsági kockázatra utalhatnak, mivel azt jelezhetik, hogy valami vagy valaki csatlakozott a helyi állomáshoz. Továbbá a szükségtelen TCP-kapcsolatok értékes rendszererőforrásokat emészthetnek fel, így lerontják a számítógép teljesítményét. A netstat parancsot kell használni az állomás nyitott kapcsolatainak vizsgálatára, amennyiben a teljesítmény visszaesését érzékeljük.

Számos hasznos opció érhető el a netstat parancshoz. Kattintsunk az 1-5. ábrák gombjaira a netstat parancs különböző kimeneteinek megtekintéséhez.

TCP és UDP szegmentálás

Egy korábbi fejezet már ismertette, hogy miként lehet az alkalmazástól származó, a különböző rétegeken keresztül lefelé továbbított adatokból olyan PDU-kat létrehozni, amelyek ezt követően már továbbíthatók az átviteli közegen keresztül. A célállomásnál a folyamat megfordul, amíg az adatok fel nem érnek az alkalmazáshoz.

Némelyik alkalmazás rendkívül nagy mennyiségű adatot továbbít, amely egyes esetekben elérheti a több gigabájtot is. Ilyen nagy méretű adatokat nem lenne praktikus egy darabban továbbküldeni. Egyrészt amíg a fájl átküldése zajlana, semmilyen egyéb forgalom továbbítása nem lenne lehetséges.

Másrészt, az ilyen nagy méretű fájlok átküldése percekig, vagy akár órákig is eltarthat. Ezen felül, bármilyen hiba előfordulásakor a teljes adatfájl elveszne, vagy újra kellene küldeni. A hálózati eszközök nem rendelkeznek olyan nagy méretű memóriapufferrel, amelyben tárolható lenne ilyen nagy mennyiségű küldött és fogadott adat. Ez a korlát az alkalmazott hálózati technológiától, valamint a használatban lévő átviteli közegtől függően eltérő lehet.

Az adatok szegmensekre történő szétbontásával biztosítható, hogy azok továbbítása az átviteli közeg korlátain belül maradjon. Továbbá garantálható, hogy a különböző alkalmazásokból származó adatok multiplexelhetők legyenek a közegen.

A TCP és az UDP eltérően szegmentál

Minden TCP-szegmens fejléce tartalmaz egy sorszámot, amely a szállítási réteg funkcióinak felhasználásával lehetővé teszi a szegmensek elküldési sorrendben történő összeállítását a célállomáson (lásd ábra). Ezzel garantálható, hogy a célalkalmazás a küldő szándékainak pontosan megfelelő formában kapja meg az adatokat.

Habár az UDP-t használó szolgáltatások szintén nyomon követik az alkalmazások közötti párbeszédeket, ezek egyáltalán nem foglalkoznak az információ átviteli sorrendjével, valamint a kapcsolat fenntartásával. Az UDP fejlécében nem szerepel sorszám. Az UDP egyszerűbb elgondoláson alapszik, így kisebb terhelést okoz a TCP-hez képest, ezért nagyobb adatátviteli sebességet eredményez.

Előfordulhat, hogy az információ nem sorrendben érkezik, mivel az egyes csomagok különböző útvonalakon juthatnak el a célhoz a hálózaton keresztül. Az UDP-t használó alkalmazásnak tolerálnia kell, ha az adatok nem az elküldés sorrendjében érkeznek meg.

TCP megbízható kézbesítés

A TCP és az UDP közötti alapvető különbség a megbízhatóság. A TCP-kommunikáció megbízhatóságát az összeköttetés alapú munkamenetek adják. Mielőtt egy TCP-t használó állomás adatokat küldene egy másik állomásnak, a TCP elindít egy kapcsolatlétesítési folyamatot a cél irányába. Az állapottartó kapcsolat lehetővé teszi a munkamenet, más néven az állomások közötti kommunikációs adatfolyam nyomon követését. Ez a folyamat garantálja, hogy mindegyik állomás tudatosan felkészüljön a kommunikációs adatfolyamra. A TCP-párbeszédhez kétirányú munkamenet létrehozására van szükség az állomások között, amit az ábrán is látható.

Miután a kapcsolat felépült, és az adatátvitel megkezdődik, a cél nyugtát küld a beérkezett szegmensekről a forrásnak. Ezek a nyugták adják a TCP-kapcsolat megbízhatóságának alapját. Ha a forrás megkap egy nyugtát, akkor tudja, hogy a kézbesítés sikeresen megtörtént, és befejezheti a kérdéses adatok nyomon követését. Ha a küldő nem kap nyugtát egy előre meghatározott időn belül, akkor újraküldi az adatokat.

A TCP használatból fakadó többletteher egy része a nyugtázásokból és újraküldésekből származó hálózati forgalom. A kapcsolatok létrehozása további szegmensek cseréjét igényli, mely szintén többletterhelést okoz. Többletterhet jelent még az egyes állomásokra nézve annak nyomon követése, hogy mely szegmensek várnak nyugtázásra, valamint melyeket kell újraküldeni.

TCP szerverfolyamatok

Az alkalmazási folyamatok futtatása szervereken történik. Egy önálló szerver is lehetővé teszi több alkalmazás egyidejű futtatását. Ezek a folyamatok egészen addig várnak, amíg egy kliens

kommunikációt nem kezdeményez valamilyen információ vagy egyéb szolgáltatás igénybevételének céljából.

A szerveren futó minden egyes alkalmazási folyamat esetében be kell állítani egy portszám használatát, amely történhet alapértelmezés szerint, de végezheti a rendszergazda is. Egy szervernek nem lehet két olyan szolgáltatása, amely ugyanannak szállítási rétegbeli protokollnak (TCP vagy UDP) ugyanahhoz portjához van rendelve. Egy webszolgáltatást és fájlátvitelt egyaránt kínáló szerveren nem állítható be mindkét alkalmazás számára ugyanazon port használata (pl.: a TCP 8080-as portja). Az aktív szerver oldali alkalmazásokhoz rendelt konkrét portokat nyitott portnak tekintjük, ami azt jelenti, hogy a szállítási réteg elfogadja és feldolgozza az ezekre a portokra küldött szegmenseket. Bármely bejövő ügyfélkérés elfogadásra kerül, amelyet megfelelő socket címmel láttak el, az adatok pedig a szerveralkalmazáshoz lesznek továbbítva. Egy szerveren számos port lehet egyidejűleg nyitva, minden aktív szerveralkalmazáshoz egy. Gyakori, hogy egy kiszolgáló egyszerre több szolgáltatást is kínál, például FTP- és webszerverként is működik.

A szerverbiztonság javításának egyik módja, hogy hozzáférést kizárólag olyan portokhoz engedélyezünk, melyek szolgáltatásait csak jogosultsággal rendelkező felhasználók vehetik igénybe.

A TCP kliens/szerver műveleteiben érintett forrás- és célportok tipikus kiosztásához nézzük végig az ábrákat!

TCP-kapcsolatok létrehozása és lezárása

Egyes kultúrákban, ha két személy találkozik, kézfogással üdvözlik egymást. Ilyenkor mindkét fél tudja, hogy a kézfogás a baráti üdvözlés jele. A hálózati kapcsolatok is hasonlóan működnek. Az első kézfogás szinkronizálást kér. A második kézfogás nyugtázza az első szinkronizálási kérést, majd egyezteti az összeköttetés paramétereit az ellenkező irányban is. A harmadik kézfogási szegmens egy nyugta a célállomás számára, amely jelzi, hogy mindkét fél egyetért az összeköttetés létrejöttében.

Ha két állomás TCP használatával kommunikál egymással, még az adatcsere megkezdése előtt létre kell hozni a kapcsolatot. Miután a kommunikáció befejeződött, a munkameneteket le kell zárni, a kapcsolatot pedig bontani kell. A kapcsolathoz és munkamenethez tartozó mechanizmusok adják a TCP megbízhatóságát. A TCP kapcsolat létrehozásának és bontásának lépéseit megtekinthetjük az ábrán.

Az állomások nyomon követik az egyes adatszegmenseket egy munkameneten belül, valamint a TCP-fejlécben szereplő adatok alapján arról is információt cserélnek, hogy mely adatok érkeztek meg. A TCP egy full-duplex protokoll, ahol minden egyes kapcsolatnak két, egyirányú kommunikációs adatfolyam, más néven munkamenet felel meg. A kapcsolat létrehozásához az állomásoknak egy háromfázisú kézfogást kell végrehajtaniuk. A TCP-fejlécben lévő vezérlőbitek jelzik a kapcsolat állapotát és előrehaladását. A háromfázisú kézfogás:

- megállapítja a céleszköz jelenlétét a hálózaton
- ellenőrzi, hogy a céleszköz aktív szolgáltatással rendelkezik, amely elfogadja az olyan célportra érkező kéréseket, amelyet a kezdeményező kliens használni kíván a munkamenet idejére
- tájékoztatja a céleszközt arról, hogy a küldő kliens kapcsolatot kíván létrehozni az adott porton

A TCP-kapcsolatok esetében mindig a kliens számítógép kapcsolódik a szerverhez. A TCP-kapcsolat létrehozásának három lépése:

1. A kezdeményező ügyfél egy kliens-szerver irányú kapcsolat létrehozását kéri a kiszolgálótól.

2. A kiszolgáló nyugtázza a kliens-szerver irányú kapcsolat létrehozását, egyúttal kéri egy szerver-kliens irányú kapcsolat létrehozását is.

3. A kezdeményező ügyfél nyugtázza a szerver-kliens irányú kapcsolat létrehozását.

A TCP-kapcsolat létrehozásának megtekintéséhez kattintsunk az ábrán látható gombokra.

A háromfázisú kézfogás folyamatának megértéséhez vessünk egy pillantást a két állomás között átküldött különféle értékekre. A TCP-szegmens fejlécén belül hat olyan 1 bites mező szerepel, amely a TCP-folyamatok kezelésére használatos. Ezek az alábbi mezők:

- URG - sürgősségi jelző
- ACK - nyugtázás
- PSH - áttöltési funkció
- RST - a kapcsolat alaphelyzetbe állítása
- SYN - sorszámok szinkronizálása
- FIN - nincs több adat a küldőtől

A háromfázisú kézfogás elemzéséhez az ACK és a SYN mezők fontosak.

A TCP-kapcsolat lezárásának elemzése

A kapcsolat lezárásához be kell állítani a szegmens fejlécében található FIN vezérlőbit értékét. Az egyes, egyirányú TCP-munkamenetek megszüntetéséhez kétirányú kézfogást kell használnunk, amely egy FIN és egy ACK szegmensből áll. Ezért egy TCP által támogatott párbeszéd bontásához, mindkét munkamenetet meg kell szüntetni, amelyhez négy adatcsere szükséges (lásd 1. ábra).

Megjegyzés: A magyarázat során az egyszerűség kedvéért a kliens és szerver fogalmakat használjuk, de a bontás folyamatát bármely, nyitott munkamenettel rendelkező állomás kezdeményezheti.

1. Amikor már nincs több átküldendő adat, a kliens egy olyan szegmenset küld, amelyben a FIN jelzőbit beállítása megtörtént.

2. A kliens-szerver irányú munkamenet bontásához a szerver egy ACK üzenet küldésével nyugtázza a FIN üzenet megérkezését.

3. A szerver-kliens irányú munkamenet bontásához a szerver egy FIN üzenetet küld a kliensnek.

4. A kliens válaszként egy ACK üzenet küldésével nyugtázza a szervertől érkező FIN üzenetet.

Amikor már nincs szállítandó adat, a kliens beállítja a szegmens fejlécében szereplő FIN jelzőbitet. Ezután a kapcsolat szerver oldali vége egy normál, adatokat tartalmazó szegmenssel válaszol, amelyben az ACK jelzőbittel érvényesített nyugtaszám igazolja, hogy minden bájt megérkezett. Miután az összes szegmens nyugtázása megtörtént, a munkamenet is lezárul.

Az ellenkező irányú munkamenet bontása is ugyanígy történik. A fogadó fél a forrásnak küldött szegmens fejlécében szereplő FIN jelzőbit beállításával közli, hogy nincs több küldendő adat. A válaszként küldött nyugta igazolja, hogy az összes adatbájt megérkezett, a munkamenet pedig lezárul.

A szegmens fejlécében szereplő FIN és ACK vezérlőbitek beállítását, ezáltal egy HTTP-kapcsolat lebontását megtekinthetjük a 2. és 3. ábrán.

A kapcsolat háromfázisú kézfogással szintén megszüntethető. Amikor már nincs több átküldendő adat, a kliens egy FIN üzenetet küld a szervernek. Amennyiben a szervernél sincs több küldendő adat, küldhet olyan választ, amelyben mindkét vezérlőbit beállítása megtörtént, egyesítve ezzel két lépést (FIN, ACK). A kliens erre egy ACK üzenettel válaszol.

TCP megbízhatóság - Sorrendben történő kézbesítés

A szegmensek sorrendjének helyreállítása

Ha egy szolgáltatás TCP-vel küld adatokat, elképzelhető, hogy a szegmensek nem a megfelelő sorrendben érnek célba. Ahhoz, hogy a fogadó fél megértse az eredeti üzenetet, a szegmensekben lévő adatokat ismét az eredeti sorrendben kell összeállítani. Ennek érdekében minden egyes csomag fejlécében szerepel egy sorszám.

A kapcsolat felépítése során a kezdősorszám (ISN) is értéket kap. Az ISN tulajdonképpen a fogadó alkalmazásnak átvitt bájtok kezdőértéke az adott munkamenet során. A kapcsolat ideje alatt továbbított adatoknak megfelelően a sorszám értéke is növekszik az átvitt bájtok számával. Az adatbájtok ilyenfajta nyomon követése lehetővé teszi az egyes szegmensek egyedileg történő azonosítását és nyugtázását. Így a hiányzó szegmensek szintén azonosíthatók.

Ahogy az ábrán is látható, a megbízhatóságot a szegmensek sorszámai azzal garantálják, hogy jelzik, miként kell ismételten összeállítani és sorba rendezni a megérkezett szegmenseket.

A fogadó TCP-folyamat a szegmensek adatait egy vételi pufferbe helyezi át. A szegmensek a sorszámuknak megfelelő sorrendben kerülnek a pufferbe, majd onnan az ismételt összeállítást követően az alkalmazási rétegbe. A nem folytonos sorszámmal érkező szegmenseket későbbi feldolgozás céljából visszatartják. Amikor megérkeznek a hiányzó bájtokat tartalmazó szegmensek, akkor ezek is sorban feldolgozásra kerülnek.

TCP megbízhatóság - Nyugtázás és ablakméret

A szegmensek beérkezésének nyugtázása

A TCP egyik feladata, hogy garantálja minden egyes szegmens célba érkezését. A célállomáson működő TCP-szolgáltatások nyugtázzák a forrásalkalmazás által küldött adatokat.

A sorszám (sequence number, SEQ) és a nyugtaszám (acknowledgement number, ACK) együttesen használatos az átvitt szegmensekben lévő adatbájtok beérkezésének nyugtázására. A SEQ értéke mutatja a munkamenet során továbbított bájtok relatív számát, beleértve a jelenlegi szegmens bájtjait is. A TCP az ACK szám visszaküldésével jelzi a forrásnak, hogy melyik bájt érkezésére számít legközelebb. Ezt várományos nyugtázásnak nevezzük.

A forrás értesül, hogy célba érkezett az adatfolyam összes bájtja, egészen az ACK szám által jelzett bájtig, de azt már nem beleértve. A forrás állomás várhatóan egy olyan szegmenset küld legközelebb, amelynek sorszáma megegyezik az ACK értékével.

Ne feledjük, hogy minden egyes kapcsolat valójában két egyirányú munkamenetnek felel meg! Ezért a SEQ és az ACK értéke mindkét irányban továbbításra kerül.

Az ábrán látható példában a bal oldali állomás adatokat küld a jobb oldali állomásnak. Jelen

munkamenet során egy 10 adatbájtot tartalmazó szegmens átküldése zajlik, amelynek fejlécében az 1-es sorszám (Seq) szerepel.

A fogadó állomás a 4. rétegbe érkező szegmensről megállapítja, hogy a sorszáma 1, adattartalma pedig 10 bájt. Ezután visszaküld egy szegmenset a bal oldali állomásnak, amelyben nyugtázza az adatok beérkezését. Ebben a szegmensben az ACK értéke 11-re van állítva, jelezvén, hogy az állomás a 11-es számú adatbájt beérkezésére számít legközelebb. Amikor a küldő állomás megkapja a nyugtát, már indíthatja is a következő adatszegmens küldését, amely a 11-es bájttal kezdődik.

A példát szemügyre véve láthatjuk, hogy amennyiben a küldő állomásnak meg kellene várni a nyugtát minden beérkező 10 bájt után, az rengeteg többletterhet jelentene a hálózatra nézve. A nyugtázásból adódó többletterhelés csökkentésének érdekében számos adatszegmens átküldhető, de nyugtázásuk egyetlen TCP-üzenettel történik meg az ellenkező irányba. Az ilyen nyugta egy olyan ACK értéket tartalmaz, amely a munkamenet során beérkezett összes bájt számán alapul. Például, ha a sorszám 2000-rel kezdődik, és 10 egyenként 1000 bájt méretű szegmens érkezik, akkor a 12001-es ACK értéket kell visszaküldeni a forrásnak.

Azt az adatmennyiséget, amelyet a forrás átküldhet, mielőtt nyugtát kellene kapnia, ablakméretnek (window size) nevezzük. Az ablakméret a TCP-fejléc egyik mezője, amely lehetővé teszi az adatfolyam-vezérlést.

TCP megbízhatóság - Adatvesztés és újraküldés

A szegmensvesztés kezelése

Bármennyire alaposan is van egy hálózat megtervezve, alkalmanként előfordul adatvesztés. Ezért a TCP módszereket biztosít a szegmensvesztések kezelésére. Ezek közé tartozik a nem nyugtázott adatszegmensek újraküldésének mechanizmusa is.

A célállomás TCP-t használó szolgáltatása rendszerint csak a folyamatos sorszámú bájtokat nyugtázza. Ha egy vagy több szegmens hiányzik, kizárólag az első folyamatos bájtsorozat nyugtázása történik meg. Például, ha a beérkezett szegmensek sorszáma 1500 és 3000, valamint 3400 és 3500 között van, az ACK értéke 3001 lesz. Ennek oka, hogy a 3001 és 3399 közötti SEQ számmal rendelkező szegmensek nem érkeztek meg.

Ha a forrásállomás TCP-je nem kap nyugtát egy előre megadott időn belül, akkor visszatér az utolsóként beérkező ACK értékéhez, és újraküldi az adatokat attól a ponttól kezdve. Az újraküldés folyamatát nem rögzíti a szabványtervezet, így az mindig a TCP konkrét megvalósításától függ.

A TCP egyik tipikus megvalósításában az állomás elküld egy szegmenset, a szegmens másolatát az újraküldési sorba helyezi, majd elindít egy időzítőt. Amikor megérkezik az adatok nyugtázása, a szegmens törlésre kerül a sorból. Ha nem érkezik meg a nyugta az időzítő lejártáig, megtörténik a szegmens újraküldése.

Az elveszett szegmensek újraküldését szemléltető animáció megtekintéséhez kattintsunk a Lejátszás gombra!

Az állomások manapság alkalmazzák a szelektív nyugtázásnak (SACK) nevezett opcionális lehetőséget is. Amennyiben mindkét állomás támogatja a szelektív nyugtázást, akkor a célállomás a nem folyamatos szegmensekben érkező bájtokat is nyugtázhatja, a másik oldalon pedig csak a hiányzó adatokat kell újraküldeni.

TCP adatfolyam-vezérlés - Ablakméret és nyugtázás

Adatfolyam-vezérlés

A TCP az adatfolyam-vezérléshez is kínál módszereket. Az adatfolyam-vezérlés azzal segíti a TCP-átvitel megbízhatóságának megőrzését, hogy mindig az adott munkamenethez igazítja az adatátvitel sebességét a forrás és a cél között. A megvalósítás része az is, hogy korlátozva van az egyszerre továbbítható adatszegmensek mennyisége, valamint az újabb adat küldése a kézbesítést igazoló nyugta visszaérkezéséig.

Az adatfolyam-vezérlés megvalósításának első lépéseként a TCP megállapítja a céleszköz által elfogadható adatszegmensek mennyiségét. Ennek céljából a TCP-fejléc tartalmaz egy 16 bites mezőt, amelynek ablakméret (window size) a neve. Ez nem más, mint egy adott TCP-munkamenet céleszköze által egyszerre fogadható és feldolgozható bájtok száma. A kezdeti ablakméret a beállítása a munkamenet felépítésekor, a forrás és cél közötti háromfázisú kézfogás alkalmával történik. Mihelyt megállapodik a két fél, a forráseszköznek az ablakméret alapján korlátoznia kell a céleszköznek küldött adatszegmensek mennyiségét. A forráseszköz kizárólag az adatszegmensek kézbesítését igazoló nyugta visszaérkezése után küldhet további adatokat az adott munkamenet során.

Amíg késik a nyugta visszaérkezése, addig a forrás nem küld további szegmenseket. Olyan időszakokban, amikor torlódik a hálózati forgalom, vagy a fogadó állomás erőforrásai túlságosan le vannak terhelve, nagyobb lehet a késés mértéke. Ahogy a késés egyre növekszik, úgy csökken az adott munkamenet tényleges adatátviteli sebessége. Ha mindegyik munkamenet adatátvitelét lassítjuk, az csökkenti az erőforrás-ütközések számát a hálózaton és a sok munkamenettel rendelkező céleszközön.

Az ábrán megtekinthető az ablakméret, valamint a nyugtázás egyszerűsített ábrázolása. A példában szereplő TCP-munkamenet kezdeti ablakmérete 3000 bájtra van állítva. Ha a feladó végzett 3000 bájt elküldésével, megvárja az ezekhez a bájtokhoz tartozó nyugta visszaérkezését, újabb adatokat csak ezt követően küld a munkamenet keretében. Miután a feladó megkapta a fogadó fél nyugtáját, további 3000 bájt küldését kezdheti meg.

A TCP azért használ ablakméreteket, hogy megpróbálja az átviteli sebességet a hálózat és a céleszköz által támogatott maximumra feltornászni, ugyanakkor minimalizálni a veszteségek és újraküldések számát.

TCP adatfolyam-vezérlés - Torlódások elkerülése

Az ablakméret csökkentése

Az adatfolyam-vezérlés egy másik módja, ha dinamikus ablakméretet használunk. Amikor a hálózati erőforrások túl vannak terhelve, a TCP csökkenti az ablakméretet, így sokkal gyakrabban igényli a beérkezett szegmensek nyugtázását. Ez ténylegesen lassítja az átviteli sebességet, mivel a forrásnak is sokkal gyakrabban kell várnia a nyugták visszaérkezésére.

A fogadó állomás az ablakméret értékének elküldésével jelzi a feladónak, hogy mennyi bájt fogadására áll készen. Ha a fogadónak le kell lassítania a kommunikáció sebességét, például a korlátozott puffermemória miatt, a nyugtázás részeként kisebb ablakméretet küld vissza a forrásnak.

Ha a fogadó állomásnál torlódás lép fel, válaszolhat a feladónak egy csökkentett ablakméretet meghatározó szegmens elküldésével (lásd ábra). A képen az látszik, hogy egy szegmens elveszett. A fogadó fél csökkentette a válasz üzenet TCP-fejlécben szereplő ablak mező értékét (window size) 3000-ről 1500-ra a jelenlegi párbeszéd során. Ebből kifolyólag a feladó is 1500-ra csökkentette az

ablakméretet.

Ha a további átvitel során nem történik adatvesztés és az erőforrások sincsenek korlátozva, a fogadó fél elkezdi növelni az ablak mező értékét. Ez csökkenti a hálózat többletterhelését, mivel így kevesebb nyugta elküldésére van szükség. Az ablakméret folyamatosan növekszik, amíg nem történik adatvesztés, az viszont az ablakméret csökkenését vonja maga után.

Az ablakméret dinamikus növelése, illetve csökkentése a TCP egyik állandó folyamata. A kimagaslóan jól működő hálózatokban egészen nagy ablakméretek is előfordulhatnak, mivel nem történik adatvesztés. Az olyan hálózatokban viszont, ahol az alapvető infrastruktúra komoly igénybevételnek van kitéve, az ablakméret valószínűleg kicsi marad.

UDP alacsony többletteher kontra megbízhatóság

Az UDP egyszerű protokoll, amely a szállítási réteg alapfeladatait látja el. Mivel nem összeköttetés-alapú és nem használja a TCP kifinomult sorszámozási, újraküldési és folyamatszabályozási mechanizmusait, sokkal kisebb többletterhelést okoz.

Ez persze nem jelenti azt, hogy az UDP-t használó alkalmazások megbízhatatlanok vagy maga a protokoll alsóbbrendűbb. Csak annyit jelent, hogy a többlet funkciókat nem a szállítási réteg biztosítja, hanem szükség esetén valahol máshol kell megvalósítani.

Bár egy tipikus hálózat összes UDP forgalma viszonylag kicsi a többihez képest, a következő alkalmazási rétegbeli protokollok mind az UDP-t használják:

- Domain Name System (DNS)
- Simple Network Management Protocol (SNMP)
- Dynamic Host Configuration Protocol (DHCP)
- Routing Information Protocol (RIP)
- Trivial File Transfer Protocol (TFTP)
- Voice over IP (VoIP)
- Online játékok

Néhány alkalmazás elvisel kismértékű adatvesztést, ilyenek például az online játékok vagy a VoIP. Ha ezek a programok TCP-t használnának, nagymértékű késéseket tapasztalnánk, amíg a TCP észleli az adatvesztést és újraküldi azokat. Az ilyen késések sokkal károsabban hatnak az alkalmazás teljesítményére, mint a kisebb adatvesztések. Némelyik alkalmazás, például a DNS, egyszerűen újraküldi a kérést, ha nem érkezik válasz, ezért nincs is szükség a TCP által garantált üzenetkézbesítésre.

Az ilyen alkalmazások számára rendkívül kívánatos az UDP alacsony többletterhelése.

UDP-adategységek újbóli összeállítása

Mivel az UDP összeköttetés nélküli protokoll, ezért a kommunikáció megkezdése előtt nem jönnek létre munkamenetek, mint a TCP esetében. Az UDP-ről azt is mondják, hogy tranzakció alapú, vagyis ha egy alkalmazás adatot szeretne küldeni, akkor egyszerűen elküldi azokat.

Sok UDP-t használó alkalmazás olyan kicsi adatmennyiséget küld, amely belefér egyetlen

szegmensbe. Ugyanakkor, néhány alkalmazás lényegesen nagyobb adatmennyiséget továbbít, amelyet több szegmensre kell szétbontani. Az UDP adategységét datagramnak nevezzük, bár a szállítási réteg adategységeinek leírásánál a szegmens és a datagram fogalmakat felváltva használjuk.

Amikor több datagramot küldünk egy célállomásnak, azok különböző útvonalat választhatnak, így előfordulhat, hogy az adatok rossz sorrendben érkeznek meg. Az UDP nem követi nyomon a sorszámokat, mint a TCP. Ahogy az ábrán is látható, az UDP semmilyen módszerrel nem rendelkezik a datagramok eredeti sorrendjének helyreállításához.

Ezért az UDP a kapott sorrendben állítja össze az adatokat, majd továbbküldi azokat az alkalmazásnak. Ha az adatsorrend lényeges az alkalmazás szempontjából, akkor az alkalmazásnak kell meghatározni a helyes sorrendet, valamint az adatok feldolgozásának módját.

UDP szerverfolyamatok és kérések

Hasonlóan a TCP-hez, az UDP-t használó szerveralkalmazásokhoz is jól ismert vagy bejegyzett portszámok vannak hozzárendelve. Amikor ezek az alkalmazások és folyamatok futnak egy szerveren, akkor csak a hozzájuk rendelt portszámra illeszkedő adatokat fogadják el. Ha olyan datagram érkezik, amelyet ezen portok egyikének címeztek, az UDP a portszám alapján továbbítja az adatokat a megfelelő alkalmazáshoz.

UDP kliensfolyamatok

A TCP-hez hasonlóan, a kliens/szerver kommunikációt itt is egy kliensalkalmazás kezdeményezi, amely adatokat kér egy szerverfolyamattól. Az UDP kliensfolyamat véletlenszerűen választ egy portszámot a dinamikus portszámok tartományából, amelyet később a párbeszéd forrásportjaként használ. A célport általában egy szerverfolyamathoz rendelt jól ismert vagy bejegyzett portszám.

A biztonsághoz a véletlenszerűen választott portszámok is hozzájárulnak. Ha a port kiválasztása kiszámítható minta alapján történik, a támadók lényegesen egyszerűbben szerezhetnek a klienshez hozzáférést úgy, hogy megpróbálnak a legnagyobb valószínűséggel nyitva lévő porthoz kapcsolódni.

Mivel az UDP esetében nem jön létre munkamenet, ezért a datagramok létrehozása rögtön azután megkezdődhet, ahogy készen állnak az adatok és megvannak a portok is. A datagramokat ezután továbbítani kell a hálózati rétegbe, hogy címzés után kiküldésre kerüljenek a hálózatba.

Miután a kliens kiválasztotta a forrás- és célportot, a tranzakció során végig ugyanaz a portpár szerepel az összes datagram fejlécében. Ami a szervertől a klienshez visszaérkező adatokat illeti, a datagramok fejlécében szereplő forrás- és célportok felcserélődnek.

Az UDP kliensfolyamatok részleteinek megvizsgálásához nézzük meg az 1-5 ábrákat!

TCP-t használó alkalmazások

A TCP megbízhatóságát és egyéb szolgáltatásait számos alkalmazás megköveteli. Ezek olyan alkalmazások, amelyek elviselik a TCP által okozott többletterhelésből adódó kisebb mértékű késést vagy teljesítménycsökkenést.

Emiatt a TCP leginkább olyan alkalmazások esetében használható, ahol fontos a megbízható szállítás és megengedett némi késés. A TCP kiváló példája annak, hogy a TCP/IP protokollkészlet különböző rétegeinek milyen konkrét szerepe van. Mivel a TCP kezeli az adatfolyam szegmensekre bontásához kapcsolódó összes feladatot, a megbízhatóságot, az adatfolyam-vezérlést, valamint a szegmensek

sorrendjének helyreállítását, ezt a terhet leveszi az alkalmazások válláról. Az alkalmazás egyszerűen átküldheti az adatfolyamot a szállítási rétegbe, majd használja a TCP szolgáltatásait.

Az ábra néhány példát tartalmaz a TCP-t használó, jól ismert alkalmazásokra:

- HTTP
- FTP
- SMTP
- Telnet

UDP-t használó alkalmazások

Az alkalmazásoknak három olyan típusa létezik, amelyek számára az UDP a legjobb választás:

- Alkalmazások, amelyek elviselnek némi adatvesztést, de megkövetelik, hogy alig vagy egyáltalán ne legyen késés.
- Alkalmazások, amelyek egyszerű kérés-válasz tranzakciókat használnak.
- Egyirányú kommunikáció, ahol a megbízhatóságot vagy meg sem követeljük, vagy le tudja kezelni az alkalmazás.

Számos videó- és multimédiás alkalmazás használja az UDP-t, ilyen például a VoIP és az IPTV. Ezek az alkalmazások oly módon képesek elviselni az adatvesztést, hogy annak nincs vagy csak minimálisan érzékelhető hatása van. A TCP megbízhatósági funkciói némi késést eredményeznek, ami érzékelhető lehet a beérkező hang és videó minőségében.

Az egyszerű kérés-válasz tranzakciókat használó alkalmazások számára szintén jó választás lehet az UDP. Ilyenkor az állomás elküld egy kérést, aztán vagy kap rá választ, vagy nem. Ide tartoznak az alábbi alkalmazások is:

- DHCP
- DNS (TCP-t is használhat)
- SNMP
- TFTP

Némelyik alkalmazás maga gondoskodik a megbízhatóságról. Ezeknek nincs szüksége a TCP szolgáltatásaira, szállítási protokollként jobban ki tudja használni az UDP-t. A TFTP is ezek közé tartozik, mivel saját módszert kínál az adatfolyam-vezérlésre, hibakeresésre, nyugtázásra, valamint a hibajavításra. Ezeket a szolgáltatásokat nem kell igénybe vennie a TCP-től.