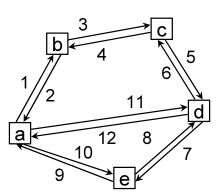
12. Liikenteenhallinta verkkotasolla

Sisältö

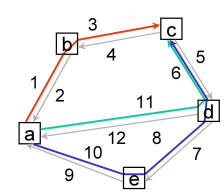
Topologia

- Verkko muodostuu joukosta solmuja ja linkkejä
  - Merk. solmujen joukkoa N:llä ja indeksoidaan niitä n:llä
  - Merk. linkkien joukkoa J:llä ja indeksoidaan niitä j:llä
- Esimerkki:
  - $N = \{a,b,c,d,e\}$
  - $J = \{1,2,3,\ldots,12\}$
  - linkki 1 solmusta a solmuun b
  - linkki 2 solmusta b solmuun a
- Merk. c_j:llä linkin j kapasiteettia (bps)



Polut

- Määritellään **polku** (= reitti)
  - joukoksi peräkkäisiä linkkejä, jotka yhdistävät kaksi verkon solmua toisiinsa.
  - Merk. polkujen joukkoa P:llä ja indeksoidaan niitä p:llä
- Esimerkki:
  - solmusta a solmuun c kolme polkua
    - punainen polku käyttää linkkejä 1 ja 3
    - vihreä polku käyttää linkkejä 11 ja 6
    - sininen polku käyttää linkkejä 10, 8 ja 6



Polkumatriisi

- Jokainen polku siis muodostuu joukosta linkkejä
- Tätä yhteyttä kuvaa **polkumatriisi** $\mathbf{A}$, jossa
  - komponentti $a_{jp} = 1$, jos $j \in p$ eli linkki j kuuluu polulle p
  - muuten $a_{jp} = 0$
- Esimerkki:
  - polkumatriisin kolme saraketta

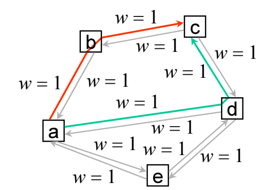
	ac1	ac2	ac3
1	1	0	0
2	0	0	0
3	1	0	0
4	0	0	0
5	0	0	0
6	0	1	1
7	0	0	0
8	0	0	1
9	0	0	0
10	0	0	1
11	0	1	0
12	0	0	0

Lyhimmät polut

- Jos kullekin linkille j määritellään linkkipaino w_j, niin polun p pituus l_p saadaan summana

$$l_p = \sum_{j \in p} w_j$$

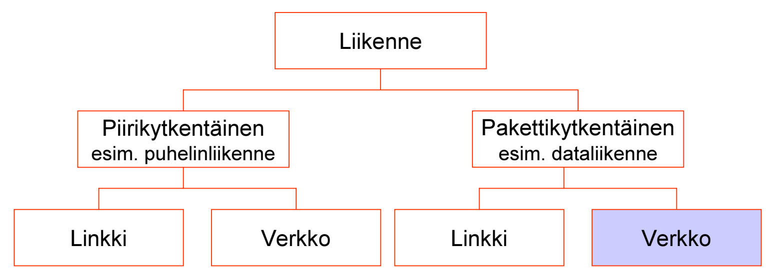
  - Jos vakiopainot $w_j = 1$, niin polun pituus = hyppyjen lkm
- Esimerkki:
  - linkkipainot 1, lasketaan siis hyppyjen lukumäärää
  - solmusta a solmuun c kaksi lyhintä polkua (pituus 2 hyppyä)



Sisältö

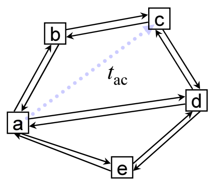
- Verkon topologia
- Liikennematriisi
- Liikenteenhallinta verkkotasolla
- Kuormantasaus

Liikenteen luonnehdinta



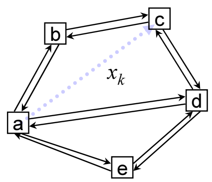
Liikennematriisi (1)

- Liikennettä verkkotasolla kuvataan **liikennematriisilla** $\mathbf{T}$, jossa
  - komponentti t_{nm} kertoo **liikenne-tarpeen** (bps) lähdesolmusta n määränpääsolmuun m
  - Aggregaatti kaikista voista joilla sama lähde ja määränpää
  - Aggregaatti myös aika-akselilla: liikenteen keskiarvo määrätyllä aikajaksolla, esim. kiiretunnin aikana tai ”tyypillisen 5 minuutin jaksossa”
- Esimerkki:
  - Liikennetarve lähteestä a määränpäähän c on t_{ac} (bps)



Liikennematriisi (2)

- Jatkossa esitämme liikenne-tarpeet myös vektorimuodossa
  - Sitä varten indeksoimme lähde-määränpää-parit eli **OD-parit**.
  - Merk. OD-parien joukkoa K:llä ja indeksoidaan niitä k:lla
- Liikennetarpeet voidaan tällöin esittää vektorina $\mathbf{x}$, missä
  - komponentti x_k kertoo OD-parin k liikennetarpeen
- Esimerkki:
  - jos OD-parin (a,c) indeksi on k, niin $x_k = t_{\mathrm{ac}}$



Sisältö

- Verkon topologia
- Liikennematriisi
- Liikenteenhallinta verkkotasolla
- Kuormantasaus

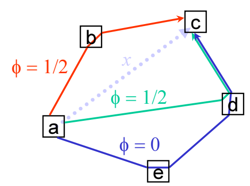
Liikennekuorman hallinta ja verkon suunnittelu

- **Liikennekuorman hallinta** (traffic engineering) = ”Engineer the traffic to fit the topology”
  - Jos topologia ja linkkien kapasiteetit on kiinnitetty ja liikennematriisi tunnetaan, miten nämä liikennetarpeet pitäisi **reitittää** läpi verkon?
- **Verkon suunnittelu** (network design) = ”Engineer the topology to fit the traffic”

Reitityksen vaikutus kuorman jakaantumiseen

- Reititysalgoritmi määrää, miten eri liikennetarpeet kuormittavat verkon linkkejä
  - IP-verkkojen reititysprotokollat (RIP, OSPF, BGP) käyttävät lyhimmän polun algoritmeja (Bellman-Ford, Dijkstra)
  - MPLS-verkoissa myös muut algoritmit mahdollisia
- Tarkemmin sanottuna: reititys-algoritmi määrää liikenne-tarpeiden x_k **jakosuhteet** ϕ_{pk} eri poluille p,

$$\sum_{p \in P} \phi_{pk} = 1 \quad \text{kaikille } k$$



Linkkikuormat

- OD-paria k yhdistävälle polulle p tuleva liikenne on siis

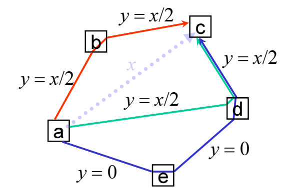
$$\phi_{pk} x_k$$

- Liikennetarpeet x_k ja jakosuhteet ϕ_{pk} yhdessä määräävät eri lin-keille j tulevat **linkkikuormat** y_j:

$$y_j = \sum_{p \in P} \sum_{k \in K} a_{jp} \phi_{pk} x_k$$

- Sama matriisimuodossa:

$$\mathbf{y} = \mathbf{A}\phi\mathbf{x}$$



MPLS

- **MPLS** (Multiprotocol Label Switching) tukee liikenteen jakamista rinnakkaisille poluille
  - MPLS-verkoissa kunkin OD-parin välille voidaan vapaasti luoda rinnakkaisia polkuja eli LSP:itä (Label Switched Path)
  - Näiden polkujen ei tarvitse olla ”lyhimpiä polkuja”
- Kutakin LSP:tä vastaa tunnus (label) ja kussakin MPLS-paketissa on tällainen polun ilmaiseva tunnus
  - MPLS-paketit reititetään näitä tunnuksia käyttäen läpi verkon
- Kuorman jakautumiseen voidaan vaikuttaa muuttamalla **suoraan** jakosuhteita ϕ_{pk} lähdesolmuissa

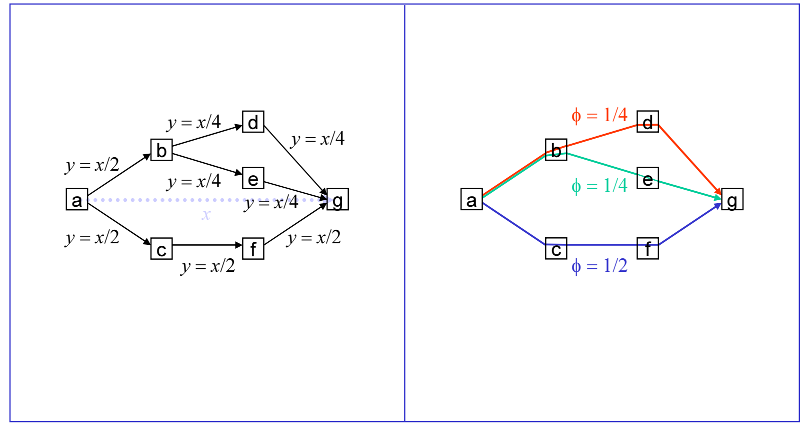
OSPF (1)

- **OSPF** (Open Shortest Path First) on alueen sisäinen (intradomain) reititysprotokolla IP-verkoissa
- Linkkitilaprotokolla (Link State Protocol)
  - Jokainen alueen solmu kertoo etäisyytensä naapurisolmuihinsa
  - Nämä etäisyydet toimivat linkkipainoina lyhimmän polun algoritmissa
  - Näistä tiedoista jokainen solmu rakentaa itselleen koko alueen topologian
  - Ko. topologia määrää lyhimmät polut kyseisestä solmusta kuhunkin kohdesolmuun
  - Lyhimmän polun algoritmina Dijkstra
- IP-paketit reititetään näitä lyhyimpiä polkuja pitkin

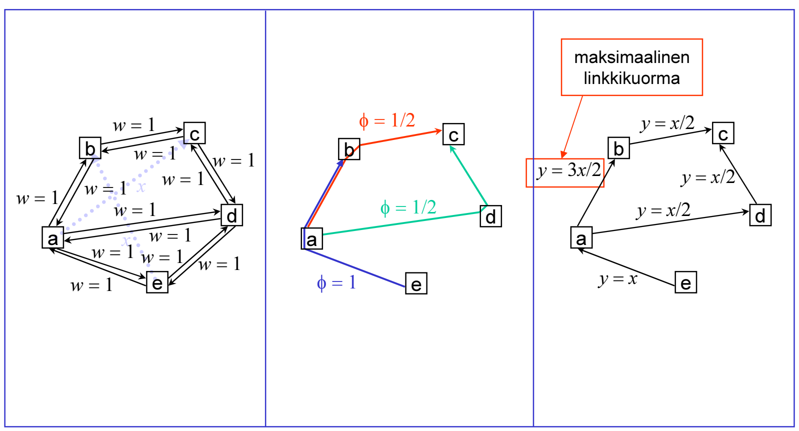
OSPF (2)

- OSPF:ssä yleensä käytössä **ECMP** (Equal Cost Multipath)
  - Jos solmusta n useita lyhimpiä polkuja solmuun m, niin liikenne pyritään jakamaan tasan niiden solmusta n lähtevien linkkien välillä, jotka kuuluvat johonkin näistä lyhimmistä poluista
  - Tästä **ei** kuitenkaan seuraa, että liikenne jakautuisi tasan rinnakkaisten lyhimpien polkujen välille! Kts. seuraavan kalvon esimerkkiä.
- Kuorman jakautumiseen voidaan vaikuttaa vain **epäsuorasti** muuttamalla linkkipainoja
  - jakosuhteita ϕ_{pk} ei voida suoraan muuttaa
  - ECMP:n vuoksi haluttuja jakosuhteita ϕ_{pk} ei välttämättä saavuteta

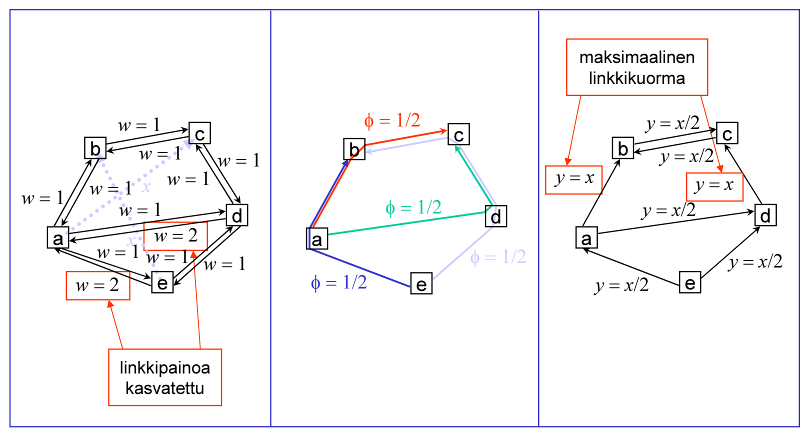
ECMP



Linkkipainojen vaikutus kuorman jakaantumiseen (1)



Linkkipainojen vaikutus kuorman jakaantumiseen (2)

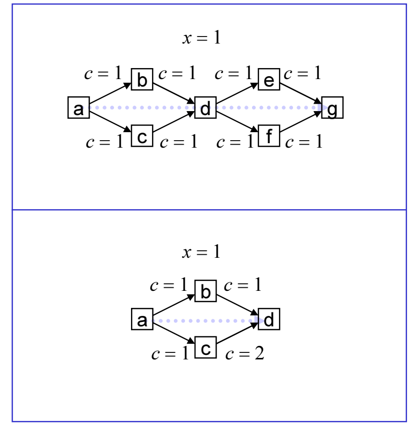


Sisältö

- Verkon topologia
- Liikennematriisi
- Liikenteenhallinta verkkotasolla
- Kuormantasaus

Kuormantasausongelma (1)

- Jos verkon topologia ja liikennematriisi ovat tiedossa, niin miten liikenne kannattaisi reitittää verkkoon?
- Eräs järkevä tapa on pyrkiä tasaamaan eri linkkien suhteellinen kuorma $\rho_j = y_j/c_j$
  - Joskus se onnistuu useallakin eri tavalla (kuten yläkuvassa)
  - Joskus taas se ei ole ollenkaan mahdollista (kuten alakuvassa)
  - Tällöin voimme kuitenkin pyrkiä niin lähelle tasajakoa kuin mahdollista, esim. minimoimalla suhteellisten kuormien maksimi (ns. kuormantasausongelma)



Kuormantasausongelma (2)

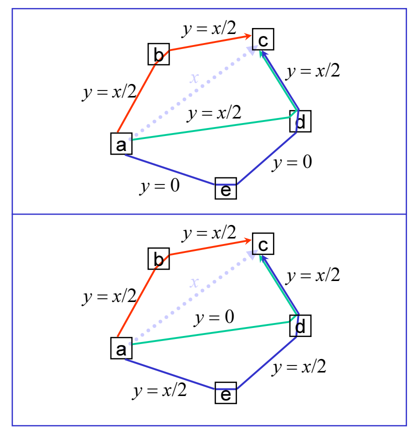
- **Kuormantasausongelma**:
  - Olkoon (N,J) verkon topologia, c_j linkkikapasiteetit sekä x_k liikennetarpeet. Tavoitteena on määrätä jakosuhteet ϕ_{pk} siten, että suhteellisten linkki-kuormien maksimi minimoituu

$$\text{Minimize} \quad \max_{j\in J} \frac{y_j}{c_j}$$

$$\text{subject to} \quad \begin{cases} y_j = \sum_{p\in P}\sum_{k\in K} A_{jp}\phi_{pk}x_k & \forall j\in J \\ \sum_{p\in P}\phi_{pk} = 1 & \forall k\in K \\ \phi_{pk} \ge 0 & \forall p\in P, k\in K \end{cases}$$

Kuormantasausongelma (3)

- Kuormantasausongelmalla on aina ratkaisu, mutta ratkaisu ei välttämättä ole yksikäsitteinen
- Esimerkki:
  - sama maksimikuorman minimi saavutetaan eripituisilla reiteillä
  - näistä ylempi ratkaisu on tietysti resurssien kokonaiskäytön kannalta järkevämpi
- Järkevään yksikäsitteiseen ratkaisuun päästään lisäämällä (häviävän) pieni kustannus polun jokaisesta käytetystä hypystä



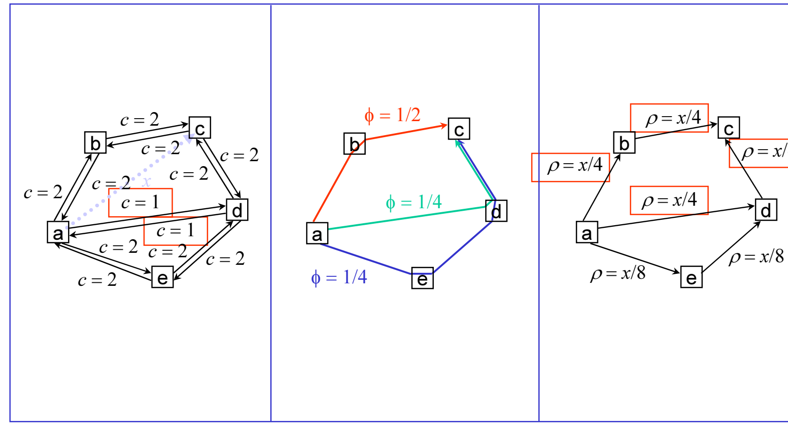
Kuormantasausongelma (4)

- Järkevä yksikäsitteinen kuormantasausongelma:
  – Olkoon (N,J) verkon topologia, c_j linkkikapasiteetit sekä x_k liikennetarpeet. Tavoitteena on määrätä jakosuhteet ϕ_{pk} siten, että suhteellisten linkkikuormien maksimi minimoituu pienimmällä mahdollisella kokonaisliikenteellä

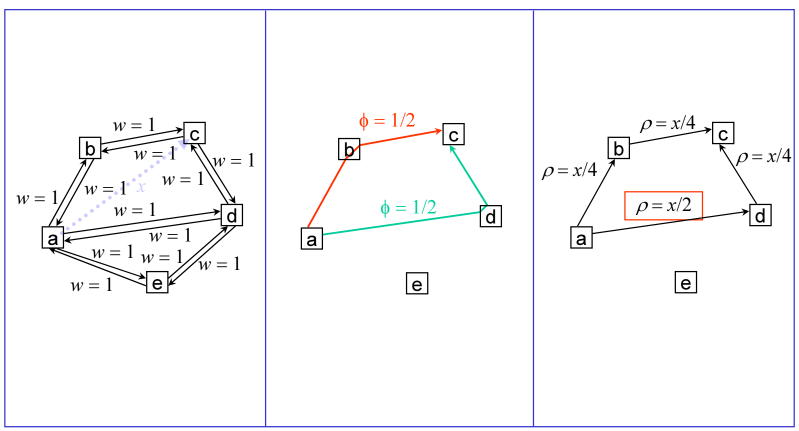
$$\text{Minimize} \quad \max_{j \in J} \frac{y_j}{c_j} + \varepsilon \sum_{j' \in J} y_{j'}$$

$$\text{subject to} \quad \begin{cases} y_j = \sum_{p \in P} \sum_{k \in K} A_{jp} \phi_{pk} x_k & \forall j \in J \\ \sum_{p \in P} \phi_{pk} = 1 & \forall k \in K \\ \phi_{pk} \geq 0 & \forall p \in P, k \in K \end{cases}$$

Esimerkki (1): optimaalinen ratkaisu



Esimerkki (2): linkkipainot w = 1



Esimerkki (3): optimaaliset linkkipainot