LUENTO 3

Käyttöjärjestelmät II

Prosessit ja säikeet
SMP, Mikroytimet
W2K säikeet
Stallings, Ch 3-4.4

Mitä KJ-I:ssä / KJ-II:ssa?

KJ I + RIO (luvut 3.1-3, 4.1, 5, 6.1-6)

- **Prosessin tilat, jonot, PCB**
- **Luonti, prosessinvaihto, prosessi vs. säie, TCB**
- **User-level vs. kernel-level threads**
- **Samanaikaisuuden hallinnan perusasiat**

Seuraavaksi KJ-II:ssa (luvut 3-6 loppuun)

- **UNIX prosessien hallinta**
- **Moniprosessorijärjestelmä, SMP**
- **Mikrokernel**
- **W2K: säikeet, SMP**

(luento 3)

- **Solaris: säikeet, SMP**
- **Linux: prosessit ja säikeet**
- **Samanaikaisuuden hallinta eri KJ:ssä**

(luento 4)

Käyttöjärjestelmät II

UNIX prosessit

Ch 3 [Stal 05]

Prosessit yleensä

- **Käsite**
- **Tilat**
- **Resurssit**
- **Prosessin tiedot**
  - kuvaaja PCB

ks. Fig 3.9 [Stal05]

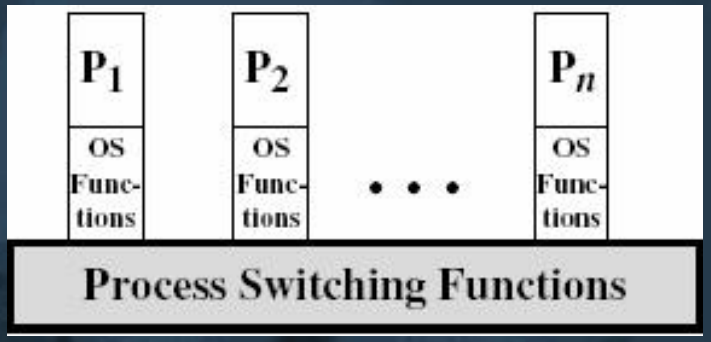
ks. Fig 3.10 [Stal05]

ks. Tbl 3.4 [Stal05]

ks. Tbl 3.5 [Stal05]

UNIX SVR4 Prosessit

(Fig. 3.15 (b) [Stal05])



- **Pääosa KJ-työstä tapahtuu käyttäjätason prosessien sisällä**
  - etuoikeutetussa tilassa (SVC, keskeytys)
- **Osa KJ-työstä tapahtuu KJ-prosesseissa**
  - etuoikeutetussa tilassa
- **Yhdeksän eri prosessin tilaa**

ks. Fig. 3.17 [Stal05]

- **Etuoikeutettu prosessi ei voi menettää suoritinta**
  - ellei se itse pyydä jotain resurssia, joka ei ole vapaana
- **Prosessi 0 – boot**
- **Prosessi 1 – init (mother of all)**

UNIX prosessin kuvaaja

- **Kääntäjän tiedoista: User Level Context**
- **Suoritinympäristö: Register Context**
- **KJ:n tiedot: System Level Context**
  - staattinen
    - prosessin tilatiedot (Process Table Entry)
    - prosessin hallintotiedot (U Area)
  - dynaaminen
    - muistialueet (Region Table)
    - oma pino etuoik. tilan koodille (Kernel Stack)

Tbl 3.10 [Stal05]

Tbl 3.11 [Stal05]

Tbl 3.12 [Stal05]

UNIX prosessin luonti

- **Ytimen systeemikutsu *pid=fork()***
  - varaa paikka prosessitaulusta Process Table
  - anna uniikki ID (nelinumeroinen)
  - kopioi vanhemman koko prosessinkuvaaja uudelle prosessille (tai luo ihan uudet rakenteet!)
    - vain linkki yhteiseen muistialueeseen
  - lisää aukiolevien tiedostojen käyttäjien määrää yhdellä (jos tavallinen ”sibling”)
  - siirrä uusi prosessi RR-jonoon
  - palauttaa lapsen pid:n vanhemmalle ja pid=0 lapselle (tämä on ainoa ero koodinäkökulmasta)
    - esimerkki seuraavalla kalvolla

UNIX fork()

```
while (TRUE) { /* repeat forever /*
        type_prompt( ); /* display prompt on the screen */
        read_command(command, params); /* read input line */
        pid = fork( ); /* fork off a child process */

        if (pid < 0) {
                printf("Unable to fork0); /* error condition */
                continue; /* repeat the loop */
        }

        if (pid != 0) {           /* parent waits for child */
                waitpid (−1, &status, 0);

        } else {                  /* child does the work */
                execve(command, params, 0);
        }
}
```

Fig. 10-7 [Tane01]. A highly simplified shell.

Käyttöjärjestelmät II

Säikeet
Symmetric MultiProcessors (SMP)
Mikroytimet

Ch 4 [Stal 05]

Prosessit ja säikeet

- **Ennen (kun ei säikeitä)**
  - prosessi:
    - resurssien hallinta
    - suorittava (laskentaa tekevä) yksikkö
- **Nyt**
  - prosessi
    - resurssien hallinta
  - säie
    - säikeet toteuttava yksikkö
    - samanaikainen suoritus usealla suorittimella?

ks. Fig 4.1 [Stal05]

ks. Fig 4.3 [Stal05]

ks. Fig 4.2 [Stal05]

ks. Fig 4.6 [Stal05]

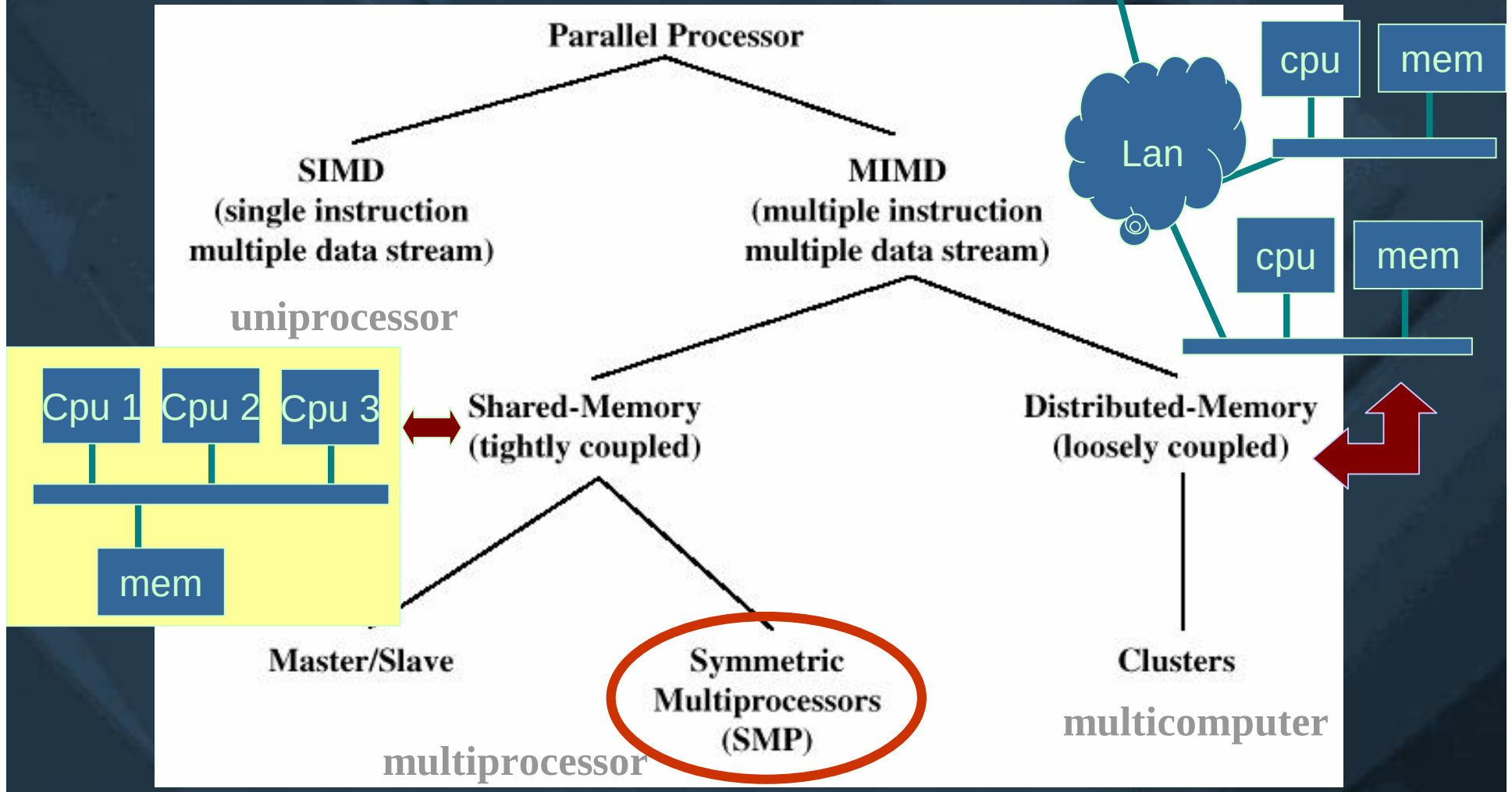
ks. Fig 4.7 [Stal05]

Flynnin jaottelu moniprosessorikoneille

Michael J. Flynn, 1966

- SISD: Single Instruction - Single Data
  - yksi käsky, yksi data
  - normaali CPU: yksi ALU, yksi CU, yksi MEM
- SIMD: Single Instruction - Multiple Data
  - yksi käsky, monta data-alkiota
  - taulukkoprosessori: monta erikoistunutta ALUa
  - vektorisuoritin: vektorikonekäskyt, vektorirekisterit
- MISD: Multiple Instruction - Single Data
  - Ei mielekäs
- MIMD: Multiple Instruction - Multiple Data
  - monta käskyä ja monta dataa
  - tiukasti kytketty: monta CPU:ta, yhteinen MEM
  - löyhästi kytketty: monta CPU:ta, kullakin oma MEM

Moniprosessorikoneet



(Fig 4.8 [Stal05])

Master-slave vs. SMP

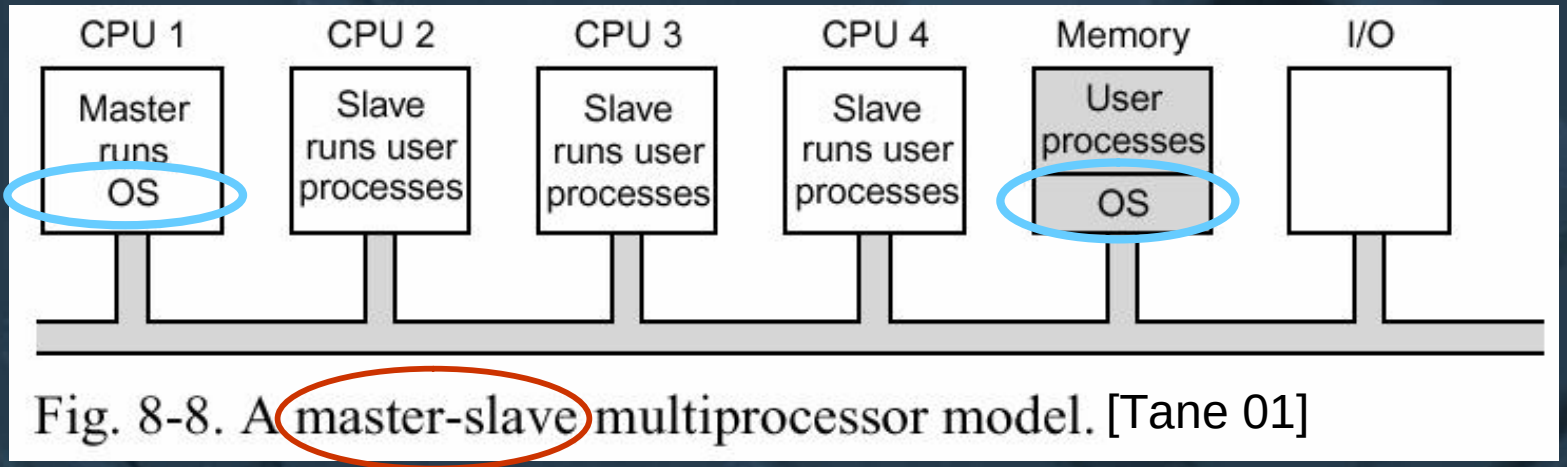


Fig. 8-8. A master-slave multiprocessor model. [Tane 01]

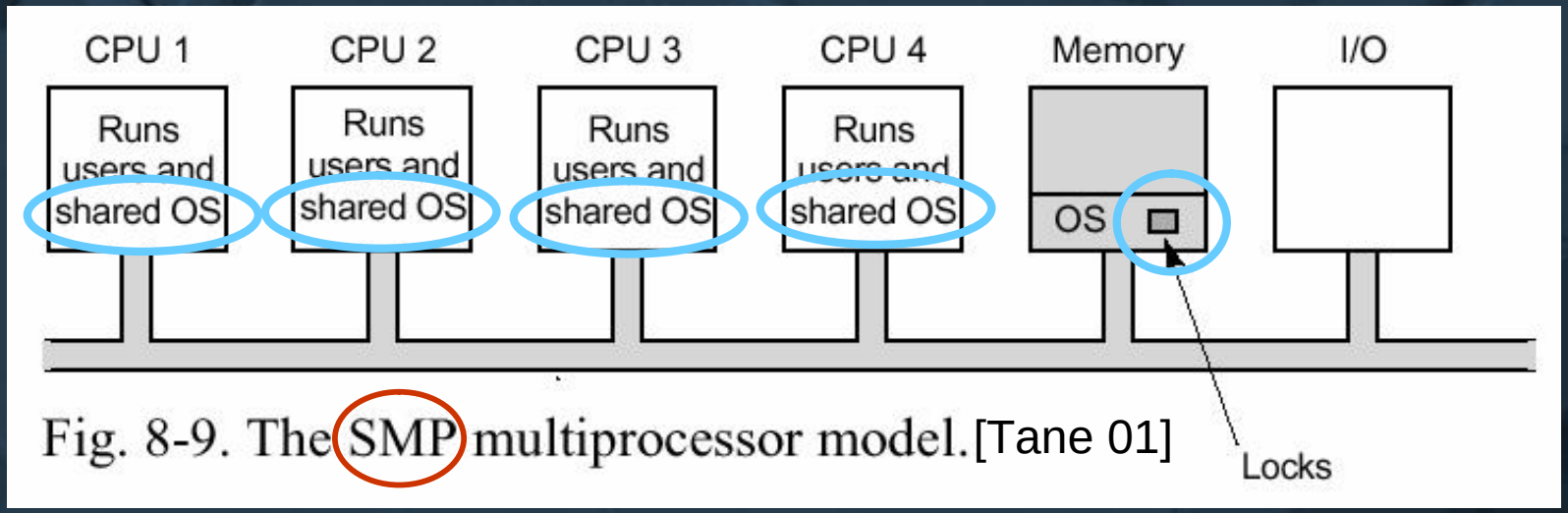


Fig. 8-9. The SMP multiprocessor model.[Tane 01]

Toinen jaottelu yhteisen muistin moniprosessorikoneille

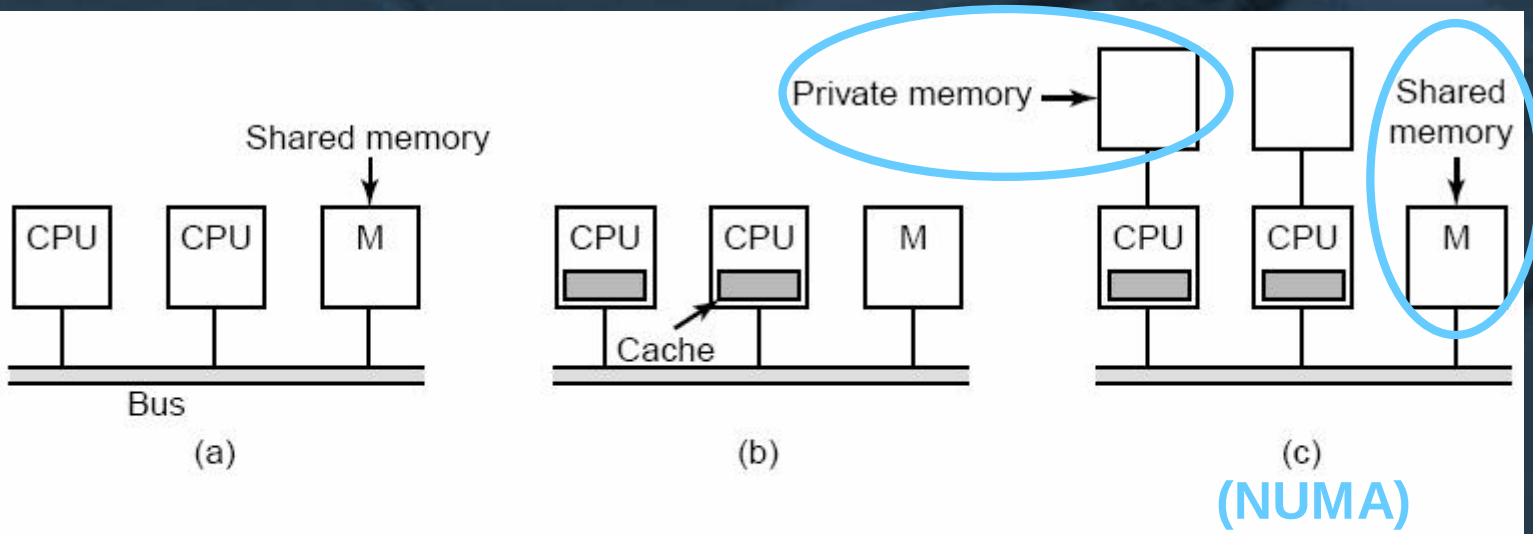
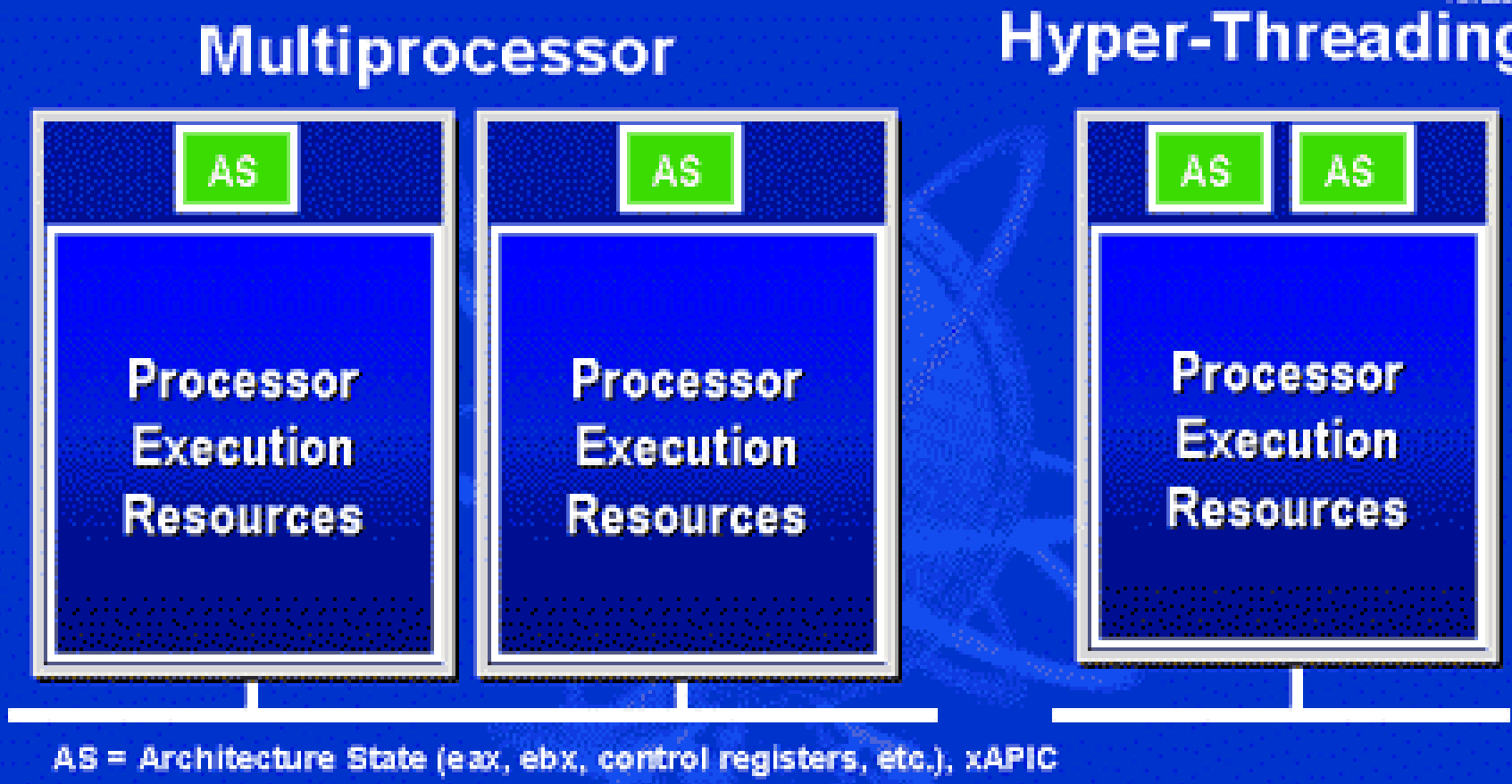


Fig. 8-2. Three bus-based multiprocessors. (a) Without caching. (b) With caching. (c) With caching and private memories.

[Tane 01]

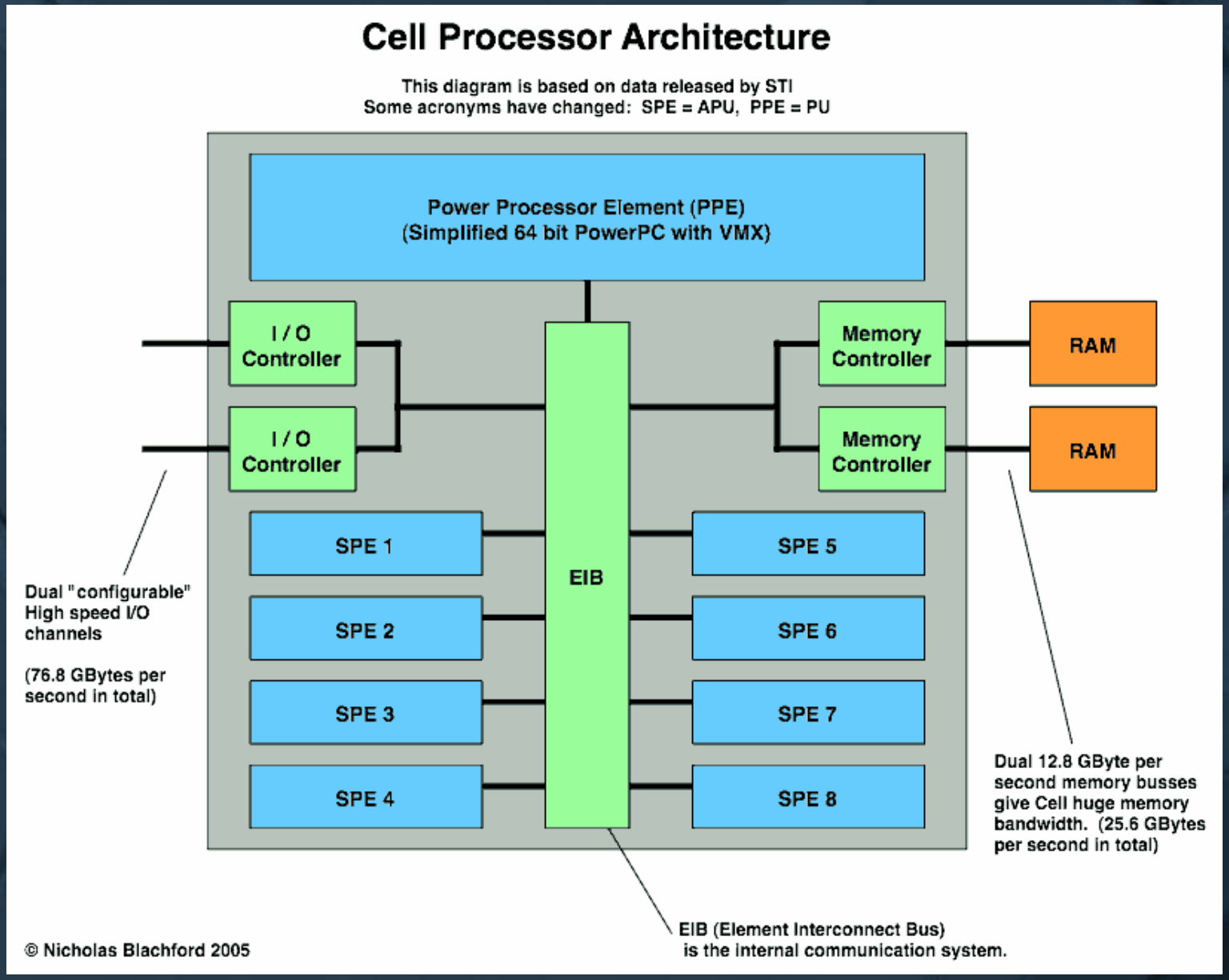
Hyper-Threading Technology

Intel Developer Forum Fall 2001



Hyper-Threading Technology looks like two processors to software

http://www.intel.com/technology/hyperthread/intro_nexgen/



http://www.blachford.info/computer/Cells/Cell1.html

Kolmas jaottelu yhteisen muistin moniprosessorikoneille

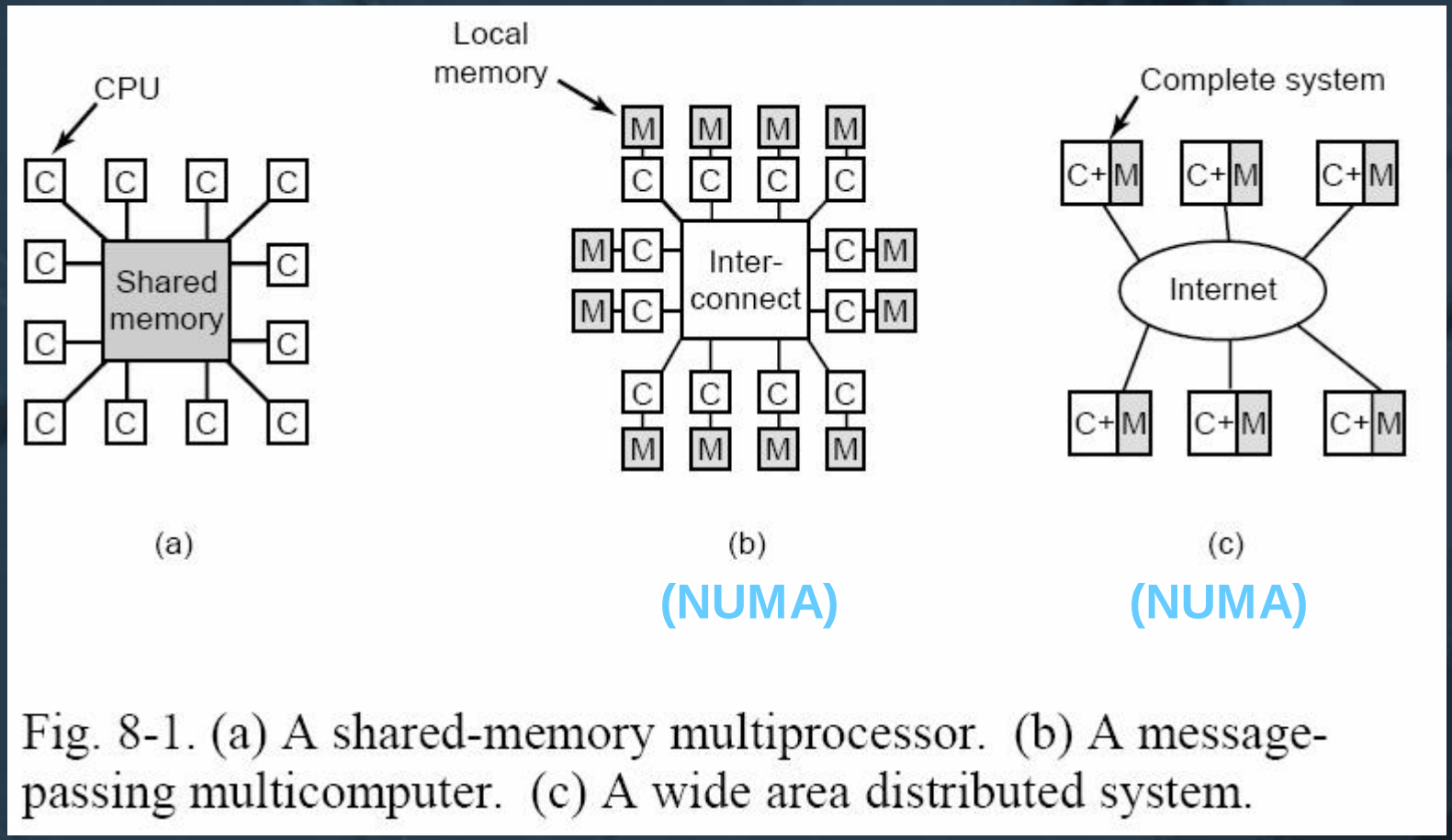
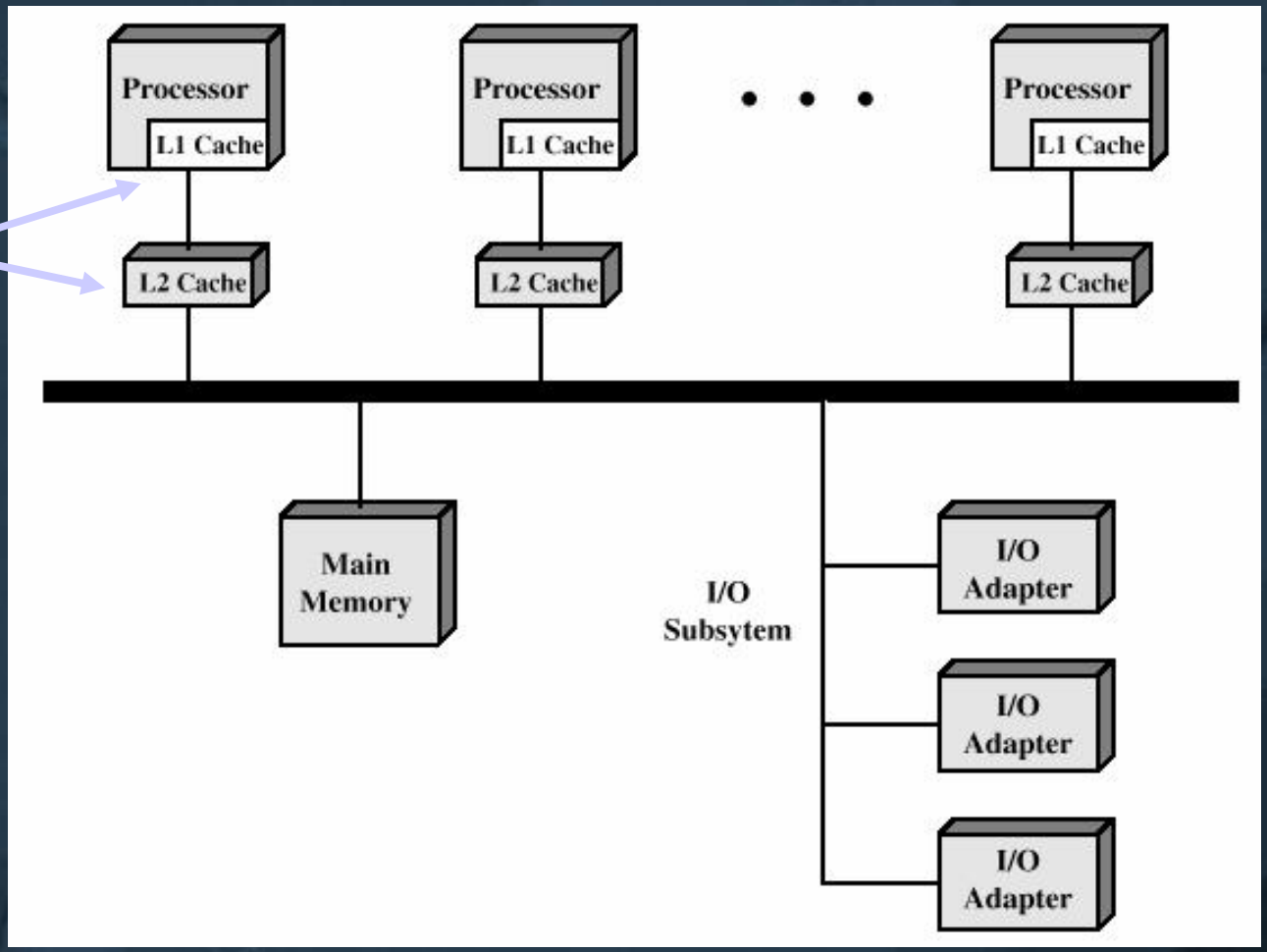


Fig. 8-1. (a) A shared-memory multiprocessor. (b) A message-passing multicomputer. (c) A wide area distributed system.

[Tane 01]

SMP: Symmetric Multiprocessing

samalla lastulla vai ei?



- **Useita suorittimia (CPU)**
- **Yhteinen muisti**
  - organisoitu lohkoihin
  - useita yhtäaikaisia viittauksia
- **Yhteiset oheislaitteet**
- **Välimuistien yhteneväisyys (coherency)?**

(Fig. 4.9 [Stal 05])

Huomioitava SMP KJ:ssä

- **Samanaikaisuuden hallinta** (Ch 5-6, RIO)
  - KJ voi olla suoritettavana usealla CPU:lla
  - KJ:n oltava vapaakäyntinen (re-entrant)
  - KJ:n tietorakenteiden käsittelyssä tarvitaan poissulkemista
- **Vuorotettavana useita CPU:ita** (Ch 10)
  - KJ:kin voi olla suorituksessa yhtäaikaa eri CPU:illa
  - saman prosessin säikeiden vuorottaminen
- **Synkronointi, lukot** (Ch 5-6, RIO)
  - poissulkeminen ja tapahtumien järjestys
- **Muistinhallinta** (Ch 7-8)
- **Ulkoisten keskeytysten käsittely**
- **Vikasietoisuus: joku CPU voi pudota pois**
  - tästä ei tarvinnut murehtia yhden CPU:n kanssa!
  - ollaanko vikasietoisia vai ei? miten?

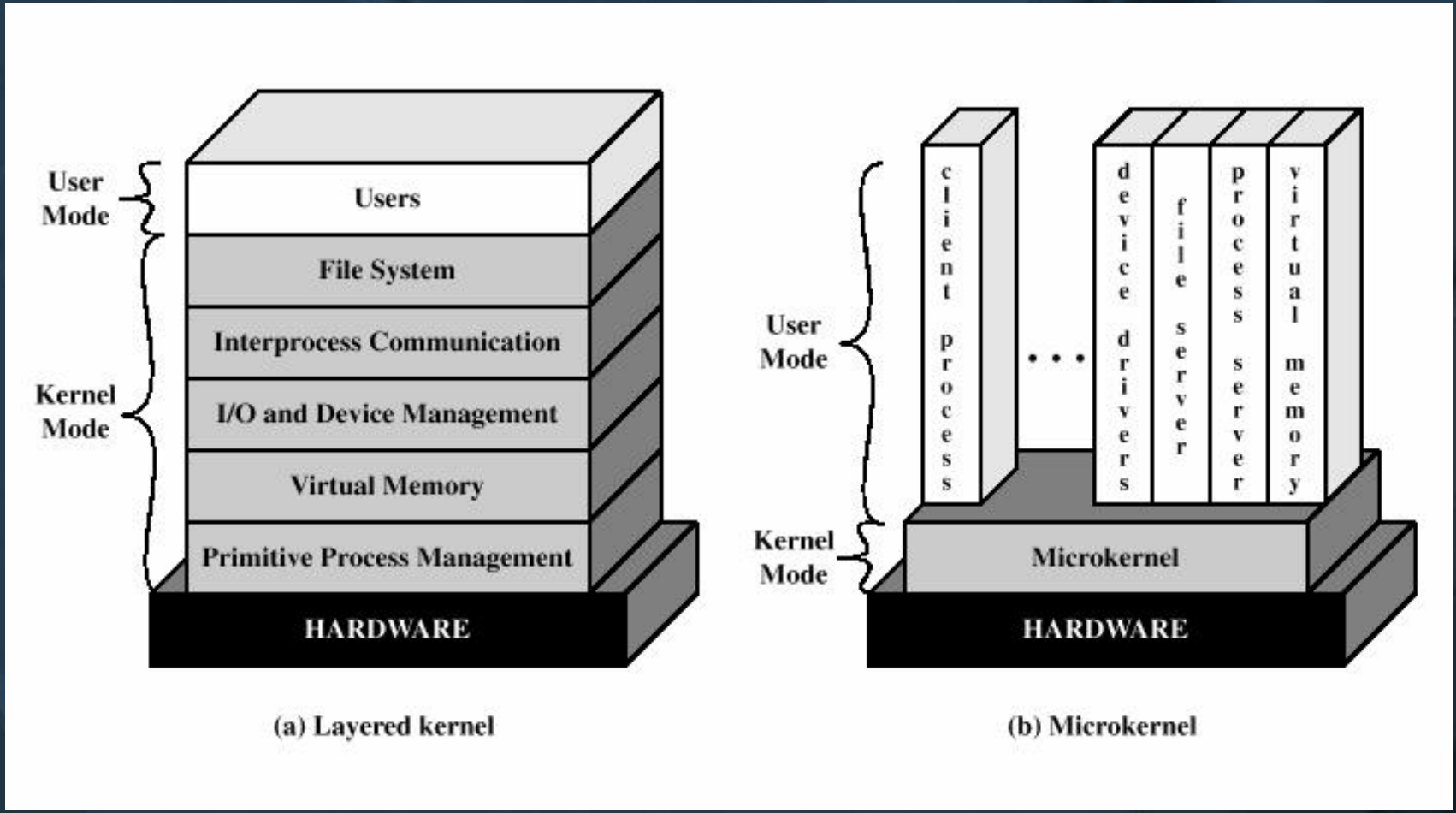
Käyttöjärjestelmät II

Mikroytimet

Ch 4.3 [Stal05]

KJ:n ydin

(Fig 4.10 [Stal 05])



Monoliittinen ydin

- **"Kokoelma KJ:hin kuuluvia funktioita", jotka käännetään yhdeksi binäärikoodiksi**
  - koko KJ etuoikeutetussa tilassa
  - funktio voi kutsua suoraan toista funktiota

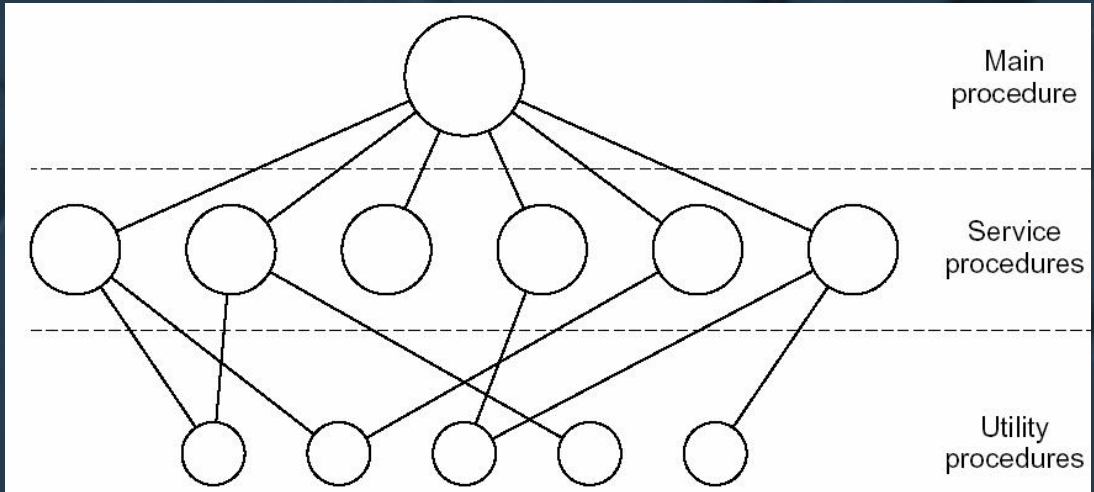


Fig. 1-24. A simple structuring model for a monolithic system.

[Tane 01]

Monoliittinen ydin

- **Periaatteessa kaikki KJ:n osat pääsevät käsiksi kaikkiin tietorakenteisiin**
  - globaalien muuttujien käyttö
    - ei edes parametrien kopiointi hidastamassa
  - kopioi itse tieto suoraan paikalleen
    - ei sanomanvälitystä
    - ei ***etuoikeutettu tila*** ⟷ ***käyttäjätila*** -vaihtoja
- **Sovellus käyttää palvelua palvelupyynnöllä**
  - parametrit esim. pinoon, keskeytys
  - ‘vastaus’ suoraan annettuihin muuttujiin tai pinossa
- **Nopeus, tehokkuus, yksinkertaisuus**
- **Ongelmat?**

Mikroydin (mikrokernel)

- **Etuoikeutetussa tilassa toimiva pieni KJ:n ydin (core functions)**
  - tärkeät laitetason liittymät
  - minimijoukko muita perustoimintoja
- **Muut alijärjestelmät käyttäjätilassa**
  - laiteajurit
  - tiedostojärjestelmät
  - virtuaalimuistin hallinta
  - ikkunoitu käyttöliittymä
  - turvallisuusosia, mm. käyttäjien tunnistus
- **Alijärjestelmät eivät pääse suoraan käsiksi laitteistoon tai toisiinsa**
  - käyttö sanomanvälityksen kautta

Mikroytimen kehuja

- **Yhtenäinen palvelujen käyttötapa**
  - sama mekanismi etuoikeutetussa ja käyttäjätilassa
  - sanomanvälitys: send ja receive
- **Laajennettavuus**
  - helppo lisätä uusia palveluja (KJ:kin kehittyy…)
  - helppo lisätä uusia liittymiä vanhoihin moduuleihin
- **Joustavuus**
  - piirteitä poistamalla saadaan räätälöity versio
  - paikkopaketit voidaan ottaa mukaan boottaamatta konetta
  - esim: sulautettu järjestelmä - minimijoukko toiminnallisuutta
    - kännykkä, kämmenmikro, videonauhuri, tv, …
- **Siirrettävyys**
  - laitesidonnaiset osat rajattu mikroytimeen
  - uusi laitteisto vaatii vain mikroytimen ronkkimista

Mikroytimen kehuja

- **Luotettavuus**
  - modulaarinen rakenne helpompi toteuttaa ja testata
  - jos joku palveluosa kaatuu, muut voivat silti toimia
- **Sopii hajautettuun järjestelmään**
  - paikallisen ja etäpalvelun käyttö samanlaista
  - ‘single microkernel image’: jos palvelut ja prosessit nimetty kaikissa koneissa yksikäsitteisesti
    - eri koneissa sama palveluiden nimentä ja osoitteet
- **Sopii oliopohjaisen KJ:n toteutukseen**
  - mikroydin yksi olio muiden (palvelija)olioiden joukossa
  - kommunikointi vain tunnettujen rajapintojen kautta
    - private data, public interfaces

Mikroytimen heikkouksia

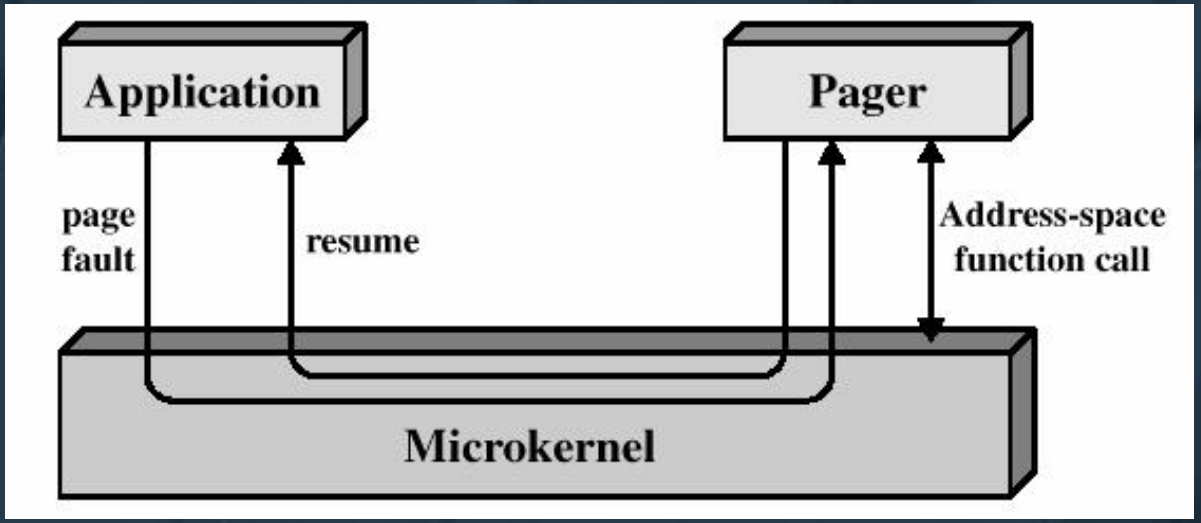
- **Tehokkuus**
  - yhteiskäytössä oleva vs. paikallinen muisti
  - sanomanvälitys vs. suora funktiokutsu
  - paljon ***user mode*** $\leftrightarrow$ ***kernel mode*** –vaihtoja
    - viestien käyttö vs. suorat funktiokutsut
- **Apu: enemmän toimintoja takaisin suoritettavaksi etuoikeutettuun tilaan**
  - esim. Mach, Chorus, W2K
  - laiteajureita takaisin etuoikeutettuun tilaan
  - muita paljon käytettyjä palveluja
  - kompromissi - mitä menetetään?

Mikroytimen peruspalvelut

- **Mitkä?**
  - ei täysin yleispäteviä sääntöjä
  - ainakin laitteistoa suoraan ronkkivat osat
  - tuki käyttäjätilan palveluille ja sovelluksille
    - kommunikointi
- **Minimi:**
  - muistinhallinnan alimmat tasot (low-level MM)
  - prosessien välinen kommunikointi (IPC)
  - siirrännän laitesidonnaiset osat
  - keskeytysten käsittely
  - prosessin vaihto (basic scheduling)

Mikroytimen peruspalvelut

- **Muistinhallinnan alimmat tasot**
  - virtuaalisivujen kuvaus fyysisiksi sivuiksi ytimessä
    - sivutaulu ja MMU:n asetukset
  - sivutus ja virtuaalimuistin hallinta mikroytimen ulkopuolella
    - sivupuutoksen käsittely
    - sivujen poisto- ja sijoitusalgoritmit



(Fig. 4.11 [Stal 05])

Mikroytimen peruspalvelut (jatkuu)

- **Prosessien välinen kommunikointi (IPC)**
  - Perusmekanismina sanomanvälitys
    - pyyntö / vastaus
    - jos prosesseilla ei yhteiskäyttöistä muistia, sanomanvälitys muistista muistiin kopiointia (etuoik. tilassa)
  - Sanoma: (keneltä, kenelle, data)
    - data = esim. palvelun nimi (‘portti’) ja sen parametrit
    - datan paikalla voi olla osoite, josta data löytyy
  - Palveluun voi liittyä oikeuksien tarkistus KJ:ssä
    - kirjanpito ‘porteista’ ja pääsyoikeuksista
    - prosessi voi muuttaa pääsyoikeuksia
  - Poissulkemisen ja synkronoinnin primitiivit

Mikroytimen peruspalvelut (jatkuu)

- **Siirräntä**
  - laiteohjaimen rekistereiden käyttäminen, DMA
  - I/O-porttien sijoittaminen prosessin osoiteavaruuteen
- **Keskeytysten käsittely**
  - mikroydin tunnistaa keskeytykset, mutta erillinen käyttäjätason palvelu käsittelee ne
    - keskeytysvektori: mikroydin pitää kirjaa keskeytysnumeroista ja niihin liittyvistä palvelijoista
    - generoi ja lähetä sanoma käsittelijälle (esim. ajurille)
      - välitä tarvittaessa ohjaimen rekistereiden arvot
    - kuittaa keskeytys huomatuksi
    - lyhyt ja nopea?

Laiteajuri

- **Käyttäjätilassa toimiva prosessi / säie**
  - saa pyyntöjä sovelluksilta ja laitteistolta

```
while (true) do {
   waitFor (msg,sender);
   if (sender == my_sw_client) {
              read/write I/O-ports
              …
              }
   if (sender == my_hw_interrupt) {
        read/write I/O-ports
        reset hardware interrupt
        }
   }
```

  - I/O-portit ajurin osoiteavaruudessa (muuten tiedot sanomina)

Käyttöjärjestelmät II

Windows 2000 (eli W2K)

Säikeet ja SMP

Ch 4.4 [Stal 05]

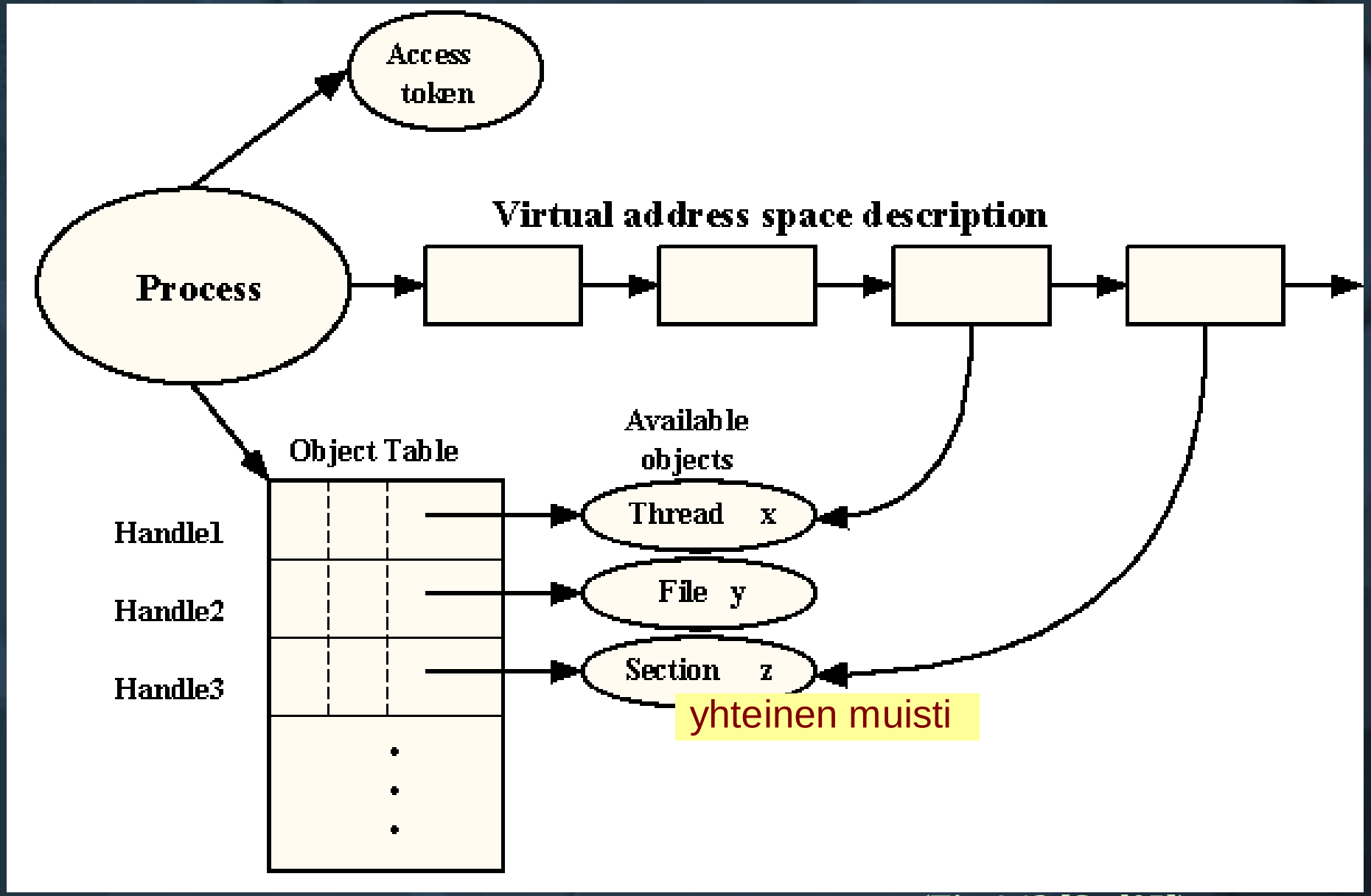
W2K resurssien ja suorituksen hallinta

- **Työ (job)**
  - prosessien kokoelma, jolla yhteiset rajoitukset
  - max prosessien lkm, max CPU aika per prosessi, ...
- **Prosessi (process)**
  - omistaa resurssit
  - koostuu useasta säikeestä
- **Säie (thread)**
  - oma access token (saatu esim. asiakkaalta)
  - suorittava vuorotuksen yksikkö
  - kaksi pinoa, oma user ja kernel tiloissa suoritusta varten
- **Kuitu (fiber) – ”kevytsäie” (nopea vuorotus)**
  - ydin ei tiedä kuiduista mitään Win32 API hoitaa kaiken
    - hyvin nopea vuorotus user tilassa (Win32 API hoitaa)
    - jos kuitu blokkaa, niin toinen saman säikeen kuitu vuorotetaan automaattisesti (säie ei siis blokkaannu)

W2K: Säikeet

- **W2K tukee useita erilaisia ajoympäristöjä (Win16, Win32, Posix?, OS/2?), eroja:**
  - prosessien nimeäminen
  - säikeitä vai ei
  - prosessin kuvaaminen
  - prosessin resurssien suojaus
  - prosessien välinen kommunikointi
  - prosessien sukulaisuussuhteet
- **Perusta**
  - prosessit ja säikeet toteutettu olioina
  - prosessi = yksi tai useampi säie
- **Sekä prosessi- että säieoliolla synkronointiprimitiivejä**

W2K: Prosessi ja sen resurssit



(Fig 4.12 [Stal05])

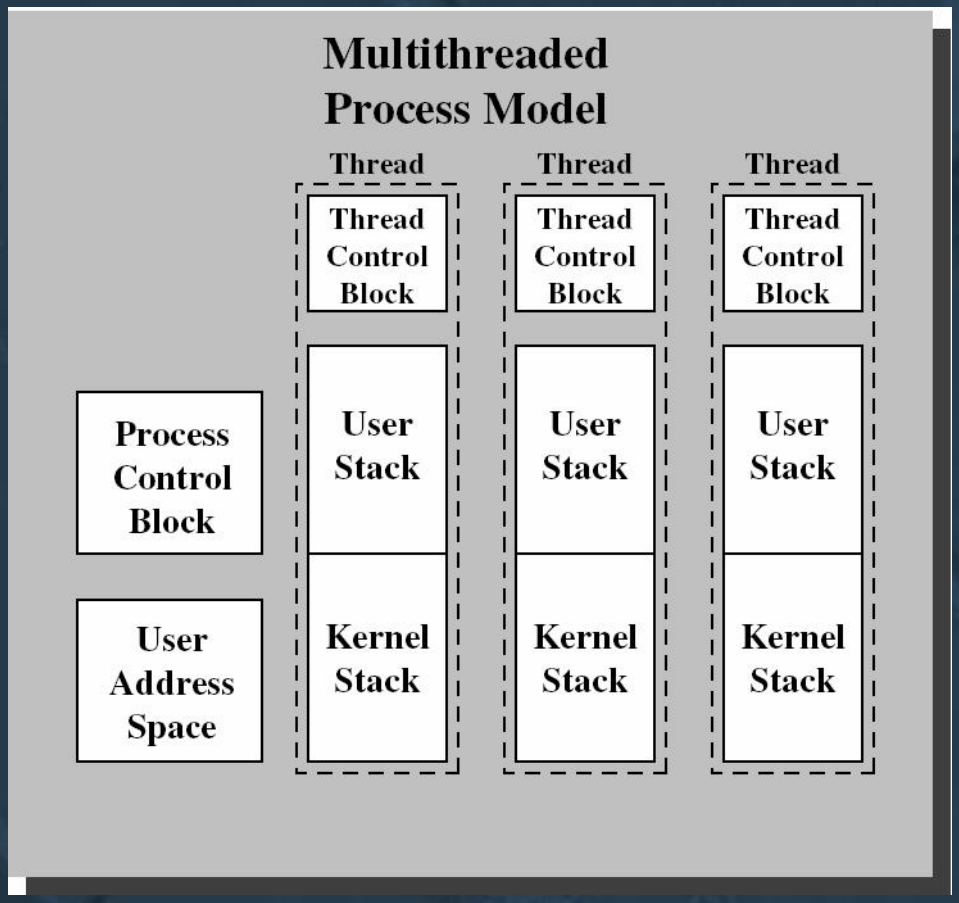
W2K: Prosessi ja sen resurssit

- **Oliotaulukko (object table)**
  - pääsy olioiden tietorakenteisiin
  - prosessiin liittyvät säikeet
- **Pääsypoletti (pääsylippu?) istuntoa käynnistettäessä**
  - määrää oletusoikeudet luotaville olioille
    - esim. säikeille
    - myöhemmin säie voi saada esim. asiakkaan pääsypoletin, jolloin lisää oikeuksia
  - rajoittaa / sallii olioiden käyttöä
    - kuka saa käyttää, kuka ei
    - käyttötapa: read, write, change, …

access token

- **Osa tietorakenteista vain KJ:n hallussa**
  - sovelluksella ei suoraa kahvaa (esim. prosessin oma pääsypoletti)

W2K Prosessi ja säikeet (Fig 4.2 [Stal 05])



Prosessi:

yhteisten resurssien kirjanpito,
passiivinen

Säie:

omien resurssien kirjanpito,
vuorottamisen yksikkö,
aktiivinen

W2K Säikeiden yhteistyö

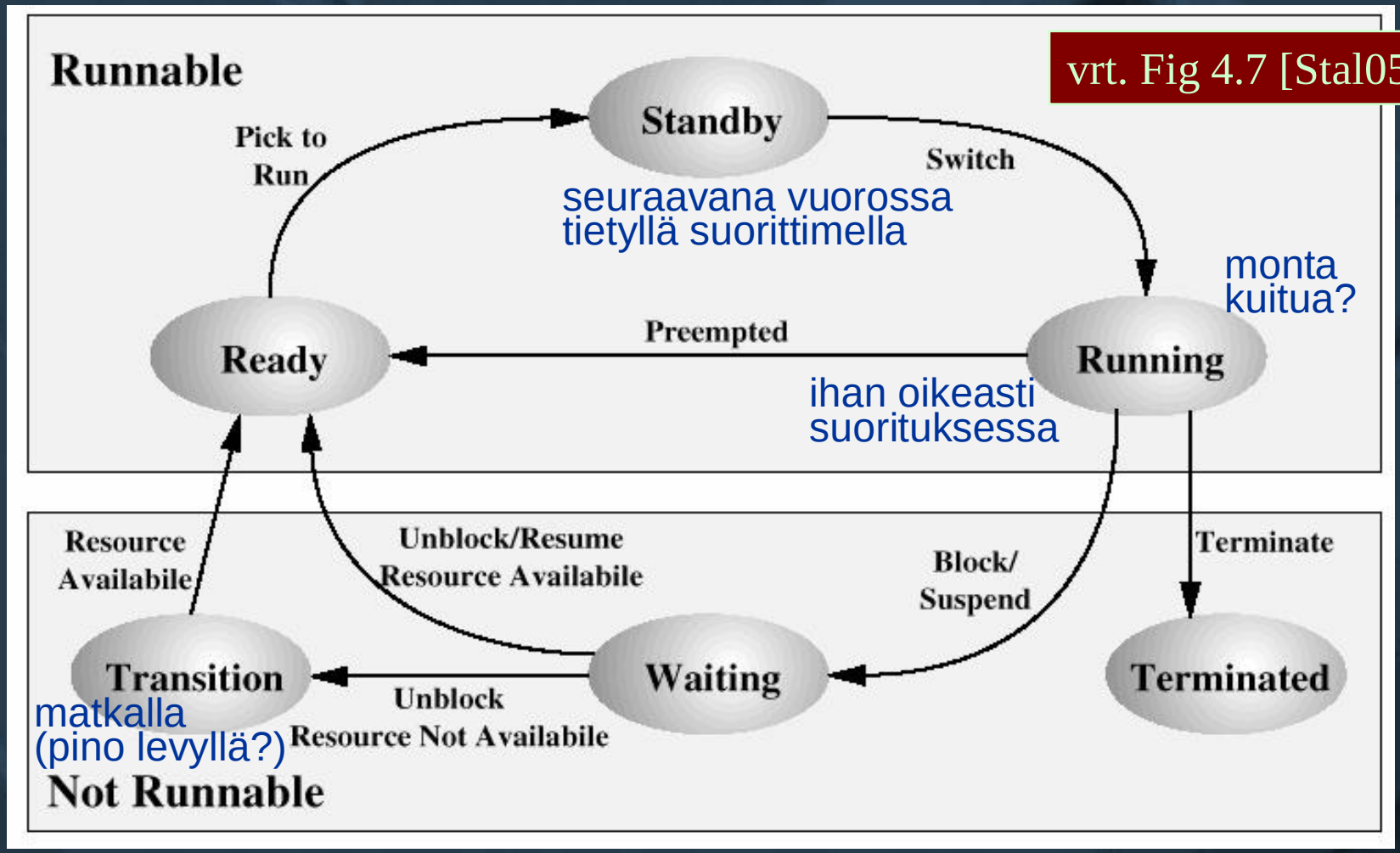
- **Yhteiskäyttöinen muisti**
  - saman prosessin säikeet voivat aina käyttää
  - järjestyy myös eri prosessien säikeiden välille
- **Sanomanvälitys**
  - kuka, kenelle, data
- **Semaforit**
  - poissulkeminen ja synkronointi

W2K Prosessi- ja säieoliot

- **prosessiolio**
  - attribuutit: I/O counters, quota limits, debug ports, …
  - palvelut
    - create process
    - terminate process
- **säieolio**
  - attribuutit: processor affinity, base priority, …
  - palvelut
    - create thread
    - suspend, resume
    - get context

Fig 4.13 [Stal05]

W2K Säietilat



vrt. Fig 4.7 [Stal05]

(Fig 4.14 [Stal05])

W2K: SMP

- **Prosessin säikeet voivat olla yhtäaikaa eri prosessoreilla**
  - myös W2K Executive
- **Prosessi/säie voidaan sitoa tietylle prosessorijoukolle** (affinity)
  - tieto kirjattu sekä PCB:hen että TCB:hen
  - vuorotetaan aina samoille prosessoreille
  - thread affinity $\subseteq$ process affinity
  - pehmeä sidonta (soft affinity) – yritä edes (välimuistin vuoksi)
  - kova sidonta (hard affinity) – muu ei kelpaa
- **Pyritään vuorottamaan uudelleen samalle CPU:lle**
  - välimuisti

Affinity: *kinship by marriage or adoption, the force attracting atoms to each other*

[hyperdictionary]

Kertauskysymyksiä

- Miksi Unix prosesseilla on tila zombie?
  Mitä kyseinen prosessi tekee enää järjestelmässä?
- Mitkä ovat suurimmat erot mikroydin KJ:llä ja monoliittisella KJ:llä
- Milloin W2K sovellus olisi parempi toteuttaa usealla kuidulla yhdessä säikeessä kuin usealla säikeellä? Miksi?
- Mitä hyötyä/haittaa on siitä, että W2K prosessit ja säikeet ovat olioita?
- Mitä eroa on Linux'illa ja W2K'lla prosessien ja säikeiden suhteen?