**TTMER35 – Hálózatszimuláció**

1. ***Milyen feladatokat kell megoldani a mérési gyakorlaton?***
* Tcl szkriptnyelv használatának gyakorlása
* Network Simulator és Network Animator használatának megismerése
* FTP/TCP forgalom szimulálása
* TCP protokoll „fairness” tulajdonságának vizsgálata
* Linkhibák hatásának vizsgálata
1. ***Mire használható a Network Simulator?***

A Network Simulator egy diszkrét eseményvezérelt, csomagszintű hálózati szimulátor, forráskódja szabadon hozzáférhető. A program segítségével lehetőség van többek között útvonalválasztó, multicast vagy TCP protokoll vizsgálatára akár vezetéknélküli hálózatok esetén is.

1. ***Milyen szerepet tölt be a Tcl szkriptnyelv az ns2 használata közben?***

A szimulációs konfiguráció megadásához, és egyszeri, nagy hatékonyságot nem igénylő feladatok megoldására használatos.

Tcl szkriptet kell készíteni a hálózati komponensek definiálásra, azok összekapcsolására, valamint az előre megadott események időzítésére. Az ns programot az „ns program.tcl” szkript segítségével hívjuk meg, ami a kimeneti fájlokban előállítja a naplóbejegyzéseket és a szimulációs eredményeket.

1. ***Melyek a IP hálózatok legfontosabb szállítási protokolljai? Melyek a legfontosabb különbségek közöttük?***
* **TCP (Transmission Control Protocol)** – megbízható, sorrendhelyes full duplex bytefolyamot biztosít. Két alkalmazás között logikai összeköttetést hoz létre. Pont-többpont összeköttetést nem tud megvalósítani. A TCP menedzsmentfunkcióit tekintve befolyással van a szegmensméretre, forgalomszabályozást és torlódásvezérlést végez.
* **UDP (User Datagram Protocol)** – lehetővé teszi a felhasználóknak, hogy összeköttetés felépítése és lebontása nélkül üzenetet küldhessenek. Nem garantálja sem az üzenetek kézbesítését, sem azok sorrendtartását.
1. ***Miként tudunk globálisan egyedi módon azonosítani egy TCP összeköttetést?***

A TCP két alkalmazás között logikai összeköttetést hoz létre. A TCP mindig két végpont közötti összeköttetést jelent, pont-többpont összeköttetést TCP-vel megvalósítani nem tudunk. Tipikus példa a webböngésző és a webszerver kapcsolata. A két végpontot azok IP-száma, míg a rajtuk futó, egymással kommunikálni kívánó alkalmazásokat a portszámok (amelyekkel tulajdonképpen interfészeket jelölünk meg) azonosítják. Ez az azonosítás globálisan egyedi.

1. ***Ismertesse a TCP kapcsolatfelépítési folyamatát!***

A TCP összeköttetés orientált protokoll, így adatátvitel előtt kapcsolatot kell felépítenünk. A TCP kapcsolatot kezdeményező felet kliensnek (client host) nevezzük, míg a másik felet kiszolgálónak (server host) hívjuk. A kapcsolat létrehozása (3-way-handshake):



A kezdeti sorszámokat (ISN – Initial Sequence Number) véletlenszerűen választjuk. A kapcsolatfelépítéskor határozzuk meg a maximális szegmensméretet (Maximum Segment Size – MSS), ami a TCP szegmensben található alkalmazási rétegbeli adat maximális mérete.

1. ***Ismertesse a TCP kapcsolatlezárási folyamatát!***

Az adatfolyam utolsó bytejának átvitele után az összeköttetést le kell zárnunk. A lezárás a két fél által kölcsönösen elküldött FIN jelzőbites szegmensekből és az arra adott nyugtákból áll, vagyis alapvetően 4 üzenetet használunk, ami nem is meglepő, ha arra gondolunk, hogy a TCP full-duplex összeköttetést nyújt.



Az ábráról leolvasható, hogy előfordulhat olyan eset, amikor az egyik fél már lezárta az összeköttetést, míg az a másik oldalról nyitva marad (half open). Lehetséges az is, hogy a két fél kölcsönösen le akarja zárni az összeköttetést, vagyis a FIN szegmenst mindketten úgy küldik el, hogy a másiktól még nem kaptak ilyet.

1. ***Hogyan működik a TCP-ben a folyamvezérlés? Hogyan számítjuk a küldő ablakát?***

Az egyidőben nyugtázatlan adatmennyiséget az Ablak hirdeti meg (Advertised Window). Ennek változtatásával a célpont egyértelműen a forrás tudtára adhatja, hogy mennyi adatot képes még fogadni, vagyis megakadályozza, hogy egy gyors küldő elárassza a vevőt (folyamvezérlés - flow control). Amikor a csomagok sorrendben, hibátlanul érkeznek, akkor az ablak mérete tulajdonképpen állandó (vagy csak kissé ingadozik a késleltetett nyugta miatt), ugyanakkor a kezdőpontja folyamatosan emelkedik ("csúszik" - innen az elnevezés). Ha nem sorrendhelyesen érkeznek a csomagok, akkor a nyugta sorszáma (vagyis az ablak kezdőpontja) változatlan, ugyanakkor a mérete csökken, hiszen a sorrenden kívül érkezett csomagokat eltároljuk (amennyiben az ablakon belülre esnek, egyébként eldobjuk). Ugyancsak eldobjuk a duplikált csomagokat.

EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)

🡪 Ez adja meg, hány byte-ot küldhet még a forrás.

1. ***Mi az MSS és hogyan adjuk meg? Milyen szempontok befolyásolják értékének megválasztását?***

MSS – Maximum Segment Size

A kapcsolatfelépítéskor határozzuk meg, a TCP szegmensben található alkalmazási rétegbeli adat maximális mérete. Az MSS értéke függ a TCP implementációtól (amit meghatároz az operációs rendszer), de legtöbbször konfigurálható. Az aktuális értéket annak megfelelően szokták beállítani, hogy az IP fragmentációt elkerüljük. Az IP fragmentációra akkor kerül sor, ha az IP csomag mérete meghaladja az adatkapcsolati réteg által a PDU méretére engedélyezett maximumot (Maximum Transmission Unit - MTU).

1. ***Mire használatos az ún. "Window Scale" TCP opció?***

A TCP forgalomszabályozása során az Ablak változó mondja meg, hogy a vevő mennyi adatot képes még fogadni. A lehetséges 65535 byte a „Window Scale” opció segítségével növelhető.

1. ***Mire használatos az ún. "Timestamp" TCP opció?***

A Timestamp (időbélyeg) opció használatával lehetőségünk van arra, hogy akár nagy ablakméret mellett is megkülönböztessünk azonos sorszámmal érkező szegmenseket. Az időbélyeg időegységeket számol, a számlálót a SYN-SYN/ACK üzenetváltás során tudjuk inicializálni és szinkronizálni. A gyakorlatban néhányszor 10 és néhányszor 100 msec közötti időalapot szoktak választani. Az időbélyeg visszaküldésével arra is lehetőségünk van, hogy az RTT-t viszonylag pontosan mérni tudjuk.

1. ***Mi a pszeudófejléc és mi a szerepe?***

A pszeudófejléc sorrendben a forrás és a célpont IP címét, egy csupa 0 byte-ot, a belső protokoll jelzését 8 biten és a TCP szegmens hosszát 16 biten ábrázolva. Ezt a 96 bitet a TCP szegmens elég tesszük, az ellenőrző összeg mezőjét csupa 0-val töltjük fel, majd elvégezzük az összegzést és az eredményt beírjuk az ellenőrző összeg mezőjébe.

1. ***Hogyan működik a kumulatív nyugtázás?***

A célcsomópont kumulatív nyugtát küld, a nem nyugtázott adatokat újraküldi a forrás.

A TCP a hatékonyság növelése érdekében lehetővé teszi, hogy egyszerre több szegmens is „kint legyen a hálózaton”, azonban a célpontnak az adott porthoz rendelt tárolói végesek és a feldolgozást többnyire mégiscsak az összefüggő bytefolyamon tudjuk elvégezni, így törekednünk kell arra, hogy a szegmensek „nagyjából” abban a sorrendben érkezzek, ahogy elküldtük őket. Emiatt a nyugtázás is az addig megérkezett összefüggő adatmennyiségre utal: a nyugta sorszám annak a byte-nak a sorszáma, amely az addig vett összefüggő bytefolyam után következne. Ha a nyugták alapján a forrás arra a következtetésre jut, hogy egy szegmens elveszett, akkor azt újraküldi.

1. ***Hogyan működik a szelektív nyugtázás?***

A jelenleg érvényes ajánlások szerint az implementációknak már kötelezően támogatniuk kell szelektív nyugtázást (SACK). Ebben az esetben az Opció mezőt vesszük igénybe és egy vagy több összefüggő sorszámtartományt tudunk nyugtázni. Ez a megoldás az újraküldés hatékonyságát jelentősen javítja.

1. ***Mit jelent az Additive Increase Multiplicative Decrease? Rajzolja le két versengő TCP folyamra!***

Congestion window (cwnd) – a küldő ablakaként a vevő meghirdetett ablaka és a torlódási ablak közül a kisebbet kell figyelembe venni.

Slow Start Threshold (ssthresh) – két algoritmus közötti váltásnak a határpontja.

Torlódás elkerülés fázis: ssthresh értéke a fele annak a küldési ablakénak, amelyben a torlódást észleltük, míg a cwnd értékét az ssthresh-ra állítjuk. Ezek után minden újonnan érkező ACK-ra a cwnd értékét **MSS\*MSS/cwnd**-vel növeljük. Ez praktikusan RTT-nként 1 MSS-sel növeli a cwnd-t (*Additive Increase*). Ha újabb csomagvesztés történik és ezt a duplikált nyugták alapján észleljük, akkor a gyors újraküldési fázisba lépünk át az aktuális küldési ablak felét tekintve új ssthresh-nak és a cwnd értékét az ssthresh-ra állítjuk (*Multiplicative Decrease*), ha az újraküldési időzítő lejár, akkor a lassú indulás fázisát kezdjük újra.



1. ***Mit jelent a TCP esetén a "slow start" (lassú indulás)?***

A torlódási ablak minden nyugta után egy MSS-sel növekszik, ami a gyakorlatban a cwnd megduplázódását jelenti RTT-ként. A fázis addig tart, amíg a küldési ablak az ssthresh értékét el nem éri vagy csomagvesztést nem tapasztalunk. Ha a küldési ablak eléri az ssthresh értékét, akkor a torlódás elkerülés fázisába lépünk át. Ha csomagvesztést tapasztalunk 3 duplikált nyugta alapján, akkor a gyors újraküldési fázisba lépünk át. Ha az újraküldési időzítő lejár, akkor a lassú indulás fázisát kezdjük újra, úgy hogy az ssthresh értéke az aktuális küldési ablak fele lesz, míg a cwnd értékét 1 MSS-re állítjuk.

1. ***Mit jelent a TCP esetén a "congestion avoidence" (torlódás elkerülés)?***

A torlódás elkerülés fázist azért használjuk, hogy a hálózaton elérhető legnagyobb sebességet minél kisebb ingadozásokkal keressük meg. A lassú indulás exponenciális növekedésével szemben itt RTT-ként lineárisan nő a cwnd. Minden újonnan érkező ACk-ra a cwnd értékét MSS\*MSS/cwnd-vel növeljük. Ez praktikusan RTT-nként 1 MSS-sel növeli a cwnd-t (Additive Increase). Ha újabb csomagvesztés történik és ezt a duplikált nyugták alapján észleljük, akkor a gyors újraküldési fázisba lépünk át az aktuális küldési ablak felét tekintve új ssthresh-nak és az a cwnd értékét az ssthresh-ra állítjuk (Multiplicative Decrease). Ha az újraküldési időzítő lejár, akkor a lassú indulás fázisát kezdjük újra.

1. ***Milyen lesz a TCP összeköttetés hosszútávú képe, ha a torlódás korlátozza az átvitelt? Rajzoljon! Hogyan határozná meg a várható átlagsebességet (throughput)?***



0.75\*cwnd/RTT

1. ***Mit jelent az RTT rövidítés? Hogyan becsüljük?***

RTT – Round Trip Time: A kommunikáció két végpontja közötti kétirányú késleltetés

Az eredeti algoritmus az RTT becslésére: minden szegmens/nyugta pár esetén mértük az aktuális körbefordulási időt (SampleRTT), amelyekből egy súlyozott összeget számítottunk:

 EstimatedRTT = a\*EstimatedRTT + b\*SampleRTT, ahol a+b=1 és a~0.8-0.9, b~0.1-0.2

Karn/Partridge algoritmus: a SampleRTT nem valós információt hordoz újraküldött csomagok esetén 🡪 nem szabad figyelembe venni. Exponenciális kihátrálás (exponential backof), ami azt jelenti, hogy minden újraküldés esetén az RTO-t kétszeresére növeljük.

A torlódás nem csak az átbocsátóképességet csökkenti, hanem a végül megérkező csomagok késleltetésének a szórását is jelentősen megnöveli. Az pedig könnyen belátható, hogy ha az RTT szórása nagy, akkor az RTO-nak jóval nagyobbnak kellene lennie az RTT átlagánál. Így a következő szabályokat használjuk:

 Difference = SampleRTT - EstimatedRTT

 EstimatedRTT = EstimatedRTT + ( d \* Difference)

 Deviation = Deviation + d ( |Difference| - Deviation)), ahol d egy 0 és 1 közötti tört.

Az RTO beállításánál vegyük figyelembe a szórást:

 Timeout = u \* EstimatedRTT + q \* Deviation, ahol u = 1 és q = 4.

1. ***Mit jelent az RTO rövidítés? Hogyan képezzük az RTO-t?***

Mivel a TCP megbízható átvitelt nyújt, minden olyan szegmenst újraküldünk, amire egy meghatározott időn belül nem érkezik nyugta. Minden szegmenshez egy újraküldési időzítőt kapcsolunk, amely az RTO (Retransmission TimeOut) lejárta után újraküldi a szegmenst. Az RTO-t az RTT alapján határozzuk meg (lásd: előző kérdés).

1. ***Tegyük fel, hogy van egy olyan TCP kapcsolatunk, ahol 14 csomagot kell átvinnünk. Rajzolja fel a csomagok idő szerinti sorszámdiagramját abban az esetben, ha az X sorszámú csomag elveszik! (X értékét a mérésvezető adja meg).***

A TCP sorszámokat az idő függvényében ábrázolva szemléletes képet kapunk az adott TCP folyam jellegzetességeiről. A léptékhelyes diagramon nemcsak a torlódáskezelés egyes lépései követhetőek nyomon, hanem a TCP összeköttetésre vonatkozó jellemző mennyiségeket is le tudjuk olvasni (például: MSS, RTT-k, RTO, ablakméretek, stb.).

Figyelem: az alábbi ábra egy olyan esetet ábrázol, amikor egyetlen csomag sem vész el! Egyszóval nem 100%-ban ez a válasz a kérdésre. De ilyen jellegű rajzot kell felvenni!

