S-38.201 ATM JA MULTIMEDIA SEMINAARI, KEVÄT -97

Internet reitittimien ruuhkanhallinta

Marko Luoma

S 39279H

Teknillinen korkeakoulu, Teletekniikan laboratorio

Otakaari 5A, 02150 Espoo

Marko.Luoma@hut.fi

TIIVISTELMÄ

Internet reitittimien ruuhkanhallinta sai alkunsa Ford yhtymän sisäisen verkon ongelmista. Ruuhkatila reitittimessä syntyy, kun resurssien tarve reitittimessä ylittää tarjolla olevan resurssien määrän. Ruuhkanhallinnan menetelmät voidaan jakaa kahteen luokkaan: ruuhkatilaa ennustaviin ja ruuhkatilaan reagoiviin. Ennustavat menetelmät pyrkivät pitämään verkon/reitittimen toimintatilan teho maksimissa, kun taas reagoivat menetelmät pyrkivät palauttamaan toimintapisteen optimiin, mikäli se sen ylittää. Monia hyviä menetelmiä on kehitetty mutta yksikään ei ole saavuttanut standardin asemaa. Suurin ongelma menetelmillä on toteutuksen yksinkertaisuus ja skaalautuvuus tulevaisuuden gigabitti internettiin, jossa yhteyksien ja käyttäjien lukumäärä kasvaa eksponentiaalisesti.

1. JOHDANTO

Internet reitittimien ruuhkanhallinta sai alkunsa Ford yhtymän sisäisen verkon ongelmista. Nämä ongelmat ilmenivät, koska silloinen ARPANET perustui identtisiin linkkeihin, joita hallittiin erillisellä vuonohjauksella. Näin mitään ruuhkatilanteita ei ollut päässyt syntymään, ainakaan siinä määrin, että niihin olisi reagoitu. Fordin verkko vastaavasti oli heterogeeninen, yhdistäen useita tuotantolaitoksia eri puolilla USAta sekä sateliitin kautta laitoksen Englannissa. Heterogeenisuus syntyi siitä, että laitoksien sisällä käytettiin ethernet verkkoa ja laitoksien välillä vuokrajohtoja, joilla liikennöintinopeus oli alhaisimmillaan 1,2kbit/s.

Mitä ongelmia havaittiin ? Ensimmäiset ongelmat liittyivät TCP:n sisäiseen vuonhallintaan, ns pienten pakettien ongelma. Pienet paketit, joita syntyy esimerkiksi merkkipohjaisessa tiedonsiirrossa, 40 tavua otsikkoa ja yksi tavu hyötykuormaa, johti verkkojen ylikuormittumiseen turhalla informaatiolla. Toinen ongelmatiikka syntyi, kun verkkojen käyttö kasvoi ja aiheutti mahdollisuuden yhteyden äkillisiin viiveiden kasvuihin; mikäli yhteyden kiertoaikaviive kasvaa nopeammin kuin lähettävän TCP-prosessin mittaama kiertoaikaviive, syntyy uudelleen lähetyksiä, jotka vastaavasti kuormittavat lisää reitittimiä aiheuttaen yhä pitemmän kiertoaikaviiveen. Näin ollen kehä on sulkeutunut ja verkko takuuvarmasti nurin. [1]

Ratkaisuja ongelmiin, joita edellisessä kappaleessa esitettiin pyrittiin ratkaisemaan TCP-prosessin modifioinnilla sekä ns ‘Source Quench’ toiminnolla, jossa ruuhkautunut reititin pyytää edellistä reititintä pakottamaan lähdettä pienentämään nopeutta. Tämä informointi tapahtuu vastavuohon aina lähteeseen saakka, joka riippuen riippuen toteutuksesta pienentää nopeuttaan tai ei. Kuten saattaa arvata lähderiippuvat ratkaisut reitittimen ruuhkanhallintaan ovat aina ehdollisia; mikäli lähde ei noudata reitittimen ohjeita mitään ei tapahdu. Näin ollen kehitys johti reitittimien sisäisiin ruuhkanhallintamekanismeihin.

2. MÄÄRITELMIÄ

2.1 Määritelmä ruuhkatilalle

Ruuhkatila reitittimessä syntyy, kun resurssien tarve reitittimessä ylittää tarjolla olevan resurssien määrän. Resursseiksi lasketaan siirto-, prosessointi- ja puskurointikapasiteetti. Toinen määritelmä sille, koska ruuhkatila katsotaan syntyneeksi, perustuu toimintapisteanalyysiin. Reititin on ruuhkautunut, kun jokin sen

resursseista toimii pisteessä, joka on suurempi kuin piste, jossa resurssiteho on maksimi. Resurssiteho on määritelty seuraavan kaavan mukaisesti

$$Teho = \frac{Läpäisy^{\alpha}}{Viive}$$

α on vakio, jolla määritellään kumpaa liikennettä preferoidaan (interaktiivista vai suuri kapasiteettista). Mikäli preferoitava liikenne on interaktiivista α on pienempi kuin yksi ja mikäli suuri kapasiteettinen liikenne on etusijalla valitaan α suurempi kuin yksi.

2.2 Ruuhkanhallinta

Ruuhkatilan esiintymistä pyritään hallitsemaan prosessilla, joka on nimeltään ruuhkanhallinta. Ruuhkanhallinnan menetelmät voidaan jakaa kahteen luokkaan: ruuhkatilaa ennustaviin ja ruuhkatilaan reagoiviin. Ennustavat menetelmät pyrkivät pitämään verkon/reitittimen toimintapisteen teho maksimissa, kun taas reagoivat menetelmät pyrkivät palauttamaan toimintapisteen optimiin, mikäli se sen ylittää. Mikäli tarkoituksena on operoida minimihallinalla on ruuhkatilaan regoivat menetelmät välttämättömiä, sillä toimintapisteen rajoittaminen maksimitehon alapuolelle takaa verkon jatkuvan toiminnan, kun taas optimointi ei ole lainkaan välttämätöntä verkon toiminnalle.

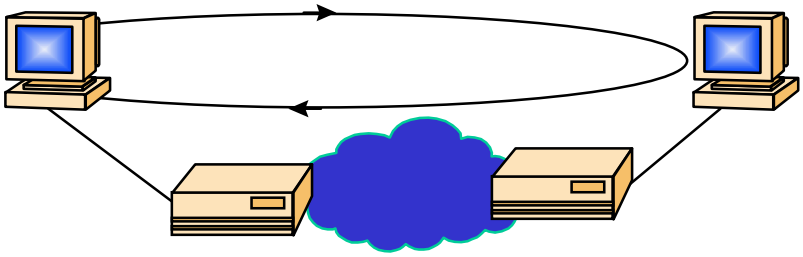
Ruuhkatila on aina riippuvainen niin lähteistä(päätelaitteista) kuin verkon kytkimistä. Lähteet tuottavat verkoon liikenteen, joka on riippuvainen käyttäjän tarpeista, verkon resursseista sekä päätelaitteen havainnoimasta verkon tilasta. Verkon tilan havainnointi perustuu, joko lähteen omaan tarkasteluun tai sitten verkon reitittimien välittämään informaatioon. Internet protokollat mahdollistavat erilaisia tapoja välittää verkon tilaa lähteille. Yksinkertaisin tapa on karsia liikennettä, joka väistämättä huomataan lähteissä. Toinen mahdollisuus on välittää lähteelle sanoma siitä mikä on tila ja näin pyytää lähdettä pienentämään nopeuttaan. Kuten menetelmistä havaitaan jakautuvat nämäkin kahteen luokkaan: yhteistoimintaa vaativiin ja yhteistoimintaa vaatimattomiin. Yhteistoiminta on välttämätöntä mikäli reititin ei kykene poistamaan ylimääräistä liikennettä vaan yrittää epätoivoisesti palvella kaikkia pyyntöjä. Toisaalta verkon operaattori ei koskaan voi luottaa siihen, että kaikki lähteet olisivat yhteistyöhaluisia ja siksi reitittimien tuleekin sisältää tapoja joilla yhteistyöhaluttomat lähteet saadaan kuriin.

2.3 Liikenteen suoritusarvojen mittaus

Mikä sitten on sopiva väli mitata ja ennustaa suoritusarvoja ? Mikäli suoritusarvoja mitataan ja ennustetaan perustuen liian tiheään tai harvaan näytteenottoon ilmenee tuloksissa väistämättä virheitä, jotka johtuvat internet-liikenteen dynamiikasta. Esimerkiksi mikäli kaksi FTP-yhteyttä toimii saman reitin yli ja niitä näytteistetään liian tiheästi, näyttävät niiden palvelun tarpeet eroavan varsin suuresti. Tämä johtuu ikkunallisten protokollien tavasta aloittaa lähetysjaksonsa eksponentiaalisella nopeuden muutoksella, mikä tasapainotilan saavutettuaan vakioituu. Liian tiheä näytteistys saattaa kuitenkin määrittää lähteen irronneen kontrollista ja pyrkiä irroittamaan lähteen verkosta. Toisaalta liian pitkä näytteistysväli mahdollistaa sen, että palveltavien yksilöiden dynamiikka jää huomaamatta, että uudet yhteydet ja päättyvät yhteydet jäävät huomaamatta. Näin ollen erheellisesti luullaan resurssien jakaantuvan vakio määrälle yhteyksiä.

Nyrkkisääntönä voidaankin pitää, että näytteistysväli olisi vähintään yhtä suuri kuin kiertoaikaviive. Tämä siksi, että reitittimien tekemät liikenteenhallintapäätökset välittyvät lähettävälle päätelaitteelle minimissään puolen kiertoaikaviiveen päästä ja päätelaitteen suorittama säätö näkyy tästä aikaisintaan puolen kiertoaikaviiveen jälkeen. Näin ollen seuraava todellinen muutos, joka riippuu reitittimen säädöstä, havaitaan aikaisintaan yhden kiertoaikaviiveen jälkeen.

2.4 Kiertoaikaviive



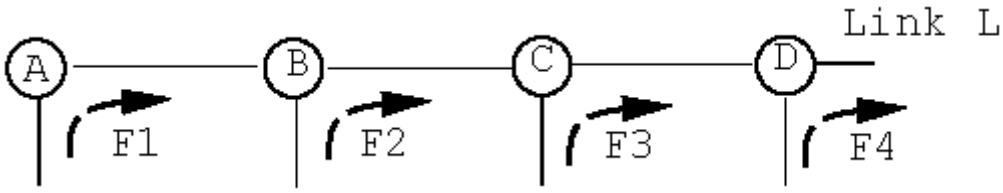
Kuva 1: Kiertoaikaviiveen määritelmä

Seuraava ongelma onkin sitten määrittää kiertoaikaviive. Mikä on reitittimen kannalta kiertoaikaviive ja miten reititin voi määrittää sen ? Yksinkertaisin tapaus olisi silloin, jos kaikki internet yhteydet kommunikoisivat yhtenevien reittien yli, jolloin kaikilla yhteyksillä olisi yhtä pitkä kiertoaikaviive. Tämä ei kuitenkaan ole totta sillä normaalitapauksessa yhteydet ovat varsin sattumanvaraisia ja suuria määriä identtisiä yhteyksiä esiintyy harvoin. Koska näin on, ei kiertoaikaviivettä voida määrittää siten, että se palvelisi optimaalisesti kaikkia yhteyksiä. Eräs ehdotus kiertoaikaviiveen määrittämiselle reitittimessä on käytää jonon regeneroitumisviivettä (allekirjoittaneen onneton suomennos termistä queue regeneration cycle). Periaate menetelmässä on hyödyntää dominoivan liikenteen dynamiikkaa. Eli mikäli jokin kiertoaikaviive dominoi liikenteen intensiteetissä, hyödynnetään sitä. Tämä dominoiva kiertoaikaviive voidaan määrittää tarkkailemalla jonon täyttymis- ja tyhjentymisprosessia. Mikäli prosessi on deterministinen, kuten ikkunallisten yhteysprotokollien tapauksessa on, voidaan tämän syklin pituudesta määrittää kiertoaikaviive.

2.5 Reiluus

Reiluus on määritelmä sille, miten oikeudenmukaisesti eri yhteyksiä kohdellaan ruuhkautuvassa reitittimessä. Mutta mikä sitten on oikeudenmukaista kohtelua ? Yksinkertaisimmillaan asiat olisivat reitittimessä, jossa kaikilla yhteyksillä olisi identtiset tarpeet. Tällöin resurssit jaettaisiin tasan kaikkien yhteyksien kesken. Reaalimaailmassa yhteyksien tarpeet kuitenkin vaihtelevat ja edellisen kaltainen tasajako ei olisi reilua. Sen lisäksi internet kehittyy kokoajan ja uudet palvelut vaativat erillaisia ominaisuuksia, kuten kiinteitä kaistanvarauksia.

Esimerkiksi mikä on seuraavassa kuvassa reilua:



Kuva 2: Parkkipaikka ongelma [9]

Onko reilua, että yksittäinen reititin jakaa resurssinsa tasan eri johtojen välillä vai että resurssit jaetaan tasan eri yhteyksien välillä. Vaikka ratkaisu tuntuukin yksinkertaiselta niin aina voi kiistellä siitä onko se välttämättä ainoa oikea.

Koko reiluus käsitettä voidaankin alkaa tarkastelemaan sosio-ekonomisista lähtökohdista; onko tarkoitus palvella kaikkia tasapuolisesti vai maksukyvyn mukaan. Mikäli kaikkia palvellaan tasapuolisesti ollaan sosialistisessa näkökannassa, jossa kaikki resurssit ovat kaikkien vapaasti hyödynntettävissä toisaalta kaikilta odotetaan myös kunnioitusta toisia kohtaan. Kapitalistisessa näkökannassa yhteyksiä käsitellään perustuen maksukykyyn, eli sen kulutat mistä maksat. Näin ollen on mahdollista suorittaa kaistan varauksia. Kapitalistista järjestelmää voidaan vielä sisällä jakaa sosiokapitalistiseen ja kapitalistiseen järjestelmään; sosiokapitalistisessa järjestelmässä uudet yhteydet pääsevät mukaan ja vanhat yhteydet saatavat kärsiä siitä, kun taas puhtaassa kapitalistisessa järjestelmässä yhteyksien tilaan ei puututa mitenkään.

Oli tilanne mikä tahansa reiluudelle on aina löydettävissä jokin politiikka ja tämän politiikan pohjalta tulisi yhteyksille jakaa resursseja. Mikäli joku yhteyksistä ei käytä resurssejaan täysin jaetaan ylijäämä muille tasapuolisesti. Tämä lopullinen jako, jossa kaikkien tarpeet tyydytetään joten kuten on nimeltään max-min reiluusjako.

3. RUUHKANHALLINTA VAATIMUKSET IN REITTIMELLE

Tämä on ote lähteestä [5], joka asettaa vaatimukset internet reitittimille. Kuten alla olevasta nähdään on ruuhkanhallinta varsin keveästi määritelty.

> *Congestion in a network is loosely defined as a condition where demand for resources (usually bandwidth or CPU time) exceeds capacity. Congestion avoidance tries to prevent demand from exceeding capacity, while congestion recovery tries to restore an operative state. It is possible for a router to contribute to both of these mechanisms. A great deal of effort has been spent studying the problem. The reader is encouraged to read* [3] *for a survey of the work. Important papers on the subject include* [2], [6], [7], *and* [8], *among others.*
>
> *The amount of storage that router should have available to handle peak instantaneous demand when hosts use reasonable congestion policies, such as described in* [7], *is a function of the product of the bandwidth of the link times the path delay of the flows using the link, and therefore storage should increase as this Bandwidth*Delay product increases. The exact function relating storage capacity to probability of discard is not known.*
>
> *When a router receives a packet beyond its storage capacity it must (by definition, not by decree) discard it or some other packet or packets. Which packet to discard is the subject of much study but, unfortunately, little agreement so far.*
>
> *A router MAY discard the packet it has just received; this is the simplest but not the best policy. It is considered better policy to*

randomly pick some transit packet on the queue and discard it (see [3]*). A router MAY use this Random Drop algorithm to determine which packet to discard.*

If a router implements a discard policy (such as Random Drop) under which it chooses a packet to discard from among a pool of eligible packets:

- *If precedence-ordered queue service (described in Section [5.3.3.1]) is implemented and enabled, the router MUST NOT discard a packet whose IP precedence is higher than that of a packet which is not discarded.*
- *A router MAY protect packets whose IP headers request the maximize reliability TOS, except where doing so would be in violation of the previous rule.*
- *A router MAY protect fragmented IP packets, on the theory that dropping a fragment of a datagram may increase congestion by causing all fragments of the datagram to be retransmitted by the source.*

To help prevent routing perturbations or disruption of management functions, the router MAY protect packets used for routing control, link control, or network management from being discarded. Dedicated routers (i.e.. routers which are not also general purpose hosts, terminal servers, etc.) can achieve an approximation of this rule by protecting packets whose source or destination is the router itself.

Advanced methods of congestion control include a notion of fairness, so that the 'user' that is penalized by losing a packet is the one that contributed the most to the congestion. No matter what mechanism is implemented to deal with bandwidth congestion control, it is important that the CPU effort expended be sufficiently small that the router is not driven into CPU congestion also.

As described in Section [4.3.3.3], this document recommends that a router should not send a Source Quench to the sender of the packet that it is discarding. ICMP Source Quench is a very weak mechanism, so it is not necessary for a router to send it, and host software should not use it exclusively as an indicator of congestion.

4. RUUHKANHALLINTA

Käytännön ruuhkanhallinta-algoritmeja on kehitetty useita. Niistä osaa käsitellään lähteessä [2]. Tässä esitellään muutama lähteen suorittamaan säätöön ja muutama reitittimen itsesuojeluun perustuva menetelmä.

4.1 Source Quench

Source Quench eli lähteen kuristus perustuu internet liikenteen hallintaprokollaan (ICMP). SQ on ainut pakollinen ruuhkan hallintamenetelmä [5]. Reititin, joka joutuu karsimaan liikennettä lähettää siitä tiedon lähettävälle päätelaitteelle käyttäen ICMP Source Quench sanomaa. Päätelaitteen tulisi reagoida tähän pienetämällä

lähetysnopeuttaan. Ongelmana on, että ICMP sanomat käsitellään IP:n päällä omana kokonaisuutena ja niillä ei välttämättä ole mitään yhteyttä yhteysprotokollaan. Näin ollen SQ-sanoman vastaanotto ei millään muotoa ole tae sille, että lähteen nopeus pienenisi. Toinen ongelma on se että SQ voidaan toteuttaa monella eri tavalla. Näin ollen päätelaite, joka vastaanottaa SQ-sanoman ei voi millään tietää mitä sanomalla yritetään viestittää (paketti pudotettu, puskurin täyttöaste ylittänyt kynnysarvon ...).

4.2 Tail Drop

Tail Drop eli häntäkarsinta on yksinkertaisin ruuhkanhallintamekanismi. Siinä rajallinen puskuri, jota palvellaan FIFO-periaatteella, ylivuotaa pitäen siten liikenteen kurissa. Ylivuotava liikenne generoi SQ-sanomia lähettäville päätelaitteille, jotka toivon mukaan vähentävät nopeuttaan. Toisaalta myös vastaanottavan päätelaitteen yhteysprotokolla reagoi pakettien katoamiseen vaatimalla uudelleen lähetystä.

Häntäkarsintaan perustuvat ruuhkanhallintamenetelmät sisältävät seuraavia ongelmia:

- Jos puskurin koko on pieni, on TCP:llä hankaluuksia päästä ‘slow-startista’, koska ikkunan kokoa kasvatetaan varsin nopeasti sekä pieni puskuri antaa heikkon vasteen verkon transienteille.
- Jos puskurin koko on suuri, koituu reeaaliaikaiselle liikenteelle kohtuuttoman suuri viive sekä verkossa ei etene hetken päästä ollenkaan liikennettä, mikäli linkin nopeus on pieni. Mikäli linkin nopeus on puskurin kokoon nähden pieni jonottavat paketit lähes niiden elinajan (TTL), jonossa mistä seuraa se, että paketit lähtevät TTL arvolla yksi ja poistetaan välittömästi seuraavassa reitittimessä. [2]
- Globaali synkronisaatio, joka seuraa siitä, että jonon täyttyessä kaikki yhteydet kokevat pakettihukkaa miltei yhtäaikaisesti ja pienentävät ikkunaa. Tämä johtaa siihen, että liikenne putoaa vain muutamaan prosenttiin tehollisesta arvostaan.
- Selkeä painotus tasaiselle liikenteelle, koska jonon ollessa kasvussa todennäköisyys, että purskeiselle yhteydelle riittää puskurikapasiteettia pienenee.

4.3 Random Drop

Random Drop eli satunnaiskarsinta perustuu olettamuskeen, että paketti joka valitaan satunnaisesti kuuluu lähteelle N todennäköisyydellä, joka on suoraan verrannollinen lähteen keskinopeuteen. Paketti valitaan tällä kertaa jonon sisältä eikä hännältä. Näin jonosta karsimalla vapautetaan paikkoja uusille paketeille. Koska menetelmä on satunnainen, kärsivät myös ne yhteydet, jotka kuluttavat alle osuutensa resursseista, pakettihukkaa.

Satunnaiskarsintaa voidaan käyttää sekä ruuhkaa ennakoivana että ruuhkaan reagoivana menetelmänä. Reagointi perustuu paketin karsintaan jokaisen uuden paketin saapuessa, kun taas ennakoinnissa jokainen tuleva paketti aiheuttaa karsinta todennäköisyyden laskemisen ja sen perusteella toimimisen. Näin ollen ennakoiva menetelmä on kahteen kertaan satunnainen prosessi. Satunnaiskarsintaan perustuvia menetelmiä on kehitelty viime aikoina paljon eteenpäin; viimeisin versio on Random Early Detection eli RED. [4]

REDissä hyödynnetään puskurin keskimääräistä tilaa karsinta todennäköisyyden laskemiseen. Keskimääräinen puskurin täyttöaste lasketaan EWMA-suodattimella eli eksponentiaalisesti painotetulla liukuvan keskiarvon suodattimella.

$$avg_t = \alpha q_t + (1-\alpha) avg_{t-1}$$

avg	puskurin keskimääräinen täyttöaste hetkellä t
q	puskurin täyttöaste hetkellä t
α	painokerroin

Käyttämällä keskimääräistä puskurin täyttöastetta ollaan eliminoitu verkon transienttiilmiöiden vaikutus pakettien karsintaan.

RED gateway algorithm:

```
for each packet arrival
   calculate the new average queue size avg
   if min_th ≤ avg < max_th
      calculate probability p_a
      with probability p_a:
         mark/drop the arriving packet
   else if max_th < avg
      drop the arriving packet
```

Variables:

avg: average queue size
p_a: packet marking/dropping probability

Fixed parameters:

min_{th}: minimum threshold for queue
max_{th}: maximum threshold for queue

Kuva 3: RED algoritmi

RED algoritmin edut häntäkarsintaan verrattuna ovat selvät.

- RED ei aiheuta globaalia synkronoitumista sillä sen toiminta perustuu eston havainnointiin jo ennen kuin sitä edes esiintyy. Näin ollen lähteitä informoidaan hallitusti eikä purskeen omaisesti.
- RED algoritmi takaa puskuriin kapasiteettia purkeiden varalle ja lisäksi purskeen todennäköisyys tulla karsituksi ei kasva lineaarisesti; sillä purske ei aikaansaa keskiarvon välitöntä nousua (EWMA).

4.4 DECbit

DECbit on menetelmä, jossa lähteitä informoidaan verkon ruuhkatilasta perustuen yhden bitin välittämiseen otsikossa. Ruuhkabitti asetetaan silloin, kun reititin havaitsee keskimääräisen jonon pituuden ylittäneen edellisellä regeneroitmisjaksolla arvon yksi.

4.5 Fair Queuing

Fair Queuing on menetelmä, jossa ryhmälle yhteyksiä muodostetaan rinnakkaiset jonot. Nämä rinnakkaiset jonot ovat tietyssä mielessä autonomisia, eli niillä ei ole suoraa interaktiota keskenään. Näin yhteydet saavat suojaa toisiaan vastaan mikäli joukossa on yhteyksiä, jotka eivät reagoi pakettihukkaan. FQ:n pohjalta on rakennettu useita erilaisia menetelmiä ruuhkan hallintaan. Ne perustuvat puskurikapasiteetin ja palvelun erilaisiin jakamisiin. Palvelua jaetaan kiertävällä periaattella perustuen:

- Tasapuolisesti
    - Paketti kerrallaan
    - Bitti kerrallaan
- Painotetusti
    - Paketti kerrallaan
    - Bitti kerrallaan

FQ:lla on mahdollista luoda teoreettisesti ihanteellinen ratkaisu mutta käytäntöön FQ-pohjaiset ratkaisut ovat sopimattomia. Niiden kompleksisuus on sidoksissa yhteyksien lukumäärään, joten ne eivät skaalaudu tuleviin verkkoihin.

YHTEENVETO

Ruuhkanhallinta internetissä on pitkälti työn alla. Monia hyviä menetelmiä on kehitetty mutta yksikään ei ole saavuttanut standardin asemaa. Suurin ongelma menetelmillä on toteutuksen yksinkertaisuus ja skaalautuvuus tulevaisuuden gigabitti internettiin, jossa yhteyksien ja käyttäjien lukumäärä kasvaa eksponentiaalisesti. Esitetyistä menetelmistä RED tuntuu soveliaimmalta laajempaan käyttöön. Sen idea on yksinkertainen ja tehokas. Lisäksi sillä kyetään saavuttamaan kohtuullinen reiluus eri yhteyksien välille. Toisaalta myös päätelaitteiden vastuu ruuhkanhallinasta korostuu mitä enemmän yhteyksiä on. Yhtenäinen linja liikenteen hallinalle internetissä on välttämätöntä, sillä erilaiset toteutukset aikaansaavat ristiriitaisia tulkintoja hallintainformaatiosta. Internet yhteisön tulisi ottaa mallia ISO TP4 protokollan spesifioinnista ja siitä miten TP4:ssä on määritelty tarkasti miten jokaisen päätelaitteen tulee vastata ohjausinformaatioon.

LÄHTEET

/1/ RFC 896, J. Nagle, Congestion control in IP/TCP internetworks, 6.1.1984, 9s.

/2/ RFC 970, J. Nagle, On packet switches with infinite storage, 1.12.1985, 9s.

/3/ RFC 1254, A. Mankin, K. Ramakrishnan, Gateway congestion control Survey, 30.8.1991, 25s.

/4/ S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking, 8/1993.

/5/ RFC 1716, P. Almquist, F. Kastenholz, Towards requirements for IP routers, 4.11.1994. 186s.

/6/ R. Jain, K. Ramakrishnan, and D. Chiu, Congestion Avoidance in Computer Networks With a Connectionless Network Layer, Technical Report DEC-TR-506, Digital Equipment Corporation.

/7/ V. Jacobson, Congestion Avoidance and Control, Proceedings of SIGCOMM '88, Association for Computing Machinery, August 1988.

/8/ G. Finn, A Connectionless Congestion Control Algorithm, Computer Communications Review, vol. 19, no. 5, Association for Computing Machinery, October 1989.

/9/ M. Shreedhar, G. Varghese, Efficient fair queuing using deficit round robin, julkaisematon - hyväksytty ACM/IEEE Trans. Networking.