S-72.253 Datasiirto ja tietokoneverkot
Laskuharjoitus 11/1999 (viimeinen)
Ratkaisut

1. ATM ja TCP/IP -pohjaiset protokollat.

ATM-yhteydellä tieto kulkee jaettuina kiinteänmittaisiksi, 53 tavua pitkiksi datagrammeiksi (Halsall, kuva 10.7), joissa on 5 tavua ohjaustietoa. Yhteydet ovat yleensä kiinteitä tai puolikiinteitä, mutta emme tässä yhteydessä puutu tarkemmin niiden muodostamiseen eikä purkamiseen. Vaikka kysymys on datagrammeista, kehykseen kuuluu otsikkoa koskeva tarkistussumma, jonka tarkoituksena on estää kehyksen eksyminen väärään paikkaan osoitteessa olevan virheen vuoksi. Lyhyet kehykset mahdollistavat hyvin monenlaista tietoa sisältävien kehysten multipleksauksen samaan tiedonsiirtoväylään. Siirtotie on yleensä vuokrattu kiinteä johto ja kehysten muodostaminen ja purkaminen jää nykyisillä järjestelyillä siirtotien käyttäjän vastuulle. Telelaitokset tarjoavat toistaiseksi lähinnä vain lähiverkkojen yhteenkytkentäpalvelua, joka saattaa olla suurimmilla siirtonopeuksilla ATMlla toteutettu, mutta tämä ei näy käyttäjälle.

AAL-kerroksen (ATM Adaptation Layer) tarkoitus on koota lyhyet, kiinteänmittaiset kehykset suuremmiksi, vaihtuvanmittaisiksi ja kulloistakin sovellutusta paremmin palveleviksi kokonaisuuksiksi. Datasiirtoon soveltuu ehkä parhaiten kehysrakenteeltaan varsin yksinkertainen versio AAL 5 (Halsall, kuva 10.14). Tässä kehyksessä voi olla varsinaista siirrettävää tietoa korkeintaan 65535 tavua ja kehyksen pituudeksi saadaan, tarvittaessa täytetavuja lisäämällä, 48 tavun monikerta. Kehyksen lopussa on tarkistussumma, mutta tällä tasolla ei ole määritelty uudelleenlähetyskäytäntöä vaan virheistä toipuminen jää AAL-kerroksen käyttäjän huoleksi. ATM-kehyksen otsikossa on bitti, joka ilmoittaa, onko tämä kehys AAL-kehyksen viimeinen.

IP-datagrammipalvelu voidaan rakentaa AAL-palvelun päälle ainakin kahdella eri tavalla. Lähiverkkotyyppisessä ratkaisussa AAL-kerroksen päällä on (lähiverkon MAC-kerrosta vastaava) yhteensovituskerros, joka saa ATM-yhteyden näyttämään ulospäin lähiverkon (esim. Ethernet-verkon) liittymältä (Halsall, kuva 10.15). Verkon käyttäjät voivat tällöin lähettää toisilleen esim. Ethernet-kehyksiä aivan kuin kysymys olisi tavallisesta lähiverkosta. Myös IP-palvelu toteutetaan samalla tavoin kuin tavallisessa lähiverkossa, siis mm. ARP-protokollaa tarvittaessa käyttäen.

Toisessa ratkaisussa käytetään (oikeaa, ei simuloitua) lähiverkkoa, johon AAL-yhteys liittyy. Lähiverkossa IP-kehykset kulkevat normaalisti lähiverkon kehysten sisällä. Sillan tavoin toimivassa liittymäsolmussa lähiverkon kehys avataan, IP-kehys sijoitetaan suoraan AAL-kehyksen sisään ja lähetetään eteenpäin vuokralinjaa tms. nopeaa WAN-verkon yhteyttä käyttäen. Loppukäyttäjän kannalta ATMn käyttö ei tosin tuo mitään lisäetua, koska molemmat mainitut kehykset ovat datagrammipohjaisia. Kyseessä onkin yksi nopean runkoverkon käyttämä tapa multipleksoida lukuisia erityyppisiä käyttäjiä samalle siirtotielle.

2. (kotitehtävä, jaossa tuplapisteet) Ollaan tekemässä protokollaa, jota käyttäen kaksi pelaajaa voi pelata shakkia (tms.) keskenään verkon välityksellä. Mieti tietoliikenneratkaisuja hierarkian eri tasoilla. Minkälaisen verkon varaan rakentaisit yhteyden? Millaista protokollaa käyttäisit kullakin tasolla? Millainen on sovellutustason protokollan ja varsinaisen sovellutusohjelman välinen rajapinta?

Huom! Tämä ratkaisu ei enää kuulu varsinaiseen tenttialueeseen, koska sovellutustason protokollia ei ehditty lainkaan käsitellä luennolla. Ratkaisu on myös huomattavasti laajempi, kuin mitä kotitehtävävastaukselta vaaditaan. Ratkaisuun kannattaakin tutustua lähinnä siinä mielessä, että tässä ovat mukana kaikki ylemmät hierarkiatasot ja ratkaisusta saa toivottavasti protokollapinon toiminnasta jonkinlaisen kokonaiskuvan. Tätä ratkaisua ei ole mistään prujattu,

vaikka samantapaisia saattaa toki maailmalta löytyä, jos jaksaa lukea news-ryhmiä yms. Jos olet kiinnostunut työstämään tätä eteenpäin, hommasta voisi varmaan hyvin tehdä erikoistyön (EI ole pelkkää koodin vääntämistä).

Vaikka pelaajien välillä kulkevien sanomien muoto on tarkkaan määrätty, tämä ei tarkoita, että itse **sovellusohjelman** olisi oltava molemmilla pelaajilla samanlainen. Peliä voi hyvin pelata vaikka tekstipohjaisilla komennoilla tyyliin

```
>e2 e4
```

Jos kuitenkin tavoitteet ovat kunnianhimoisemmat, esim. että käyttäjän näytöllä on shakkilauta, nappuloita siirretään hiirellä jne., joudutaan tutustumaan Windowsin (tms.) varsin paksuun ohjelmoijan käsikirjaan. Tässä käsikirjassa on (turhankin) lukuisa joukko ohjelmoijan avuksi laadittuja aliohjelmia, jotka piirtävät ikkunan näytölle, avaavat menun, tutkivat, mihin kohtaan hiiren "klikkaus" osui, kirjoittavat ikkunaan, piirtävät mustia tai valkoisia neliöitä... Itse tietoliikenneohjelman tekoon tarvitaan kuitenkin lähinnä vain IO-aliohjelmia.

Nyt luonnostelemassamme ratkaisussa käyttäjän näytöllä on pelilauta nappuloineen, kaksi kelloa ja painonapit "Tie?" (ehdotan tasapeliä), "Break?" (ehdotan taukoa) ja "I surrender" (antaudun). Näytöllä voi myös olla kaksi teksti-ikkunaa, joista toiseen pelaaja voi kirjoittaa vapaamuotoisia viestejä vastapelaajalle ja toiseen ilmestyvät vastapelaajan kommentit. Lisäksi esiin ponnahtaa toisinaan viesti-ikkunoita erilaisia kysymyksiä (teksti ja painonapit "Yes" ja "No") tai ilmoituksia varten (teksti ja OK-nappi).

Tarkoitus on, että peli voidaan välillä keskeyttää ja sitä voidaan jatkaa myöhemmin. Tällöin viimeinen siirto ennen taukoa tehdään kilpailupelien tapaan "kirjekuoreen" eli pelaaja ei voi enää muuttaa sitä, mutta vastapelaaja saa siirron tietoonsa vasta pelin jatkuessa. Käynnissä oleva peli tallennetaan käyttäjän hakemistoon samaan tapaan kuin esim. tekstitiedosto. Ikonina voisi olla esim. shakkihevonen ja tiedostotyyppinä .chs. Käyttäjä voi myös kesken pelin tallentaa tilanteen ("Save"-painonappi tai menuvalinta). Tilanne tallennetaan samalla myös vastapelaajan hakemistoon.

Tietoliikenneohjelmisto on tarkoitus rakentaa internetin varaan. Internet-maailmassa sovellus rakennetaan usein suoraan TCP-käyttäjärajapinnan päälle. OSI-verkkokaavan mukainen jaottelu varsinaiseen sovellukseen sekä sovellustason, esitysmuototason ja istuntotason tietoliikenneprotokolliin auttaa kuitenkin jäsentämään ongelmaa ja pilkkomaan tehtävän helpommin hallittaviin osiin. Käytännössä eri tasot voivat hyvinkin olla saman ohjelman sisäkkäisiä aliohjelmakerroksia. Tähän ohjelmaan kuuluvat vielä TCPn aliohjelmakutsut, mutta varsinainen TCP/IP-tietoliikenneohjelmisto hankitaan erikseen ja se voi samanaikaisesti palvella muitakin käyttäjiä samaan tapaan kuin tavanomaiset IO-aliohjelmat.

Sovellustason ohjelmisto liittyy läheisesti kulloiseenkin sovellukseen, tässä siis shakkipeliin. Asiat, jotka eivät edellytä kommunikointia vastapelaajan kanssa, kuten vaikkapa shakkilaudan ja nappuloiden piirtely kuvaruudulle tai hiiren seuraaminen kuuluvat kuitenkin varsinaiseen sovellusohjelmaan, vastapelaaja haluaa tietää vain mitä nappulaa lopulta siirsimme ja minne. Palvelupyynnöt voisivat olla seuraavan tapaisia:

START.request, ehdotetaan peliä vastapelaajalle, parametreina

- vastapelaajan Internet-osoite
- tieto, halutaanko pelata mustilla vai valkoisilla (Jos molemmille pelaajille käy kumpi tahansa, arvonta tapahtuu automaattisesti. Jos molemmat pelaajat haluavat ehdottomasti pelata samanvärisillä nappuloilla, saadaan virheilmoitus ja peli päättyy.)

- haluttu peliaika minuutteina tai tieto, että halutaan pelata ilman kelloa ja
- paluuparametrina pelin tunniste, jota käytetään tästä eteenpäin kaikissa palvelupyynnöissä.

Toinen ja kolmas näistä sisältyvät vastapelaajalle lähetettävään sovellustason START-kehykseen. Vastapelaajalta saadaan aikanaan START.confirm -vastaus. Jos vastapelaaja ehdottaa eri peliaikaa, tähän suostutaan tekemällä ensimmäinen siirto tai lopetetaan peli.

MOVE.request, tavallinen siirto, parametreina lähtö- ja tuloruutu, linnoituksesta vain kuninkaan siirto, ja tarvittaessa käytetty aika. Vastapelaajalle ei lähetetä MOVE-tyyppistä kehystä, jota ei sovellustasolla ole, vaan esitetään palvelupyyntö esitysmuototasolle, jolle huolenpito osapuolten yhteisestä virtuaalisesta tietorakenteesta eli pelilaudasta kuuluu.

TIE.request, ehdotetaan tasapeliä, vastauksena saadaan aikanaan TIE.accept tai TIE.refuse -sanoma. Tämä on tehtävä ennen pelaajan omaa siirtoa.

BREAK.request, ehdotetaan taukoa, parametrina kuorisiirto, vastaukset kuten edellä. Myös keskeytysten ja uudelleenaloitusten käsittely kuuluu OSI-hierarkiassa alemmalle tasolle, joten välitetään vain palvelupyyntö esitysmuototasolle.

SAVE.request, ei keskeytetä peliä, mutta talletetaan asema varmuuden vuoksi. Käsitellään samalla tavalla kuin edellinen.

SURRENDER.request, heitetään oma kuningas jorpakkoon.

MESSAGE.request, lyhyt, vapaamuotoinen viesti vastapelaajalle.

Esitysmuototason tarkoituksena on luoda välitettävälle tiedolle standardisoitu esitysmuoto, jonka molemmat osapuolet tuntevat, siis huolehtia esim. sellaisista asioista kuin montako bittiä kokonaisluvussa on ja tuleeko eniten vai vähiten merkitsevä bitti ensin. Osapuolten omista koneista ei välttämättä kumpikaan käytä sisäisesti tätä esitysmuotoa. Tällä kertaa standardoitavia asioita on vähän, vain siirtojen esitystapa. Siirto voitaisiin esittää neljällä ISO Latin 1 -kirjaimella tyyliin "e2e4". Sekä isoja että pieniä kirjaimia saa käyttää. Palvelupyyntöjä ovat

START.request, aloituspyyntö parametreinaan

- vastapelaajan Internet-osoite, joka siis vain välitetään eteenpäin
- datakenttä, johon sijoitetaan sovellustason START-kehys, joka on nyt tavallista, vastapelaajan sovellustason ohjelmalle välitettävää dataa ja
- paluuparametrina jälleen pelin tunniste, joka luodaan esitysmuototasolla.

Pyynnön saatuaan esitysmuototason ohjelma luo pelilaudan. Tähän saadaan aikanaan vastauksena esitysmuototason START.confirm.

FINISH.request, sovellustason ilmoitus pelin päättymisestä, jolloin pelilauta voidaan hävittää. Annetaan vasta, kun myös vastapelaajan sovellustaso tietää pelin päättyneen. Ei siis välitetä vastapelaajalle.

DATA.request, tavallinen lähetyspyyntö, parametrina ylemmän tason välitettäväksi antamat tiedot (eli TIE-, SURRENDER- tai MESSAGE-kehys, mutta nyt ei välitetä siitä, mitä tiedot ovat, ne vain välitetään sellisenaan vastapuolen vastaavalle tasolle).

MOVE.request, joka muunnetaan nyt vastapelaajan esitysmuototasolle lähetettäväksi, edellä esitetyllä tavalla koodatuksi MOVE-kehykseksi. Tämän saatuaan vastapelaaja tekee siirron omalla pelilaudallaan.

BREAK.request, taukopyyntö, jolloin peliasema ja kuorisiirto tallennetaan. Kuorisiirto ilmoitetaan vastapelaajalle ENVELOPE-tyyppisen kehyksen parametrina ja vastapelaajaa pyydetään myös tallentamaan asema lähinnä alemman tason, istuntotason palvelupyynnöllä, koska asia kuuluu tälle tasolle.

SAVE.request, tallennuspyyntö eli tallennetaan peliasema ja annetaan istuntotason palvelupyyntö, jotta vastapelaaja tekisi samoin.

COMPARE.request, jolla pelaajat vertailevat tallentamiaan peliasemia (tarkistussummia) keskenään esim. uudelleenaloitustilanteissa. Parametrina tarkistussumma.

Istuntotason tehtävänä on valvoa vuorottelua vuorosuuntaisilla, half-duplex yhteyksillä, jollaisia ovat luonteeltaan juuri monet pelit sekä huolehtia tarkistus- ja uudelleenaloituspisteistä. Palvelupyyntöjä ovat

START.request, aloituspyyntö, joka aloittaa tällä tasolla kertaluonteisen istunnon, ei mahdollisesti useista istunnoista koostuvaa peliä, parametreinaan

- vastapelaajan Internet-osoite, jota lähinnä alemman tason, kuljetustason, TCP-protokolla tarvitsee IP-datagrammien osoiteosaa varten
- pelin tunniste tiedoksi vastapuolelle (kumpikin puoli generoi oman tunnisteensa)
- datakenttä, johon voidaan sijoittaa esitysmuototason START-kehys ja
- paluuparametrina istunnon tunniste, jota käytetään seuraavissa tähän istuntoon liittyvissä palvelupyynnöissä.

RESTART.request, uudelleenaloituspyyntö, eroaa edellisestä siinä, että nyt jatketaan aikaisemmin aloitettua peliä tauon tai vikatilanteen jälkeen viimeksi tallennetusta asemasta. Parametreina

- vastapelaajan internet-osoite
- vastapelaajan käyttämä pelin tunniste
- datakenttä, johon voidaan sijoittaa esitysmuototason COMPARE-kehys ja
- paluuparametrina istunnon tunniste.

SAVE.request, pyydetään vastapelaajaa tallentamaan oma asemansa. Tämä kuuluu istuntotasolla hoidettaviin asioihin, joten pyyntö aiheuttaa SAVE-kehyksen lähettämisen vastapelaajan istuntotasolle.

DATA.request, tavallinen lähetyspyyntö parametreinaan

- datakenttä, johon sijoitetaan ylemmän tason kehyksiä ja
- tieto siitä, luovutetaanko samalla pelivuoro (luovutetaan, jos tehdään siirto, ei luovuteta, jos esim. ehdotetaan tasapeliä ja jäädään odottamaan vastausta).

FINISH.request, tallennetaan asema ja lopetetaan istunto, peliä voidaan jatkaa myöhemmin.

Tämän tason palvelupyynnöt eroavat edellisistä tasoista yhdessä olennaisessa suhteessa: ne eivät varsinaisesti liity käsillä olevaan sovellukseen, vaan ovat varsin yleiskäyttöisiä ja toistuvat samanlaisina lukuisissa tiedonsiirto-ohjelmissa. Tarkoitukseen voisi siten käyttää jotakin valmista

aliohjelmapakkausta, joskaan näitä ei ole Internet-maailmassa välttämättä tehty paljoakaan, ITU-Tn protokollien pohjalle sensijaan kyllä.

Kuljetustasolla käytetään Internet-ympäristössä vakiintunutta yhteyspohjaista TCP-protokollaa, jossa on varauduttu mm. siirtovirheisiin ja kehysten katoamiseen. Yhteydenottoja odotellessaan toisen osapuolen protokolla päivystää sovitun numeroista TCP-porttia. TCP lähettää puolestaan toiselle osapuolelle IP-datagrammeja. Nämä taas voivat matkallaan käyttää hyväkseen monenlaisia paikallis- ja WAN-verkkoja. Tästä on ollut joitakin esimerkkejä edellisissä laskuharjoituksissa.