

Etunimi Sukunimi

TYÖN OTSIKKO

Kuvaava alaotsikko

TIIVISTELMÄ

Etunimi Sukunimi
Työn otsikko: Kuvaava alaotsikko
Tampereen yliopisto
Tutkinto-ohjelma
Diplomityö
Tammikuu 2026

Tiivistelmä on suppea, itsenäinen esitys työstä: mikä oli ongelma, mitä tehtiin ja mitä saatiin tulokseksi. Kuvia, kaavioita ja taulukoita ei käytetä tiivistelmässä. Kandidaatintöissä, diplomitöissä ja pro graduissa tiivistelmän pituus tulee rajata yhteen sivuun, mutta lisensiaatintöissä ja väitöskirjoissa tiivistelmä saa olla pidempi. Laita työn pääkielellä kirjoitettu tiivistelmä ensin ja käännös sen jälkeen. Suomenkieliselle opinnäytteelle pitää olla myös englanninkielinen nimi arkistointia varten.

Avainsanat: avainsana1, avainsana2, ...

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality -ohjelmalla.

ABSTRACT

Etunimi Sukunimi
Thesis Title: Descriptive Subtitle
Tampere University
Thesis programme
Master's Thesis
January 2026

The abstract is a concise, self-containing description of the work: what was the problem, what was done, and what are the results. Do not include charts or tables in the abstract. In Bachelor's and Master's theses the abstract length is restricted to 1 page, but in Licentiate and Doctoral theses the abstract is allowed to be longer. Place the abstract in the primary language of your thesis first followed by its translation, if required. International students do not need to include an abstract in Finnish.

Keywords: keyword1, keyword2, ...

The originality of this thesis has been checked using the Turnitin Originality service.

ALKUSANAT

Alkusanoissa kiitetään yleensä niitä, jotka ovat vaikuttaneet työhön, kuten työn tarkastajia. Alkusanoissa ei ole soveliaista arvostella ketään, vaikka alkusanat eivät vaikutakaan arvosanaan. Alkusanojen on mahdollista yhdelle sivulle. Kirjoita alkusanojen loppuun päivämäärä, jonka jälkeen et ole enää muuttanut tekstiä.

Tampereella 1. tammikuuta 2026,

Etunimi Sukunimi

SISÄLLYS

JOHDANTO.....	1
1 DOKUMENTTIEN YHTEISMUOKKAUS.....	2
1.1 Järjestelmän vaatimukset.....	2
2 SYNKRONOINNIN LÄHESTYMISTAVAT.....	4
2.1 Operaatiomuunnos.....	4
2.1.1 Muunnosfunktio.....	4
2.2 Konfliktittomat replikoidut tietotyypit.....	4
3 SUUNNITTELU JA TOTEUTUS.....	5
3.1 Historian tallentaminen.....	5
4 ARVIOINTI.....	6
4.1 Tulosten tarkastelu.....	7
YHTEENVETO.....	8
LÄHTEET.....	9

LYHENTEET JA MERKINNÄT

TUNI Tampereen yliopisto

TUT Tampereen teknillinen yliopisto

KUVALUETTELO

Kuva 1.1. Dokumentin yhteisen kirjoittamisen tyypillinen kulku.....	2
Kuva 4.1. Muokkausten määrä viikonpäivän ja kellonajan mukaan seuratuissa istunnoissa.....	6

TAULUKKOLUETTELO

Taulukko 4.1. Keskimääräinen latenssi ja tallennetun datan määrä käyttäjämäärän mukaan.....	6
--	---

TEKOÄLYN KÄYTTÖ TÄSSÄ TYÖSSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

☐ Kyllä

☒ Ei

Riskien tiedostaminen

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien osat, joissa on hyödynnetty tekoälyä, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

JOHDANTO

Tekstidokumentteja laatii nykyään vain harvoin yksi ainoa kirjoittaja. Opiskelijat kirjoittavat seminaaritöitä pareittain, kehittäjät ylläpitävät teknistä dokumentaatiota yhdessä ja tutkimusryhmät viimeistelevät artikkeleita viime hetkellä useasta paikasta yhtä aikaa. Jotta tällainen kirjoittaminen toimisi, editorin on varmistettava, että kaikki osallistujat näkevät saman tekstin eikä yksikään muutos katoa.

Tämän työn tavoitteena on verrata kahta keskeistä lähestymistapaa dokumenttien synkronointiin reaaliaikaisessa yhteismuokkauksessa, operaatiomuunnosta ja konfliktittomia replikoituja tietotyyppejä, sekä arvioida niitä verkkopohjaisen editorin prototyypillä. Ensimmäinen luku määrittelee tällaiselle järjestelmälle asetettavat vaatimukset, toinen kuvaa molemmat lähestymistavat, kolmas prototyypin suunnittelun ja neljäs mittausten tulokset.

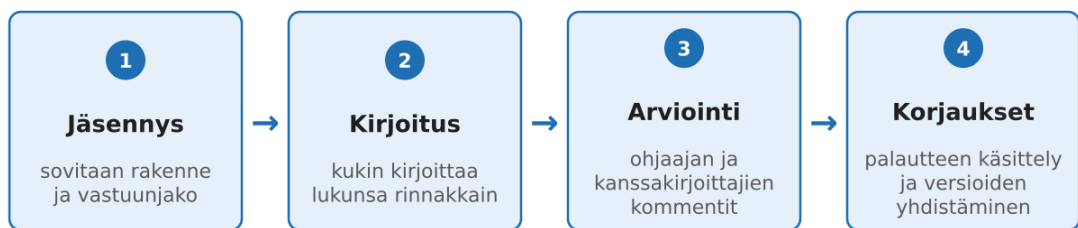
1 DOKUMENTTIEN YHTEISMUOKKAUS

Yhteismuokkauksella tarkoitetaan tilannetta, jossa useat käyttäjät muokkaavat samaa dokumenttia ja heidän muutoksensa välittyvät toisilleen ilman käsin tehtävää yhdistämistä. Sen mukaan, kuinka nopeasti muutokset leviävät, erotamme kolme perustyytä:

- **asynkroninen muokkaus**, jossa kirjoittajat välittävät toisilleen kokonaisia dokumentin versioita ja yhdistävät ne vasta jälkikäteen,
- **lukitukseen perustuva muokkaus**, jossa tiettyä dokumentin osaa saa kerrallaan muokata vain yksi kirjoittaja,
- **reaaliaikainen muokkaus**, jossa jokainen muutos näkyy välittömästi muille käyttäjille.

Tämä työ käsittelee yksinomaan viimeksi mainittua työtä. Se on käyttäjälle mukavin, mutta asettaa järjestelmän suunnittelulle suurimmat vaatimukset, koska muutoksia syntyy samanaikaisesti monessa paikassa ja ne saapuvat eri järjestyksessä.

Reaaliaikainen muokkaus ei kuitenkaan muuta pelkästään tekniikkaa vaan myös koko yhteistyön etenemisen. Haastateltujen käyttäjien kuvaamaa yhteisen kirjoittamisen tyyppillistä kulkua havainnollistaa Kuva 1.1.



vaiheita 2–4 toistetaan niin kauan kuin arviointi tuo esiin olennaisia huomautuksia

Kuva 1.1. Dokumentin yhteisen kirjoittamisen tyyppinen kulku

1.1 Järjestelmän vaatimukset

Nykyisten editorien käyttäjien haastatteluista ja alan kirjallisuudesta johdimme neljä vaatimusta, jotka yhteismuokausjärjestelmän on täytettävä:

1. konvergenssi, eli kun kaikki muutokset on toimitettu, dokumentin kaikilla kopioilla on sama sisältö,
2. tarkoituksen säilyminen, eli muokkauksen tulos vastaa sitä, mitä kirjoittaja näki muokaushetkellä,
3. vasteaika, eli paikallinen muokkaus näkyy välittömästi odottamatta palvelinta,
4. vikasietoisuus, eli lyhytaikainen yhteyskatkos ei johda keskeneräisen tekstin menettämiseen.

Kaksi ensimmäistä vaatimusta koskevat oikeellisuutta, kaksi jälkimmäistä käytettävyyttä. Käytännössä juuri välittömän vasteen ja taatun konvergenssin yhdistäminen osoittautuu koko suunnittelun vaikeimmaksi osaksi.

2 SYNKRONOINNIN LÄHESTYMISTAVAT

Reaaliaikaiset yhteismuokkausjärjestelmät suorittavat jokaisen muutoksen ensin paikallisesti ja vasta sitten lähettävät sen muille osallistujille. Ne eroavat toisistaan ennen kaikkea siinä, miten ne käsittelevät tilanteen, jossa kaksi muutosta syntyy samanaikaisesti ja kukin osallistuja vastaanottaa ne eri järjestyksessä.

2.1 Operaatiomuunnos

Operaatiomuunnos pohjautuu Ellisin ja Gibbsin työhön [1], jossa he kuvasivat ensimmäisinä, miten samanaikaista operaatiota voidaan muokata niin, että se voidaan suorittaa myös sellaisen toisen operaation jälkeen, josta sen tekijä ei tiennyt. Jokainen muokkaus kirjataan lisäys- tai poisto-operaatioksi tiettyssä tekstin kohdassa.

Jos esimerkiksi yksi käyttäjä lisää sanan kappaleen alkuun ja toinen samanaikaisesti poistaa sanan sen lopusta, poiston kohtaa on siirrettävä lisätyn sanan pituuden verran. Tämän siirron laskee muunnosfunktio.

2.1.1 Muunnosfunktio

Muunnosfunktio saa syötteenään samanaikaisten operaatioiden parin ja palauttaa niiden muunnetut versiot. Jotta dokumentin kaikki kopiot konvergoisivat, funktion on täytettävä niin sanotut muunnosominaisuudet, joiden toteutumista on yleisten operaatioiden kohdalla vaikea todistaa. Useimmat käytössä olevat järjestelmät tukeutuvat siksi keskitettyyn palvelimeen, joka määrää operaatioille yhden yhteisen järjestyksen.

2.2 Konfliktittomat replikoidut tietotyypit

Toisen lähestymistavan muodostavat konfliktittomat replikoidut tietotyypit, joihin viitataan yhteisesti lyhenteellä CRDT [2]. Operaatioiden jälkikäteisen muokkaamisen sijaan ne käyttävät dokumentista sellaista esitystapaa, jossa samanaikaiset operaatiot kommuoivat eikä niiden järjestys vaikuta lopputulokseen.

”Replikoitu tietotyyppi on konfliktiton, jos mitkä tahansa kaksi replikaa, jotka ovat vastaanottaneet saman joukon päivityksiä, ovat samassa tilassa riippumatta päivitysten toimitusjärjestyksestä.”

Dokumentin jokainen merkki saa yksikäsitteisen tunnisteiden, joka pysyy samana koko merkin olemassaolon ajan. Lisäyskohtaa ei tällöin ilmaista numerona vaan viittauksena viereisen merkin tunnisteeseen, joten muualla tekstissä tehdyt samanaikaiset muokkaukset eivät voi siirtää sitä.

3 SUUNNITTELU JA TOTEUTUS

Editorin prototyyppi koostuu verkkoasiakkaasta, synkronointipalvelimesta ja tietokannasta, johon palvelin tallentaa dokumenttien tilan jatkuvasti. Sekä asiakas että palvelin on toteutettu niin, että synkronointikerros voidaan vaihtaa ja molempia kuvattuja lähestymistapoja verrata saman käyttöliittymän avulla.

Asiakas ylläpitää dokumentin paikallista replikaa ja muuntaa jokaisen käyttäjän muokkauksen replikoidun tietotyyppin operaatioksi. Operaatiot lähetetään palvelimelle erissä, jotta nopea kirjoittaminen ei ruuhkauttaisi verkkoa. Saapuvan erän yksinkertaistettu käsittely palvelimella on esitetty seuraavassa listauksessa:

```
1 def kasittele_era(dokumentti, era):
2     for operaatio in era.operaatiot:
3         if operaatio.id in dokumentti.kasiteltyt:
4             continue
5         dokumentti.sovella(operaatio)
6         dokumentti.kasiteltyt.add(operaatio.id)
7     laheta_muille(dokumentti.id, era)
```

Koska operaatiot kommutoivat, palvelimen ei tarvitse järjestää eikä muuntaa eriä. Riittää, että se varmistaa, ettei mitään operaatiota suoriteta kahdesti ja että jokainen erä tavoittaa kaikki yhdistetyt asiakkaat.

3.1 Historian tallentaminen

Palvelin ei tallenna pelkkää lopullista tekstiä vaan koko operaatiolokin. Lokista voidaan palauttaa mikä tahansa dokumentin aiempi versio ja koota uudelleen pidempään yhteydettömänä olleen asiakkaan replikan tila. Tämä lähestymistapa vastaa muuttumattomaan tapahtumalokiin perustuvaa arkkitehtuuria, jota Kleppmann kuvaa [3]. Jotta loki ei kasvaisi rajattomasti, palvelin tiivistää sen säännöllisin väliajoin tilannevedoksi.

4 ARVIOINTI

Vertasimme prototyyppiä operaatiomuunnoksen referenssitoteutukseen todellisista muokkausistunnoista tallennetun aineiston avulla. Kustakin istunnosta mittasimme latenssin eli ajan, joka kuluu kirjoittajan tekemästä muokkauksesta siihen, kun muut osallistujat näkevät sen, ja istunnon keskimääräinen latenssi laskettiin kaavalla, jonka antaa (4.1):

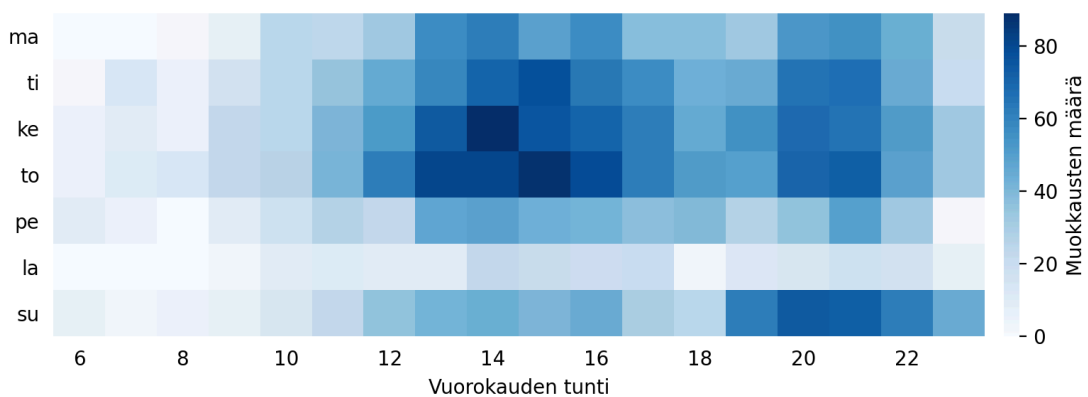
$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{toimitus}} - t_i^{\text{muokkaus}}) \quad (4.1)$$

missä n on istunnon operaatioiden lukumäärä. Eri määrillä samanaikaisia käyttäjiä mitatut arvot kokoaa yhteen Taulukko 4.1.

Taulukko 4.1. Keskimääräinen latenssi ja tallennetun datan määrä käyttäjämäärän mukaan

Käyttäjät	OT:n latenssi (ms)	CRDT:n latenssi (ms)	Data (kt)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

Käyttäjämäärän kasvaessa operaatiomuunnoksen latenssi kasvaa huomattavasti nopeammin, koska palvelimen on muunnettava jokainen operaatio kaikkia samanaikaisia operaatioita vastaan. CRDT:n pienemmän latenssin hintana on suurempi tallennetun datan määrä, sillä jokainen merkki kantaa omaa tunnistettaan. Kuormitustestin perustana olevaa muokkausten ajallista jakaumaa havainnollistaa Kuva 4.1.



Kuva 4.1. Muokkausten määrä viikonpäivän ja kellonajan mukaan seuratuissa istunnoissa

4.1 Tulosten tarkastelu

Tulokset vahvistavat, että pienemmissä ryhmissä molemmat lähestymistavat ovat tasavertaisia ja ratkaisevampaa on toteutuksen yksinkertaisuus. Kun samanaikaisia käyttäjiä on enemmän, CRDT:llä on selvä etu sekä latenssissa että vikasietoisuudessa yhteyskatkoksia vastaan. Täydentävät käyttäjähaastattelut, jotka toteutettiin laadullisen tutkimuksen periaatteiden mukaisesti [4], osoittivat, etteivät käyttäjät havaitse alle sadan millisekunnin eroa latenssissa, mutta keskeneräisen tekstin menettämistä yhteyskatkoksen jälkeen he pitivät vakavana ongelmana.

YHTEENVETO

Työssä verrattiin operaatiomuunnosta ja konfliktittomia replikoituja tietotyyppejä kahdena lähestymistapana dokumenttien synkronointiin reaaliaikaisessa yhteismuokkauksessa. Verkkopohjaisen editorin prototyypillä osoitimme, että CRDT:t mahdollistavat yksinkertaisemman palvelimen ilman operaatioiden keskitettyä järjestämistä ja tarjoavat suuremmilla käyttäjämäärillä pienemmän latenssin.

CRDT:iden pääasiallisena haittana on edelleen tallennettavan datan suurempi määrä. Seuraavaksi onkin tarkoitus selvittää, miten datan määrä vaikuttaa lokin säännölliseen tiivistämiseen tilannevedoksiksi ja miten prototyyppi käyttäytyy pitkäaikaisessa käytössä kymmenien tuhansien dokumenttien kanssa.

LÄHTEET

- [1] C. A. Ellis ja S. J. Gibbs, "Concurrency control in groupware systems", teoksessa *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York: ACM, 1989, s. 399–407. doi: [10.1145/67544.66963](https://doi.org/10.1145/67544.66963).
- [2] M. Shapiro, N. Preguiça, C. Baquero, ja M. Zawirski, "Conflict-free replicated data types", teoksessa *Stabilization, Safety, and Security of Distributed Systems*, Berlin: Springer, 2011, s. 386–400. doi: [10.1007/978-3-642-24550-3_29](https://doi.org/10.1007/978-3-642-24550-3_29).
- [3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol: O'Reilly Media, 2017.
- [4] J. Hendl, *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.