OHJ-5010 Hajautettujen järjestelmien perusteet — Tentti 19.05.2012

Luennoitsija: Kari Systä

Tentissä ei saa käyttää kirjallista materiaalia, laskimia, tietokoneita tai muita lunttausvälineitä.

1 Esittele lyhyesti kurssin aihealuetta tuntemattomalle mitä seuraavat asiat ovat: (6p)

a. Socket
b. Käsittelytuntumattomuus
c. Virtapohjaisessa kommunikoinnissa (stream-oriented communication) esiintyjä jitter
d. Nimeäminen ja nimipalvelu

---

2 Prosessi P tarvitsee mahdollisimman tarkan ajan, ja tietää että Oraakkelin kello on tarkassa ajassa. Jos Oraakkelille lähettää viestin se vastaa lähettämällä tarkan kellonajan takaisin kysyjälle. Mikään ei kuitenkaan riko suhteellisuusteoriaa eikä ole äärettömän nopeaa. Kuvaa algoritmi jolla P saa kellonsa mahdollisimman tarkkaan aikaan. Pohdi myös kuinka tarkka menetelmä on, ja mitkä olisivat epätarkkuuden syyt?

---

3 Replikoinnista ja konsistenssista (6p)

Seuraavassa W(x)a tarkoittaa että globaalin muuttujaan x kirjoitetaan arvo a. R(x)b tarkoittaa että globaalista muuttujasta x luetaan arvo b. Oletetaan että meillä on seuraavat prosessien P1-P4 käyttäytymiset (globaali kello etenee vasemmalta oikealle):

P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)b	R(x)a

(a)

P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

(b)

Toteuttaako (a) tai (b), vai molemmat, tai ei kumpikaan sarjallisen konsistenssin (sequential consistency)? Entä onko ovatko ne kausaalisesti konsistentteja? Perustele.

---

4 Transaktioista (6p)

Oletetaan, että *A ja B* ovat kaksi prosessia jotka operoivat muuttujilla *x* ja *y*.

A haluaa tehdä transaktion:

```
if (x > y) {
    x = y;
}
```

B haluaa tehdä transaktion:

```
if (y > x) {
    y = x;
}
```

Transaktio on toteutettu kaksi-vaiheisella lukituksella. Onko tässä lukkiutuman vaara? Missä tilanteessa lukkiutuma voi syntyä?

---

5 Oletetaan, että sinulla on joukko prosesseja P. Kukin prosessi sisältää listan naapureista ja jonon tulevia viestejä. Käytettävissäsi on seuraavat primitiivit

- P.neighbours() – palauttaa listan naapureista, eli prosesseista joihin on kommunikointiyhteys.
- send (P, m) – lähetä viesti naapurille P. Viestin sisältö on vapaa.
- receive() – Lue jonosta seuraava vastaanotettu viesti. Jos jonossa ei ole viestiä, palautaa arvon *null*.
- m.id() – kullakin viestillä on uniikki numeerinen id.

Tehtävänäsi on toteuttaa epideeminen kommunikointi jossa uusi viesti välitetään kaikille saavutettavissa oleville prosesseille niin, että kaikkien saatua viestin kommunikointi pysähtyy, ja ylimääräistä kommunikointia muutenkin vältetään. Operaation käynnistää jonkin prosessin saama viesti. Käynnistävällä viestillä uusi id jonka arvo on suurempi kuin yksikään aikaisempi id.

Kirjoita pseudokoodi.