KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Julijus Kerys

Laiko ir duomenų sinchronizavimo metodai rungtynių monitoringo sistemose

Magistro darbas

Darbo vadovas
doc. E. Karčiauskas

Kaunas, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. E. Bareiša
2005 01

Laiko ir duomenų sinchronizavimo metodai rungtynių monitoringo sistemose

Magistro darbas

Vadovas

doc. E. Karčiauskas
2005 01

Recenzentas

doc. D. Rubliauskas
2005 01

Atliko

J. Kerys
2005 01

Kaunas, 2005

Turinys

Santrauka

Darbe analizuojamos informacijos sinchronizavimo problemos. Apžvelgiami du aspektai – laikrodžio sinchronizavimas, algoritmai ir jų panaudojimas, ir duomenų sinchronizavimas bei programos veikimo užtikrinimas, nutrūkus ryšiui su duomenų baze. Apžvelgiamos jau egzistuojantys produktai, jų panaudojimas, privalumai ir trūkumai.

Šiame darbe pasiūlomi sprendimai, panaudoti darbe „Krepšinio rungtynių registravimo ir analizės paskirstyta sistema“, bei kiti teoriniai sprendimai. Sinchronizavimo modeliai palyginami su kitais rinkoje jau kurį laiką esančiais panašiais sprendimais, aprašomi metodai ir algoritmai.

Summary

Time and data synchronization methods in competition monitoring systems

Information synchronization problems are analyzed in this thesis. Two aspects are being surveyed – clock synchronization, algorithms and their use, and data synchronization and maintaining the functionality of software at the times, when connection with database is broken. Existing products, their uses, cons and pros are overviewed.

There are suggested models, how to solve these problems, which were implemented in "Distributed basketball competition registration and analysis software system", and other theoretical solutions. Synchronization models are compared with other available solutions, the detail algorithms and methods are reviewed in this document.

1 Įvadas

Siekiant gauti platesnės informacijos apie krepšinio rungtynes, jos registruojamos, apdorojamos, ir pateikiami analizių rezultatai. Šiuo metu rinkoje nėra automatizuotos įrangos, galinčios atpažinti vaizdą ir tokiu pagrindu analizuoti rungtynes. Vienas žmogus spėja užregistruoti tik dalį per rungtynes įvykstančių įvykių. Siekiant užregistruoti daugiau informacijos, kyla poreikis padalinti registravimo procesą, kad jame galėtų dalyvauti keletas žmonių.

Registruojant keliese, reikia pasiskirstyti, kurią informacijos dalį kuris registruotojas registruos. Galima pasiskirti, kad kiekvienas registruotojas registruos atskirą komandą, arba kiekvienas registruotojas tam tikrus veiksmus. Esant daugiau žmonių, galima skaidyti dar labiau – kiekvienam registruotojui duoti tik tam tikrus kurios nors komandos veiksmus. Taip pat reikia paskirti, kuris registruotojas valdys rungtynių laikrodį. Pasidalijant registravimo darbais, reikia įvertinti registruotojų sugebėjimus. Praktika rodo, kad patyręs registruotojas spėja užregistruoti vienos komandos pagrindinius veiksmus, kartu valdydamas rungtynių laikrodį. Pagal veiksmų pasiskirstymus galima padaryti tam tikras modifikacijas vartotojo sąsajoje, kad būtų matomi tik tie veiksmai ar komandos, kurie jam yra paskirti registruoti. Tai leistų greičiau pasiekti reikiamą informaciją. Priklausomai nuo padalinimo, veiksmų aibės gali tiek persikirsti, tiek nepersikirsti.

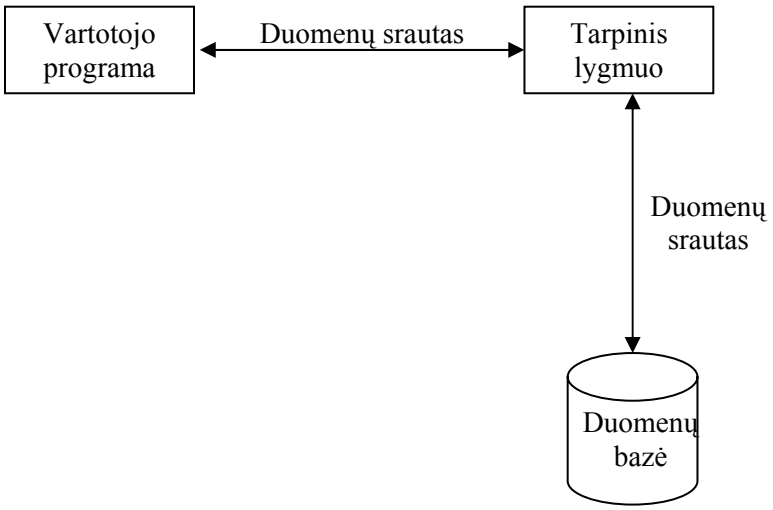
Šiame darbe nagrinėjamos dvi – laiko ir duomenų – sinchronizavimo problemos.

Laiko sinchronizavimas yra viena pagrindinių problemų paskirstyto skaičiavimo uždaviniuose. Laiko sinchronizavimo tikslas yra tai, kad atskiri nutolę procesoriai eitų tuo pačiu laiku, naudodami nuosavus laikrodžius ir apsikeitimą pranešimais per tinklą su kintančiomis informacijos perdavimo trukmėmis. Laiko sinchronizavimas plačiai taikomas programose, veikiančiose kompiuterių tinkluose. Vienas iš tokių pavyzdžių yra kai komunikaciniai protokolai nutraukia veiksmą, jei praeina jam skirtas maksimalus laikas. Taip pat duomenų bazių versijų valdymas ir lygiagretumas paprastai priklauso nuo laiko sinchronizavimo. Sinchronizuoti laikrodžiai leidžia naudoti paskirstytus algoritmus, kurie veikia tam tikrais laiko intervalais, taip supaprastindami jų struktūrą ir analizę. Keletas tokių (du iš jų buvo realizuoti) paskirstytų algoritmų aprašomi ir šiame darbe.

Kaip galimas duomenų saugojimo formas paminėsiu XML failus ir duomenų bazes. XML failuose duomenis saugoti nepatogu, nes tokiu būdu saugant informaciją, labai sunku realizuoti paskirstytą kelių vartotojų darbą. Duomenų bazė informacijos saugojimui yra žymiai efektyvesnė. Duomenų saugojimas DB (duomenų bazėse) daugeliu atveju pranašesnis už paprastus failus dėl tokių priežasčių:

- Informacija saugoma struktūrizuota ir suprantamesnė.
- Greičiau pasiekiami duomenys (atrenkama reikalinga informacija).
- Keli vartotojai gali lygiagrečiai dirbti su ta pačia informacija.
- Duomenims pasiekti naudojama standartizuota kalba. Tai paprastai leidžia su nedidelėmis modifikacijomis keisti įvairių gamintojų DB.

Tipinėse DB naudojančiose sistemose, nutrūkus ryšiui su nutolusiu kompiuteriu, kuriame įdiegta DB, darbas sutrinka. Reikia, kad nutrūkus ryšiui, programos duomenys išliktų, ir su jais būtų galima toliau dirbti. Tam paprastai tarp vartotojo programos ir duomenų bazės įterpiamas lygmuo, kuris apdoroja duomenų perdavimą į DB.



Pav. 1. Duomenų bazė su tarpiniu lygmeniu

Tarpinis lygmuo gali būti ir kita duomenų bazė, ir paprasta duomenų struktūra, tai priklauso nuo pasirinktos realizacijos.

Yra įvairių sprendimų, kaip užtikrinti pilną sistemos veikimą nutrūkus ryšiui. Juos galima suskirstyti į dvi grupes:

- Programavimo kalbų įrankiai,
- Duomenų bazių ir kiti įrankiai.

Programinės kalbos įrankių galima rasti kai kuriose programavimo platformose, tokiose kaip Microsoft .NET, konkrečiau jos dalyje ADO.NET (skirtoje darbui su duomenų bazėmis ir kitomis informacijos laikymo formomis). Java programavimo kalbos dalis JDBC, skirta darbui su duomenų bazėmis, neturi tiesioginių įrankių užtikrinti darbui dingus ryšiui, tačiau turi panašią architektūrą, mano manymu, tinkamą tam, kad ateityje JDBC galėtų nagrinėjamą funkciją realizuoti.

Taip pat atskirose duomenų bazėse yra specialių įrankių, galinčių užtikrinti tokias funkcijas.

2 Tikslai

Vystant projektą „Krepšinio rungtynių registravimo ir analizės paskirstyta sistema“ ir lyginant su egzistuojančiais sprendimais buvo susidurta su sinchronizacijos problemomis. Šio darbo tikslas – atskleisti šias problemas, parodyti realizuotus ir nerealizuotus sprendimus, įvertinti jų stipriąsias ir silpnąsias puses, taip pat apžvelgti situaciją rinkoje, ir galimas panaudojimo alternatyvas. Siekiant darbo tikslų, bus plačiau apžvelgiamas populiariausias šiuo metu laiko sinchronizavimo protokolas – NTP, jo pavyzdžiu parodoma, su kokiomis problemomis susiduriama sinchronizuojant laiką.

3 Laiko sinchronizacija

3.1 Sinchronizavimo problemos

Laikinė informacija turi tendenciją būti iškraipyta. Tai nutinka dėl dviejų priežasčių. Pirma, nutolę laikrodžiai gali eiti neteisingu greičiu – tada laikrodžių sinchronizacija per laiką silpnėja, ir antra, perduodant pranešimus per tinklą, pranešimų ėjimo laikas gali kisti, dėl ko laikrodžiai gali būti netiksliai sinchronizuoti. Praktikoje, šitoms paklaidoms visada yra tam tikros ribos: paprastai yra priimama, kad yra žinomas tam tikras minimalus ir maksimalus laikrodžių ėjimo greitis, ir kad pranešimai tinkle turi minimalų ir maksimalų laiką, per kurį jie gali pasiekti adresatą. Visų laiko sinchronizavimo algoritmų pagrindas yra tai, kaip pasinaudojama šitomis minimaliomis ir maksimaliomis ribomis.

Yra įvairių laiko sinchronizavimo uždavinių, iš jų galima išskirti du tipus:

- Išorinis sinchronizavimas, kai yra pagrindinis laiko šaltinis. Kiekvieno nutolusio laikrodžio uždavinys yra gauti patį mažiausią laiko intervalą $[a, b]$, kad tuo metu šaltinio rodomas laikas pakliūtų į intervalą $[a, b]$,
- Vidinis sinchronizavimas, kai yra laikrodžių sistema, kurie veikia savo greičiu, ir jų rodomi laikai turi kuo mažiau skirtis.

Yra įvairių teorinių ir praktinių darbų apie laiko sinchronizavimą. Tipiniuose teoriniuose algoritmuose laikrodžiai tarpusavio ryšio linijomis surenka reikiamą informaciją, po to siunčia vienam centriniam procesoriui apdoroti, kuris po to apdorotą informaciją siunčia atgal laikrodžiams, ir pagal atsiųstą informaciją laikrodžiai koreguoja savo rodomą laiką. Praktinės realizacijos labiau orientuotos į tinklinius algoritmus. Paprastai menkai susijusios sistemos naudoja išorinį sinchronizavimą, ir labai susijusios sistemos naudoja vidinį sinchronizavimą. Bene labiausiai paplitęs išorinio sinchronizavimo protokolas yra NTP („Network Time Protocol“), plačiai naudojamas internete. Dar galima paminėti vieną praktiškai naudojamą algoritmą „Probabilistic clock synchronization“ („Tikimybinis laiko sinchronizavimas“), kuriame priimama, kad pranešimų ėjimo laikai yra pasiskirstę pagal tam tikrą tikimybę, ir skirtingų pranešimų perdavimo laikai nėra tarpusavyje susiję. Pagal šias prielaidas galima išvesti tam tikrus tikimybinius laiko pasiskirstymus.

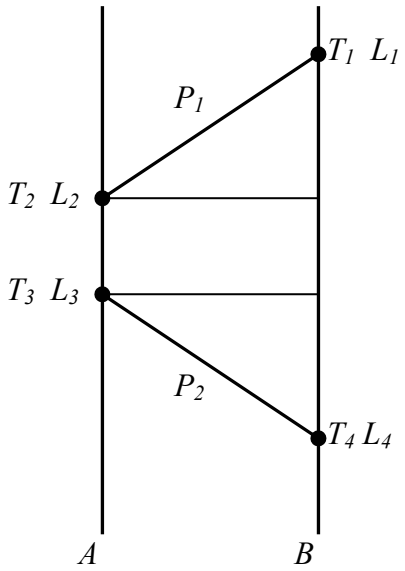
Kalbant apie laiko sinchronizavimo tikslumą, galima išskirti dvi tokias sąvokas:

- „Taiklumas“ – tai, kiek maksimaliai laikrodžio parodymas gali skirtis nuo realaus pasaulio laiko.
- „Precizija“ – tai, kiek maksimaliai gali skirtis keli iš eilės paimti laikrodžio parodymai.

Laikrodžio „precizijai“ viršijus kažkokią tai ribą, yra pagrindas atnaujinti laikrodžio „taiklumą“. Šituos dalykus sprendžia konkretūs algoritmai.

3.2 Supaprastintas NTP (SNTP)

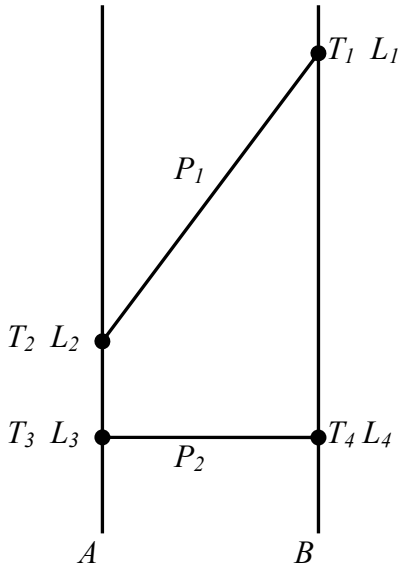
Kaip pavyzdį pateiksime supaprastintą NTP versiją, kurioje bus tik du laiko šaltiniai, iš kurių vienas ims laiką iš kito. Šaltiniai bus vadinami *A* ir *B*, ir šaltinio *B* laikas turi būti pririštas prie šaltinio *A* laiko. Šaltinis *B* visada turi laiko intervalą [I_1, I_2], tokį, kad bet kokiu laiko momentu šaltinio *A* laikas patenka į šį intervalą. Periodiškai iš šaltinio *B* siunčiamas pranešimas šaltiniui *A*, ir šaltinis *B* iškart atsako siųsdamas pranešimą šaltiniui *B*. Ryšys tarp šaltinių yra toks, kad visi pranešimai yra pristatomi nesumaišant jų eilės, ir jų vėlavimo laikai yra intervale [0, +∞]. Priimama, kad abu laikrodžiai eina vienodu greičiu, tai yra, laiko skirtumas tarp jų savaime nekinta. Tada tam, kad sužinoti *A* laiko ribas, užtenka žinoti *B* laiką, ir *A* ir *B* laiko skirtumo ribas bet kokiu momentu.



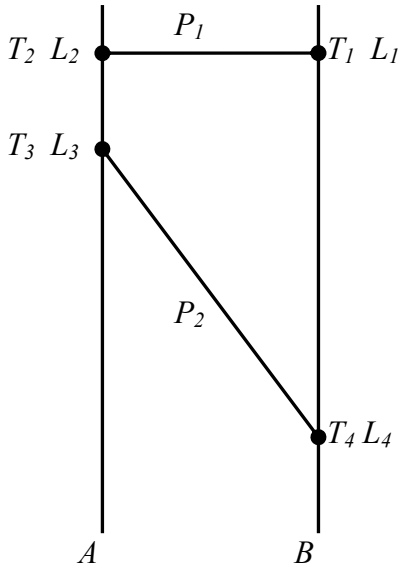
Pav. 2. Pranešimų ėjimo laikai

Pagal piešinį xx a, *A* siunčia pranešima *P1* šaltiniui *B*, ir *B* atsako pranešimu P_2 šaltiniui *A*. Kai P_1 yra išsiunčiamas (taškas T_1), šaltinis *B* įsimena išsiuntimo laiką L_1. Kai *A* gauna pranešimą P_1 (taškas T_2), jis įsimena gavimo laiką L_2. Kai *A* siunčia atsakymą P_2, jame būna įrašytas laikas L_2, ir išsiuntimo laikas L_3 (taškas T_3). Kai *B* gauna atsakymą P_2 (taškas T_4), pasižymi gavimo laiką L_4, ir apskaičiuoja bendrą pranešimų ėjimo laiką (pav. 2)

$LL = (L_4 - L_1) - (L_3 - L_2)$.



Pav. 3. Pranešimų ėjimo laikai. P_1 eina LL laiko.



Pav. 4. Pranešimų ėjimo laikai. P_2 eina LL laiko.

Tada nustatomos A ir B laiko skirtumo ribos. Iš vienos pusės, kadangi pranešimas keliavo bent 0 laiko vienetų (pav. 3), tai kai $P2$ buvo gautas, A laikrodis rodė bent L_3, ir iš kitos pusės, kadangi P_2 maksimaliai galėjo užtrukti LL laiko (pav. 4), tai reiškia, kad maksimalus A laikas, kai buvo gautas P_2 pranešimas, galėjo būti $L_3 + LL$. Iš to išplaukia, kad A ir B laiko skirtumas yra ribose $[L_3 - L_4, L_3 + TT - L_4]$.

Kaip minėta, pranešimai siunčiami periodiškai, ir jei kurio nors sekančio siuntimo metu gaunamas mažesnis laiko skirtumas, tai jis įsimenamas vietoj seno turimo. Šis algoritmas nėra geriausias pateiktam pavyzdžiui, kadangi jame atsižvelgiama tik į suminį abiejų pranešimų keliavimo laiką. Darbe [1] pateiktas sinchronizacijos grafų algoritmas, kuris suranda

trumpiausius tiek P_1, tiek P_2 pranešimų siuntimo laikus. Įvykus pav. 3 ir pav. 4 įvykiams, pagal šį algoritmą būtų nustatytas visiškai tikslus laikas.

3.3 NTP veikimo metodika

3.3.1 NTP apžvalga

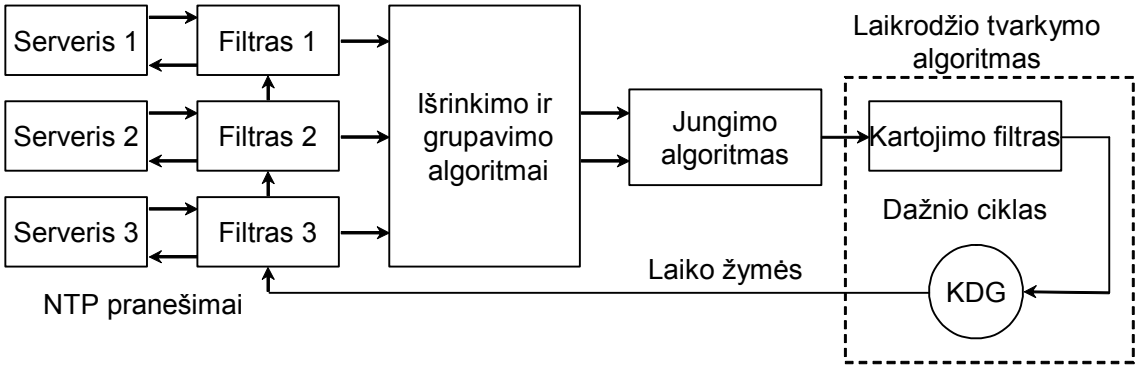
NTP ("Network Time Protocol") naudojamas internete kompiuterių mazgų ir maršrutizatorių laiko sinchronizavimui. Šiuo metu pasaulyje yra apie 10-20 milijonų NTP serverių ir klientų. Visose Microsoft Windows XP operacinėse sistemose irgi yra NTP klientas. Priklausomai nuo vietos, kur naudojamas NTP, tikslumas siekia nuo dešimčių milisekundžių iki mažiau nei mikrosekundės. Šiam protokolui jau apie 20 metų, ir jis vis dar tobulinamas. Tikslaus laiko poreikis egzistuoja įvairiose srityse – duomenų bazių tranzakcijos, biržų prekyba, kriptografija, aviacija, telekonferencijos ir daug kitų. Pvz., elektroninio parašo protokoluose labai svarbu, kada išsiųstas ir gautas pranešimas, tam, kad būtų galima apsisaugoti nuo įsilaužimo, kurio metu būtų pasiųstas kitas pranešimas vietoj tikro. Pranešime yra milisekundžių tikslumo laiko žymė, sudaranti tokias sąlygas, kad įsilaužėlis nespėja perimti originalo ir pasiųsti savo pranešimo.

Galima būtų apibrėžti keletą sąvokų apie NTP veikimą.

- Pirminiai serveriai laiką sinchronizuoja per radijo, palydovinį arba modeminį ryšį su nacionaliniais laiko standartais.
- Antriniai ir tolesni serveriai sinchronizuoja laiką su pirminiais serveriais per hierarchinę struktūrą.
- Patikimumas užtikrinamas pertekliniais serveriais ir ir skirtingais tinklo pranešimų ėjimo keliais.
- Sukurti algoritmai sumažina iškraipymus, sujungia informaciją iš kelių šaltinių ir išvengia neteisingai veikiančių serverių.
- Sistemos laikrodis yra koreguojamas naudojant prisitaikantį algoritmą, atsakingą už tinklo iškraipymų ir laikrodžio greičio nukrypimų ištaisymą.

Naujausios NTP versijos yra paremtos intervaline laiko sinchronizacija, plačiau apie tai galima pasiskaityti [4] darbe. Toliau pateiksiu naujausios versijos NTP struktūros, algoritmų ir sprendžiamų problemų apžvalgą.

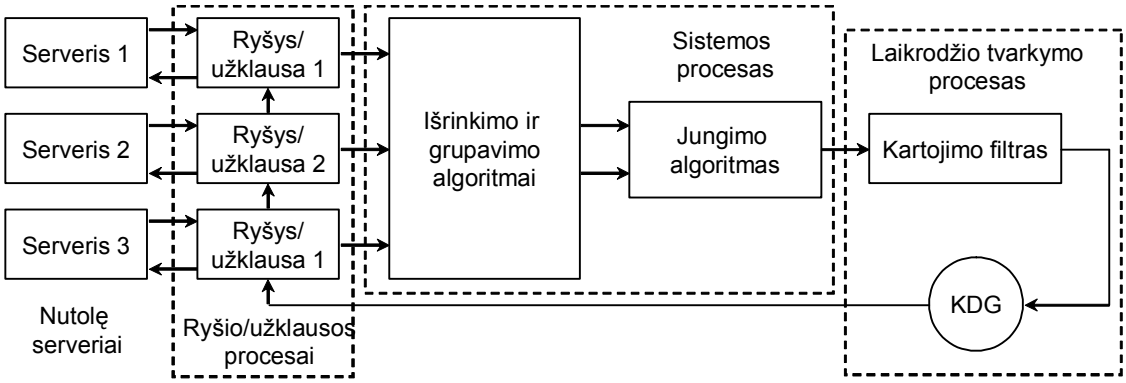
3.3.2 NTP architektūra



Pav. 5. NTP struktūra.

- Keletas serverių suteikia pertekliškumą ir įvairumą.
- Laiko filtrai išrenka geriausią iš aštuonių laiko nuokrypio pavyzdžių.
- Išrinkimo ir grupavimo algoritmai išrenka geriausius nuokrypius ir atmeta klaidingus nuokrypius.
- Jungimo algoritmas suskaičiuoja vidutinį laiko nuokrypį.
- Kartojimo filtras ir kintamo dažnio generatorius (KDG) įgyvendina dažnio atgalinio ryšio ciklą, kuriame sumažinami laikrodžio nuokrypiai.

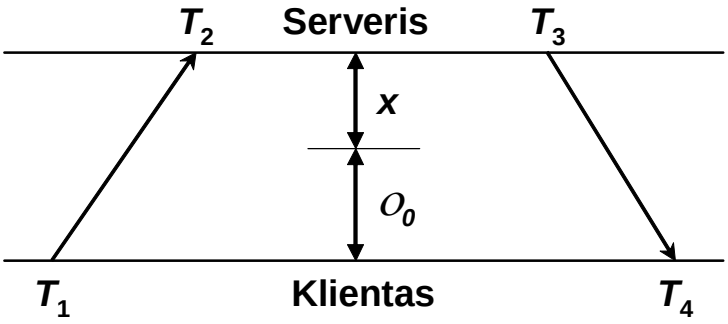
3.3.3 NTP proceso veikimas



Pav. 6. NTP procesas.

- Ryšio procesas įsijungia kai gaunamas pranešimas.
- Užklausos procesas siunčia pranešimus laiko intervalais, kuriuos nustato laikrodžio tvarkymo procesas ir nutolę serveriai.
- Sistemos procesas įsijungia kai gaunamas atnaujinimo pranešimas iš ryšio proceso.
- Laiko tvarkymo procesas įsijungia intervalais, kurie nustatomi pagal išmatuotus tinklo nuokrypius ir laikrodžio generatoriaus (KDG) dažnio nuokrypius.
- Laikrodžio korekcijos procesas (KDG) veikia 1 sekundės intervalais.

3.3.4 Laiko filtravimo algoritmas

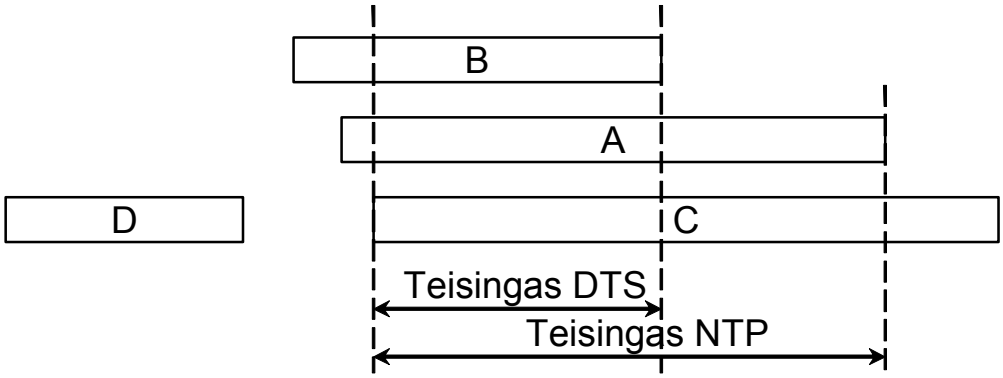


$$o = \frac{1}{2}[(T_2 - T_1) + (T_3 - T_4)]$$
$$d = (T_4 - T_1) - (T_3 - T_2)$$

Pav. 7. Laiko filtravimas.

- Pats tiksliausias nuokrypis o_0 yra išmatuojamas pagal mažiausią vėlavimą d_0.
- Tikslus laikas o turi būti $o \pm (d - d_0)/2$.
- d_0 yra įvertinamas kaip mažiausias iš paskutinių aštuonių vėlavimo matavimų ir (o_0, d_0) tampa nauju tinklo atnaujinimu.
- Kiekvienas tinklo atnaujinimas gali būti panaudotas tik kartą ir turi būti naujesnis nei prieš tai buvęs tinklo atnaujinimas.

3.3.5 Laiko išrinkimo principai

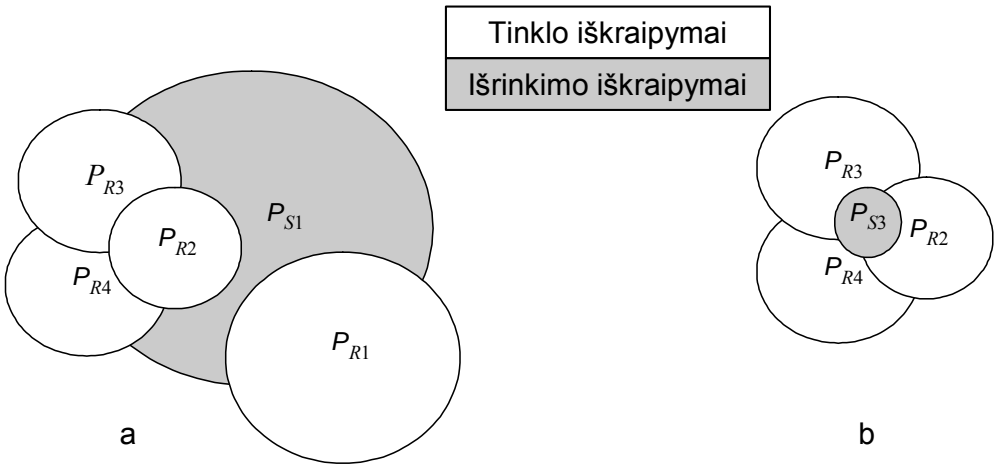


Pav. 8. Laiko išrinkimas.

- Kiekvieno kandidato korektiškumo intervalas yra taškų aibė intervale, kurio ilgis yra dvigubai didesnis nei sinchronizavimo atstumas, išcentruotas pagal suskaičiuotą nuokrypį.
- DTS intervale yra taškai iš didžiausio kiekio korektiškumo intervalų, tai yra, intervalų susikirtimas.
- Į NTP intervalą įeina DTS intervalas, kartu su suskaičiuotu kiekvieno kandidato nuokrypiu.

- Pagal korektiškumo reikalavimus, bent pusė kandidatų turi patekti į NTP intervalą.

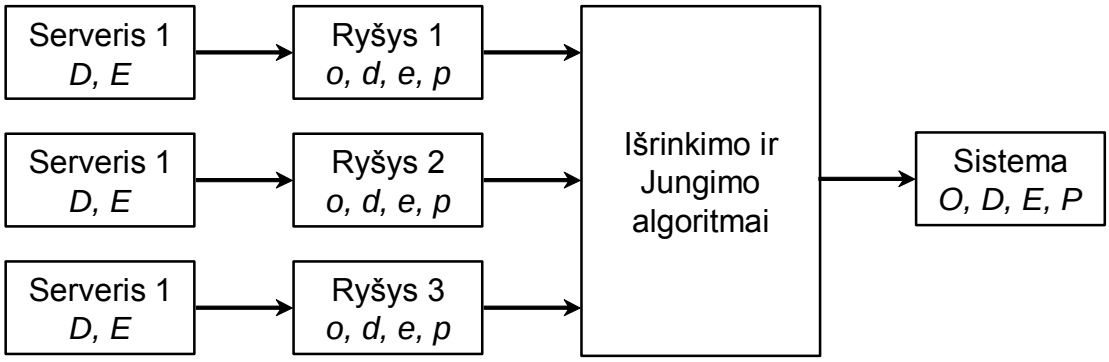
3.3.6 Grupavimo principai



Pav. 9. Iškraipymai.

- 1 kandidatas yra toliausiai nuo kitų, todėl jo išrinkimo iškraipymas P_{S1} yra didžiausias.
- (a) $P_{max} = P_{S1}$ ir $P_{min} = P_{R2}$. Kadangi $P_{max} > P_{min}$, algoritmas pašalina 1 kandidatą, kad sumažinti išrinkimo iškraipymą ir tęsia toliau.
- (b) $P_{max} = P_{S3}$ ir $P_{min} = P_{R2}$. Kadangi $P_{max} < P_{min}$, papildomų kandidatų pašalinimas nebesumažintų išrinkimo iškraipymo. Algoritmas baigiasi palikdamas P_{R2}, P_{R3} ir P_{R4}.
- Algoritmas sustoja ir tada, kai likusių kandidatų kiekis pasiekia minimalią normą arba iškraipymai pasiekia nustatytą minimalią ribą.

3.3.7 Duomenų srauto analizė

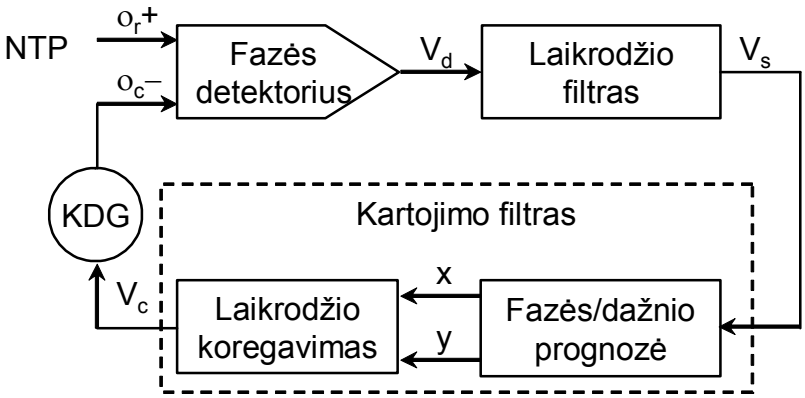


Pav. 10. Duomenų srautas.

- Kiekvienas serveris perduoda vėlinimą D ir dispersiją E, kurie yra apskaičiuoti pagal šakninį serverį, esantį sinchronizacijos medyje.

- Atėjus NTP pranešimui, ryšio procesas atnaujina ryšio nuokrypį *q*, vėlinimą *d*, dispersiją *e* ir iškraipymą *j*.
- Užklausos intervalais, laikrodžio išrinkimo ir jungimo algoritmai atnaujina sistemos nuokrypį *O*, vėlinimą *D*, dispersiją *E* ir iškraipymus *J*.
- Dispersijos *e* ir *E* didėja per laiką greičiu, priklausančiu nuo nustatytos dažnio tolerancijos.

3.3.8 Laikrodžio tvarkymo algoritmas



Pav. 11. Laikrodžio tvarkymo algoritmas.

- V_d yra funkcija, nusakanti fazės skirtumą tarp NTP ir KDG.
- V_s yra funkcija, priklausanti nuo laikrodžio filtro.
- *x* ir *y* yra fazės ir dažnio atnaujinimas atitinkamai, suskaičiuotas prognozės funkcijų.
- Laikrodžio koregavimo procesas veikia kartą per sekundę, suskaičiuodamas funkciją V_c, kuri kontroliuoja vietinio laiko generatoriaus (KDG) dažnį.
- KGD fazė yra palyginama su NTP faze, tuo užbaigiant ciklą.
- Laikrodžio tikslinimas tęsiamas ir lūžus ryšiui su NTP serveriais.

3.4 Protokolo laiko sinchronizavimas

Įvairiais būdais remiantis laikrodžio laiko sinchronizavimu, galima gauti labai tikslius protokolo įvykių laikus, rungtynes registruojant keliems asmenims. Yra keletas darbo metodų, besiskiriančių savo kokybe.

3.4.1 Laiko registravimas atskirai

Paprasčiausias (ir mažiausiai efektyvus) metodas yra toks, kai kiekvienas registruotojas turi rungtynių laikrodį, kuris rodo likusį kėlinio laiką, ir pats jį valdo. Toks variantas buvo naudojamas pačioje pirmoje tinklinėje „Kregio“ versijoje. Jo trūkumai:

- Kiekvienas registruotojas valdo laikrodį atskirai, dėl to didėja darbo sąnaudos ir mažiau dėmesio lieka rungtynių sekimui ir registravimui; dėl to gali sumažėti registravimo tikslumas.
- Kadangi laikrodžiai valdomi atskirai, nėra garantijos, kad jie rodo vienodą laiką, dėl to gali susidaryti situacijos, kai susikeičia įvykių eilė. Pvz., jei vienas žmogus registruoja netikslų metimą, o kitas registruoja atkovotą kamuolį, ir jei pirmojo registruotojo laikrodis atsilieka keliomis sekundėmis, tai protokole bus užregistruota, kad pirma buvo atkovotas kamuolys, o tik po to mestas metimas.

Kaip praktiškai parodė pirmoji tinklinė „Kregio“ versija, šitaip registruojant, nuokrypiai siekia po kelias sekundes. Toks variantas neracionalus.

3.4.2 Netiesioginis laiko dalinimas

Tai paprasčiausias metodas, kai laikrodis valdomas tik vieno žmogaus. Šis variantas yra realizuotas dabartinėje „Kregio“ versijoje. Kadangi dirbant tinkle informacija yra saugoma duomenų bazėje, su kuria kiekvienas registruotojas palaiko ryšį, galima tuo pasinaudoti laikrodžio sinchronizavimui. Vienas registruotojas (tai gali būti ir vienas registruojančių protokolą, bet geriau atskiras žmogus) valdo rungtynių laikrodį. Tam tikrais laiko tarpais laikrodžio parodymai įrašomi į duomenų bazę. Kartu įrašoma ir informacija, ar tuo metu laikrodis paleistas ar stovi. Likę registruotojai tokiais pat laiko tarpais (nebūtinai) ima laikrodžio parodymus iš duomenų bazės ir pagal juos koreguoja savo laikrodžio parodymus, įskaitant tai, ar laikrodis eina ar stovi. Laikrodžio ėjimo/stovėjimo informacija reikalinga papildomai registravimo korekcijai, kadangi žmonės paprastai apie pusantros sekundės vėluoja registruoti įvykius, ir, jeigu įvykis užregistruotas einant laikrodžiui, pakoreguojamas įvykio registracijos laikas.

Šitoks metodas tinkamas, kai visi registruotojai turi greitą ryšį su duomenų baze.

Metodo trūkumai:

- Didelės paklaidos, jei tarp registruotojų ir duomenų bazės yra prastas ryšys.
- Praktiškai tinkamas naudoti tik kai registruotojai ir duomenų bazė yra ypatingai greitame tinkle.
- Algoritmas visiškai netinkamas tuo atveju, jeigu didelė tikimybė, kad nutrūks ryšys su duomenų baze.

Metodo privalumai:

- Gana paprastas realizuoti.

Maksimali laiko fiksavimo paklaida priklauso nuo laiko intervalo, kuriuo į duomenų bazę įrašomas laikrodžio laikas, paimamas laikas ir nuo ryšio vėlinimo. Bendru atveju, paklaida siekia

*o= (įrašymo intervalas + nuskaitymo intervalas) / 2 + ryšio vėlinimas * 2.*

Realus skaitymo/rašymo intervalų vėlinimas yra abiejų intervalų suma, tačiau, kad subalansuoti šį vėlinimą, laiko gavėjas iš gauto laiko atima ½ intervalo vertės, todėl intervalų paklaida sudaro ±*(įrašymo intervalas + nuskaitymo intervalas) / 2*.

Ryšio vėlinimo paklaidos pateiktos žemiau esančioje lentelėje.

Tinklo tipas	Atstumas	Pralaidumas (Mbps)	Vėlinimas (ms)
LAN	1-2 km	10-1000	1-10
WAN	Visas pasaulis	0,01-600	100-500
MAN	2-50 km	1-150	10
Belaidis LAN	0,15-1,5 km	2-11	5-20
Belaidis WAN	Visas pasaulis	0.01-2	100-500
Internetas	Visas pasaulis	0.01-2	100-500

1 lentelė. Tinklo pralaidumai ir vėlinimai

Kad būtų korektiškai fiksuojamas protokolas, laikrodžių skirtumai negali sudaryti daugiau nei 0,1-0,2 sekundės. Jei skirtumas pasiekia 0,5 sekundės, dauguma atveju dar nebus iškraipyta įvykių eilė, tačiau, pridėjus žmogaus registravimo netikslumus, susidaro gana solidus apie sekundės ir daugiau registravimo netikslumas. Tai žymiai keičia kai kuriuos rodiklius, tokius kaip suskaičiuoti atakų laikai, ir kita.

Dabartinėje „Kregio“ versijoje laikrodžio skaitymo ir rašymo intervalai yra po 0,1 sekundės, tai reiškia, laikrodžio sinchronizavimo algoritmo paklaida sudaro ±0,1 sekundės. Kaip matome iš 1 lentelės ir iš anksčiau minėtų metodo trūkumų, toks algoritmas tinkamas naudoti tik LAN, MAN ir belaidžio LAN tinkluose, kuriuose vėlinimas siekia iki 0,02 sekundės. Kituose tinkluose vėlinimai siekia iki 0,5 sekundės (į abu galus tai sudaro 1 sekundę), o kaip rodo praktika, kartais ir daugiau (interneto atveju), dėl to jie tinkami naudoti tik tais atvejais, kai yra žinoma, kad ryšys yra kokybiškas ir jo vėlavimai sunkiai viršys 0,1 sekundės.

3.4.3 Tiesioginis laiko dalinimas

Šis metodas galimybėmis panašus į netiesioginį laiko dalinimą, nors jo realizacija gerokai skiriasi. Kaip ir 3.4.2 skyriuje aprašytame metode, vienas registruotojas valdo

rungtynių laikrodį, kurio informacija perduodama kitiems vartotojams. Skirtumas toks, kad informacija perduodama ne per duomenų bazę, o tiesioginiu ryšiu.

Metodo trūkumai:

- Didelės paklaidos, jei tarp registruotojų yra prastas ryšys.
- Praktiškai tinkamas naudoti tik kai registruotojai yra ypatingai greitame tinkle.
- Algoritmas visiškai netinkamas tuo atveju, jeigu didelė tikimybė, kad nutrūks ryšys tarp vartotojų.
- Kiekvienas vartotojas turi susitvarkyti prisijungimą ne tik prie duomenų bazės, bet ir prie registruotojo, kuris valdys laikrodį.

Metodo privalumai:

- Dvigubai mažesnės laikrodžio informacijos siuntimo intervalo paklaidos nei netiesioginio laiko dalinimo metode.
- Dvigubai mažesnės tinklo vėlinimo paklaidos (kadangi informacija siunčiama tik į vieną pusę).
- Ypač tinka tuo atveju, kai registruojama į viešo priėjimo duomenų bazę, su kuria ryšys gali būti lėtas dėl apkrovimo, rungtynių analizes stebint dideliam žmonių kiekiui.

Vėlinimų skaičiavimo teorija, išdėstyta 3.4.3 skyriuje, tinka ir šiam skyriui, tik šiame metode nuskaitymo intervalo vėlinimas yra lygus nuliui. Todėl paklaida lygi

o= (įrašymo intervalas) / 2 + ryšio vėlinimas.

3.4.4 Netiesioginis laikrodžio valdymo signalų dalinimas

Šis metodas yra 3.3.2 skyriuje aprašyto netiesioginio laiko dalinimo metodo patobulinimas. Skirtumas nuo netiesioginio laiko dalinimo metodo yra toks, kad vietoj laikrodžio rodomo laiko yra siunčiama informacija apie tai, kada laikrodis buvo paleistas ir kada sustabdytas. Informacija siunčiama į duomenų bazę tik tada, kai pasikeičia laikrodžio ėjimo būsena. Dėl to iš siuntėjo pusės nėra jokio laiko siuntimo intervalo, ir paklaida lygi

*o= (nuskaitymo intervalas) / 2 + ryšio vėlinimas * 2.*

Kaip matome, paklaida atitinka 3.4.3 skyriaus tiesioginio laiko dalinimo metodo paklaidą ir yra mažesnė už netiesioginio laiko dalinimo metodo paklaidą.

Metodo privalumai, lyginant su tobulinamu metodu:

- Mažesnės paklaidos.

- Algoritmas dalinai atsparus ryšio nutrūkimams – jei ryšys su duomenų baze nutrūks tik tuo momentu, kai nesikeis laikrodžio būsena (eina ar stovi), tai nepaveiks registravimo.

3.4.5 Tiesioginis laikrodžio valdymo signalų dalinimas

Šis metodas yra 3.3.3 skyriuje aprašyto tiesioginio laiko dalinimo metodo patobulinimas, ekvivalentus 3.4.4 skyriuje aprašytam patobulinimui. Šiame metode išvis nelieka laiko skaitymo / rašymo intervalų, ir jo paklaida lygi

o=ryšio vėlinimas.

Metodo privalumai, lyginant su tobulinamu metodu:

- Ypač mažos paklaidos.
- Kaip ir 3.4.4 skyriaus patobulintas algoritmas, šis algoritmas dalinai atsparus ryšio nutrūkimams – jei ryšys tarp registruotojų nutrūks tik tuo momentu, kai nesikeis laikrodžio būsena (eina ar stovi), tai nepaveiks registravimo.

Ypač maža vėlinimo paklaida sudaro realią galimybę šį metodą naudoti ir kiek lėtesniuose tinkluose (pagal 1 lentelę).

3.4.6 Laiko dalinimas su NTP ir SNTP

3.3 skyriuje buvo apžvelgtas šiuo metu populiariausias interneto laiko sinchronizavimo algoritmas NTP, taip pat 3.2 skyriuje – sena supaprastinta jo versija, vadinama SNTP. NTP yra per daug sudėtingas protokolas, kad būtų verta jį dar sykį realizuoti, ypač dėl to, kad yra įrankių, kurie realizuoja šį protokolą, ir kuriais galima sinchronizuoti patį kompiuterio laikrodį. 3.4.3 ir 3.4.5 skyriuose minėtuose metoduose galime pasinaudoti tiek NTP ar kitu protokolu tam, kad sulyginti registruotojų kompiuterių laikrodžius. Buvusi tinklo paklaida išnyksta, vietoj jos lieka protokolo paklaida. Turint minty, kad NTP protokolas internete laiką sulygina kelių dešimčių milisekundžių tikslumu, 3.4.5 skyriuje aprašytame metode mes gauname paklaidą:

o = sinchronizavimo protokolo paklaida.

Ši paklaida ne visada išlieka tokio dydžio, retesniais atvejais, kai vienas registruotojas fiksuoja laikrodžio būsenos pasikeitimą, ir kuris nors kitas registruotojas užfiksuoja protokolo įvykį, paklaida gali pasiekti ryšio vėlinimo paklaidą.

Šiuo atveju 3.4.5 skyriaus metodas duoda pakankamai mažas paklaidas, tačiau jis nėra labai atsparus tinklo trūkiams. Kaip pamatysime sekančiuose skyriuose, yra ir geresnių algoritmų.

3.4.7 Laikrodžio valdymo sekos fiksavimas

Šiame skyriuje aprašysiu algoritmą, kuris yra visiškai atsparus neriboto ilgio tinklo trūkiams. Jo pagrindinis principas yra tas, kad kartu su rungtynių protokolu turi būti išsaugota laikrodžio valdymo signalų seka. Į šią seką įeina visi valdymo signalai, kada buvo sustabdytas ir paleistas rungtynių laikrodis kartu su tuo metu rodytu laiku, ir kartu fiksuojamas registruotojo kompiuterio laikas. Turint šią veiksmų seką, bet kuriuo metu galima atkurti laiko tekėjimo seką, vykusią su rungtynių laikrodžiu. Pademonstruosiu tai pavyzdžiu.

Eil.Nr.	Kompiuterio laikas	Laikrodžio laikas	Signalas
1	18:00:00	10:00	Paleisti
2	18:00:15	09:45	Sustabdyti
3	18:00:25	09:45	Atgal 2 sek.
4	18:00:30	09:47	Paleisti
5	18:00:35	09:42	Pirmyn 3 sek.
6	18:00:40	09:34	Sustabdyti

2 lentelė. Laikrodžio valdymo seka.

Kaip matome, lentelėje pavaizduoti 4 galimi valdymo signalai – laikrodžio stabdymas, paleidimas, ir laiko korekcija į dvi puses. Tarus, kad gavusio seką registruotojo kompiuterio laikrodis rungtynių pradžioje rodė 0:00, iš sekos galima atkurti tokius laikrodžio parodymus:

Eil.Nr.	Realaus laiko intervalas	Laikrodžio laiko intervalas
1	0:00:00 – 0:00:15	10:00 – 9:45
2	0:00:15 – 0:00:25	9:45 – 9:45
3	0:00:25 – 0:00:30	9:47 – 9:47
4	0:00:30 – 0:00:35	9:47 – 9:42
5	0:00:35 – 0:00:40	9:39 – 9:34
6	0:00:40 – 0:10:06	9:34 – 0:00

3 lentelė. Atkurta laiko seka

Šitokioje atkurtoje sekoje dar reikia padaryti papildomas korekcijas tuo atveju, jei laikas buvo koreguojamas sustabdžius rungtynių laikrodį. Tokios laiko korekcijos būna daromos tais atvejais, kai pavėluojama ar paskubama sustabdyti laikrodį.

Eil.Nr.	Realaus laiko intervalas	Laikrodžio laiko intervalas

1	0:00:00 – 0:00:15	10:00 – 9:45
2	0:00:15 – 0:00:2**3**	9:4**7** – 9:4**7**
3	-	-
4	0:00:**23** – 0:00:35	9:47 – 9:42
5	0:00:35 – 0:00:40	9:39 – 9:34
6	0:00:40 – 0:10:06	9:34 – 0:00

4 lentelė. Pakoreguota laikrodžio seka.

4 lentelėje matome pakoreguotą seką. Pakoregavus išnyko 3 eilutė, o 2 ir 4 eilutėse padaryti keitimai paryškinti.

Šiame algoritme nepaminėta viena problema – tai, kaip gaunamas atskaitos taškas, pagal kurią sulyginami realūs registravimo laikai. Tai nagrinėsiu sekančiame skyriuje.

Algoritmo privalumai:

- Galima keliese registruoti su vienu rungtynių laikrodžiu, net jei tuo metu visiškai nėra tinklo ryšio. Tinklo trūkiai neturi įtakos.
- Nėra papildomų vėlinimo paklaidų, išskyrus tas, kurios susidaro dėl rungtynių ataskaitos taško sulyginimo.
- Atsižvelgiama į nukrypimus, kurie susidaro dėl laiko korekcijos sustabdžius rungtynių laikrodį.

3.4.8 Laikrodžio valdymo seka su ataskaitos tašku

3.4.7 skyriuje minėtam algoritmui trūksta dalies, aprašančios tai, kaip sulyginamas gavusio seką registruotojo kompiuterio laikas su užregistravusio laikrodžio seką kompiuterio laiku. Bendrai imant, turime du variantus.

Pirmas, tai yra visiems registruotojams užfiksuoti kažkokį sutartinį laiko tašką, pvz., tai galėtų būti susitarimas tiksliai užregistruoti rungtynių pradžios signalą. Tai yra tolygu laikrodžių sinchronizavimui „pagal susitarimą“, ir akivaizdžiai netikslu. Atkuriant laikrodžio seką, pagal sutartinį tašką būtų „paslenkami“ realūs laikai, kad registravusio laikrodžio valdymą kompiuterio laikas sutaptų su gavusio laikrodžio valdymą seką kompiuterio laiku.

Antras, tai yra kompiuterių laikrodžių susinchronizavimas naudojantis tobulais protokolais, tokiais kaip NTP. Į šį variantą įeina ir atvejis, kai turime kompiuterius, neprijungtus prie pasaulinio interneto tinklo. Tokiu atveju kurio nors registruotojo kompiuteris turi būti tas, iš kurio kiti tobulais protokolais susilygintų laiką. Reikia pastebėti, kad bet kuriuo atveju, galima naudotis tiek išorinio, tiek vidinio sinchronizavimo protokolais.

Taip yra todėl, kad mums nėra būtina, kad visi kompiuteriai rodytų tikslų laiką, svarbiausia, kad jų rodomas laikas sutaptų.

Renkantis protokolą, yra ne vienas variantas, bet manau, kad patogiausia naudotis tuo, kas patikrinta laiko ir lengvai prieinama. Toks protokolas yra NTP. Šiuo metu visos Microsoft Windows XP operacinės sistemos tiekiamos kartu su įdiegtu NTP protokolu, ir automatiškai sinchronizuoja laiką iš interneto. Jeigu registracija vyksta operacinėje sistemoje, kuri pati nemoka sinchronizuoti laiko, galima pasinaudoti kitų gamintojų produktais tam atlikti. Paskutinis variantas, tai pačiam įgyvendinti kurį nors laiko sinchronizavimo protokolą, tačiau, kaip matome iš ankstesniuose skyriuose buvusios analizės, geri protokolai yra ypatingai sudėtingi ir nelabai įmanomi esamomis galimybėmis.

4 Duomenų sinchronizavimas

4.1 Alternatyvų apžvalga

Kai kurios duomenų bazės turi galimybių, leidžiančių realizuoti duomenų struktūras, galinčias dirbti autonominiu (*off-line*) režimu. Pirma technologija – duomenų sandėlis.

- Duomenų sandėlis (*data warehouse*) – duomenų centrų rinkinys, atspindintis laikinius duomenis iš skirtingų kompanijos operacijų. Duomenų centras – atskira duomenų sandėlio dalis, teikianti informaciją apie atskirą kompanijos dalį ar padalinį (kartais tai būna atskiri duomenų sandėliai). Duomenų sandėlio informacija saugoma struktūroje, optimizuotoje duomenų paėmimui ir analizei. Tai skiriasi nuo reliacinių duomenų bazių, kurių pagrindinis tikslas yra apdoroti daug užklausų. Į duomenų sandėlį informacija surenkama iš įvairių rūšių šaltinių, išsimėčiusių kompanijos viduje.
- PivotTable® servisas – komponentas, prijungiantis klientus prie Microsoft® SQL Server™ 2000 analizių serverio. Leidžia klientams sukurti autonominius OLAP kubus naudojant šį komponentą kaip OLAP serverį. Šie kubai palaikomi ir valdomi pačių vartotojų, kurie patys turi atnaujinti jų duomenis.
- MOLAP (*Multidimensional OLAP*) kubai – apskaičiuoti duomenys ir faktinių duomenų kopija saugoma analizių serveryje [7].

Duomenų kopijavimas (*replication*) – technologija, apimanti keletą duomenų bazės kopijavimo aspektų. Pagrindiniai jų:

- Momentinis (*snapshot*) vaizdas – tiesioginė duomenų kopija, gaunamas toks DB vaizdas, koks buvo kopijavimo momentu. Pasikeitimai duomenyse nėra tikrinami.
- Sujungimo (*merge*) – skirtingose DB šis kopijavimas vadinamas skirtingai, su skirtingomis galimybėmis. Šis kopijavimo būdas leidžia keisti duomenis tiek ryšio, tiek autonominiame režime, ir po to apjungti pakeitimus esant galimybei (ryšio režime).

Kai kurios duomenų bazės leidžia DB turinį atsisiųsti XML formatu. Turint tokį failą, galima naudotis jame esančia informacija ir neturint interneto ryšio. Praktiškai, tai yra beveik tas pats duomenų bazės kopijavimas (*replication*). Pagrindinis skirtumas yra tik tame, kokiu formatu gauname ir apdorojame informaciją.

Smulkiau apžvelgsiu duomenų kopijavimo galimybes, randamas Microsoft SQL Server 2000. Duomenų kopijavimas (*replication*) gali būti daromas 3 būdais. Pirmas, tai yra

momentinis (*snapshot*) vaizdas. Visi duomenys nukopijuojami į kitą kompiuterį. Duomenys iš pirminės duomenų bazės perrašomi ant viršaus į antrinę duomenų bazę. Į duomenų pokyčius nereaguojama. Toks būdas labiausiai tinka duomenims, kurie retai keičiasi arba kur nėra būtina, kad duomenys visada būtų patys naujausi. Duomenys kopijuojami kas kažkiek laiko.

Antras būdas tai yra transakcinis (t*ransactional*). Eilučių įterpimo, atnaujinimo ir ištrynimo operacijos yra perduodamos į kitą kompiuterį. Transakciniame kopijavime pirma perduodamas momentinis vaizdas į antrines duomenų bazes, ir kai pirminėje duomenų bazėje kas nors pasikeičia, pokyčiai perduodami į antrines duomenų bazes. Toks kopijavimo būdas patogus, kai:

- Duomenų pokyčiai turi iškart matytis antrinėse duomenų bazėse.
- Duomenų pokyčiai turi palaikyti ACID reikalavimus.
- Antrinės duomenų bazės palaiko patikimą arba bent pakankamai dažną ryšį su pirmine DB.

Sujungimo (*merge*) kopijavimas – duomenų pokyčiai perduodami nebūtinai iškart. Duomenis galima keisti tiek pirminėje, tiek antrinėse duomenų bazėse ir ryšio, ir autonominiame režime; ryšio režime duomenys sujungiami tarp duomenų bazių į vientisą rezultatą. Pradžioje (kaip ir anksčiau minėtais atvejais) iš pirminės duomenų bazės perduodama duomenų kopija į antrines DB. Galima pasirinkti, kada sinchronizuoti duomenis – nuolatos, nustatytu laiku, ar pagal konkretų nurodymą. Kadangi tie patys duomenys gali būti redaguojami keliose DB, galimi konfliktai. Jiems spręsti yra pasirinkimų, juos galima nustatyti reguliuojant sujungimo nustatymus. Toks kopijavimo būdas tinka kai:

- Pokyčiai vienose duomenų bazėse turi atsispindėti kitose DB.
- Vartotojams reikia galimybės dirbti autonominiu režimu ir gauti informacijos pokyčius iš kitų vartotojų.
- Tikimasi nedaug konfliktų. Įvykus konfliktams, galimi ACID reikalavimų pažeidimai [8].

Kaip matome, Microsoft SQL Server 2000 turi vertingų ir panaudojamų galimybių. Tokių galimybių turi ir kitos didelės duomenų bazės. Tačiau čia susiduriame su faktoriumi, kuris dažnai neleidžia panaudoti geriausių dalykų – kaina. Microsoft SQL 2000 kaina priklauso nuo versijos ir licencijos tipo. Standartinės versijos licencija 1 kompiuteriui kainuoja apie 15000 Lt. Licencija serveriui su 5 klientais – apie 4000 Lt, papildomi 5 klientai serveriui – apie 3000 Lt. Platesnės versijos kainuoja dešimtimis tūkstančių litų. Kitos galingos duomenų bazės (Oracle, DB2) kainuoja panašiai. Palyginimui – duomenų bazė MySql yra nemokama nekomerciniam naudojimui, o komerciniam naudojimui licencija kainuoja apie

1500 Lt vienam kompiuteriui. Tačiau, tokios paprastos duomenų bazės kaip MySql neturi daugelio papildomų galimybių (įskaitant ir anksčiau minėtą duomenų kopijavimą).

4.2 Alternatyvų panaudojimas

Apžvelgtos alternatyvos turi ir privalumų, ir trūkumų, lyginant su programiniais (ne „gimtais“ duomenų bazių) įrankiais.

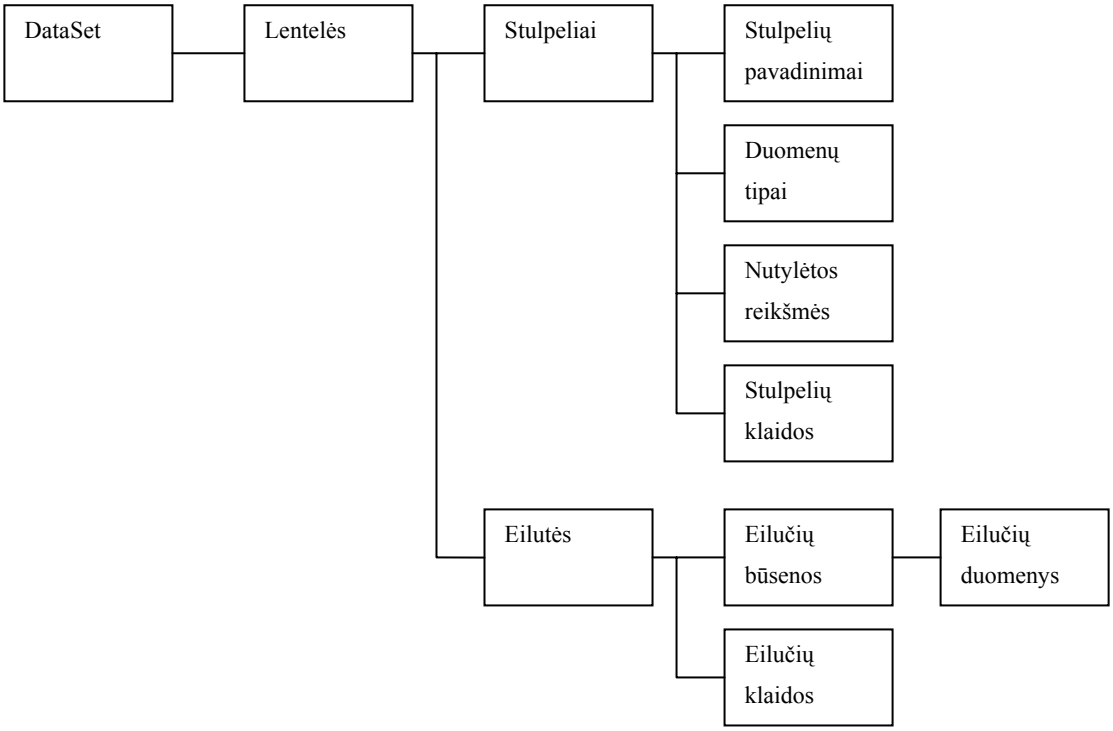
- Programiniai įrankiai gali veikti lėčiau, nes tikėtina, kad jie bus mažiau optimizuoti nei duomenų bazių įrankiai.
- Programiniams įrankiams panaudoti reikia papildomų programos kodo sąnaudų, duomenų bazių įrankiams – papildomų instaliavimo ir konfigūravimo sąnaudų.
- Naudojantis duomenų bazių įrankiais, sistema gali tapti priklausoma nuo konkrečios duomenų bazės. Norint pakeisti duomenų bazę, reiktų perdirbti sistemą. Yra galimybių duomenų perkėlimams iš vienų DB į kitas, su kai kuriomis DB įmanoma padaryti duomenų kopijavimą (*replication*) tarp skirtingų gamintojų DB. Tačiau tai sudaro papildomas sąnaudas eksploatavimo metu.

4.3 .NET ir Java ryšio su duomenų baze architektūra

Kaip jau buvo anksčiau minėta, programines priemones nepertraukiamam darbui su duomenų baze palaikyti turi tik .NET platforma. Pati .NET platforma yra gana naujas dalykas, pakol kas niekas daugiau tokio funkcionalumo šiuo atžvilgiu nėra pateikęs. Asmeniškai tikiuosi tokių galimybių sulaukti iš Java programavimo kalbos, nes, kaip toliau apžvelgsiu, duomenų bazės valdymo architektūra panaši į esančią .NET, reikėtų ją tik išplėsti.

.NET platformoje darbui su duomenų bazėmis skirta dalis vadinama ADO.NET. Java turi panašios architektūros atitikmenį, vadinamą JDBC.

ADO.NET pagrindinė klasė, suteikianti galimybę ryšio ir autonominiu režimu dirbti su duomenų baze, yra DataSet. Tai yra gana tiksli duomenų bazės kopija, kurioje galima patalpinti DB lenteles, lentelių ryšius, raktus ir kitą informaciją. Bendrą struktūrą matome 3 paveiksle.



Pav. 12. DataSet architektūros diagrama

Kaip matome iš diagramos, turima informacija atitinka esančią DB.

Darbo pradžioje paprastai DataSet užpildomas duomenimis iš DB. Yra galimybės nusiskaityti norimas informacijos dalis – nuo atskirų eilučių iki visos duomenų bazės. Duomenų užpildymui yra specialios tam skirtos procedūros. Toks darbo metodas leidžia pasiekti gana unikalų dalyką – darbą su duomenų baze nenaudojant SQL (duomenų bazių valdymo kalbos). Tai reiškia, kad programinis kodas gali būti visiškai nepriklausomas nuo duomenų bazės. Užbėgdamas už akių pasakysiu, kad su JDBC išsiversti be SQL negalima.

JDBC irgi turi klasę pavadinimu ResultSet, kuri yra užklausos duomenų bazei rezultatas. Atskiru atveju tai gali būti ir DB lentelė.

	Stulpelis	Stulpelis	……	Stulpelis
Eilutė	DuomA	DuomB	……	DuomE
Eilutė	DuomC	DuomD	……	DuomF
……	……	……	……	DuomG
Eilutė	DuomK	DuomJ	DuomI	DuomH

Pav. 13. ResultSet struktūra

SQL užklausas galima suskirstyti į 2 tipus – kurios grąžina lentelės pavidalo rezultatą (pvz. select funkcija) ir kurios negrąžina (pvz. delete funkcija). Jei grąžintas lentelės pavidalo rezultatas, tai gali nebūtinai būti visa lentelė (kaip parodyta 4 paveiksle). Kartu su tokia

lentelės struktūra grąžinama ir papildoma informacija apie lentelės struktūrą, bet ji saugoma atskirai (vartotojui to praktiškai nereikia, ta informacija panaudojama automatiškai per ResultSet). Nesvarbu, ar buvo grąžinta visa lentelė, ar tik jos dalis, atgalinis ryšys išlieka. Galime ResultSet keisti eilučių informaciją, trinti ar įterpti eilutes – viskas atsispindės duomenų bazėje. Tai yra didelis pranašumas, leidžiantis iš didelių lentelių atsisiųsti tik reikiamą informacijos gabaliuką ir jį apdoroti. Reikia pažymėti, kad ne visos JDBC tvarkyklės palaiko galimybę modifikuoti duomenų bazę per ResultSet. Pagal JDBC dokumentacijos nurodymus, šitoks atgalinis ryšys nėra būtinas, bet yra sukurta sąsaja, kad tvarkyklių gamintojai galėtų tai realizuoti.

Lyginant .NET ir Java duomenų bazių galimybes, matome, kad .NET yra daugiau pažengusi į priekį. Java turi pagrindą tam, kad sukurti įrankį, kurio pagalba būtų galima rašyti programas, palaikančias nepertraukiamą ryšį su duomenų baze. Mano manymu, užtektų išplėsti ResultSet – padaryti, kad duomenys būtų ne tiesiai rašomi į DB, o saugomi tarpiniuose modeliuose.

4.4 Sinchronizacijos problemos

Dirbant keliems vartotojams autonominiu režimu su duomenų baze, labai tikėtina, kad atsiras sinchronizacijos problemų. Įsivaizduokime, kad rungtynes registruoja du registruotojai – A ir B. A nutrūko ryšys, jis perėjo į autonominį darbo režimą. Praėjus 10 minučių A pataisė įvykio X atributus. Dar po 10 minučių B pataisė to paties įvykio atributus. A atsirado ryšys, iš jo informacija atnaujinama į duomenų bazę. Gaunasi, kad nors B atributus taisė vėliau, bet duomenų bazėje lieka A pataisyti atributai.

Kaip matome, tokiu atveju, du žmonės keičia tą pačią informaciją, duomenyse atsiranda klaidos.

Duomenų bazėse yra galimybė išspręsti tokią problemą. Vartotojas, prieš keisdamas informaciją, „užrakina“ duomenų eilutę, lentelę ar kitą keičiamą informacijos vienetą, tam, kad kol jis su juo nebaigė dirbti, niekas kitas negalėtų jo pakeisti. Deja, kai susiduriame su autonominiu darbo režimu, taip paprastai problemos išspręsti nebegalime, kadangi tikroji duomenų bazė autonominiame režime yra nepasiekiama ir jos negalima „užrakinti“. Lieka du pasirinkimai:

- Jei leistini netikslumai, galime ignoruoti sinchronizacijos klaidas.
- Jei negalime leisti tokių netikslumų, reikia papildomų operacijų, tam, kad patikrinti, ar neįvyko sinchronizacijos klaidos.

ADO.NET automatiškai tikrina, ar nebuvo sinchronizacijos klaidų, ir, jei jų buvo, praneša apie klaidą. Praktiškai, ADO.NET nepatikrina, ar sinchronizacija nebuvo teisinga tuo

atveju, kai vartotojai laiko požiūriu teisinga tvarka surašė duomenis. Norint tai patikrinti, reikia rašyti papildomą programos kodą. Geras tokio kodo pavyzdys pateiktas „Collision Form“ [10]. Kaip matome šito straipsnio pavyzdyje, išsaugoma operacijos sukūrimo, modifikavimo ir tikroji data, taip pat keli papildomi atributai. ADO.NET išsaugo 2 duomenų kopijas, ir pagal jas tikrina, ar duomenys korektiški ir nebuvo sinchronizacijos klaidų.

4.5 Sinchronizacijos sprendimas

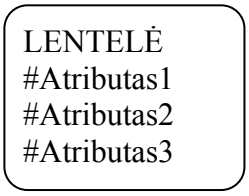
Savo siūlomą modelį remiu modeliu, sukurtu mūsų grupiniame projekte „Krepšinio varžybų registravimo ir analizės paskirstyta sistema.“ Pagal šio darbo 4.2 punktą, sinchronizacijos klaidos ignoruojamos. Taip galima daryti dėl projekto specifikos. Informacijos srautas, keliaujantis tarp vartotojų, yra toks, kad vienam vartotojui perrašius kito vartotojo duomenis, negali atsirasti nekorektiškų duomenų. Duomenys visada yra vienareikšmiai – jei ir galimas atvejis, kad dviejų vartotojų norima įvesti informacija skirsis, tai tik dėl to, kad kažkuris iš vartotojų padarė klaidą. Dalis informacijos nustatoma susijungimo su duomenų baze metu (komandos, žaidėjų informacija, bendra informacija apie rungtynes), kita informacija leidžiama keisti registruojant rungtynes (protokolas, informacija apie kėlinius ir veiksmus). Komandos, žaidėjų ir bendra informacija nustatoma iš karto, nes rungtynių eigoje ji nekinta, ir nekyla poreikis ją keisti. Informacija apie kėlinius ir veiksmus atitinka išdėstytą 4.2 punkto 1 pavyzdyje, tai yra, keitimai būna labai retai, ir sinchronizacijos problemos praktiškai neįmanomos. Su protokolu situacija kiek kitokia. Galima daryti dvi prielaidas – pirma, kad vartotojai taisys klaidas tik savo užregistruotuose veiksmuose, antra, kad vartotojai, taisydami savo ar svetimas klaidas, nebepadarys dar kartą klaidų. Jei nors viena prielaida išpildoma (praktiškai jos išpildomos abi), sinchronizacijos problemų neturi likti. Išskirtinis atvejis būtų, jei registruojant būtų paskirtas atskiras žmogus stebėti ir taisyti klaidoms, tada aišku, kad pirma prielaida neišpildoma.

Kyla dar viena problema – kaip sukurti unikalius raktus kiekvieno vartotojo registruojamai informacijai. Kad autonominiame režime būtų galima redaguoti informaciją, ir prisijungus prie duomenų bazės vienareikšmiškai būtų atrenkami tie patys įrašai, net jei jie ir pakitę, įrašai turi turėti unikalų raktą. Tai reikalinga, kad nesusidarytų situacija, jog tuo pačiu metu du vartotojai sugeneruoja tokį patį raktą – tada savaime aišku, kad vieno vartotojo įrašoma informacija dingtų. Su paties vartotojo sugeneruotais raktais galima visada patikrinti, ar tokio rakto dar nėra, tiek vartotojui prisijungus prie DB, tiek neprisijungus. Tam, kad raktai nesutaptų, reikia įtraukti kažkokią unikalią informaciją apie vartotoją, paprasčiausias variantas yra į raktą įtraukti vartotojo vardą. Tada atsiranda papildomas reikalavimas, kad tuo pačiu

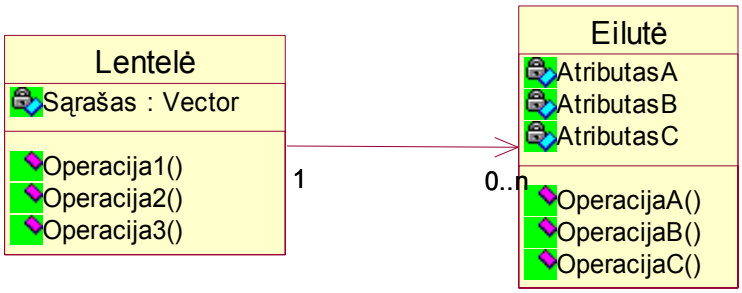
vartotojo vardu vienų rungtynių neregistruotų du žmonės, bet šis reikalavimas kartu ir logiškas, ir nesunkiai įvykdomas.

4.6 Duomenų bazės transformacija į klases

Siūlomame modelyje duomenų bazė atvaizduojama klasėmis. Duomenų bazių lentelei atvaizduoti naudojamos dvi klasės – viena yra lentelės eilučių sąrašas, kita yra konkrečios eilutės. Eilučių informacija inkapsuliuojama, toks priėjimas suteikia lankstumo.



Pav. 14. Transformuojama duomenų bazės lentelė



Pav. 15. DB lentelės atvaizdas klasėmis

Kaip parodė praktika, toks transformavimas pranašesnis už duomenų saugojimą masyvuose ar medžio struktūrose. Darbas „Krepšinio rungtynių registravimo ir analizės paskirstyta sistema“ pirma versija buvo bakalauro baigiamasis darbas, joje informacija buvo saugoma medžio struktūra. Augant programos dydžiui, vis sunkiau darėsi aptarnauti duomenų struktūrą (apdorojančios klasės buvo per daug atskirtos nuo duomenų struktūros), iki atėjo laikas, kai teko struktūrą pakeisti. Šitokia transformacija padaryta pagal objektinio projektavimo principus, ir ji leido išskaidyti atskiras duomenų struktūros dalis kartu su susietomis funkcijomis.

Ryšiams tarp lentelių realizuoti specialaus modelio nėra sukurta. Ryšiai „rankiniu“ būdų realizuojami programos viduje. Dėl to gali būti sunku atvaizduoti sudėtingus duomenų bazės ryšius.

4.7 Bendras modelis

Remiantis 5.1 punkte išdėstytomis prielaidomis, nustačiau, kad informacijos vienetui (DB eilutei) reikia trijų loginių atributų:

- Ar nustatytas kodas yra unikalus realios duomenų bazės atžvilgiu (atributas „nustatytas“). Jei nenustatytas, vadinasi unikalus tik pas vartotoją esančios informacijos atžvilgiu.
- Ar informacija įrašyta į realią duomenų bazę (atributas „įrašyta“).
- Ar informacija ištrinta iš realios duomenų bazės (atributas „ištrinta“).

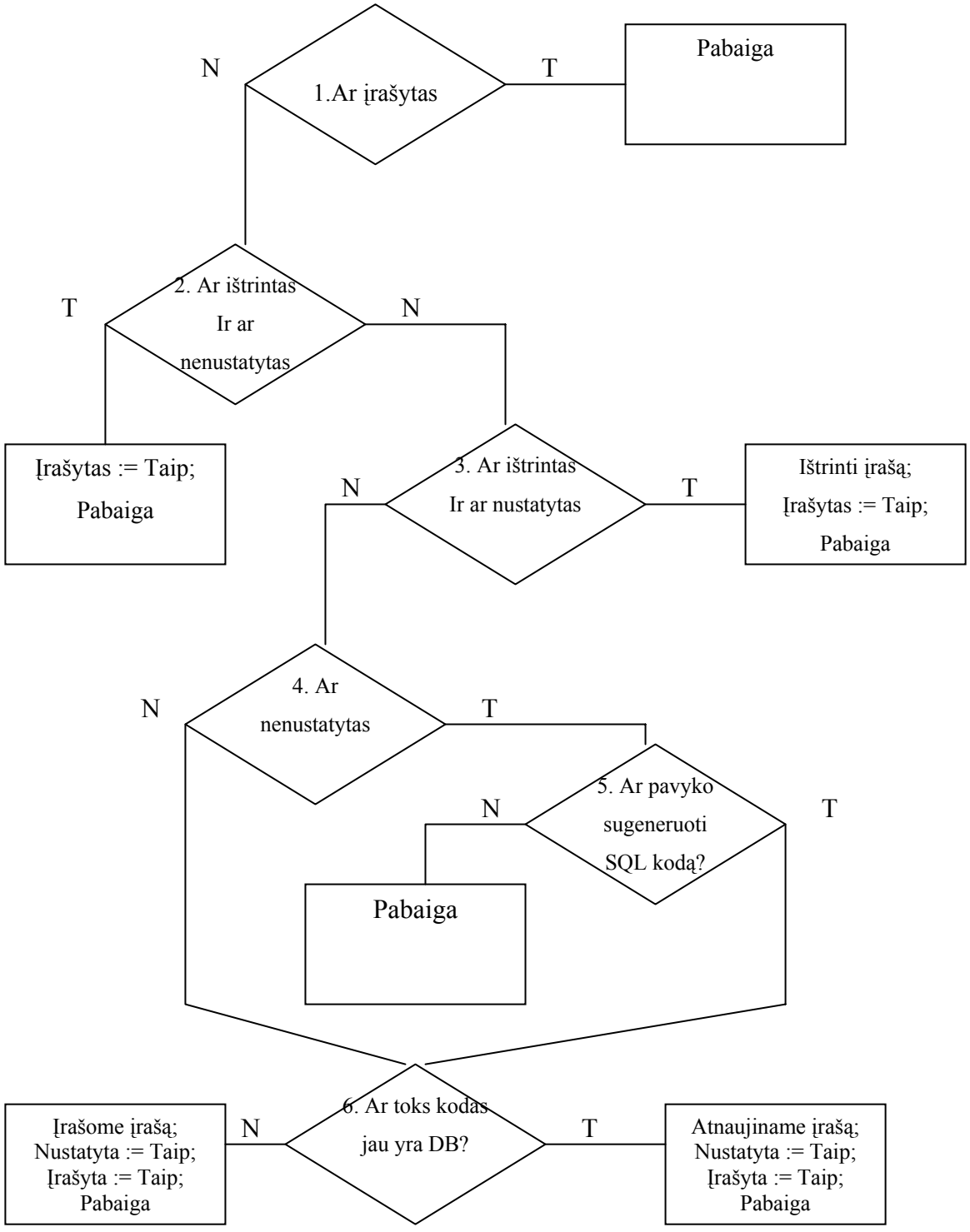
Remiantis šiais trimis atributais, sukūriau algoritmą, kaip elgiamasi eilinio atnaujinimo metu, kai sulyginama informacija su duomenų baze.

Kaip matom 5 paveiksle, 1 punktas, patikrinama, ar informacija įrašyta. Jei šio atributo reikšmė „taip“, vadinasi, įrašo liesti nebereikia, jis tvarkingai sulygintas su duomenų baze. 2 punktu tikriname, ar įrašas nebuvo ištrintas ir nenustatytas, t.y., ištrintas taip ir nepasiekęs realios duomenų bazės. Tokiu atveju pažymime, kad įrašas įrašytas, sekančio tikrinimo metu jis bus pašalintas. 3 punktas, jei įrašas ištrintas ir nustatytas, vadinasi, įrašas laukia pašalinimo iš duomenų bazės. Šaliname, jei pavyksta, pažymime, kad įrašas įrašytas. 4 punktas, jei nenustatytas (ir neištrintas), bandoma sugeneruoti unikalų SQL raktą. Jei pavyksta (nenulūžta tuo momentu ryšys su DB ir panašiai), 6 punkte vykdomas informacijos įrašymas. Informacijos įrašymas vykdomas ir tuo atveju, kai 4 punkte paaiškėja, kad SQL raktas jau nustatytas (iš anksčiau, galbūt lūžo ryšys ir nepavyko įrašyti informacijos).

Šis algoritmas vykdomas tam tikrais laiko tarpais. Kokio ilgio laiko tarpai, pasirinkti reikia priklausomai kaip greitai atsinaujinančią informaciją norima matyti. Darbe „Krepšinio rungtynių registravimo ir analizės paskirstyta sistema“ tai daroma kas 1 sekundę.

Prieš sprendžiant, ar verta naudotis mano sukurtu modeliu, reikėtų įvertinti:

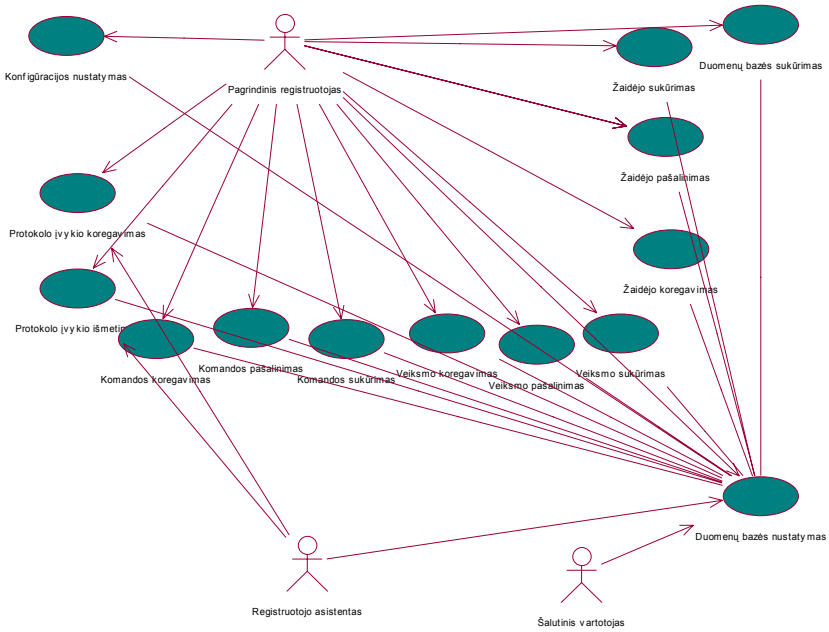
- Ar apsimoka pasinaudoti duomenų bazių galimybėmis.
- Ar duomenų struktūra nėra labai sudėtinga.
- Ar galima pereiti prie .NET platformos ir pasinaudoti jos teikiamomis galimybėmis.



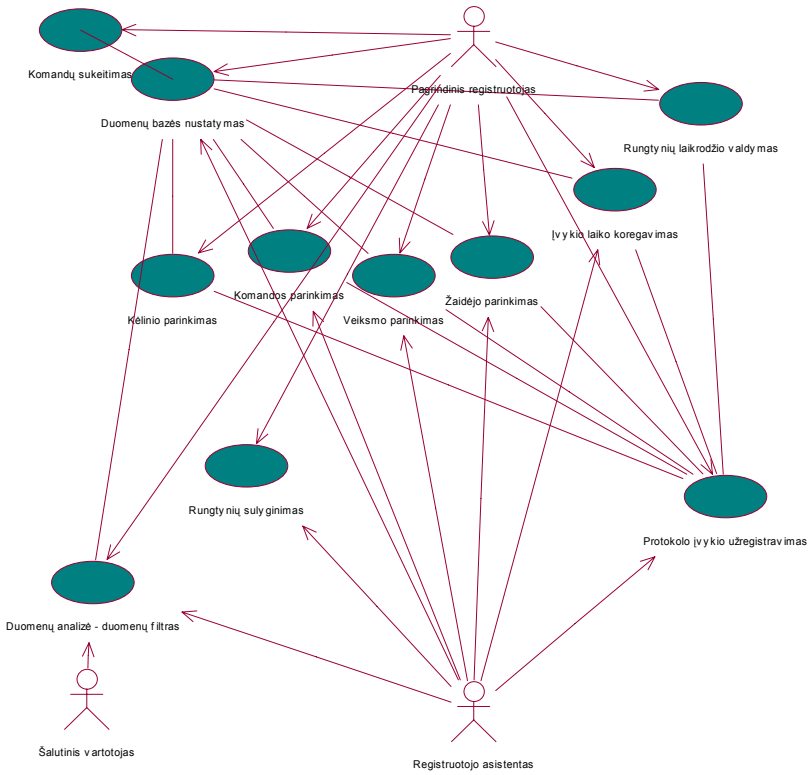
Pav. 16. Duomenų sinchronizacijos algoritmas

4.8 Realizacija

Programos „Krepšinio rungtynių registravimo ir analizės paskirstyta sistema“ atliekamos funkcijos pateiktos pav. 17 ir pav. 18 panaudojimo atvejų diagramose.

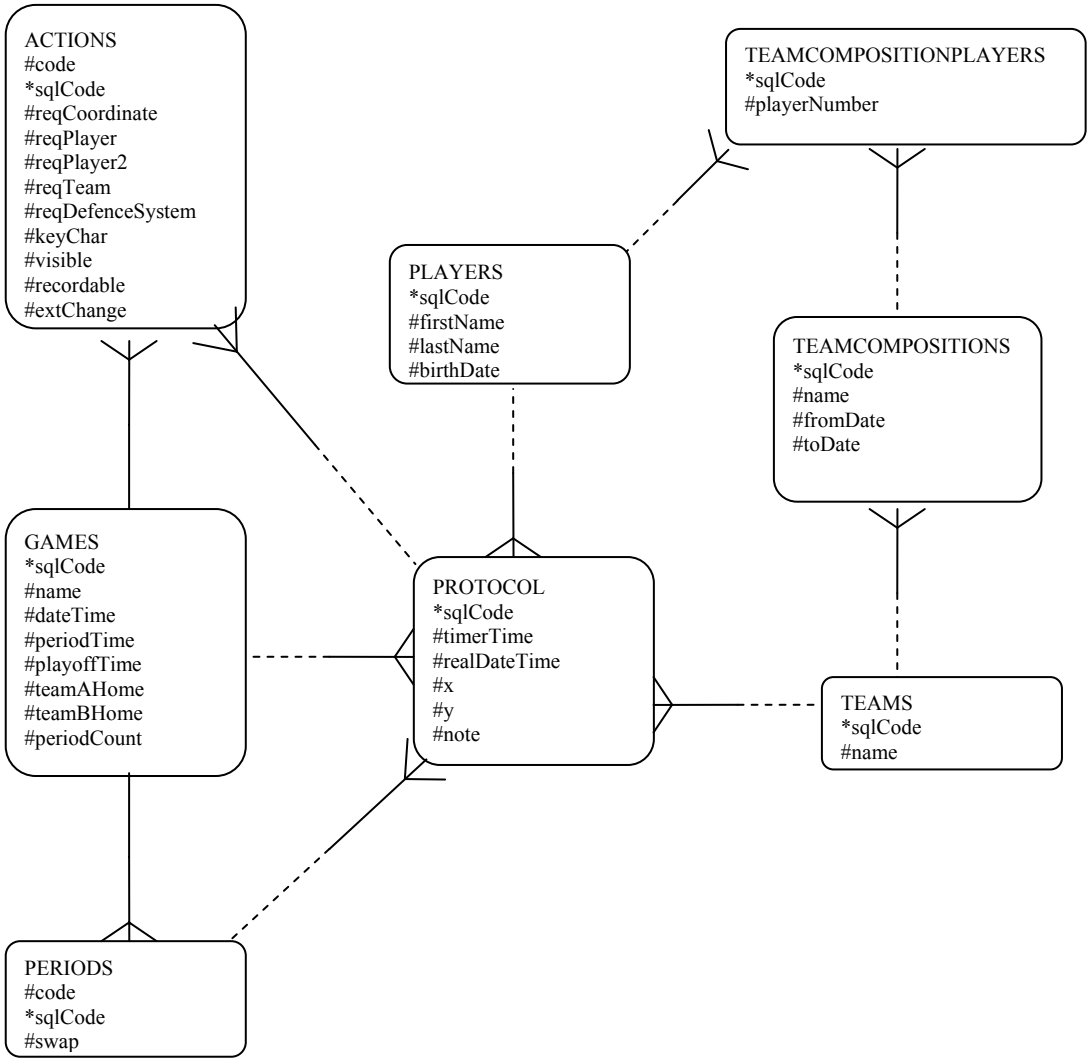


Pav. 17. Panaudojimo atvejų diagrama (1 dalis)



Pav. 18. Panaudojimo atvejų diagrama (2 dalis)

Panaudojimo atvejų diagramose matome, kokius darbus atlieka vartotojai dirbdami su sistema. Atliekami darbai yra susiję su naudojamomis klasėmis (iš panaudojimo atvejų diagramos modeliuojama klasių diagrama).



Pav. 19. Duomenų bazės schema

Pagal 5.2 punkte paaiškintą principą, DB lentelės transformuojamos į klases. Kiekvienai lentelei sukuriamos 2 klasės – viena sąrašo klasė (lentelės atitikmuo) ir viena eilutės klasė (lentelės eilutės atitikmuo).

5 Išvados

1. Siekiant padidinti per rungtynes užfiksuojamų įvykių kiekį ir jų tikslumą, rungtynes turi registruoti keli registruotojai su nepriklausoma įranga.

2. Apžvelgti laiko sinchronizavimo protokolai, pateiktas populiariausio iš jų (NTP) veikimas, parodoma, su kokiomis sudėtingomis problemomis susiduriama tiksliai sinchronizuojant laiką ir kaip jos sprendžiamos.

3. Pateikti galimi rungtynių įvykių sinchronizavimo laike algoritmai, du iš jų buvo praktiškai realizuoti. Algoritmai įvertinti pagal:

- Registravimo laiko paklaidas (atsirandančias dėl algoritmų trūkumų ir vėlavimo laikų).
- Tinkamumą naudojimui, priklausomai nuo ryšio greičio ir išdėstymo.
- Atsparumą ryšio trūkiams.
- Realizacijos sudėtingumą.

4. Sukonstruotas algoritmas, visiškai atsparus bet kokiems ryšio trukdžiams (net ir jo nebuvimui), paremtas kompiuterių laikrodžių sinchronizavimu ir rungtynių laikrodžio valdymo sekos išsaugojimu ir laikrodžio rodomo laiko atkūrimu.

5. Antrojoje darbo dalyje pasiūlytas modelis ir algoritmas, kaip realizuoti programą, kad ji veiktų nutrūkus ryšiui su duomenų baze. Modelio realizacija paremta tarpinių duomenų struktūrų sukūrimu ir jų sinchronizavimu su duomenų baze, esant ryšiui. Pasiūlytas modelis gali būti realizuotas nepriklausomai nuo programavimo kalbos.

6. Apžvelgti didelių duomenų bazių nuosavi įrankiai užtikrinti darbui nutrūkus ryšiui. Taip pat nustatyta, kad .NET platformoje yra universalesnis programinis įrankis, tačiau, norint juo pasinaudoti, būtinai reikia dirbti minėtoje platformoje, kuri pilnai veikia tik Windows aplinkoje.

7. Nustatytos galimos tobulinimo kryptys:

- Optimizuoti duomenų atnaujinimo algoritmą, kad su mažesnėm laiko sąnaudom būtų patikrinami duomenys; tikrinti tik tas SQL duomenų dalis, kurios matomos vartotojo.
- Išplėsti algoritmą, kad būtų stebima, ar nebuvo sinchronizacijos klaidų.

6 Literatūra

1. Boaz Patt. A theory of clock synchronization. – Massachusetts Institute of Technology, 1994
2. Network Time Synchronization Project. Iš http://www.eecis.udel.edu/~mills/ [interaktyvus]. Prieiga per internetą http://www.eecis.udel.edu/~mills/ntp.html.
3. Ulrich Schmid. Synchronized Universal Time Coordinated for Distributed Real-Time Systems. Control Engineering Practice 3(6), 1995, pp. 877-884. Prieiga per internetą http://www.auto.tuwien.ac.at/Projects/SynUTC/documents/cep3-6.ps.gz.
4. Ulrich Schmid, Klaus Schossmaier. Interval-based Clock Synchronization. Journal of Real-Time Systems 12(2), March 1997, pp. 173-228. Prieiga per internetą http://www.auto.tuwien.ac.at/Projects/SynUTC/documents/cep3-6.ps.gz.
5. Klaus Schossmaier, Ulrich Schmid, Martin Horauer, Dietmar Loy. Specification and Implementation of the Universal Time Coordinated Synchronization Unit (UTCSU). Journal of Real-Time Systems 12(3), May 1997, pp. 295-327. Prieiga per internetą http://www.auto.tuwien.ac.at/Projects/SynUTC/documents/jrts12-3.ps.gz.
6. Martin Horauer, Ulrich Schmid, Klaus Schossmaier. NTI: A Network Time Interface M-Module for High-Accuracy Clock Synchronization. Proceedings of the 6th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS‘98), Orlando, Florida, USA, March 30 – April 3, 1998, pp. 1067-1076. Prieiga per internetą http://www.auto.tuwien.ac.at/Projects/SynUTC/documents/wpdrts98.ps.gz.
7. Ulrich Schmid. High-Accuracy Time Services and Fault-Tolerant Clock Synchronization. 37th Meeting of the IFIP Working Group 10.4, Workshop on Time and Dependability, Martinique, January 21-25, 2000. Prieiga per internetą http://www.auto.tuwien.ac.at/Projects/SynUTC/documents/slides_ifip_wg10_4_00.ps.gz.
8. Wijegunaratne I. And Fernandez G. Distributed applications engineering. – Springer, 1998
9. Gabrick K. A., Weiss D. B. Java 2EE and XML Development. – Manning, 2000
10. Sun Microsystems Designing Enterprise Applications with the J2EE Platform, Second Edition. Iš Java.sun.com [interaktyvus]. Prieiga per internetą: http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/index.html
11. Mathew Reynold’s .NET 247™. Iš http://www.dotnet247.com [interaktyvus]. Prieiga per internetą: http://www.dotnet247.com/247reference/articles/1/9942.aspx

12. Using ADO.NET – First principles. Iš http://visualbasic.about.com [interaktyvus]. Prieiga per internetą: http://visualbasic.about.com/library/weekly/aa041203c.htm
13. Welcome to Abstract ADO.NET. Iš http://abstractadonet.sourceforge.net [interaktyvus]. Prieiga per internetą: http://abstractadonet.sourceforge.net
14. A data warehouse tutorial [interaktyvus]. Prieiga per internetą: http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=378&lngWId=5
15. SQL Server 2000 replication overview [interaktyvus]. Prieiga per internetą http://www.microsoft.com/sql/evaluation/features/replication.asp
16. Oracle Replication FAQ [interaktyvus]. Prieiga per internetą http://www.orafaq.com/faqrepl.htm
17. Collision Form [interaktyvus]. Prieiga per internetą http://www.windowsforms.net/TaskVision/SourceViewer.aspx?tabIndex=7&tabId=44&path=CollisionForm.src&file=CollisionForm&rows=1

7 Terminų žodynas

„Kregis“	Krepšinio rungtynių registravimo ir analizės paskirstyta sistema – magistrinio darbo projektas
Išorinė sinchronizacija	Laiko sinchronizacija, kai kompiuterių grupė sulygina savo laiką su vieno šakninio kompiuterio laiku.
Vidinė sinchronizacija	Laiko sinchronizacija, kai kompiuterių grupė sulygina savo laiką tarpusavyje, kad jų laikai kuo mažiau skirtųsi.
NTP (*network time protocol*)	Labiausiai internete paplitęs laiko sinchronizavimo protokolas
SNTP (*simplified network time protocol*)	Supaprastinta NTP versija, pateikiama tiek kaip atskiras protokolas, tiek kaip supažindinimas su NTP
DTS (*digital time service*)	Skaitmeninio laiko servisas
KGD	Kintamo dažnio generatorius – laikrodžio ėjimo generatorius, su netolygiu dažniu, dėl ko galimi jo rodomo laiko nuokrypiai
„Taiklumas“ (*accuracy*)	Tai, kiek maksimaliai laikrodžio parodymas gali skirtis nuo realaus pasaulio laiko.
„Precizija“ (*precision*)	Tai, kiek maksimaliai gali skirtis keli iš eilės paimti laikrodžio parodymai.
LAN (*local area network*)	Tinklo standartas – vietinis tinklas; šiuo metu populiariausias pasaulyje
MAN (*metropolitan area network*)	Tinklo standartas
WAN (*wide area network*)	Tinklo standartas
DB (*database*)	Duomenų bazė
Kubas (*cube*) -	Duomenų sandėlio duomenų apdorojimo elementas.
OLAP (*online analytical processing*) -	Duomenų bazių technologija greitiems atsakymams į sudėtingas užklausas.
JDBC (*Java database connectivity*)	Universalus Java programavimo kalbos interfeisas darbui su duomenų bazėmis

.NET	Platforma (konkretus programavimo kalbų ir aplinkų rinkinys)
ADO.NET	.NET įrankių rinkinys darbui su duomenų bazėmis
XML (*extensible markup language*)	Duomenų objektų aprašymo kalba
ACID (*atomicity, consistency, isolation, durability*)	Reikalavimai, užtikrinantys duomenų bazės vientisumą