TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TEEMU LAAKSO

MIKROYTIMIEN KÄYTTÖ MODERNEISSA JÄRJESTELMISSÄ

Kandidaatintyö

Tarkastaja: Yliopisto-opettaja Marko Helenius

Jätetty tarkastettavaksi 26. syyskuuta 2018

TIIVISTELMÄ

TEEMU LAAKSO: Mikroytimien käyttö moderneissa järjestelmissä
Tampereen teknillinen yliopisto
Kandidaatintyö, 19 sivua
Syyskuu 2018
Tietotekniikan kandidaatin tutkinto-ohjelma
Pääaine: Ohjelmistotekniikka
Tarkastaja: Yliopisto-opettaja Marko Helenius

Avainsanat: käyttöjärjestelmät, käyttöjärjestelmäytimet, mikroytimet, monoliittiset ytimet, hybridiytimet, käyttökohteet, L4

Tässä työssä tarkastellaan mikroytimien ja niihin perustuvien järjestelmien kehitystä ja käyttökohteista moderneissa järjestelmissä. Työ pyrkii löytämään reunaehtoja mikroytimien käytön mielekkyydelle ja selvittämään, millaisissa kohteissa todellisuudessa mikroytimiä käytetään.

Mikroytimien nähdään kehittyneen selkeästi viimeisten vuosikymmenten aikana. Niille on olemassa selkeitä käyttökohteita esimerkiksi terveydenhuollossa ja pankeissa, mutta niiden käyttö yleiskäyttöisiin järjestelmiin, kuten kotikäyttöön, ei ole mielekästä. Mikroytimien toiminnassa suurin käyttökohteisiin vaikuttava ominaisuus on prosessien välisen kommunikoinnin hitaus.

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

ABI	Ohjelman sisäinen rajapinta (engl. Application Binary Interface)
CPU	Prosessori (engl. Central processing unit)
IPC	Prosessien välinen kommunikointi (engl. Inter-process communication)

1. JOHDANTO

Käyttöjärjestelmä voidaan mieltää järjestelmän toimintaa eniten määrittteleväksi osaksi. Käyttöjärjestelmän ytimellä tarkoitetaan sitä osaa käyttöjärjestelmästä, joka suorittaa koodia ydintilassa (engl. kernel space). Ydintilassa suoritettavalla prosessilla on vapaa pääsy laitteistoresursseihin. Vastakohtana ydintilalle voidaan esittää käyttäjätila (engl. user space), jossa suoritettavat prosessit saavat tarvitsemansa resurssit ytimen allokoimina. [26]

Käyttöjärjestelmien ytimillä (engl. kernel) on kaksi selkeää arkkityyppiä: monoliittiset ytimet (engl. monolithic kernel) ja mikroytimet (engl. microkernel) [26]. Yleisesti käytössä on myös ytimiä, joissa on ominaisuuksia molemmista. Näitä kutsutaan hybridiytimiksi.

Kandidaatintyön aihetta motivoi mikroytimien kehittyminen viimeisten vuosikymmenten aikana. Mikroytimet kehityskaarensa alkupäässä olivat järjestelmiä, joiden käyttöä ei nähty sopivana, mutta ne ovat kehittyneet tilaan, jossa niille on olemassa sopivia käyttökohteita.

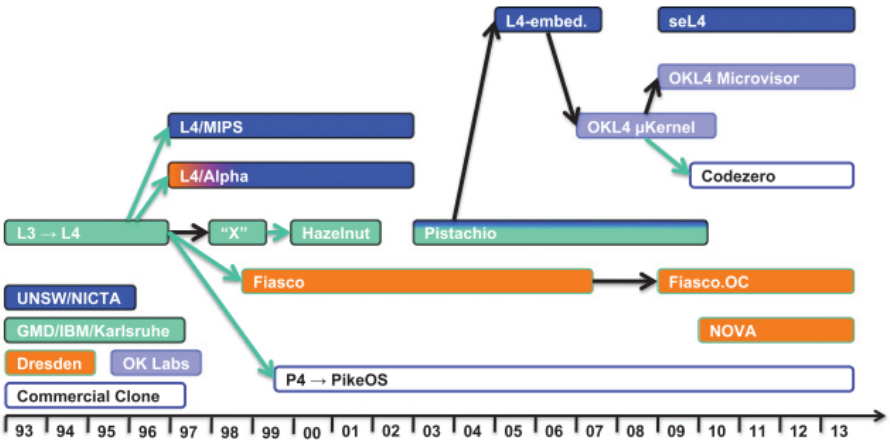
Tässä kandidaatintyössä keskitytään mikroytimiin ja niiden käyttöön. Työn tarkoituksena on selvittää, millaisiin käyttötapauksiin mikroytimet ovat sopiva valinta, ja käyttötapausten sekä mikroydinten ominaisuuksien kautta pyritään luomaan reunaehtoja järjestelmälle, jossa mikroytimen käyttö olisi edullista. Useissa tapauksissa esimerkkinä käytetään seL4-mikroydintä, sillä sen lähdekoodi on avointa ja ytimen dokumentaatio helposti saatavilla.

Työ esittelee luvussa 2 lukijalle suppeasti käyttöjärjestelmien historiaa. Luvussa 3 esitellään erilaisia ydintyyppejä, joista tarkempi katsaus luodaan mikroytimiin alaluvussa 3.2. Luvussa 4 tarkastellaan erilaisia käyttökohteita mikroytimille. Lopuksi luvussa 5 esitellään johtopäätöksiä ja luvussa 6 suppea yhteenveto työn sisällöstä.

2. MIKROYDINTEN KEHITYKSEN HISTORIA

Ensimmäiset tietokoneet olivat valtavia laitteita, ja niiden käyttömahdollisuudet olivat rajallisia. Reikäkortit olivat työläämpiä kuin koodin kirjoittaminen nykyaikana. Mutta tietokoneet ovat kehittyneet sen jälkeen.

Kuva 1. *Laitteiston, käyttöjärjestelmän ja käyttäjän ohjelmien hierarkia.*



Kuva 2. *L4-mikroydinperheen yksinkertaistettu sukupuu, jossa mustat nuolet viittaavat koodiin ja vihreät nuolet ytimen sisäisen ohjelmointirajapinnan (ABI, Application Binary Interface) periytymistä. Ytimien taustat on värikoodattu alkuperän mukaan. [7]*

Käyttöjärjestelmä on se osa järjestelmäkokonaisuutta, joka toimii käyttäjän ohjelmien ja laitteiston välissä. Käyttöjärjestelmä yksinkertaisimmillaan on prosessi, joka abstraktoi laitteiston, käyttäjän ohjelmilta. Tätä hierarkiaa kuvataan kuvassa 1.

Monoliittiset ytimet olivat luonnollinen jatkumo prosessissa, jossa käyttöjärjestelmät rakentuivat hiljalleen saaden lisää ominaisuuksia. Ensimmäisiä askeleita kohti mikroytimen kehittymistä oli vuonna 1969 esitelty moniajojärjestelmä RC 4000. Moderniin käyttöjärjestelmäytimeen verrattuna yksinkertainen ja rajallinen, mutta muistutti mikroydintä siten, että se toimi monitorina laitteiston ja prosessien välissä. [6]

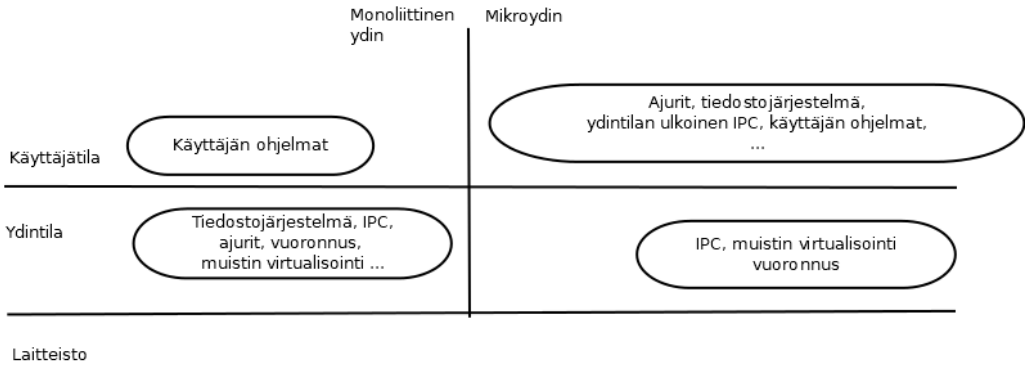
Tietotekniikan kehitys oli kuitenkin nopeaa, ja ohjelmiston suorituskyvystä tuli merkittävää. Vuonna 1993 Jochen Liedtken tutkimus- ja kehitystyö johti mikrokontrollien suorituskyvyn parantamiseen prosessien välistä viestintää tehostamalla. Liedtken kehittämä L4-mikroydin on pohjana useille moderneille mikroytimille, ja siitä on kehitetty muodollisesti varmistettu versio. [7, 15, 17]

L4 on mikroydinperhe, jolla on nähtävissä selkeä kehityshistoria, joka on verrattain hyvin dokumentoitua. Kuvassa 2 esitellään L4-mikroydinperheen kehitystä esimerkkinä mikroytimistä.

3. KÄYTTÖJÄRJESTELMÄYTIMET

Käyttöjärjestelmien arkkitehtuurissa ydin on se osa, joka suorittaa koodiaan ydintilassa. Tällöin ytimen suorittamalla koodilla on oma muistiavaruutensa, johon käyttäjätilassa suoritettavalla ohjelmakoodilla ei ole pääsyä. [26]

Järjestelmillä on kyky jakaa suoritettava ohjelmakoodi eri suoritustiloihin sen mukaan, kuinka paljon oikeuksia ohjelmalle halutaan antaa. Yleisesti CPU-arkkitehtuureihin on sisällytetty kyky suorittaa ohjelmia eri tiloissa [22], ja vastaavia rakenteita on mahdollista tuottaa esimerkiksi virtualisoiduissa suoritusympäristöissä, joskaan ei ongelmitta [1]. Yksinkertaisimmillaan suoritustilat voidaan erottaa toisistaan siten, että jaetaan oikeudet käyttäjätilaan ja ydintilaan. Tällöin ydintilassa suoritetaan järjestelmäydin, jonka osilla on täysi pääsy laitteistotasolle sekä ydintilan omaan muistiin. Käyttäjätila voi vuorovaikuttaa laitteiston kanssa vain ydintilasta tarjottavien palveluiden kautta ja tyypillisesti käyttää virtualisoitua muistia, joka tarjotaan ydintilasta. [13, 26]



Kuva 3. *Monoliittiset ja mikroytimet eroavat merkittävästi niiltä järjestelmän osilta, joita suoritetaan ydintilassa.*

Tyypillisesti käyttöjärjestelmäydin tarjoaa vähintään muistin virtualisoinnin, vuorontamispalvelut sekä matalan tason IPC:n. Näiden lisäksi ydin tarjoaa abstraktiotason laitteiston käyttöön. Erityyppiset ytimet eroavat merkittävästi juuri ydintilassa suorittamiensa ja tarjoamiensa palveluiden osalta, kuten esitetään kuvassa 3. Esimerkiksi Singularity-ytimen suorittamia tehtäviä ovat muistinhallinta, säikeiden vuoronnus ja luonti sekä tuhoaminen, mutexien tarjoaminen säikeille, viestikanavien tarjoaminen prosesseille sekä laitteiston abstraktointi. [31]

3.1 Monoliittiset ytimet

Monoliittiset ytimet sisällyttävät ytimeen kaikki käyttöjärjestelmän palvelut ja prosessit, jolloin ytimen ulkopuolella suoritetaan vain käyttäjän prosesseja. Tällöin ytimessä suoritetaan muun muassa IPC, tiedostojärjestelmä (engl. file system), laiteajurit ja muistinhallinta. [26]

Monoliittisia ytimiä käytetään yleiskäyttöisissä järjestelmissä, joissa palvelujen karsiminen ei ole mielekästä. Yleisiä monoliittisiin ytimiin perustuvia moderneja käyttöjärjestelmiä ovat Linux [18] ja BSD [20]. Myös MS-DOS käytti monoliittista ydintä, mutta uudempi NT-järjestelmä kehitettiin hybridiytimellä [29].

Ytimen toimiessa tiiviinä kokonaisuutena, sen virheensietokyky on heikompi. Monoliittinen rakenne johtaa tilanteisiin, joissa yhden käyttöjärjestelmän osan pysähtyminen saattaa pysäyttää koko järjestelmän toiminnan, ja vaatia laitteen uudelleenkäynnistämisen. [31]

Monoliittisten ytimien heikkoutena voidaan joissakin käyttökohteissa nähdä myös eri järjestelmän osien kuuluminen ytimeen, jolloin ei ole mahdollista korvata järjestelmän palvelua erilaisella toteutuksella ilman, että koko ydin joudutaan kääntämään uudelleen. Monoliittisesta ytimestä ei myöskään voida siten riisua tarpeettomia osia vain kytkemällä niitä pois päältä. [26]

3.2 Mikroytimet

Mikroytimissä pyritään suorittamaan ydintilassa vain ne osat järjestelmästä, jotka ovat välttämättömiä yksinkertaisellekin käyttöjärjestelmälle [12]. Näitä palveluita ovat IPC ytimen sisäisille osille, muistin virtualisointi, sekä vuorontaminen. [26]

Ytimen ulkopuolelle jäävät palvelut suoritetaan käyttäjätilassa, jolloin ne joutuvat kommunikoimaan keskenään järjestelmän tarjoamien ytimen ulkopuolisten kommunikaatioväylien avulla, ja käyttämään ytimen tarjoamia palveluita ytimen ulkopuolelle tarjottavien rajapintojen avulla. Ytimen ulkopuolisiin palveluihin lukeutuu esimerkiksi tiedostojärjestelmä ja laitteistoajurit. [26]

seL4-ydin tarjoaa rajapintansa kautta 7 erilaista oliota, joiden avulla ytimen palveluita käytetään. Näitä olioita luomalla voidaan varata resursseja prosessille, säädellä säikeitä, välittää viestejä ja hallinnoida muistia. [33]

Seuraavissa alaluvuissa käsitellään tarkemmin joitakin mikroytimien toimintamalleja ja ominaisuuksia. Lopulta alaluvuissa 3.2.4 ja 3.2.5 käsitellään mikroytimien selkeitä hyötyjä ja ongelmia.

3.2.1 Järjestelmäkutsut

Mikroydinten tarjoamien palveluiden vähäisyys johtaa tilanteeseen, jossa vaadittavien järjestelmäkutsujen määrä on vähäinen. Kun tiedostojärjestelmä ja laiteajurit suoritetaan

käyttäjätilassa, ei niiden käyttöön tarvita erillisiä järjestelmäkutsuja.

seL4-ytimen dokumentaatio esittelee pienen määrän erilaista järjestelmäkutsua. Järjestelmäkutsut seL4-ytimessä on toteutettu saman viestinvälityspalvelun avulla, jolla säikeet keskustelevat keskenään. Näin ollen kaikki 8 kutsua pohjautuvat kolmeen arkkityyppiin, joiden avulla viestinvälitys toimii: lähetys (engl. send), vastaanotto (engl. receive) ja luovutus (engl. yield). Näistä lähetys ja vastaanotto ovat estäviä viestikutsuja, ja luovutus katkaisee kutsujan suorituksen ja jää odottamaan seuraavaa sille allokoitua prosessoriaikaa. [33]

Kolmen peruskutsun lisäksi seL4-ydin tarjoaa lähetykselle ja vastaanotolle ei-estävät kutsut, jotka eivät jää odottamaan muiden prosessien toimenpiteitä. Loput tarjotut järjestelmäkutsut ovat yhdistelmiä peruskutsuista, esimerkiksi kutsu seL4_Call(), joka lähettää viestin, ja jää odottamaan vastausta, estäen kanavan käytön odotuksensa aikana.[33]

Järjestelmäkutsujen määrä luonnollisesti pienenee, kun ytimen toimintoja vähennetään, ja kasvaa, kun niitä lisätään. seL4-dokumentaatiossa listataan 11 järjestelmäkutsua debugkutsujen lisäksi, kun taas esimerkiksi Linux-ytimen tarjoama järjestelmäkutsujen määrä on huomattavasti suurempi. [30, 33]

3.2.2 Muistinhallinta

seL4-mikroytimessä ytimen osille ei varata muistia dynaamisesti. Kun järjestelmä käynnistyy, se varaa tarvittavan määrän tilaa ytimen osille, ja käynnistää yhden käyttäjätilassa suoritettavan säikeen, joka saa hallintaansa loput käynnistäjän muistista. Tällöin käyttäjätilassa olevalla säikeellä on oma osoiteavaruutensa, ja sille on luovutettu oikeus jakaa saamaansa muistia lapsiprosesseilleen. Käynnistettävällä prosessilla, joka saa joltain vanhemmaltaan muistia, on muistialueen luovutuksen jälkeen kyky luovuttaa saamaansa muistia edelleen omille lapsilleen. [33]

Muisti on seL4-ytimessä jaettu kahteen tyyppiin: yleiskäyttöiseen (engl. general purpose memory) sekä laitemuistiin (engl. device memory). Yleiskäyttöinen muisti on nimensä mukaisesti prosessien käytettävissä miten tahansa, kun se on niille luovutettu. Laitemuistille ei voida antaa tyyppiä, joten sitä ei voida käyttää esimerkiksi viestinvälityksen päätepisteelle.[33]

3.2.3 Prosessien välinen viestintä

Prosessien välinen viestintä (engl. Inter-process communication, IPC) on mikroytimille olennainen ominaisuus. Koska monet palvelut kuten tiedostojärjestelmä eivät toimi ydintilassa, ei niillä ole pääsyä ydintilan muistiin, ja siten ne joutuvat kommunikoimaan samoja kanavia pitkin, kuin mitkä tahansa käyttäjän suorittamat ohjelmat.

IPC:n toteutuksesta esimerkkinä voidaan esittää esimerkkinä seL4:n toteutus. Järjestelmä tarjoaa rekistereitä viestien välitykseen siten, että ensimmäiset rekisterit tarjotaan suoraan

prosessorilta, ja lopuille on puskuri varatulla muistialueella. Jokaiseen IPC-viestiin on liitetty nelikenttäinen rakenne (tag), joka kertoo tietoja viestistä, kuten sen pituuden. Näiden tietojen perusteella järjestelmä määrittää muun muassa tarvittavien rekisterien määrän. Viesteissä on myös tieto lähettäjän kyvyistä (engl. capability), jotka ovat seL4:ssä käytetty mekanismi oikeuksien hallintaan. Prosessit voivat lähettää viestejä ytimelle, tai päätepisteiden kautta muille suoritettaville säikeille. Päätepisteet ovat synkronisia, joten niitä voi käyttää vain yhteen lähetykseen ja sen vastaanottamiseen kerrallaan. seL4:n käyttämän IPC-viestiformaatin tarkempi kuvaus löytyy taulukosta 1. [33]

Taulukko 1. *Prosessien välisen viestin (IPC) ja niille varatun puskurin muoto seL4-mikroytimessä. Perustuu lähteeseen [33]*

Tyyppi	**Nimi**	**Kuvaus**
seL4_MessageInfo_t	tag	Viestin tagi
seL4_Word[]	msg	Viestin sisältö
seL4_Word	userData	Tallennetun viestin muistiosoite
seL4_CPtr[]	caps	Kyvyt jotka siirretään
seL4_CapData_t[]	badges	Merkinnät päätepisteisiin liittyvistä kyvyistä, jotka on saatu
seL4_Cptr	receiveCNode	Viittaus CNode -solmuun, josta haetaan slotia
seL4_Cptr	receiveIndex	Viittaus slotiin suhteessa CNode -solmuun ylemmässä kentässä
seL4_Word	receiveDepth	Kuinka monta bittiä receiveIndexistä luetaan

Hieman seL4:n toteutuksesta eroava viestinvälitysmekanismi löytyy Singularitystä jonka viestinvälitys pohjatuu sopimuksiin (engl. contract). Singularityn viestit toimivat kahden prosessin välisessä kanavassa jolla on tarkalleen kaksi päätepistettä. Päätepisteillä on omat puskurinsa ja vain päätepisteen omistava säie voi poistaa viestejä puskurista. [11]

3.2.4 Hyödyt

Koska mikroytimet ovat suppeita eivätkä sisällä sellaisia palveluita kuin tiedostojärjestelmä, voidaan järjestelmä räätälöidä käyttökohteen vaatimusten mukaisesti. Kaikki ytimen ulkopuoliset palvelut voidaan lisätä haluttuina toteutuksina ilman, että ydintä joudutaan kääntämään uudelleen, kuten monoliittisten ytimien tapauksessa. Tämä mahdollistaa myös helpommin uusien palveluiden toteutuksen.

Mikroytimien rakenne mahdollistaa myös turvallisuuden kannalta olennaisen toiminnon: järjestelmän palveluiden suorittamisen käyttäjätilassa. Tällöin palveluilla ei ole pääsyä ydintilan muistiin, eli ydin on eristetympi kuin monoliittisen ytimen tapauksessa. Käyttäjätilassa suoritettavat palvelut tarjoavat myös mahdollisuuden vakaampaan järjestelmään, sillä käyttäjätilassa väärin toimiva ohjelma ei kykene niin helposti kaatamaan koko järjestelmää, kuin ydintilassa suoritettava ohjelma.[26]

Ydintilassa suoritetut ohjelmat omaavat vapaan pääsyn laitteistoresursseihin kuten muistiin. Mikäli yksi ydintilassa suoritettu ohjelma toimii virheellisesti esimerkiksi viruksen

tai bugin vuoksi, on sen mahdollista esimerkiksi käyttää muistia väärin, ja kaataa ydin. Käyttäjätilassa ajettavilla ohjelmilla on pääsy vain omaan virtualisoituun muistiinsa ja ne saavat laitteistoresursseja ytimen sääteleminä, jolloin itsenäisen prosessin toimiminen väärin tai yllättävillä tavoilla ei uhkaa ytimen hallinnoimaa muistiavaruutta.

Mikroytimiin perustuvat järjestelmät ovat yleensä myös pieniä ja vaativat vähän resursseja. Esimerkiksi MINIX 3-järjestelmän laitteistovaatimukset [21] ovat varsin vaatimattomat. Myös varsinaisen koodin määrä voi olla pieni, MINIX 3 :n koostuessa noin 12 tuhannesta koodirivistä [35]. Taulukossa 2 esitellään eri L4-ytimien koodirivien määriä.

***Taulukko 2**. Eri L4 -mikroytimien koodirivien määriä. Ilmoitetut luvut ovat tuhansia rivejä. Perustuu lähteeseen [7]*

Nimi	Arkkitehtuuri	Rivit (C/C++)	Rivit (asm)	Rivit kokonaisuudessaan
Alkuperäinen	486	0,0	6,4	6,4
L4/Alpha	Alpha	0,0	14,2	14,2
L4/MIPS	MIPS64	6,0	4,5	10,5
Hazelnut	x86	10,0	0,8	10,8
Pistachio	x86	22,4	1,4	23,0
L4-embedded	ARMv5	7,6	1,4	9,0
OKL4 microkernel v3.0	ARMv6	15,0	0,0	15,0
Fiasco.OC	x86	36,2	1,1	37,6
seL4	ARMv6	9,7	0,5	10,2

Taulukosta 2 on nähtävissä, jotkin L4-mikroytimet on toteutettu alle kymmenellä tuhannella koodirivillä. Kymmenissä tuhansissa koodiriveissä on toteutettu kaikki taulukoidut esimerkit, joista yksikään ei ylitä 40 tuhannen rajaa.

3.2.5 Ongelmat

Mikroytimien rakenne luo ytimen käytölle rajoitteita, jotka rajaavat tietynlaisia käyttökohteita selkeästi. Kuitenkaan sopivissakaan käyttökohteissa ei vältytä ongelmilta, sillä mikroytimen arkkitehtuurille ominaiset rajoitteet vaikuttavat myös niissä.

Mikroytimien käyttökelpoisuuden kannalta merkittävää on IPC:n nopeus, koska ytimen ulkopuolinen osa järjestelmästä kommunikoi sen välityksellä. Välitysmekanismin tarve jaetun muistin sijaan hidastaa järjestelmän toimintaa. L4-mikroydinperheen kehitystä edeltänyt L3-mikroydin saavutti noin 100 μs nopeuden viestinvälityksessä, kun taas ensimmäinen L4 -ydin 486 -arkkitehtuurilla viestinvälitykseen kuluva aika oltiin saatu kestämään vain 5 μs. Vuonna 2005 Pistachio-ydin pystyi saavuttamaan 1,5 Ghz Itanium 2-prosessorilla jopa 0.02 μs viestinvälitysajan. Kernelien suorituskyky on yleisestikin parantunut IPC :n osalta, kuten nähdään taulukosta 3. Merkittävää viivettä ei kuitenkaan olla pystytty eliminoimaan, eikä prosessorien kehitys ole juuri tuonut lisää nopeutta. [7]

Kuten nähdään, on IPC kehittynyt paljon nopeammaksi. Tämä on mahdollistanut laajempaa käyttöä mikroytimille, mutta sen kuluttama aika on silti olemassaoleva rajoite, jo-

Taulukko 3. *Eri L4-mikroytimien suoriutuminen yksisuuntaisesta, osoiteavaruudet ylittävästä viestistä. Perustuu lähteeseen [7]*

Nimi	Vuosi	Arkkitehtuuri	Prosessori	Taajuus (MHz)	Kesto (μs)
Alkuperäinen	1993	486	DX50	50	5,00
Alkuperäinen	1997	x86 (32-bit)	Pentium	160	0,75
L4/MIPS	1997	MIPS64	R4700	100	1,00
L4/Alpha	1997	Alpha	21064	433	0,17
Hazelnut	2002	x86 (32-bit)	Pentium II	400	0,68
Hazelnut	2002	x86 (32-bit)	Pentium 4	1400	1,38
Pistachio	2005	IA-64	Itanium 2	1500	0,02
OKL4	2007	ARM v5	XScale 255	400	0,64
NOVA	2010	x86 (32-bit)	Core i7 (Bloomfield)	2660	0,11
seL4	2013	x86 (32-bit)	Cire i7 4770 (Haswell)	3400	0,9
seL4	2013	ARM v6	ARM11	532	0,35
seL4	2013	ARM v7	Cortex A9	1000	0,32

ka on otettava huomioon mikroytimiä tarkasteltaessa. Kun viestit joudutaan välittämään muistiavaruudesta toiseen, ei voida välttyä viivästyksiltä verrattuna jaettuun muistiin.

IBM kehitti oman mikroytimensä 1990-luvulla. Tämän mikroytimen avulla pyrittiin rakentamaan monipalvelinjärjestelmää, mutta mikroydin ei ollut soveltuva sellaiseen käyttöön. Mikroytimen käytöstä tehty tutkimus totesi kohdistetumpien järjestelmien olevan sopivampia, ja monimutkaisten olioiden käytön ohjelmakoodissa haittaavan järjestelmän toimintaa.[25]

Ytimen ulkopuolella toimiessaan prosesseilla on oma muistiavaruutensa. Käytännössä tämä tarkoittaa prosessien olevan eristettyjä ytimestä sekä toisista prosesseista. Tämä tarjoaa teoriassa vakaamman järjestelmän, sillä prosessi ei kaatuessaan tuki ydintä. Eristyvyydestä seuraava luotettavuus voi olla kyseenalaista. Minix 3-järjestelmästä on dokumentoitu mahdollisuus koko järjestelmän kaatumiseen viallisen laiteajurin vuoksi [14]. Eristyvyyttä rajoittavat siis prosessien väliset riippuvuudet, jolloin myös mikroytimeen on tarve luoda mekanismeja viallisen toiminnan valvomiseksi. [31]

Mikroytimien käyttöä hankaloittaa niille ominainen ominaisuuksien puute. Yleisesti tämä näkyy laitteistoajurien puutteena: esimerkiksi Minix 3:ssa ei ole lainkaan tukea USB-laitteille [21]. Tämä on luonnollinen seuraus ytimen ulkopuolella suoritettavista laitteistoajureista, mutta hankaloittaa järjestelmien käyttöönottoa. Laitteistotuen puute voidaan kuitenkin nähdä merkityksettömänä puhuttaessa sulautetuista järjestelmistä, sillä omalle laitteistolleen joutuisi toteuttamaan omat laiteajurit käyttöjärjestelmästä riippumatta.

3.2.6 Mikroytimistä tehty tutkimus

Mikroytimien historia alkaa vuodesta 1969, jolloin tanskalainen P. B. Hansen julkaisi RC 4000-tietokoneelle käyttöjärjestelmän RC 4000 multiprogramming system, joka rakensi

hierarkian, jossa ohjelmia suoritetaan erilaisissa tiloissa. Järjestelmä mahdollisti tarjoamansa monitorin päällä useiden käyttöjärjestelmien suorittamisen omina prosesseinaan, ja tarjosi näille keskeytyksiä ja laitteistoresursseja. [6]

Mikroytimiin liittyvä tutkimus on 80-luvun alun jälkeen siirtynyt kysymyksestä "voiko mikroytimiä käyttää"kysymään pikemminkin "miten parantaa mikroytimiä". Kokonaiskuvaa tällaisesti kehityksestä esittää Heiser et al. artikkeli "L4 Microkernels: The Lessons from 20 Years of Research and Deployment". Merkittävä askel kehityksessä oli L4 -mikroytimen syntyminen ja sen osoittautuminen huomattavasti nopeammaksi kuin aiemmat mikroytimet. [7]

Myös suuret ohjelmistotalot ovat osoittaneet mielenkiintoa mikroytimiin: Microsoft kehitti vuosien 2003 ja 2010 välillä mikroytimeen perustuvaa käyttöjärjestelmää nimeltä Singularity, joka perusti turvallisuutensa kielellisiin mekanismeihin. Projektin tarkoituksena oli kehittää järjestelmä, joka korjaisi nykyisten käyttöjärjestelmien ongelmia muun muassa turvallisuuden ja vakauden osalta. Singularityn kehitysperiaatteina oli unohtaa yhteensopivuus vanhempien järjestelmien kanssa, ja tuottaa vain "tarpeeksi hyvää"suorituskykyä. Näillä ehdoilla järjestelmästä saatiin ennalta-arvattava ja luotettava, mutta menetettiin mahdollisuus käyttää ohjelmakoodia muista järjestelmistä. [11]

Merkittävä vaihe mikroytimiin kohdistuvassa tutkimuksessa oli L4-mikroytimen muodollinen varmentaminen, joka oli ensimmäinen käyttöjärjestelmäydin, jolle varmennus on tehty. Yhdysvaltain asevoimien tutkimusorganisaatio DARPA tutki seL4-mikroytimeen pohjautuvan järjestelmän käyttöä lentokoneiden ohjausjärjestelmissä tarkoituksenaan selvittää, olisiko varmistetusta järjestelmästä korvaamaan käytössa olevia reaaliaikaisia käyttöjärjestelmiä. Tutkimus näki ongelmallisena lähdekoodin avoimuuden, nuoren järjestelmän pienen ekosysteemin, sekä seL4:n varmistuksessa käytetyn Isabelle-ohjelman tarpeen varmistukselle. [8, 15, 34]

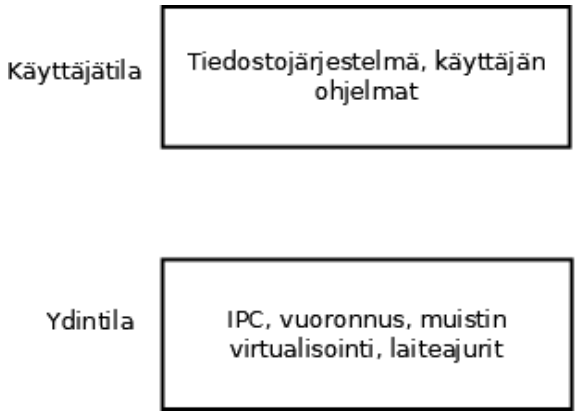
QNX-mikroytimen avulla on rakennettu järjestelmä nimeltä mTags, joka lisää IPC-viesteihin metadataa. Tämä mahdollistaa erilaisia seuranta- ja profilointimekanismeja ilman, että suoritettavien ohjelmien lähdekoodia pitää muuttaa. [23] mTagsin synty tarjoaa näkökulman IPC:n hyödyntämiseen työkaluna, eikä vain pakollisena hidasteena mikroytimien käytössä.

Mikroytimiin kohdistuva tutkimus näyttää keskittyneen tällä vuosituhannella pääasiassa erilaisiin kokeellisiin järjestelmien osiin joita ytimien avulla voidaan toteuttaa sen sijaan, että tutkittasiin yleisesti ytimien kehitystä. Tutkimuksissa on nähtävissä myös keskittyminen sulautettuihin ympäristöihin, ja niiden piirissä onkin tutkittu esimerkiksi dynaamista virranhallintaa [3] ja usean muistinhallinnan käyttämistä [16].

3.3 Nano-, ekso- ja hybridiytimet

Mikroytimet ja monoliittiset ytimet ovat hyvin erilaiset tavat lähestyä järjestelmien rakennetta. Molemmilla on omat etunsa ja ongelmansa. Ei siis ole yllättävää, että on kehitetty

myös ytimiä, jotka yhdistävät näiden ominaisuuksia. Tällaisia ytimiä kutsutaan hybridiytimiksi (engl. hybrid kernel), ja säilyttävät osan mikroytimien kyvystä suorittaa käyttöjärjestelmän palveluita ytimen ulkopuolella, sisältäen kuitenkin ytimen, joka ei ole yhtä suppea kuin mikroytimet. Esimerkki hybridiytimestä voisi olla käyttöjärjestelmäydin, joka suorittaa laiteajurit ydintilassa, mutta tiedostojärjestelmän käyttäjätilassa, kuten kuvassa 4.



***Kuva 4**. Mahdollinen rakenne ytimelle, jota voitaisiin kutsua hybridiytimeksi.*

Mikroytimien alatyypeiksi mahdollisesti luettavia ytimiäkin on kehitetty: nano- ja eksoytimiä. Nanoytimet ovat ytimiä, jotka toimivat pohjana mikroytimen rakentamiselle, tarjoten laitteistosta (kuten prosessoriarkkitehtuuri) riippuvat osat omana kokonaisuutenaan [2]. Eksoytimet puolestaan ovat MIT:ssä kehitetty ydintyyppi, joissa pyritään minimoimaan tai poistamaan laitteiston abstraktointi ja tarjoamaan siten ohjelmille mahdollisimman suora pääsy laitteistotasolle [5]. Nanoytimet vaativat siis ympärilleen muut ytimen palvelut, ja tarjoavat itsessään vain laitteistoriippuvaiset osat, jolloin muut osat järjestelmää voivat käyttää nanoytimen luomaa laitteistoabstraktiota. Eksoytimet sopivat järjestelmiin, joissa ajetaan räätälöityjä prosesseja, jotka hoitavat itse laitteistoresurssiensa käytön.

Tässä luvussa mainituista ydintyypeistä hybridiytimet ovat yleisiä, sillä Microsoftin käyttämä Windows NT perustuu hybridiytimeen [29]. Ekso- ja nanoytimien yhteydessä on mahdollista kyseenalaistaa, voiko niitä kutsua ytimiksi, sillä ne eivät ole kykeneviä toimimaan ilman koodia, joka niiden lisäksi toimii ydintilassa [8].

4. MIKROYTIMIEN KÄYTTÖ

Mikroytimillä ei ole ominaisuuksia, joilla ne voisivat kilpailla monoliittisten ja hybridiytimien kanssa kuluttajatason tietokonejärjestelmissä. Sen sijaan mikroytimien käyttö keskittyy erilaisiin järjestelmiin, joissa niiden ominaisuuksien edut korostuvat. Erityisesti mikroydinten rakenteen tarjoama vikasietoisuus, minimoitava järjestelmän koko ja ytimen eristyneisyyden tuoma turvallisuus ovat tietyissä käyttötapauksissa merkittäviä etuja.

Mikroytimien vikasietoisuus kannustaa mikroytimien käyttöön järjestelmissä, joiden toimintavarmuus pitää pystyä takaamaan. Toimintavarmuuden kannalta kriittisiä järjestelmiä ovat sellaiset, joiden halutaan olevan käyttökuntoisia niin suuren osan ajasta kuin mahdollista. Tällaisia järjestelmiä löytyy esimerkiksi sairaanhoidon piiristä sekä pankeista.

Mahdollisuus riisua ytimestä tarpeettomat osat ja siten saada ytimestä pienikokoinen mahdollistaa järjestelmän räätälöinnin suppeita ominaisuuksia tarvitseviin käyttökohteisiin. Tällaisista ominaisuuksista hyötyviä käyttökohteita voivat olla esimerkiksi jotkin sulautetut järjestelmät, mobiililaitteet ja reaaliaikajärjestelmät.

Tyypillisten, pelkkään mikroytimeen perustuvien järjestelmien lisäksi on kehitetty myös järjestelmiä, joissa hyödynnetään sekä mikroydintä että hybridi- tai monoliittista ydintä. Applen kehittämissä OS X-järjestelmissä ytimenä toimii Darwin, joka pitää sisällään Mach-mikroytimen sekä räätälöidyn toteutuksen BSD-ytimestä. [19]

Pankit ovat organisaatioita, joiden järjestelmät ovat usein toimintakriittisiä (engl. mission critical), mutta niiltä oletetaan myös verrattain korkeaa turvallisuutta. Monet pankkien järjestelmät ovat kuitenkin iäkkäitä [10]. Tämänkaltaisiin järjestelmiin mikroytimet ovat hyvä pohja, sillä ne voidaan nähdä turvallisina ja vakaina [31].

Seuraavissa alaluvuissa käsitellään mikroytimien käyttöä tietyn tyyppisten käyttökohteiden kannalta. Alaluvuille tehty jako ei ole muita alueita poissulkeva, sillä esimerkiksi sulautetut järjestelmät voivat olla toimintakriittisiä, mutta eri alaluvut pyrkivät tarkastelemaan eri tavalla käyttökohteen luomia vaatimuksia.

4.1 Sulautetut ja reaaliaikaiset järjestelmät

Ominaisuuksiltaan mikroytimet ovat sopivia sulautettuihin järjestelmiin, sillä niihin on verrattain helppo liittää vain järjestelmän tarvitsemat osat. Kaupallisessa käytössä on esimerkiksi BlueBerryn kehittämä QNX neutrino-ydin, jota on käytetty muun muassa matkapuhelimissa [24]. Laajassa käytössä on myös Green Hills Softwaren Integrity OS, joka on mikroytimeen perustuva reaaliaikakäyttöjärjestelmä. Integrity OS on löytänyt käyttöä monenlaisissa sovellutuksissa lentokoneista tulostimiin [27].

Mikroytimet soveltuvat myös sulautettuihin järjestelmiin joissa on reaaliaikavaatimuksia. L4-ydintä on käytetty reaaliaikajärjestelmissä, jonka lisäksi L4-ytimeen perustuvaa reaaliaikajärjestelmää nimeltä DROPS on kehitetty Dresdenin teknillisessä yliopistossa. [4, 26]

4.2 Toimintakriittiset järjestelmät

Toimintakriittisellä järjestelmällä tarkoitetaan sellaista järjestelmää, jonka toimiminen on olennaista käyttäjäyrityksen tai -organisaation toiminnan kannalta [32]. Tällaisen järjestelmän vioittuminen johtaa toiminnan keskeytymiseen tai merkittävään hidastumiseen.

Koska prosessien irrallisuus ytimestä luo järjestelmään vakautta, voi mikroytimiä hyödyntää järjestelmissä, joiden toiminnassa tapahtuvat katkokset pyritään välttämään keinolla millä hyvänsä. Modernissa yhteiskunnassa on paljon palveluita, joiden oletetaan olevan kokoajan saatavilla, kuten esimerkiksi pankkien sähköiset järjestelmät. Esimerkiksi maksuliikenteen häiriöt voivat aiheuttaa laajalti hankaluuksia sekä yrityksille että kuluttajille [9].

Toimintakriittinen ei kuitenkaan välttämättä tarkoita aina käynnissä olevia järjestelmiä. Muun yhteiskunnan mukana terveydenhuoltokin on sähköistynyt, ja esimerkiksi potilaiden valvontaan käytetään sähköisiä järjestelmiä [28]. Voidaan helposti kuvitella myös esimerkiksi kirurgisissa operaatioissa käytettävien laitteiden olevan toimintakriittisiä, vaikka ne ovatkin vain tarvittaessa käytettäviä.

Mikroytimien merkittävät ominaisuudet vikasietoisuuden kannalta ovat prosessien eristyneisyys ja ytimen pienuus. Pieni ydin tarkoittaa pienempää määrää prosesseja, jotka voivat toimia väärin ydintilan oikeuksilla, ja prosessien eristyneisyys ytimestä palvelee samaa tarkoitusta. Molemmat ominaisuudet rajoittavat yksittäisen prosessin kykyä pysäyttää ytimen ja sitä myötä järjestelmän toiminta. [31]

5. JOHTOPÄÄTÖKSIÄ

Mikroytimien käytössä rajoittavana tekijänä on vielä nykyäänkin prosessien välisen kommunikoinnin hitaus johtuen erillisistä muistiavaruuksista. Jotta mikroytimien toimintaa voitaisiin merkittävästi tehostaa, pitäisi luoda tehokkaampia viestinvälitysmekanismeja.

Seuraavissa alaluvuissa käsitellään mikroytimille soveltuvia käyttökohteita, niille soveltumattomia käyttökohteita, sekä esitellään joitakin selkeitä reunaehtoja käytölle.

5.1 Yleiset ja soveltuvat käyttökohteet

Mikroytimien ominaisuudet selvästi ohjaavat käyttöä tietyn tyyppisiin sovellutuksiin. Käyttökohteiden rajallisuus on ohjannut vahvasti myös mikroytimiin perustuvien käyttöjärjestelmien kehittymistä, jotka usein ovat erikoistuneet esimerkiksi sulautettuihin järjestelmiin. Samanaikaisesti ominaisuuksien luomat rajoitteet myöskin estävät mikroytimiä kilpailemasta monoliittisten ytimien kanssa yleiskäyttöisissä järjestelmissä kuten kotikäyttöjärjestelmät, joskin yleiskäyttöisyyteen ei ole havaittavissa selkeää pyrkimystäkään, vaan mikroytimiä pyritään kehittämään nimenomaan räätälöityihin järjestelmiin, joissa yleiskäyttöisyys ei ole tarpeen, vaan saattaa jopa olla haitallista ylimääräisen koodin vuoksi.

Tyypillisiä käyttökohteita mikroytimille ovat reaaliaikajärjestelmät ja sulautetut järjestelmät. Nämä ympäristöt korostavat mikroytimien vahvuuksia, ja voivat siten hyötyä mikroytimien sisällyttämisestä järjetelmään enemmän, kuin monoliittisuuteen nojaavat järjestelmät.

5.2 Soveltumattomat käyttökohteet

Mikroytimet eivät sovellu käyttökohteisiin, joissa niiden viestinvälityksen aiheuttamat viiveet ovat ongelma. Tällaisiksi kohteiksi voidaan ajatella kuluttajalaitteet, joissa suorituskyky on merkittävässä osassa.

Käyttökohteita rajoittaa myös mikroytimille ominainen ominaisuuksien puute. Yleiskäyttöisissä järjestelmissä, kuten monissa palvelin- ja kuluttajalaitteissa ei ole syytä korvata laajempien järjestelmien kuten Windows tai Linux tuomia palveluita itse toteutetuilla ytimen osilla.

5.3 Käytön reunaehdot

Mikroytimen käytölle voidaan määrittää karkeita reunaehtoja, joiden sisällä ytimien käyttö olisi mielekästä. Reunaehdoissa pitää lähteä liikkeelle rajauksista, joiden toteutumatto-

muus olisi toiminnalle haitallista, ja edetä sitten lievempiin seikkoihin, joiden toteutumattomuus vaikuttaa lähinnä mielekkyyteen ja käyttöönoton helppouteen.

Järjestelmän toiminnan on pystyttävä sisällyttämään itseensä prosessien välisen viestinnän viive. Mikäli tämä ei ole mahdollista, ei mikroytimiä voida käyttää.

Järjestelmän toteutuksessa mielekkyyteen vaikuttaa laitteistoajurien tarve. Mikäli järjestelmä ei olennaisesti hyödy mikroytimen käytöstä, ja laitteistoajureita olisi valmiiksi saatavilla toisenlaiselle alustalle, lienee syytä harkita toisen alustan käyttöä.

6. YHTEENVETO

Mikroytimien kehittyminen varsinkin IPC:n nopeuden kannalta on ollut merkittävä tekijä ytimien käytön mahdollistamisessa. Kehitys on johtanut varsin selkeään joukkoon mahdollisia käyttökohteita, ja nostanut näissä mikroytimien käytön edut ohi monoliittisten ytimien tai eksoytimiä muistuttavien järjestelmien edelle. On selvää, että mikroytimien kehityksen alkuvaiheita varjostanut hitaus ja epäily käyttökelpoisuudesta on pystytty ohittamaan, ja mikroytimille on olemassa selkeitä käyttökohteita.

Mikroytimien ominainen hitaus viestinvälitysmekanismeissa on silti este mikroydinten hyödyntämiselle yleiskäyttöisissä järjestelmissä, joissa esim. Windows-järjestelmät ovat merkittävässä osassa. Näin ollen mikroytimien käytön mielekkyys rajoittuu spesifimpeihin järjestelmiin.

LÄHTEET

[1] K. Adams, O. Agesen, A Comparison of Software and Hardware Techniques for x86 Virtualization, 2016. Saatavissa (viitattu 19.8.2018): https://www.vmware.com/pdf/asplos235_adams.pdf

[2] M. de Champlain, An architectural pattern for constructing nanokernels, teoksessa: Engineering Solutions for the Next Millennium. 1999 IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No.99TH8411), May, 1999, nide 1, s. 316–319 vol.1.

[3] J. Chen, Y. Y. Ke, A Dynamic Power Management Mechanism for Embedded System with Micro-Kernel Operating System, Applied Mechanics and Materials, vsk. 325-326, 2013, s. 916–921.

[4] DROPS - The Dresden Real-Time Operating System Project. Saatavissa (viitattu 26.8.2018): http://os.inf.tu-dresden.de/drops/

[5] D. R. Engler, M. F. Kaashoek, J. O'Toole, Jr., Exokernel: An Operating System Architecture for Application-level Resource Management, SIGOPS Oper. Syst. Rev., vsk. 29, nro 5, jou. 1995, s. 251–266.

[6] P. B. Hansen, RC 4000 SOFTWARE: MULTIPROGRAMMING SYSTEM, 1969. Saatavissa (viitattu 26.8.2018): http://brinch-hansen.net/papers/1969a.pdf

[7] G. Heiser, K. Elphinstone, L4 Microkernels: The Lessons from 20 Years of Research and Deployment, ACM Trans. Comput. Syst., Vol. 34, Iss. 1, Apr. 2016, pp. 1:1–1:29.

[8] G. Heiser, K. Elphinstone, I. Kuz, G. Klein, S. M. Petters, Towards Trustworthy Computing Systems: Taking Microkernels to the Next Level, SIGOPS Oper. Syst. Rev., vsk. 41, nro 4, hei. 2007, s. 3–11.

[9] S. Helin, J. Virtanen, Luottokorttijätti Visan maksuliikenne tökkii ympäri Eurooppaa – pitkiä jonoja muodostuu isoon helsinkiläiseen markettiin. Saatavissa (viitattu 25.8.2018): https://yle.fi/uutiset/3-10235266

[10] J. P. Honkanen, Vanhat it-järjestelmät hidastavat pankkeja – Nordea ja Aktia ryhtyivät jättipäivityksiin. Saatavissa (viitattu 25.8.2018): https://www.tivi.fi/Kaikki_uutiset/vanhat-it-jarjestelmat-hidastavat-pankkeja-nordea-ja-aktia-ryhtyivat-jattipaivityksiin-654880

[11] G. C. Hunt, J. R. Larus, Singularity: Rethinking the Software Stack, SIGOPS Oper. Syst. Rev., vsk. 41, nro 2, huh. 2007, s. 37–49.

[12] Indiana University, What are kernels and microkernels? Saatavissa (viitattu 25.8.2018): https://kb.iu.edu/d/agsv

[13] P. A. Karger, A. J. Herbert, An Augmented Capability Architecture to Support Lattice Security and Traceability of Access, teoksessa: 1984 IEEE Symposium on Security and Privacy, April, 1984, s. 2–2.

[14] J. Kinoshita, H. T. Omoto, P. d. F. Franco de Sa Barb, W. Leao, Is Minix more robust because it is microkernel?, IEEE Latin America Transactions, vsk. 9, nro 5, Sept, 2011, s. 838–844.

[15] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, S. Winwood, seL4: Formal Verification of an OS Kernel, teoksessa: Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles, New York, NY, USA, 2009, ACM, SOSP '09, Big Sky, Montana, USA, s. 207–220.

[16] Y.I. Klimiankou, A method for supporting runtime environments simultaneously served by multiple memory managers for operating systems based on second-generation microkernel, Programming and Computer Software, Vol. 41, Iss. 1, Jan, 2015, pp. 31–40.

[17] J. Liedtke, Improving IPC by Kernel Design, teoksessa: Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, New York, NY, USA, 1993, ACM, SOSP '93, Asheville, North Carolina, USA, s. 175–188.

[18] The Linux Kernel documentation. Saatavissa (viitattu 25.8.2018): https://www.kernel.org/doc/html/latest/

[19] Mac Technology Overview: Kernel and Device Drivers Layer, 2015. Saatavissa (viitattu 25.8.2018): https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html#//apple_ref/doc/uid/TP40001067-CH207-BCICAIFJ

[20] M. K. McKusick, K. Bostic, M. J. Karels, J. S. Quarterman, The Design and Implementation of the 4.4BSD Operating System, 1996. Saatavissa (viitattu 25.8.2018): https://www.freebsd.org/doc/en_US.ISO8859-1/books/design-44bsd/overview-kernel-organization.html

[21] MINIX 3 Hardware Requirements. Saatavissa (viitattu 25.8.2018): https://wiki.minix3.org/doku.php?id=usersguide:hardwarerequirements

[22] Modes of Addressing Used by Intel® Processors. Saatavissa (viitattu 19.8.2018): https://www.intel.com/content/www/us/en/support/articles/000007194/processors.html

[23] A. B. de Oliveira, A. Saif Ur Rehman, S. Fischmeister, mTags: Augmenting Microkernel Messages with Lightweight Metadata, SIGOPS Oper. Syst. Rev., vsk. 46, nro 2, hei. 2012, s. 67–79.

[24] QNX Software Systems Limited, QNX operating systems, development tools, and professional services for connected embedded systems, 2017. Saatavissa (viitattu 5.8.2018): https://blackberry.qnx.com/en

[25] F. L. Rawson, Experience with the development of a microkernel-based, multiserver operating system, teoksessa: Proceedings. The Sixth Workshop on Hot Topics in Operating Systems (Cat. No.97TB100133), May, 1997, s. 2–7.

[26] B. Roch, Monolithic kernel vs. Microkernel. Saatavissa (viitattu 26.8.2018): http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.9877&rep=rep1&type=pdf

[27] See What Our Customers Say About Us. Saatavissa (viitattu 26.8.2018): https://www.ghs.com/Quotes.html

[28] E.H. Shortliffe, J.J. Cimino, Biomedical Informatics: Computer Applications in Health Care and Biomedicine, 3rd ed., Springer, New York, NY, 2006.

[29] D. A. Solomon, The Windows NT kernel architecture, Computer, vsk. 31, nro 10, Oct, 1998, s. 40–47.

[30] syscalls(2) - Linux manual page. Saatavissa (viitattu 26.8.2018): http://man7.org/linux/man-pages/man2/syscalls.2.html

[31] A. S. Tanenbaum, J. N. Herder, H. Bos, Can we make operating systems reliable and secure?, Computer, vsk. 39, nro 5, May, 2006, s. 44–51.

[32] Technopedia, Mission Critical System. Saatavissa (viitattu 25.8.2018): https://www.techopedia.com/definition/23583/mission-critical-system

[33] Trustworthy Systems Team Data61, seL4 Reference Manual version 10.0.0, CSIRO, tou. 2018.

[34] S. H. VanderLeest, The open source, formally-proven seL4 microkernel: Considerations for use in avionics, teoksessa: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sept, 2016, s. 1–9.

[35] What is MINIX 3? Saatavissa (viitattu 25.8.2018): https://wiki.minix3.org/doku.php?id=www:documentation:read-more