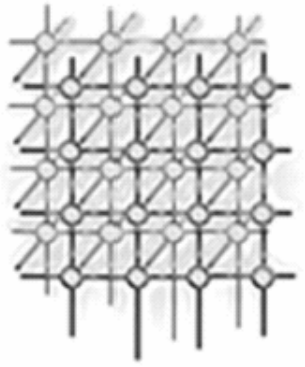
KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMACIJOS SISTEMŲ KATEDRA

TINKLINIŲ DUOMENŲ BAZIŲ SANDARA TINKLO PASLAUGŲ SURADIMUI



MAGISTRO DARBAS

Vadovas: dėst. B. Paradauskas

Magistrantai: A. Čupačenko, G. Stirbys

KAUNAS, 2004

TURINYS

SANTRAUKA

Tinklinės informacinės sistemos yra didelio masto paskirstytos sistemos, kurioms būdinga nevienalytiškumas, centralizuotos kontrolės trūkumas, daugialypiai autonominiais administraciniai domenai, nepatikimi komponentai ir dažni dinaminiai pasikeitimai. Tokiose sistemose pageidaujama išlaikyti ir užklausti dinaminę ir laikui bėgant besikeičiančią informaciją apie tokius aktyvius dalyvius kaip paslaugas, resursus ir vartotojų bendrijas. Tinklo paslaugų vizija teikia vilčių, kad programos taps daug lankstesnės, prisitaikančios ir galingenės užklausdamos internetines duomenų bazes (registrus) vykdymo eigoje, kad surastų informaciją ir tinklo komponentus, įgalindamos aukšto lygio paskirstytų komponenčių surinkimą.

Remdamiesi šia vizija, mes pristatome *Tinklo paslaugos atradimo sandarą* (WSDA); pristatome saugyklą, kuri yra centralizuotas duomenų bazės mazgas, skirtas dinaminio paskirstyto turinio suradimui, kuris palaiko XQuery užklausas iš dinaminio XML duomenų modelio.

SUMMARY

Grids are collaborative distributed Internet systems characterized by large scale, heterogeneity, lack of central control, multiple autonomous administrative domains, unreliable components and frequent dynamic change. In such systems, it is desirable to maintain and query dynamic and timely information about active participants such as services, resources and user communities. The *web services* vision promises that programs are made more flexible, adaptive and powerful by querying Internet databases (registries) at runtime in order to discover information and network attached building blocks, enabling the assembly of distributed higher-level components.

In support of this vision, we introduce the *Web Service Discovery Architecture (WSDA)*, we introduce the hyper registry, which is a centralized database node for discovery of dynamic distributed content. It supports XQueries over a tuple set from a dynamic XML data model.

TERMINŲ ŽODYNAS

XML dokumentas – XML programavimo kalba aprašytas dokumentas

XML schema – XML kalba parašytas dokumentas, skirtas XML dokumentui apibrėžti

Duomenų bazė – kartu saugomų ir susijusių duomenų visuma, kuriai būdinga integruotumas, nepertekliškumas, nepriklausomumas

Duomenų bazių valdymo sistema – programų rinkinys, atliekantis duomenų apdorojimo operacijas.

SUTRUMPINIMŲ SĄRAŠAS

XML – išplėstinė žymių kalba

RDB – reliacinė duomenų bazė

DB – duomenų bazė

DBVS - duomenų bazių valdymo sistema

IS – informacinė sistema

DTD – dokumento tipų aprašymas

OP – objektų savybių modelis

SQL – struktūrizuota užklausų kalba

CAL – komunikacinės binarinės kilpos

SGML – standartinė apibendrinta žymių kalba

HTML – hiperteksto žymių kalba

SIL – standartinė keitimosi kalba

PĮ – programinė įranga

1. ĮŽANGA

Ilgą laiką pagrindinis kompiuterinių sistemų uždavinys buvo automatizuoti teisingai apibrėžtus pasikartojančius veiksmus. Atsiradus paskirstytam skačiavimui, o būtent interneto ir WWW technologijoms, šis uždavinys prasiplėtė. Vis daugiau kompiuterinės sistemos naudojamos kaip įrankiai, įgalinantys efektyviai komunikuoti ir bendradarbiauti per didelį atstumą. Kolegos (ir programos) su bendrais interesais gali lengviau dirbti kartu neatsižvelgdami į savo ir reikalingos įrangos bei mechanizmų fizinę padėtį. Tradicinė skyriaus grupė yra papildoma kryžminėmis virtualiomis grupėmis, kurios veikia atviru skaidriuoju būdu. Tokios grupes yra vadinamos *virtualiomis grupėmis*. Tokia galimybė toliau išlėsti dabartines žinias yra būdinga mokslo bendruomenėms, nes savo stiprybes jos semiasi stimuliuodamos bendradarbiavimą peržiangiant administracines ribas. O būtent tinklinis skaičiavimas, paskirstytos duomenų bazės ir *web servisai* įveda pagrindines sąvokas ir technologijas, kurios paverčia realybe *Globalią Infrastuktūrą*.

Tinklinė technologija bando palaikyti lanksčios, saugios, kordinuotos informacijos padalinimą tarp individų, institucijų ir šaltinių dinamiškų rinkinių.

Tai apima duomenų padalinimą, taip pat prieigą prie kompiuterių, programinės įrangos ir prietaisų, reikalaujančių skaičiavimo ir duomenų padidinimo sprendžiant bendrą problemą [1]. Šie ir kiti duomenų apdorojimo perėjimai yra reikalingi vis labiau sudaryti galimybę sujungti laisvai susietus žmones ir išteklius iš sudėtinių organizacijų. Tinkleliai yra bendros paplitusios interneto sistemos, charakterizuojamos plačiu masteliu, įvairiarūšiskumu, centrinės kontrolės stoka, sudėtiniais savarankiškais valdymo domenais, nepatikimomis sudėtinėmis dalimis ir dažnu dinamišku keitimusi.

Pavyzdžiui, Europos branduolinių tyrimų organizacija, pagrindusi Europos Duomenų Tinklelio konstrukciją (EDG), yra pasaulinės programinės įrangos infrastruktūra, jungianti kartu didelę duomenų apdorojimo šaltinių ir žmonių grupę, šimtą laboratorijų ir universiteto departamentų. Tai apima tūkstantį tinklo paslaugų, dešimtis tūkstančių procesorių, WAN gigabaitų perduodant, taip pat petabaitus disko vietos [3]. Daugybė subjektų jau gali bendradarbiauti tarp vienas kito suteikti Aukštos Energijos Fizikos (HEP) eksperimentinių duomenų analizę: HEP vartotojo bendrija ir institucijų dauguma, tiekėjų sandėliavimas, taip pat tinklas, pritaikomumo ir skaičiavimo ciklo tiekėjai. Vartotojai naudoja nutolusio pritaikomumo tiekėjų rinkinio paslaugas darbų pateikimui, kurios iš eilės yra vykdomos skaičiavimo ciklo paslaugų tiekėjų, naudojant sandėliavimą ir tinklo tiekėjų paslaugas IŠ/Į.

Paslaugų būtinumas vykdyti duotą užduotį dažnai nebūna tame pačiame valdymo domene bendradarbiavimas gali turėti šiek tiek statišką konfiguraciją, arba jie gali būti labiau dinamiški ir nestabilūs, su vartotojais ir paslaugų tiekėjais prisijungiančiais ir dažnai atsijungiančiais, ir dažnas konfiguracijų keitimasis taip pat vartojimo elgsenų.

Programinės įrangos sudedamosios dalies išsivystymas pasiekė būseną, kur reikalaujama nevaržomo veikimo, tipiniams pritaikymams prieinama per trečią grupę bibliotekų, sistemų ir įrankių. Šios sudedamos dalys dažnai yra patikimos, patvirtintos dokumentais ir prižiūrimos, ir surinktos su ketinimu pakartotinai panaudoti ir papildyti. Daugumai programinės įrangos kūrėjų pagrindinsi įgudis yra daugiau ne pagrindinis programavimas, bet galimybė rasti, apskaičiuoti ir sujungti blokus iš plačios įvairovės trečių grupių.

Programinės įrangos pramonė turi nuolat judėti link programinės įrangos vykdymo lankstumo. Pavyzdžiui, dinamiškas susiejimas leidžia lengviau papildyti ir padidinti pritaikymus, nei statinis susiejimas. Modernios programavimo kalbos kaip JAVA naudoja net lankstesnį ryšio modelį, kuris atideda susiejimą iki paskutinio galimo momento (kreipimosi laiko metodas). Dar vis dauguma programavimo įrangos tikisi susieti ir paleisti fone trečios grupės veikimo instaliuoto vietiniame kompiuteryje, vykdant programą. Pavyzdžiui, teksto procesorius yra instaliuotas kartu su vidiniais blokais tokiais kaip rašybos korektorius, vertėjas, sinonimų žodynas ir moduliai įvairiems duomenų formatams importuoti ir eksportuoti. Tinklas nėra pagrindinė programinės įrangos modelio dalis, tuo tarpu vietinis diskas ir operacinė sistema, žinoma, yra.

Gerai apgalvotos interneto technologijos atnešė padidintą naudojimo paprastumą ir abstrakciją per aukšto lygio protokolo dėklus, išbandytą APIs, labiau modulines ir pakartotinai panaudojamas serverio struktūras ir atitinkamai efektyvius įrankius. Kelias yra jau nutiestas sekančiam žingsniui link padidinto programinės įrangos vykdymo lankstumo. Šiame scenarijuje, keletas sudėtinių dalių yra pririštas tinklas ir prieinama tinklo paslaugų forma visuomenės, bendradarbių ir biznio klientų vartojimui. Interneto paslaugų tiekėjai (ISPs) siūlo paleisti ir prižiūrėti patikimas paslaugas per vadovaujančias terpes. Verčiau, kada kreipiantis į vietinę biblioteką, pritaikymas jau pasitelkia funkcijas nutolusiose sudėtinėse dalyse, geriausiu atveju į tą patį rezultatą, Paslaugos pavyzdžiai :

Katalogas implemetuoja sąsają, kuri pagal duotą loginį failo pavadinimą ir grąžina duoto failo kopijų vietas pasaulyje.

Valdymo programos kopija palaiko failo kopijos kurimą, ištrinimą ir valdymą taip pat kaip nuotolinis sustabdymas ir pranešimo keitimas per skelbimo/aprašymo sąsajas.

Atmintinės paslaugos siūlo FTP perdavimo tinklą, aiškus TCP buferis, dydį nustatanti sąsaja, taip pat kaip valdymo sąsajos failų valdymui vietinėse atmintinės sistemose. Pagalbinė sąsaja palaiko užklausimus per prieigos žurnalus ir statistikas, laikomas registre - tai suformuota didelių galimybių serveryje ir padalinta į sudėtines tokias atmintinės skaičiavimo grupių paslaugas.

Genų seka, kalbos vertimas ar greitos naujienos ir pranešimų paslauga.

Nuotolinis kreipimasis yra visada reikalingas keletai užklausimo pritaikymų, tai negali būti (išimtinai) paleista vartotojo kompiuteryje, kadangi jie priklauso nuo išteklių aibės išmėtytos per sudėtinius nuotolinius duomenis. Pavyzdžiai apima genų sekos užklausimą, biznio prognozes, klimato keitimosi imitaciją ir astronominį dangaus matavimą taip pat kaip Aukštos Energijos Fizikos analizės duomenys plintantys Terabaitais. Tokie pritaikymai tik protingai gali būti paleidžiami nuotoliniame super kompiuteryje, arba kelios skaičiuojamos grupės su galingu CPU (centriniu procesoriumi), tinklu, disku ar disketės talpa, taip pat kaip atitinkama programinės įrangos aplinka, atitinkanti minimalius minimalius standartus.

Daugiausia paslaugų persiųsti, bet taip pat daugiausia nelanksčių konfiguracijų pasirodymas yra per sunkus nusiųsti vietą, sąsają, reakciją ir kitas nuotolinių paslaugų savybes į vietinį pritaikymą. Lengvai sujungtos, necentralizuotos sistemos kviečia sprendimams – tai yra lanksčiau ir gali prisitaikyti prie sąlygų keitimo. Pavyzdžiui, jei vartotojas pasirodo mažiau nei patenkintas pastebėta tekstų procesoriaus, nuotolinio kalbos korektoriaus kokybe, jis/ji norės įjungti kitą korektorių. Toks dinamiškas jungimas gali tapti įmanomu, jei paslaugos įrankiai laikosi keletos įprastų sąsajų ir tinklo protokolų ir jei įmanoma suderinti paslaugas prieš sąsają ir tinklo protokolo specifikaciją.

Tada kyla įdomus klausimas: kokia infrastruktūra reikalinga įgalinti programą turėti galimybę ieškoti interneto alternatyvioms bei panašioms paslaugoms ir dinamiškai paleisti jas?

Nors bendravimo protokolai ir pranešimų formatai yra standartizuoti internete, tampa vis labiau įmanoma ir svarbu galėti aprašyti bendravimo mechanizmus keliais struktūrizuotais keliais. Paslaugos aprašymo kalba skiria šį poreikį, apibrėždama gramatiką tinklapio paslaugų aprašymui kaip grupę sąsajų, paslaugos galinčios vykdyti operacijas per tinklo protokolus iki pabaigos taškų. Paslaugos aprašymai aprūpina sistemas dokumentacija ir tarnauja kaip receptas automatizuojant visą informaciją įtrauktą į bendravimo pritaikymą [4]. Skirtumas, tinklapio paslauga yra nei reikalaujanti išlaikyti XML pranešimus, nei gali būti sulaikyta SOAP ar HTTP protokolo, nei paleidžiama su NET aplinka, nors visos šios technologijos gali būti naudingos įrankiams.

Aiškumui, paslaugos aprašymai šiame skyriuje yra suformuluoti paprasta paslaugos aprašymo kalba (SWSDL), kaip pristatyta mūsų ankstesniuose mokymuose [6]. SWSDL aprašo sąsajas paslaugos objekto sistemos. Tai yra pedagoginė mašina, išmainanti lanstumą į aiškumą, ji nebando pakeisti WSDL [4] standarto. Kaip pavyzdys, manykite mes turime paprastą planuojamą paslaugą, tai siūlo pateikto darbo operaciją, tai duoda darbo aprašymą kaip argumentą. Funkcija gali būti nukreipta per HTTP protokolą.

Galiojantis SWSDL paslaugos aprašymas skaitomas taip:

Svarbu pažymėti, kad paslaugos samprata yra šiek tiek loginė nei fizinė (matereali). Našumui, virtualios aplinkos sudėtinis rodinys, toks kaip Apache Tomcat sudėtinis rodinys gali būti naudojamas paleisti daugiau nei vieną paslaugą ar sąsają tuo pačiu apdorojimo procesu.

Paslaugos sąsajos gali, bet nereikalauja, būti išskleistos tuo pačiu vardu. Jie gali paplisti per sudėtinius vardus per LAN ar WAN ir net jungti valdymo domenus. Ši sąvoka leidžia kalbant iš abstrakčios pusės apie nuoseklios sąsajos visumą nežiūrint į fizinį įgyvendinimą ar išskleidimo sprendimus. Čia kalbama apie paplitusią (vietinę) paslaugą, jei mes žinome ir norime apibrėžti tai, paslaugos sąsajos, iš tikrųjų, yra išskleistos per vardus (ar tuo pačiu vardu). Paprastai paslauga yra išliekanti (ilgai vykstanti), bet gali taip pat būti pasikeitusi (trumpai vykstanti, laikinai įvykusi vartotojui užklausus). Sekantis žingsnis link pagerinto vykdymo lankstumo yra (dar nepilnai išvystytas ir taigi dažnai) tinklapio paplitusio skaičiavimo paslaugų vizija [6, 7] kur programos yra daugiau nekonfiguruotos su statine informacija. Tiksliau sakant, pažadas yra, kad programos yra daug lankstesnės, prisitaikančios ir efektyvios užklausiant interneto duomenų bazių (registrų), paleidžiant informacijos suradimui ir tinklas surištas trečios grupės blokais. Paslaugos gali pasiskelbti (reklamuoti) save pačios ir susiję metaduomenys per tokias duomenų bazes įgalina aukšto lygio sudėtinių dalių sujungimą. Kai pasiekimai neseniai buvo įvykdyti paslaugų specifikacijos [4],

kreipimosi [5] ir registracijos [8] lauke, problema kaip naudoti turtingą ir išraiškingą pagrindinio tikslo – užklausimas kalbą, kad surasti paslaugas, kurios siūlo funkcionalumą atitinkantį smulkę specifikaciją, kol kas gavo mažai dėmesio. Iškyla paprastas klausimas.

Kaip tiksliai vietinė programa gali rasti svarbias, nuotolines paslaugas?

Pavyzdžiui, Aukštos Energijos Fizikos intensyvių duomenų analyzės programa ieško nuotolinių paslaugų, tai parodo tinkamą charakteristikų kombinaciją, įtraukiant tinkamas sąsajas, operacijas ir tinklo protokolus taip pat kaip tinklo įkėlimas, tinkama disko dalis, prieigos taisyklės ir galbūt Paslaugos Kokybė ir piniginės išlaidos. Tai kritinės svarbos būdas išvystyti pajėgumą paslaugos atradimui taip pat kaip užklausimo kalbos, tarpininkaujant ištekliams. Kas dar, dažnai svarbu naudoti keletą paslaugų, įgyvendinti užklausimo operacijas kombinacijoje. Pavyzdžiui, užklausimas gali įtraukti kombinuoto paslaugos perdavimo failo naudojimą (inscenizuoti įvesties ir išvesties duomenis iš nuotolinės svetainės), kopijos katalogo paslauga (rasti įvesties failo kopiją su gerais vietos duomenimis), užklausimo vykdymo paslauga (paleisti analizės programą), ir pagaliau vėl failo perdavimo paslauga (inscenizuoti išvesties grįžtančius duomenis į vartotojo darbalaukį). Tokiais atvejais dažnai naudinga taikyti koreliacijas. Pavyzdžiui, planuoklis, duomenų užklausoms gali ieškoti failo kopijos padėčių greituoju tinklo keliu į vykdymo paslaugą, kur užklausimas vartotų įvesties duomenis. Jei užklausa įtraukia, skaitydama didelį kiekį įvesties duomenų, tai gali būti prastas pasirinkimas naudoti vardą vykdymui, tai turi prastą duomenų padėtį, tikintis įvesties duomenų šaltinio, net jei labai lengvai įkelta. Kaip galima rasti rinkinį koreliuotų paslaugų, pritaikant kompleksinį raštą reikalavimų ir polinkių?

Jei vienas paslaugos atvejis gali būti galimas, paprastas sekantis žingsnis yra turėti daugiau nei vieną identišką, paplitusį atvejį, pavyzdžiui patobulinti galimybę ir įvykdymą. Besikeičiančiomis sąlygomis sistemoje įtraukia vėlavimas, dažnių juostos plotis, disponavimas, vieta, prieigos taisyklės, piniginės išlaidos ir asmeniniai polinkiai. Pavyzdžiui, prisitaikę vartotojai ar programos gali norėti pasirinkti skirtingą paslaugos parsisiuntimo turinio atvejį, priklausantį nuo įvertinto dažnių juostos pločio parsisiųsti. Jei dažnių juostos plotis pažeminamas pusiau parsisiuntus, vartotojas gali norėti įsijungti kitą parsisiuntimo paslaugą ir tęsti kur jis/ji paliko. Kokiu pagrindu galėtume atskirti vieną tarp keletos paslaugų atvejų?

Įvairioje sistemoje sujungti sudėtiniai valdymo domenai, pageidaujama prižiūrėti ir užklausti informacijos apie aktyvius dalyvius tokius kaip paslaugas, išteklius ir vartotojų bendrijas. Pavyzdžiai yra paslaugos atradimo tinklelio duomenims infrastruktūra, Domeno Vardo Sistema (DNS), elektroninis paštas, pasaulinis žiniatinklis, stebėjimo infrastruktūra ar

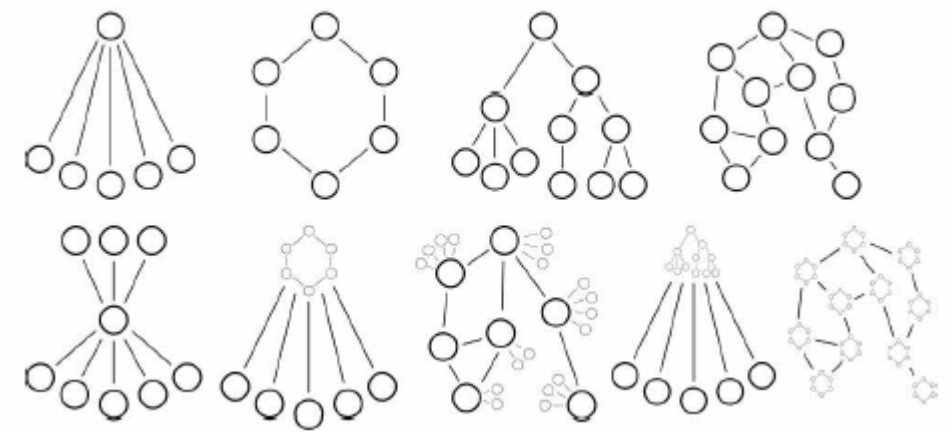
greitos žinių paslaugos. Bendrai naudojama informacija gali irgi įtraukti Paslaugos Kokybės aprašymą, failus, esamo tinklo įkėlimą, vardo informaciją, veiksmų citatas ir t.t. Tačiau, informacijos rinkinys pasaulyje padalintas per vieną ar daugiau duomenų bazių mazgų iš sistemos topologijų plataus rėžio, paaiškinimui įtraukiant autonomiją, mastelį, pasiekiamumą, įvykdymą ir apsaugą.

Kaip ir duomenų papildymo programoje [9,10,11], tikslas yra naudoti keletą nepriklausomų informacijos šaltinių kaip vieną. Tai įgalina informacijos užklausas, šaltinį ir atradimo paslaugą ir bendravimo funkcionalumą veikti sistemoje kaip vieną visumą, negu duotą jos dalį. Pavyzdžiui, tai leidžia ieškoti paslaugų aprašymų failo subendrinimo sistemos, nustatyti bendrą parsisiuntimo gebą, visų dalyvių organizacijų vardus ir t.t.

Tačiau, tokioje plačiai paplitusioje sistemoje sunku išlaikyti metaduomenų keitimąsi aprašant dalyvius tokius kaip paslaugas, išteklius, vartotojų bendrijas ir duomenų šaltinius. Nuspėjama, tinkama, teisinga ir patikima globalinės būsenos priežiūra yra negalima. Susumuota ir papildyta informacija gali būti išvesta, neteisinga, ar neprieinama ir viskas. Gedimas, blogas veikimas, apsaugos apribojimai ir nuolatinis keitimasis yra norma negu išimtis. Problema kaip palaikyti ryškų pagrindinį tikslą užklausimų atradimui per ekrano vaizdą, kad papildyti savarankiški, dinamiški duomenų bazių nodai iš paplitusios sistemos topologijų plačių rėžių iki šiol nebuvo adresuoti. Panagrinėkime greitų naujienų paslaugą, kuri sumuoja naujienas iš didelės įvarovės savarankiškų nuotolinių duomenų šaltinių būnant viduje sudėtinių valdymo domenų. Nauji duomenų šaltiniai dažnai ir sunaikinti išmetami. Vienas negali priverstinai kontroliuoti per sudėtinius valdymo domenus. Duomenų šaltinio rekonfiguracija ar fizinis judėjimas yra norma negu išimtis. Tada yra klausimas: Kaip gali vienas prarasti orientaciją ir užklausti metaduomenų aprašydamas organizuotų sistemų dalyvius patiriant dažną keitimąsi?

Neaišku, kaip įgalinti galingą užklausos atradimo priežiūrą ir bendradarbiavimo funkcionalumą, kurie veikia sistemoje kaip viena visuma, negu duotoje jos dalyje. Toliau, neaišku kaip leisti paieškos rezultatus, kurie yra nauji, leidžiant dinamišką turinį. Paplitusios duomenų sistemos prisiima suspaustą, pastovią centrinę kontrolę ir yra neįmanoma Tinklelio aplinkoms, kurios yra charakterizuotos nevienalitiškumu, plačiu mąsteliu, centrinės kontrolės stoka, sudėtiniais, savarankiškais valdymo domenais, nepatikimomis sudėtinėmis dalimis ir dažna dinamiška kaita. Pasirodo, kad lygiavertis duomenų tinklas (P2P) gali tikti prižiūrėti dinamišką duomenų bazių paiešką, pavyzdžiui, paslaugų atradimui.

Tokiose sistemose kaip Gnutella [13], Freenet [14], Tapestry [15], Chord [16] ir Globe [17] bendra P2P idėja yra tokia. Verčiau turima centralizuota duomenų bazė, paplitusi schema yra naudojama kur egzistuoja vienas ar daugiau savarankiškų duomenų bazės lizdų, jų priežiūra nuosava, potencialiai nevienalitiška, duomenų. Užklausos ne ilgai laikosi centrinės duomenų bazės, užuot jos yra pakartotinai paskleistos per tinklą keliems ar visiems duomenų bazės lizdams, ir rezultate yra surinktos ir atsiųstos klientui. Nodas laiko savo duomenų bazėje rinkinį sekų. Nodai yra susieti su saitais bet kokiu keliu. Saitas įgalina nodą užklausti kito nodo. Saito topologija aprašo saito struktūrą tarp mazgų. Centralizuotas modelis turi tik vieną nodą.



Pav.1 P2P tinklų topologijos[18]

Pavyzdžiui, paslaugos atradimo sistemoje, saito topologija gali rišti kartu rinkinį valdymo domenų, kiekvienas užvadintas registro nodas palaiko paslaugų aprašymus į domeną. Keletas saito topologijos modelių dengiančių gama iš centralizuotų modelių gali būti numatyti pilnai paplitę modeliai, tarp jų vienetiniai nodai, žvaigždė, žiedas, medis, diagrama ir hibridiniai modeliai [18]. Paveiksle 1 parodyta keletas topologijų pavyzdžių.

Bet kokioje P2P tinklo rūšyje, nodai gali pristatyti save į kitus nodus, tuo būdu formuodami topologiją. P2P tinkle paslaugos atradimui, nodas yra paslauga, kuri atskleidžia

sąsajas pristatymui ir P2P užklausoms. Čia, nodai, paslaugos ir kiti turinio tiekėjai gali pristatyti (jų) paslaugos aprašymus ir/arba kitus metaduomenis į vieną ar du nodus. Paskelbimas įgalina nodo topologijos konstrukciją (pvz. Žiedas, medis ar diagrama) ir tuo pačiu metu konstruoti sujungtą duomenų bazę ieškomą užklausų. Kituose pavyzdžiuose, nodai gali prižiūrėti kopijos vietą [19], kopijos valdymą ir optimizavimą [20, 21], sąveikaujanti prieiga į tinklelio įgalintas, susijusias duomenų bazes [22], genų seka ar daugiakalbis vertimas, aktyviai naudojant tinklą paslaugų radimui tokiu kaip kopijos katalogai, nuotoliniai genų planai ar kalbos žodynai.

Šis skyrius pateikia ir apibendrina būtinas atradimo problemos savybes ir tada suranda sprendimus, kuriuos taiko plačioms interneto sistemoms. Parodo kaip prižiūrėti išraiškingas pagrindinio tikslo užklausas per vaizdo ekraną, tai papildo savarankiškus, dinamiškus duomenų bazės nodus iš plačios sistemos topologijų. Mes aprašome pirmą žingsnį link tinklino skaičiavimo, duomenų bazių ir tinklo paslaugų apjungimo. Šio skyriaus likutis organizuotas taip:

2 paragrafas kreipia dėmesį į dinamiškos priežiūros ir savalaikės informacijos, esančios plačios įvairovės nepatikimuose, dažnai besikeičiančiuose, savarankiškuose ir nevienalyčiuose nuotoliniuose duomenų šaltiniuose esančias problemas. Mes kuriame duomenų bazę XQuery užklausoms per dinamišką turinį – vadinamą hyper registru.

3 paragrafas apibrėžia *Tinklo Paslaugų Atradimo Architektūrą* (WSDA)**,** kuri rodo internetą kaip paslaugų platų rinkinį su plačiu rinkiniu gerai apibrėžtų sąsajų. Tiksliai apibrėžia mažą rinkinį sudėtinio tikslo bendravimas primityvų (blokus), atradimui. Šie primityvai dengia paslaugos identifikaciją, paslaugos aprašymo atgavimą, duomenų publikavimą taip pat kaip minimalią ir galingą užklausos priežiūrą. WSDA parodo sąveiką, apima pramonės standartus.

4 ir 5 paragrafai aprašo Unifikuotą Lygiavertę Duomenų Bazės struktūrą (UPDF) ir atitinkantį Duomenų Bazės Protokolą (PDP) pagrindinio tikslo užklausos priežiūrai plačiose, nevienalytėse sistemose jungiant keletą valdymo domenų. Jie yra sujungti, tam tikra prasme, kad leistų išreikšti specifinias atradimo programas plačiam rėžiui duomenų tipų, nodo topologijoms, užklausos kalboms, užklausų atsakymo būsenoms, kaimyno pasirinkimo būdams, laiko tėkmėms ir kitoms galiojimo srities parinktims.

6 paragrafas aptarinėja susijusį darbą. Pagaliau, 7 paragrafas reziumuoja ir pateikia išvadas. Mes taip pat nusakome bendrais bruožais įdomius nurodymus būsimims tyrinėjimams.

2. XML IR XML SCHEMOS

Norint atlikti heterogeninių duomenų šaltinių integraciją, visų pirma reikia turėti visiems integruojamiems šaltiniams bendrą schemą, kuri leistų atlikti integraciją. Šiame darbe nusprendžiau pasirinkti XML kalbą ir šios kalbos struktūrą apibrėžiančias XML schemas. Norėčiau panagrinėti XML kalbos ir schemų privalumus bei trūkumus, aptarti jų sandarą bei panaudojimą.

2.1 XML programavimo kalba

XML (*Extensible Markup Language*) buvo sukurta 1996 metais. Ją sukūrė XML darbo grupė, padedama W3C (World Wide Web Consortium) konsorciumo.

XML aprašo duomenų objektų klasę, vadinamą XML dokumentu ir iš dalies aprašo šiuos dokumentus apdorojančių kompiuterinių programų elgesį. XML yra SGML (Standart Generalized Markup Language) [ISO 8879] apribota forma. Savo struktūra XML dokumentai atitinka SGML dokumentus.

XML dokumentai yra sudaryti iš elementų, vadinamų esybėmis, kuriose yra išnagrinėti (*parsed*) arba neišnagrinėti duomenys. Išnagrinėti duomenys yra sudaryti iš simbolių. Vieni simbolių junginiai sudaro duomenis, kiti junginiai sudaro žymes. Žymėmis yra koduojama dokumento sandara ir loginė struktūra. XML pateikia sandaros ir loginės struktūros apribojimų pritaikymo mechanizmą. [1]

XML yra naudojama kaip žymių kalbų sudarymo šablonas:

- XML naudoja neribojamą žymių skaičių, t.y. skirtingai nei HTML nėra jokių apribojimų jų kiekiui – vartotojas pats gali susikurti žymes atsižvelgdamas į informacijos pobūdį.
- Kiekviena XML kalba yra skirta savo taikomajai sričiai, tačiau kalbos pasižymi bendromis savybėmis
- Dokumentams apdoroti yra skirti bendri įrankiai

2.1.1 XML paskirtis

Galima būtų išskirti šiuos pagrindinius XML kalbos paskirties bruožus:

- Atskirti sintaksę nuo semantikos. Tai yra reikalinga todėl, kad yra kuriamas bendras informacijos struktūrizavimo mechanizmas (naršyklės vaizduojamos semantikos yra apibrėžiamos stilių lentelėmis).
- Leisti kurti žymes bet kokiai įsivaizduojamai taikomųjų programų sričiai.

- Palaikyti internacionalizaciją ir nepriklausomybę nuo platformų
- Ateityje vaizduoti struktūrizuotą informaciją, kuri apimtų ir duomenų bazes.

2.1.2 XML dokumento fragmento pavyzdys

```
<?xml version="1.0" encoding="windows-1257"?>
<klientas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="klientas.xsd">
    <vardas>Jonas</vardas>
    <pavarde>Jonaitis</pavarde>
    <adresas>
        <gatve>Savanorių</gatve>
        <namo_nr>15</namo_nr>
        <buto_nr>3</buto_nr>
        <miestas>Kaunas</miestas>
        <indeksas>3000</indeksas>
        <valstybe>Lietuva</valstybe>
    </adresas>
    <telefonas>
        <namu>37788523</namu>
        <darbo>37777555</darbo>
        <mobilus>61444444</mobilus>
    </telefonas>
</klientas>
```

Kreipinys į XML schemos dokumentą

6 pav. XML dokumento pavyzdys

Kaip matome šiame pavyzdyje XML dokumento žymės yra sukurtos taip, kad kuo tiksliau atitiktų loginę žymimų duomenų prasmę. Šis XML dokumentas yra sukurtas pagal XML schemos dokumente apibrėžtus apribojimus. XML dokumentas yra pateikiamas 3.2.6 skyriuje.

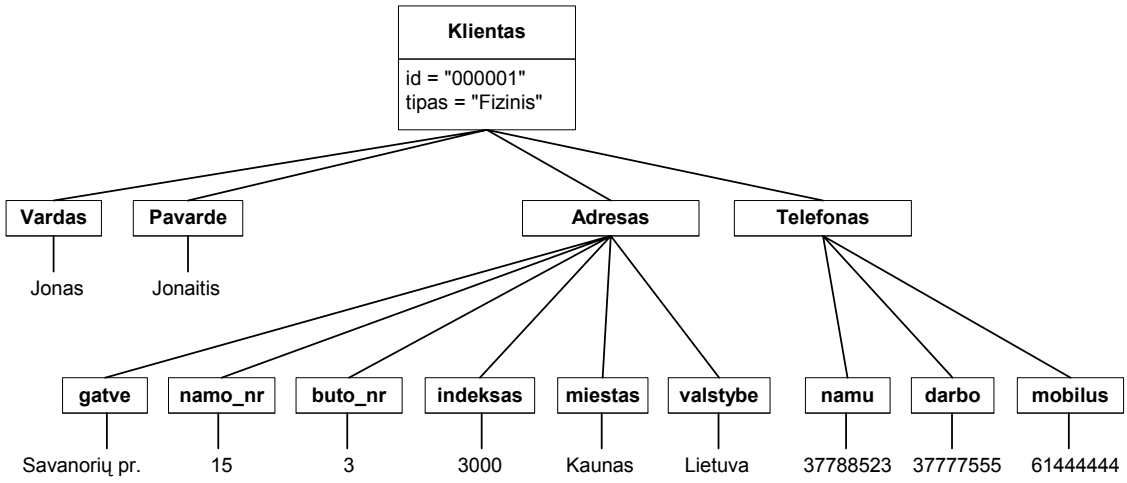
2.1.3 Konceptualus XML vaizdas

XML yra surūšiuotas, sužymėtas medis:

- Iš simbolių sudarytų duomenų mazguose saugomi tikri duomenys (simbolių eilutės) – dažniausiai šie mazgai turi būti netušti ir būti nesusieti su kitais simbolių duomenų mazgais
- Elementų mazgai yra pažymėti:
    - vardu (dažniausiai vadinamu elementų tipu)
    - atributų rinkiniu. Kiekvienas atributas yra sudarytas iš vardo ir reikšmės.

  ir gali turėti vaikinius mazgus.

Pateikiu 3.1.2 skyriuje aprašyto XML dokumento fragmento pavyzdžio medžio tipo vaizdą:



7 pav. XML dokumento struktūrinės schemos vaizdas

Dokumento medinę struktūrą galima peržiūrėti naudojantis Internet Explorer naršykle.

Be to XML medžius gali sudaryti ir kitų rūšių mazgai:

- Vykdymo instrukcijos – skirtingiems procesoriams skirti pažymėjimai
- Komentarai – kaip ir programavimo kalbose
- Dokumento tipo aprašymas DTD

2.2 XML schemos

XML 1.0 apraše buvo įtrauktas ir XML dokumentų struktūras aprašančių įrankių rinkinys, vadinamas DTD (*Document Type Definition*). DTD suteikė priemones, kurios leido apibrėžti, kurios elementų ir atributų struktūros yra leistinos dokumente. Be to DTD aprašė mechanizmus, kurie suteikė atributams pirmines (*default*) reikšmes, nusakė pakartotinio panaudojimo turinį (esybes) ir tam tikrą metaduomenų informaciją (žymėjimus). Tačiau XML kūrėjams greitai nebeužteko DTD galimybių ir dėl to XML kalbos kūrėjas, W3C konsorciumas, nusprendė kurti naują XML dokumentų aprašymo kalbą. Ji turėjo tiksliau aprašyti dokumento struktūras ir jų turinį, taip pat palaikyti XML vardų sritis (*namespace*) ir XML žodynams aprašyti panaudoti XML žodyną. [2]

XML suteikia tam tikro failo ar srauto formatui nagrinėti skirtą gramatiką. Ši gramatika apima pagrindinę sintaksę, kuri daugiausia kreipia dėmesį į elementų žymes ir atributų specifikacijas. Dauguma taikomųjų programų reikalauja daug tikslesnės struktūros. XML schemos suteikia mechanizmą, kuris leistų specifikuoti labiau išplėstus gramatinius apribojimus. Galimybė nustatyti XML dokumento struktūrą padaro XML dokumentus kur kas labiau nuspėjamus. Programuotojas gali išnagrinėti XML schemą ir tiksliai žinoti kaip atrodo

atitinkami XML dokumentai. Analogiškai programa gali nepriimti XML dokumento, jei jo struktūra neatitiks nurodomos XML schemos.

XML schema specifikuota teisingus XML dokumento elementus ir atributus. Be to, XML schema aprašo tikslią elementų hierarchiją (lizdiniams elementams). Schema taip pat specifikuoja XML dokumente esančias parinktis (tokias kaip elemento tipo panaudojimo limito nustatymas) ir kitus įvairius apribojimus. XML schema gali nustatyti ir elemento reikšmių ribas.

2.2.1 XML schemos dokumentas

XML schemos dokumentas yra XML schemos visų dalių atvaizdavimas XML formatu. XML schemą dažniausiai aprašo vienas ar keli XML schemos dokumentai. Dauguma schemos dokumento elementų yra schemos elementų atvaizdavimas XML formatu, kiti elementai padeda sujungti schemą iš kelių schemos dokumentų.

2.2.2 XML schemų privalumai

XML schemos suteikia nemažai privalumų kūrimo aplinkai. Dauguma šių privalumų yra neapčiuopiami, tačiau netiesiogiai didinantys produktyvumą ir sumažinantys produktų kūrimo laiką.

Labiausiai neapčiuopiamas XML schemos aspektas yra tas, kad XML schema nustato kontraktą tarp programinės įrangos taikomųjų programų arba PĮ taikomųjų programų dalių. XML generuojančios programos kūrėjas arba darbuotojas, kuriam reikia rankomis sukurti XML dokumentą, žino koks jis turi būti. XML gaunančios programinės įrangos kūrėjas ne tik žino ko laukti, bet ir gali patikrinti gaunamus XML duomenis pagal turimą XML schemą. XML schema gal ir nėra tokia paprasta kaip įprastinis dokumentas, tačiau ji yra labai tiksli. Be to kūrėjas gali papildyti XML schemą XML komentarais ir XML schemų žymėjimais.

Didelės apimties taikomąsias programas kuriančioms įmonėms, kontraktų žymėjimai supaprastina moduliarizaciją, resursų paskirstymą, testavimą ir išdėstymą. Moduliarizuoti kodą yra paprasčiau dėl to, kad yra iš anksto žinomi tam tikri apribojimai (kiekvienas apribojimas yra XML dokumentas). Tai iš karto palengvina ir resursų paskirstymą: individualūs kūrėjai gauna specifines užduotis, kuriose yra aiškiai apibrėžti įėjimo ir išėjimo duomenys. Testavimas taip pat tampa daug lengvesnis: XML generuojantis kūrėjas užtikrina, kad sugeneruotas XML atitiks XML schemą ir XML gaunantis kūrėjas gali lengvai kurti testinius XML dokumentus paprastu redaktoriumi. Dėl to integracijos testavimas yra daug sėkmingesnis nei tradicinių duomenų (toks kaip objektų ir struktūrų perdavimas). Išdėstymas taip pat yra paprastesnis: XML generuojančio kodo versijos gali būti išdėstomos nebūtinai tuo pat metu kaip ir XML gaunančios kodo versijos – priimant kad schema yra stabili.

Šiuolaikinės paskirstytos aplinkos sujungia aukščiau paminėtas problemas. Dažniausiai XML apdorojančią taikomąją programą kuriantis kūrėjas gali kurti Web servisą; bet kas Internete gali potencialiai generuoti XML, kuris bus siunčiamas į Web servisą. Šiuo atveju XML schema yra esminis dokumentacijos akcentas. XML yra tarsi kontraktas kurio visi turi laikytis. Tarkim kūrėjas gali keisti Web servisą tol, kol nesikeičia XML schema. Tokiu būdu kūrėjai tiksliai žino ko XML schemą reikalauja.

2.2.3 XML schemų trūkumai

Žiūrint iš praktinės perspektyvos, vienas didžiausių su XML schemomis susijusių sunkumų yra mokymosi pobūdžio. XML schemas nėra lengva kurti; sukurti gerą XML schemą yra dar sudėtingiau. Geroje schemoje turi būti atsižvelgta į visas apribojimų sąlygas.

XML schemos yra nepaprastai tikslios; tačiau jos yra ir daugiareikšmės. Tai neturėtų stebinti tų, kurie bent kiek susiduria su paprastu žodiniu XML formatu, kuriuo yra pagrįsta XML schemų struktūra. Dėl daugiareikšmiškumo problemos skirtingi projektuotojai ir vartotojai tuos pačius elementus ar atributus gali interpretuoti visiškai skirtingai.

XML naudojimas sukelia ir tam tikrų pastebimų papildomų išlaidų. Šiame skyriuje paminėtas XML daugiareikšmiškumas reikalauja papildomų išlaidų, reikalingų XML iššifruoti. Norint visiškai suprasti XML reikalingas pilnas nagrinėtojas (*parser*). Tuo tarpu paprastos programavimo struktūros (kaip C_{struct} arba $Java_{class}$) yra kur kas efektyvesnės. XML yra tikrinamas pagal XML schemą. Šis tikrinimas reikalauja tam tikro laiko. Dar blogiau yra tai, kad XML schemų dokumentus reikia taip pat nagrinėti, nes jie yra parašyti XML kalba. Kompiuteriams tobulėjant ir spartėjant, kūrėjai dažniausiai nebeatsižvelgia į sunaudojamus procesoriaus resursus. Kai kuriais atvejais kūrėjas, norėdamas išgauti didesnį greitį, gali schemas užkrauti į atmintį (*cache*). Tačiau net ir šiuo atveju vartotojas praranda šiek tiek laiko, kuris reikalingas atlikti vieną tikrinimą. Taigi, vykdant keletą tikrinimų, vartotojas praras nemažai laiko. [14]

2.2.4 XML schemų palyginimas su DTD

Kaip jau buvo minėta 2.2 skyriaus pradžioje, XML schemos išsivystė iš DTD. Šiame darbe nehomogeninių duomenų bazių integracijai naudosiu XML schemas. Norėdamas pagrįsti savo pasirinkimą palyginsiu DTD ir XML schemų galimybes bei savybes:[12]

1 lentelė. DTD ir XML schemų palyginimas

Savybė	DTD	XML schema
Naudojama sintaksė	Naudojama savita, su XML nesuderinta sintaksė (EBNF)	XML schemos aprašomos XML kalba
Vardų srities palaikymas	DTD gali būti susieta tik su viena vardų sritimi.	Gali būti susieta su viena arba daugiau vardų sričių.
Duomenų tipai	Duomenų tipų kiekis yra ribotas	Duomenų tipai yra labai įvairūs, be to vartotojas gali pats susikurti duomenų tipus.
Dokumentų papildymas	DTD gali naudoti tik uždarą modelį	XML schemos naudoja atvirą modelį, kuriame galima papildyti žodyną ir paveldėjimą.
Dokumento sudėtingumas	DTD yra glausti ir paprasti	XML schemos yra daugiareikšmės
Dokumento tikrinimas	DTD patikrinti negalima	Gali būti patikrinamos ir valdomos programiškai kaip bet kuris XML dokumentas.
Schemai skirtos priemonės	XML tikrinti pagal DTD yra sukurta daug įrankių	XML tikrinti pagal XML schemas yra tik keletas įrankių, tačiau jie yra kuriami.
Atributų grupių palaikymas	DTD palaiko tik labai apribotą atributų grupę	XML palaiko neribojamas atributų grupes.

Taigi apibendrinant XML schemų savybes galima būtų išskirti šiuos požymius:

- Platus duomenų tipų pasirinkimas
- Galimybė kurti nuosavus duomenų tipus
- Išplėstiniai tipai
- Atviro, uždaro arba tobulinamo turinio modeliai
- Grupavimas
- Vardų sričių palaikymas

2.2.5 XML schemų sandara

Šiame skyriuje norėčiau paminėti pačius svarbiausius XML schemų sandaros elementus.

- **Atributų aprašymas <attribute>**

  Atributai aprašomi norint:

  - Panaudojant paprastus tipų aprašus lokaliai patikrinti atributų informacijos reikšmes
  - Atributų informacijos reikšmėms nustatyti standartines arba fiksuotas reikšmes

- **Elementų aprašymas <element>**

Elementų aprašymas skirtas šiems tikslams:

- panaudojant tipų aprašymus lokalus patikrinti elemento informacijos reikšmes
- nustatyti standartines arba fiksuotas elemento informacijos reikšmes.
- Sukurti unikalumo ir ryšių apribojimus tarp susietų elementų ir atributų reikšmių.
- Naudojant elementų keitimo grupių mechanizmą valdyti elementų pakeičiamumą

- **Sudėtinio tipo aprašymas <complexType>**

Sudėtinis tipas aprašomas norint:

- Apibrėžti ar elemento informacijos reikšmė bus tuščia, ar atitiks vieno elemento arba mišraus turinio modelį. Sudėtinis tipas taip pat gali būti aprašomas norint, kad elementas-vaikas atitiktų nurodytą paprasto tipo aprašymą.
- Naudojant tipų aprašymo mechanizmus gauti kompleksinį tipą iš kitų paprastų arba kompleksinių tipų.
- Apriboti galimybę gauti papildomus tipus iš duoto kompleksinio tipo.
- Valdyti elementų pakeitimo apribojimus
- Suteikiant atributų aprašymą apriboti elementų informacijos reikšmių tvarką ir atributų turinį

- **Atributų grupės aprašymas <attributeGroup>**

Atributų grupės aprašymas yra skirtas nuorodoms iš schemos komponentų XML atvaizdavimo.

- **Modelių grupės aprašymas <Group>**

Modelių grupės aprašymas susieja vardus ir papildomus žymėjimus su modelių grupe

- **Modelio grupė <all>, <choice>, <sequence>**

Kai elementų-vaikų informacija nėra apibrėžiama kaip tuščia arba nuoroda į paprasto tipo aprašymą, šių elementų sekos turinys gali būti labiau apibrėžiamas modelio grupe

- **Pastabos**

Pastabos skirtos žmonėms ir kompiuteriams skirtų schemos elementų pastaboms aprašyti

- **„Laukinė korta“ (*wildcard*)**

„Laukinė korta“ leidžia tikrinti su vardų sritimi susijusias atributų ir elementų informacijos reikšmes nepriklausomai nuo jų lokalių vardų

- **Paprasto tipo aprašymas**

Paprastų tipų aprašymas yra skirtas apriboti atributų ir elementų vaikinių elementų informacijos reikšmes. [13]

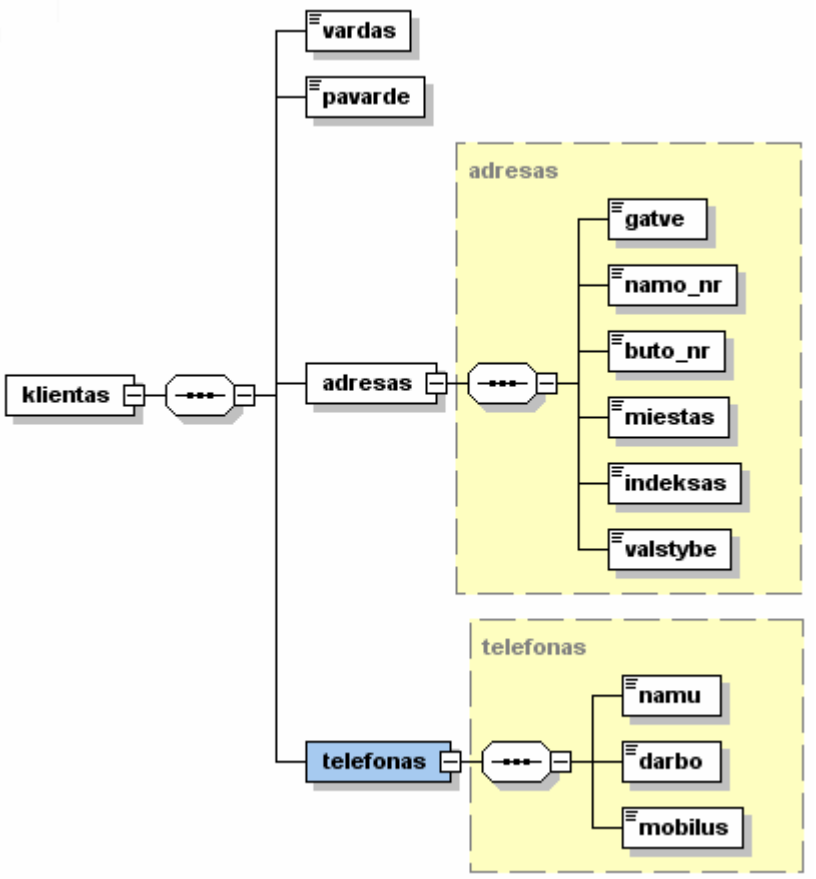
2.2.6 XML schemos dokumento pavyzdys

Šiame skyriuje norėčiau pateikti 2.1.2 skyriuje pateikto XML dokumentą aprašantį XML schemos dokumentą:

```
<?xml version="1.0" encoding="windows-1257"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="klientas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="vardas" type="xs:string"/>
                <xs:element name="pavarde" type="xs:string"/>
                <xs:element name="adresas" type="adresas"/>
                <xs:element name="telefonas" type="telefonas"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="adresas">
        <xs:sequence>
            <xs:element name="gatve" type="xs:string"/>
            <xs:element name="namo_nr" type="xs:integer"/>
            <xs:element name="buto_nr" type="xs:integer"/>
            <xs:element name="miestas" type="xs:string"/>
            <xs:element name="indeksas" type="xs:integer"/>
            <xs:element name="valstybe" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="telefonas">
        <xs:sequence>
            <xs:element name="namu" type="telnr"/>
            <xs:element name="darbo" type="telnr"/>
            <xs:element name="mobilus" type="telnr"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="telnr">
        <xs:restriction base="xs:string">
            <xs:length value="8"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

8 pav. XML schemos dokumento, pagal kurį tikrinamas 6 pav. vaizduojamas XML dokumentas, pavyzdys

2.2.7 XML schemos dokumento grafinis struktūros vaizdas



9 pav. 8 paveiksle pavaizduoto XML schemos dokumento grafinis vaizdas

Kaip matome paveiksle XML schemos struktūra taip pat yra medžio tipo, nors atskiras medžio dalys buvo aprašytos kaip atskiri objektai. Šiame pavyzdyje kompleksinis elementas yra šakninis elementas, kuris turi savo vaikinius elementus, o šie savo ruoštu taip pat gali turėti savo vaikinius elementus. Kiekvienas elementas gali turėti atributus, kurie grafiškai nėra atvaizduojami.

3. DINAMINIO TURINIO DUOMENŲ BAZĖ

Didelėje paskirstytoje sistemoje, autonominės daugelio administracinių domenų dalelės yra aprašomos įvairia informacija. Dalelės dažnai susijungia, išsiskiria ir elgiasi taip, kad pakeltų sistemos našumą. Neįmanoma iš anksto numatyti ir patikimai užtikrinti būsenos reguliavimą tam tikru laiko momentu. Renkama informacija gali būti jau pasenusi, nepilna ir nevientisa. Didžiausia problema yra:

Kaip duomenų bazės mazgas turi išlaikyti informacija, surinktą iš daugelio kitų nepatikimų, dažnai besikeičiančių, autonominių nuotolinių duomenų šaltinių? Ir konkrečiai kaip jis tai turi atlikti nepakenkdamas sistemos stabilumui ir paprastumui? Kaip gali būti vykdomos galingos užklausos į laikui jautrią informaciją?

Yra išrastas duomenų bazės tipas, kuris adresuoja problemą. Duomenų bazė XQuery užklausoms per dynamiškai platinamą turinį yra sumanyta ir specializuota – vadinama *hyper registru.* Registras kelias svarbias savybes. XML duomenų modelis leidžia struktūrizuotus ir pusiau struktūrizuotus duomenis, kurie yra svarbūs papildymui ir nevienalyčiam turiniui. Xquery užklausos kalba [23] leidžia veiksmingą paiešką, kuri yra kritinė nereikšmingoms programoms. Duomenų bazės būsenos priežiūra yra bazuojama ant softo būsenos, kuri įgalina patikimą, nuspėjamą ir paprastą turinio papildymą iš plataus skaičiaus autonomiškų, platinamų turinio tiekėjų. Saito turinys, atmintinės turinys ir hibridinis išeiti/judėti bendravimo modelis leidžia platų rėžį dinamiško turinio atsisiuntimo būsenas, kurios gali būti valdomos visų trijų sistemų sudėtinių dalių: turinio tiekėjo, registro ir kliento.

Hyper registras turi duomenų bazę, kuri laiko sekų rinkinį gali turėti dalį bet kokio turinio. Turinio pavyzdžiai įtraukia paslaugos aprašymą, išreikštą WSDL [4], Paslaugos aprašymo kokybę, failą, failo kopijos vietą, esamo tinklo įkėlimą, hosto informaciją, akcijos dalis, t.t. Sekos daromos su saito turinio rodymu į įdėto turinio duomenų šaltinį.

3.1 Turinio nuoroda ir turinio tiekėjas

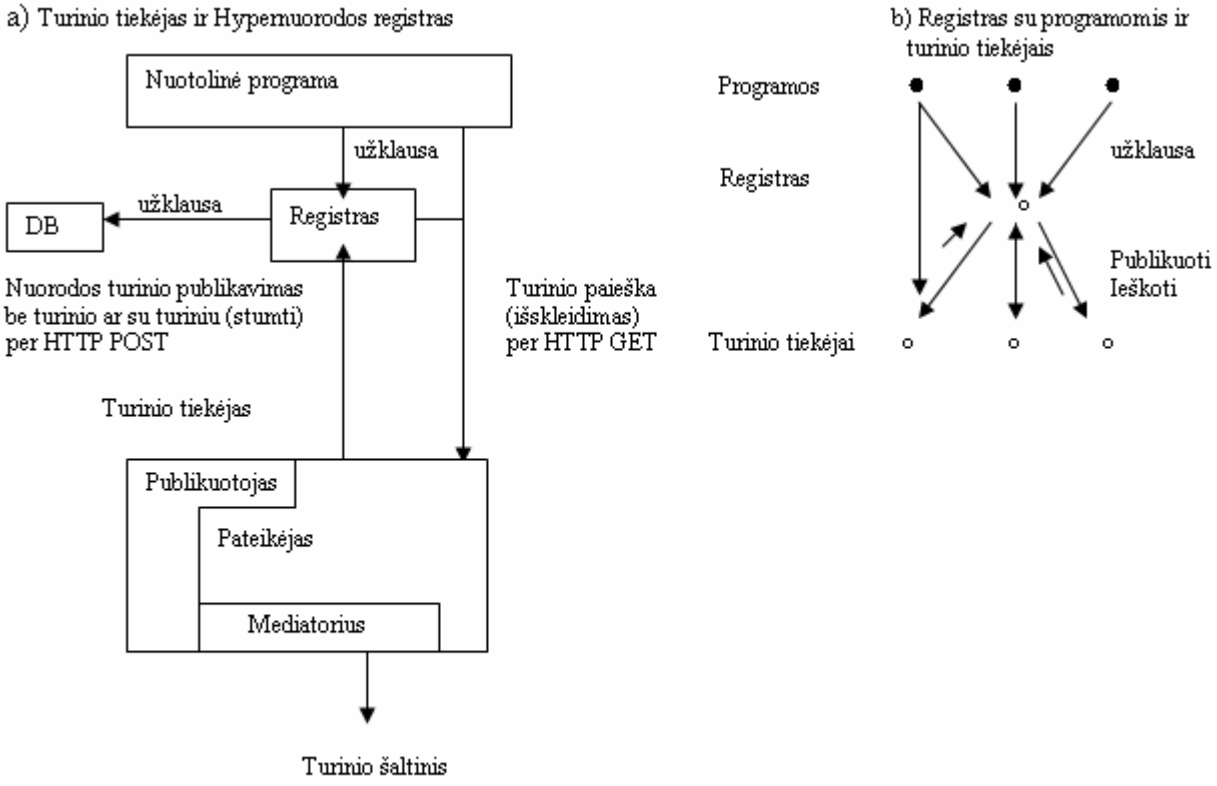
Turinio nuoroda. Turinio nuoroda gali būti bet koks URI. Tačiau, dažniau pasitaikantis yra HTTP(S) URL, kuris, šiuo atveju, rodo į turinio tiekėjo turinį, ir HTTP(S) GET reikalauja nuorodos grįžti į ęsamą (atnaujinimą) turinį. Kitais žodžiais, paprasta nuoroda yra naudojama. Paslaugos atradimo kontekste mes naudojame terminą paslaugos nuoroda reikšti nuorodos turiniui, kuris rodo į paslaugos aprašymą. Turinio nuorodos gali būti laisvai pasirenkamos kol jos atitinka URI ir HTTP URL specifikaciją [24]. Nuorodų turinio pavyzdžiai:

```
urn:/iana/dns/ch/cern/cn/techdoc/94/1642-3
urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
http://sched.cern.ch:8080/getServiceDescription.wsdl
https://cms.cern.ch/getServiceDesc?id=4712&cache=disable
http://phone.cern.ch/lookup?query="select phone from book where phone=4711"
http://repcat.cern.ch/getPFNs?lfn="myLogicalFileName"
```

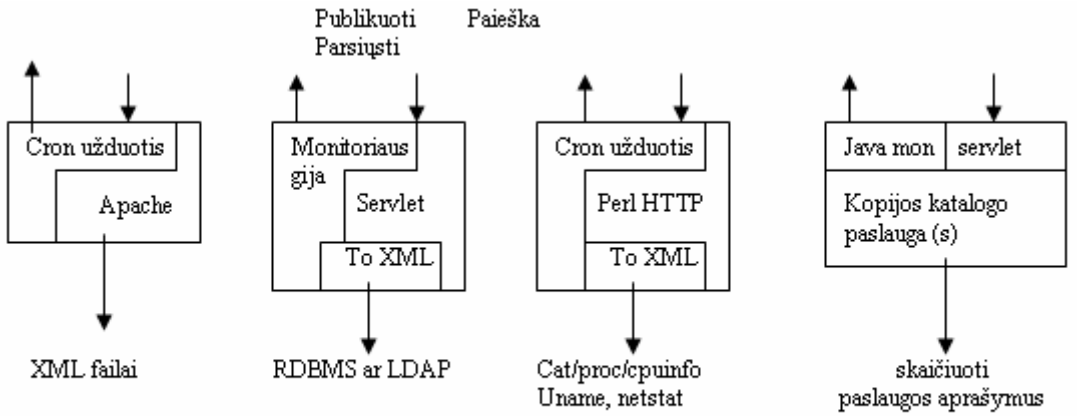
Turinio tiekėjas. Turinio tiekėjas siūlo informaciją atitinkančią nevianalytį, visuotinio tinklo duomenų modelį. Darant ką nors, tai paprastai naudoja keletą vidinio tarpininko rušių perduoti informaciją iš vietinio ar nuosavo duomenų modelio į visuotinio tinklo duomenų modelį. Turinio tiekėjas gali būti matomas kaip tinklų sietuvas į nevienalytiškus turinio šaltinius. Turinio tiekėjas yra tam tikras skėtis dviems sudėtinėms dalims, vadinamoms pateikėjas ir platintojas. Pateikėjėjas yra paslauga ir atsakymas į HTTP(S) GET paieškos turinio užklausas iš registro ar kliento. Platintojas yra kodo dalis, kuri platina nuorodos turinį ir galbūt taip pat turinį, į registrą. Platintojas nereikalauja būti paslauga, nors naudoja HTTP(S) POST bendravimų siuntimui. Turinio tiekėjo struktūra ir sąveika su registru ir klientu yra pavaizduota 2 (a) paveiksle. Pastaba, kad klientas gali aplenkti registrą ir tiesiai traukti ęsamą turinį iš tiekėjo. 2 (a) paveikslas iliustruoja registrą su keletu turinio tiekėjų ir klientų.

Kaip tik dinamiškame interneto voratinklyje leidžiama plati įvairovė įgyvendinimų duotam protokolui, kaip pateikėjas naudoja turinį paieškoje. Turinys gali būti statinis ar dinaminis (sugeneruotas). Pavyzdžiui, pateikėjas gali išsaugoti turinį tiesiai iš failo ar duomenų bazės, ar iš potiancialiai išvestos atmintinės. Tikslumo padidinimui, gali taip pat dinamiškai iš naujo panaudoti turinį, reikalaujant. Panagrinėkime tiekėjų pavyzdį 3 piešinyje. Paprastas, labai naudingas turinio tiekėjas naudoja HTTP serverio tokio kaip Apache reikmenį, pateikti XML turinį iš failo sistemos. Paprastas darbas žiūrėti Apache serverio būseną ir skelbti ęsamą būseną registrui. Kitas turinio tiekėjo pavyzdys yra JAVA, kuri įgalina duomenis laikyti reliacinėje ar LDAP duomenų bazės sistemoje. Turinio tiekėjas gali vykdyti likusios komandos linijos įrankius, skelbdamas sistemos būsenos informaciją, tokią

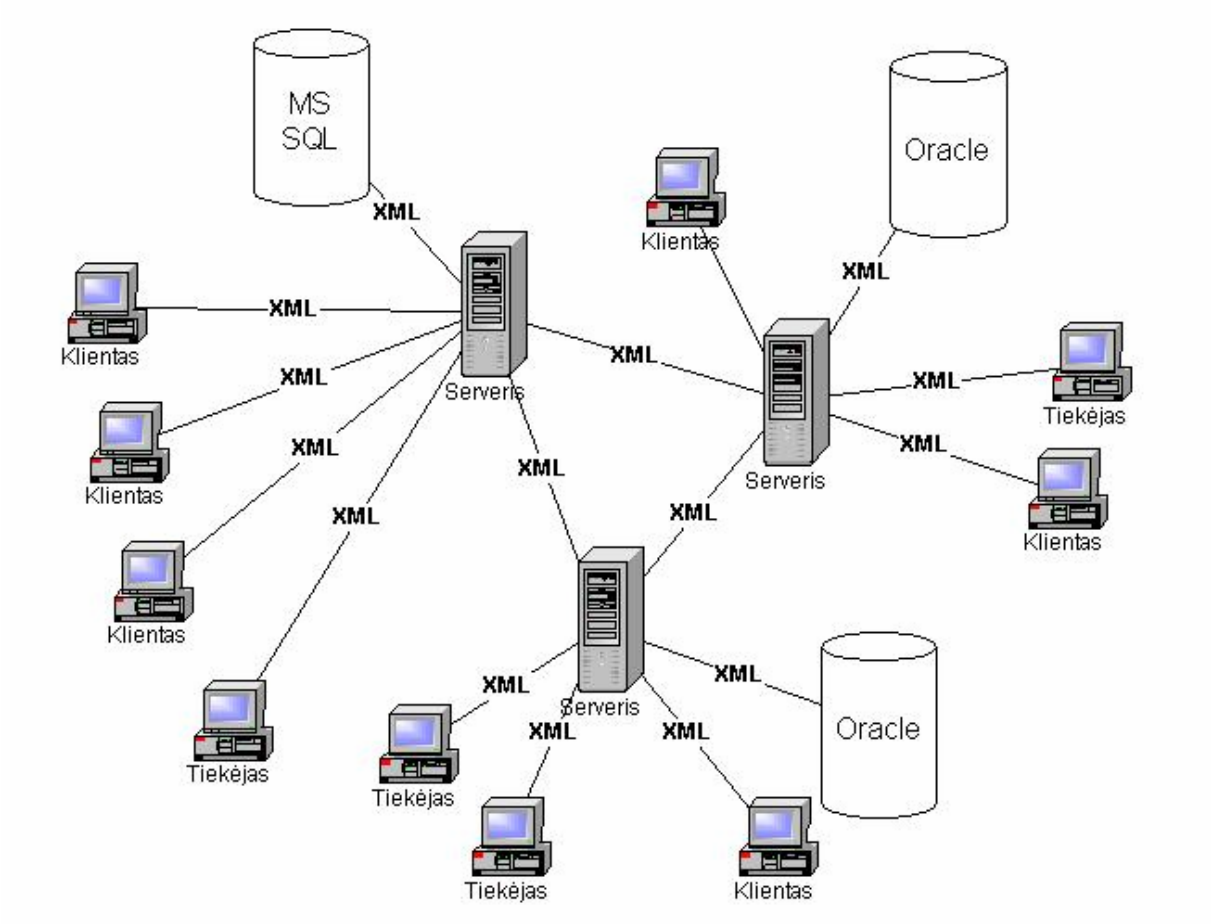
kaip, tinklo statistikos, veikianti sistema ir procesoriaus tipas. Kitas turinio tiekėjo pavyzdys yra tinklo paslauga, tokia kaip, kopijų katalogas, kuris skelbia paslaugos aprašymą ir/arba nuoroda, ką klientas gali surasti ir po to kreiptis į ją.



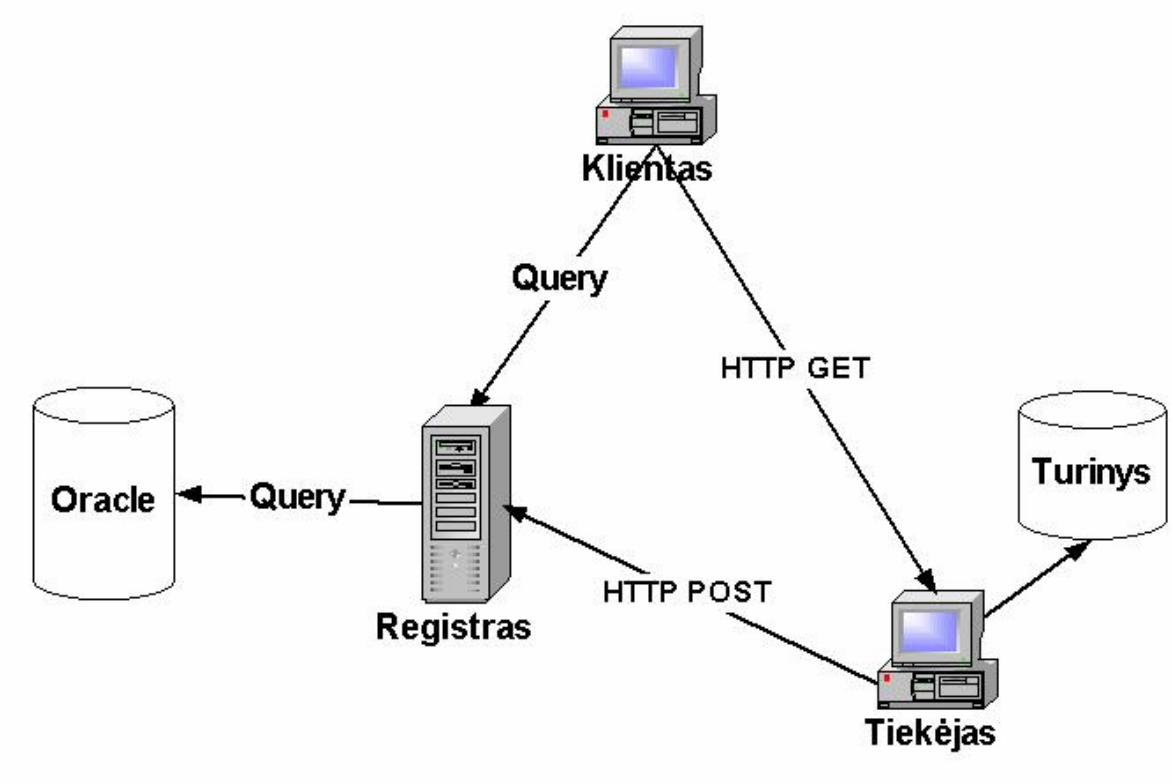
2. Pav. Turinio tiekėjas ir registras.



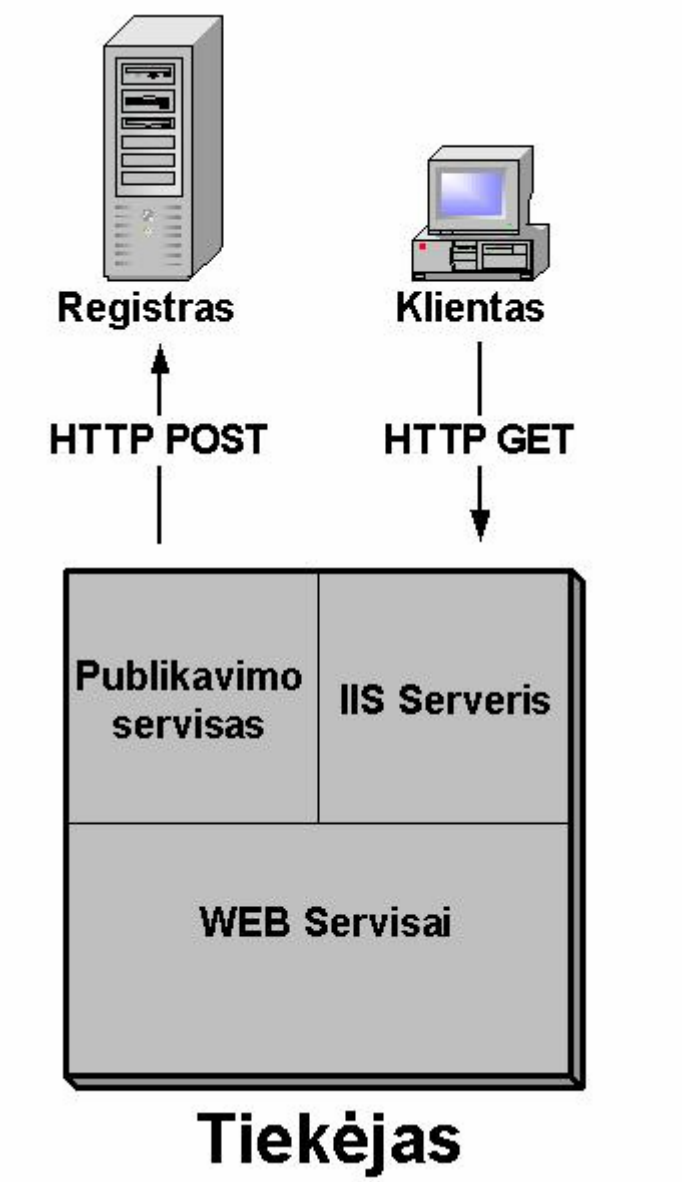
3. Pav. Turinio tiekėjų pavyzdžiai.



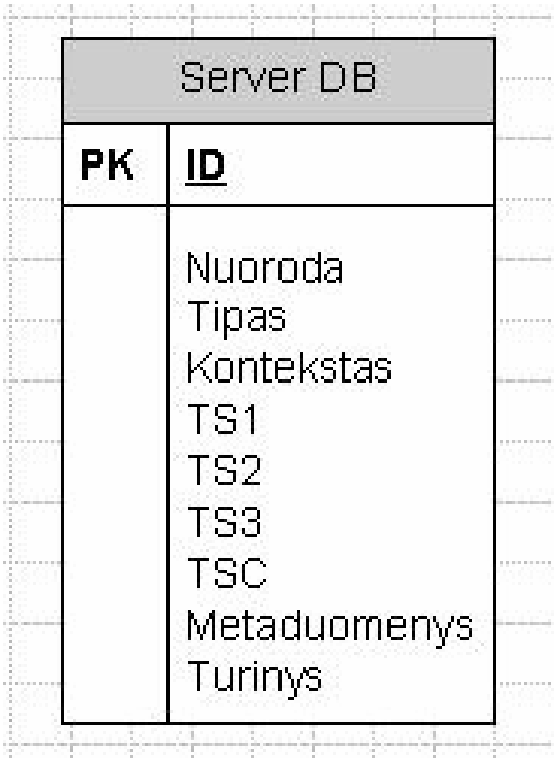
4 pav. Tinklo paslaugų registras, tiekėjai ir klientai.



6 pav. Turinio tiekėjas, klientas, registras.



7 pav. Turinio tiekėjo, registro ir kliento realizacijos pavydys.

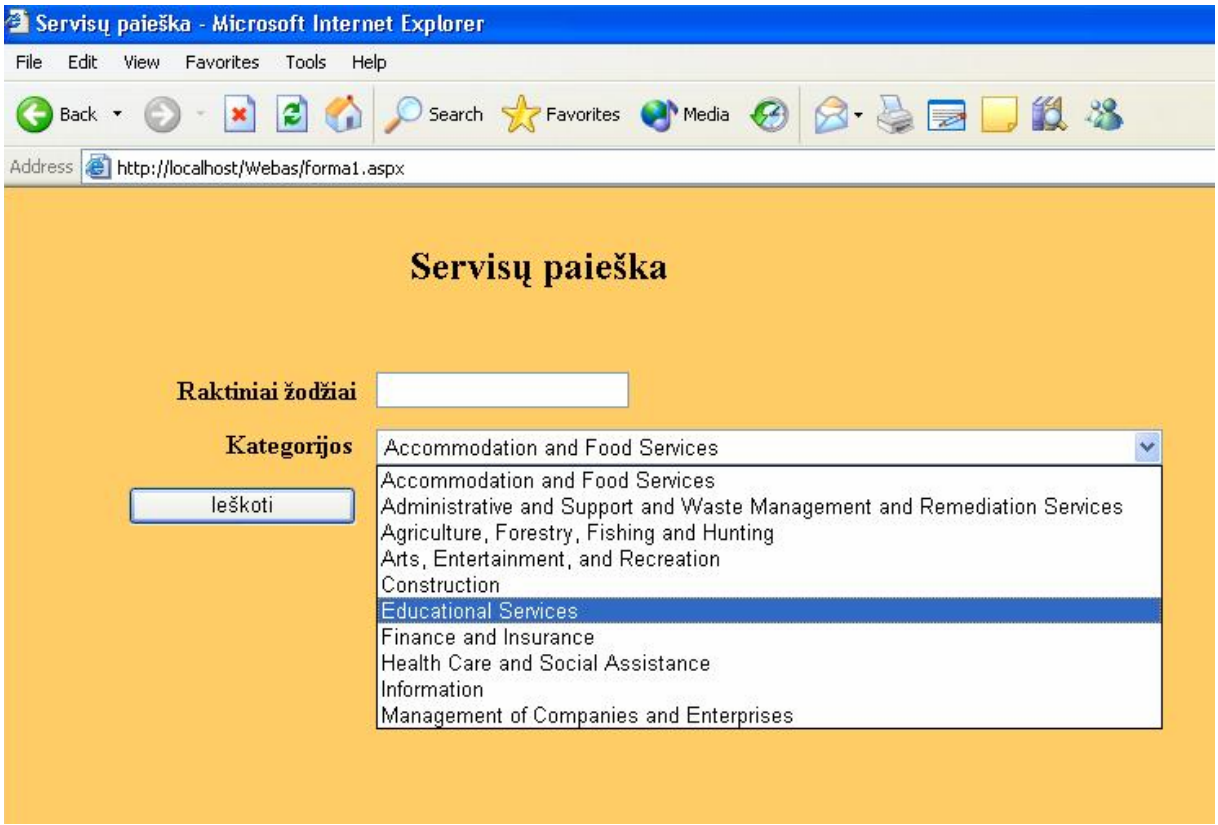


8 pav. Registro duomenų bazės struktūra

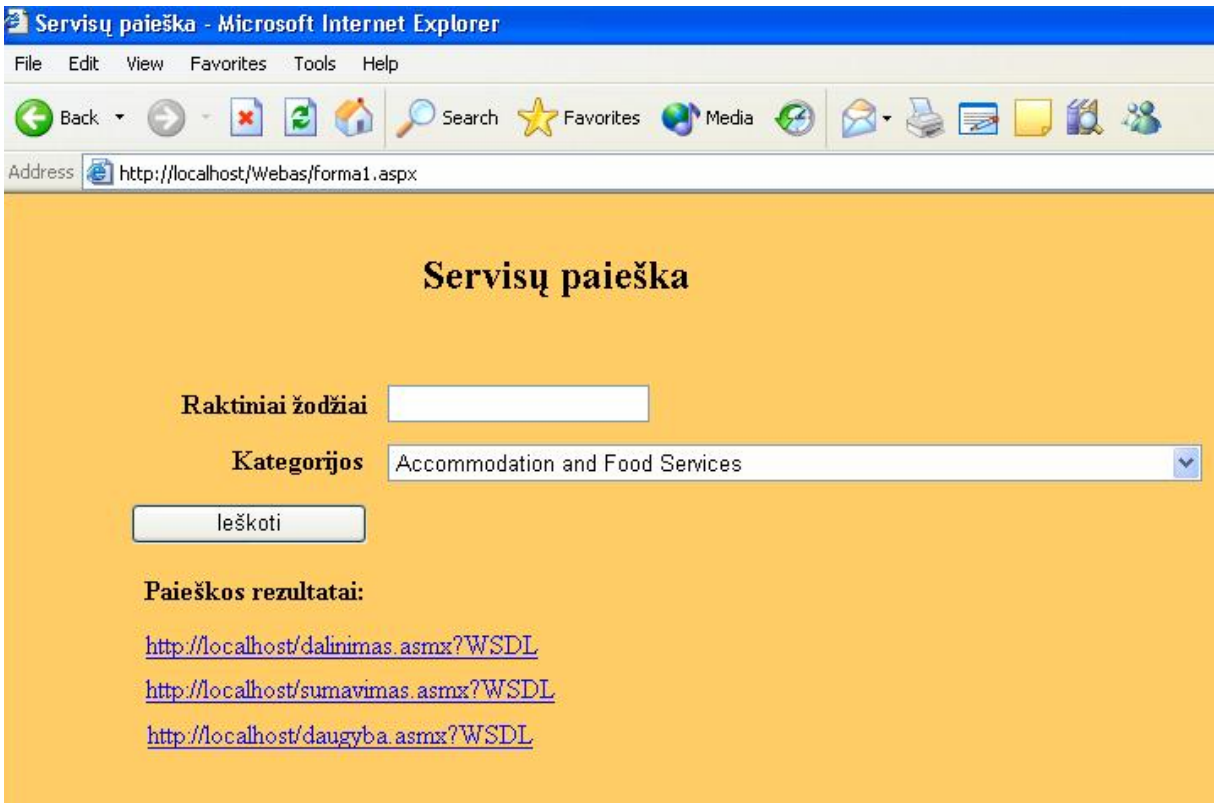
Nuoroda	Kontekstas	Tipas	TS1	TC	TS2	TS3	Meta duomenys	Turinys
http://sched001.cern.ch/ getServiceDescription	Parent	Service	10	15	20	30	<owner name = "http://cms.cern.ch"/>	<service> A < /service>
http://sched.infn.it:8080/ pub/getServiceDescription	Child	Service	20	25	30	40	null	<service> B < /service>
http://repcat.cern.ch/pub/ getServiceDescrip- tion?id=4711	Child	Service	30	0	40	50	null	null
http://repcat.cern.ch/pub/ getStatistics	Null	RepStats	60	65	70	80	null	<repcatStats> ... </repcatStats>

9 pav. Registo duomenų bazės lentelės fragmentas

10 pav. Tinklo paslaugos tiekėjo sąsaja



11 pav. Tinklo paslaugų paieskos sąsaja



12 pav. Tinklo paslaugų paieškos rezultatai

3.2 Publikavimas

Turinio tiekėjas gali skelbti duoto tipo turinį į vieną ar daugiau registrų. Tiksliau, turinio tiekėjas gali platinti dinamišką žymeklį vadinamą turinio nuoroda, kuri savo ruožtu įgalina registrą atnaujinti ęsamą turinį. Dėl efektyvumo, skelbimo operacija paima kaip įvestį rinkinį nulių ar daugiau sekų. Ką mes siūlome vadinti (DDM) Dinaminis Duomenų Modelis, kiekviena XML seka turi turinio nuorodą, tipą, kontekstą, keturis softo būsenos laiko žymes, metaduomenis ir turinį. Seka yra anotuotas, sudėtinio tikslo softo būsenos duomenų sudėtinis rodinys, kuris gali turėti dalį turinio ir leisti persisiųsti šį turinį bet kada, kaip parodyta 13 ir 14 paveiksluose.

Ląstelės sandara:

Nuoroda	Tipas	Kontekstas	Laiko žymės	Metaduomenys
Turinys (neprivalomas)				

HTTP(S) nuorodos semantika : = HTTP(S) GET (ląstelės.nuorodos) → ląstės.turinys

Kita URI nuoroda : = nespecifikuota

13 Pav. Ląselės sandara

3.2.1 Nuoroda

Turinio nuoroda yra, paprastai, URI, kaip parodyta aukščiau. Jei tai HTTP(S) URL, tada ęsamas turinio tiekėjo turinys gali būti parsiųstas bet kada.

3.2.2 Tipas

Tipas aprašo kokia turinio rūšis pateikiama (pvz. Paslaugos programa/oktetas-srautas, paveikslas/jpeg, tinklo įkėlimas, kliento informacija).

3.2.3 Kontekstas

Kontekstas aprašo kodėl turinys skelbiamas ar kodėl naudojamas (pvz. Vaikas, tėvas, gnutella, žiūrėjimas). Kontekstas ir tipas leidžia užklausą diferencijuoti lemiamu požymiu net jei turinio atmintinė nėra prižiūrima ar autorizuota.

3.2.4 Laiko žymės TS1, TS2, TS3, TC

Bazuojasi įdėtoje softo laiko būsenos žymose, apibrėždamos gyvavimo savybes, seka gali pagaliau būti pašalinta ‚jei ne iš naujo parsiųsta stream'o patvirtinimo pranešimų. Laiko žymos leidžia efektyvias atmintines, keletas iš kurių aprašytos 2.5 skyriuje.

3.2.5 Metaduomenys

Neprivalomas metaduomenų elementas be to aprašo turinį ir/arba parsisiuntimą už ką gali buti išreikštas su buvusiais požymiais. Pavyzdžiui, metaduomenys gali apsaugoti turinio skaitmeninį XML parašą [25]. Gali aprašyti turinio tiekėją ar turinio savininką. Kitas metaduomenų pavyzdys yra Tinklo Paslaugos Apžiūros Kalba (WSIL) dokumentas [26] ar fragmentas, specifikuojantis papildomus turinio parsiuntimo mechanizmus, išskyrus HTTP turinio nuorodos parsiuntimą. Metaduomenų argumentas yra elementas įgalinantis papildymą ir lanksčią raidą.

3.2.6 Turinys

Duodant nuorodą esamas turinio tiekėjo turinys gali būti parsiųstas bet kada. Turinio tiekėjas gali taip pat įtraukti ęsamo turinio kopiją kaip skelbimo dalį. Turinys ir metaduomenys gali būti struktūrizuoti ar pusiau struktūrizuoti duomenys bet kokio gerai suformuoto XML dokumento ar fragmento forma.


```
<tupleset>
<tuple link="http://registry.cern.ch/getDescription" type="service" ctx="parent"
TS1="10" TC="15" TS2="20" TS3="30">
<content>
<service>
<interface type="http://cern.ch/Presenter-1.0">
<operation>
<name>XML getServiceDescription()</name>
<bind:http verb="GET" URL="https://registry.cern.ch/getDesc"/>
</operation>
</interface>
<interface type = "http://cern.ch/XQuery-1.0">
<operation>
<name> XML query(XQuery query)</name>
<bind:beep URL="beep://registry.cern.ch:9000"/>
</operation>
</interface>
</service>
</content>
<metadata> <owner name="http://cms.cern.ch"/> </metadata>
</tuple>
<tuple link="http://repcat.cern.ch/getDesc?id=4711" type="service" ctx="child"
TS1="30" TC="0" TS2="40" TS3="50">
</tuple>
<tuple link="urn:uuid:f81d4fae-11d0-a765-00a0c91e6bf6"
type="replica" TC="65" TS1="60" TS2="70" TS3="80">
<content>
<replicaSet LFN="urn:/iana/dns/ch/cern/cms/higgs-file" size="10000000" type="MySQL/ISAM">
<PFN URL="ftp://storage.cern.ch/file123" readCount="17"/>
<PFN URL="ftp://se01.infn.it/file456" readCount="1"/>
</replicaSet>
</content>
</tuple>
<tuple link="http://monitor.cern.ch/getHosts" type="hosts" TC="65" TS1="60" TS2="70" TS3="80">
<content>
<hosts>
<host name="fred01.cern.ch" os="redhat 7.2" arch="i386" mem="512M" MHz="1000"/>
<host name="fred02.cern.ch" os="solaris 2.7" arch="sparc" mem="8192M" MHz="400"/>
</hosts>
</content>
</tuple>
</tupleset>
```

14 Pav.

Individualus elementas gali, bet nereikalauja, turėti schemą (XML schema [28]), tokiu atveju tai gali galioti pagal schemą. Visi elementai gali, bet nereikalauja, bendrai naudoti shemą. Lankstumas yra svarbus nevienalyčio turinio papildymui.

Registro skelbimo operacija turi negaliojantį skelbimo skaitmeninį parašą. Remiantis sekos rinkiniu, seka unikaliai identifikuojama pagal sekos šifrą, kuris yra pora (nuorodos turinys, kontekstas). Jei šifras jau neegzistuoja skelbime, seka yra įterpiama į registro duomenų bazę. Egzistuojanti seka gali būti atnaujinta skelbiant kitas reikšmes po tuo pačiu sekos šifru. Egzistuojanti seka yra pasisavinama turinio tiekėjo, kuris sukurtas su pirmu skelbimu. Rekomenduojama, kad turinio tiekėjui su kita tapatybe nebūtų leidžiama publikuoti ar atnaujinti seką.

3.3 Užklausa

Aptardami duomenų modelį ir kaip publikuoti sekas, dabar mes nagrinėjame užklausos modelį. Siūlomos dvi sąsajos, vadinamos MinQuery ir XQuery.

3.3.1 MinQuery

MinQuery sąsaja teikia paprasčiausią, galimą užklausos priežiūrą ("pažymėti viską" – stilius). Gražina sekas įtraukiant ir pašalinant paslėptą turinį. *Paimti seką* užklausos operacija neargumentuojama ir gražina pilną visų sekų rinkinį "kaip yra". Tai yra, užklausos išvesties formatas ir publikacijos įvesties formatas yra tokie patys (žiūr. 5 Paveikslą). Jei prižiūrima, išvestis įtraukia paslėptą turinį. *Gauti nuorodas* užklausos operacija taip pat neargumentuojama ir gražina pilną visų sekų rinkinį. Tačiau, tai visada pakeičia paslėpto turinio elutę. Kitais žodžiais, turinys yra praleistas sekų, potencialiai saugant dažnių juostos plotį. Sekanti seka 5 paveiksle turi tokią formą.

3.3.2 XQuery

XQuery sąsaja teikia veiksmingą XQuery [23] priežiūrą, kuri yra svarbi realiai paslaugai ir šaltinio atradimo atvejais. XQuery yra standartinė XML užklausos kalba sukurta remiantis W3C. Tai leidžia efektyvų ieškojimą, kuris yra kritinis nepaprastoms programoms. Viskas kas gali būti išreikšta su SQL gali, taip pat, būti išreikšta su XQuery. Tačiau, XQuery yra raiškesnė kalba negu SQL, pavyzdžiui, yra palaikoma kelio išraiškos hierarchiniam naršyme. XQuery paslaugos atradimas yra pavaizduotas 6 paveiksle. Detalesnis plataus rėžio, vidutinio ir kompleksinio užklausų atradimo ir jų pristatymo XQuery [23] kalba aptarimas duotas [6]. XQuery gali dinamiškai papildyti išorinius duomenų šaltinius per dokumento (URL) funkciją, kuri gali būti naudojama apdoroti nuotolinių operacijų XML rezultatus, kreipiantis per HTTP. Pavyzdžiui, duotas paslaugos aprašymas su operacija *getPhysicalFileNames (LogicalFileName)* , užklausa gali lyginti reikšmes, dinamiškai sukurtas šios operacijos.

- *Rasti visas (prieinamas) paslaugas.*

```
RETURN /tupleset/tuple[@type="service"]
```

- *Find all services that implement a replica catalog service interface that CMS members are allowed to use, and that have an HTTP binding for the replica catalog operation "XML getPFNs(String LFN).*

```
LET $repcat := "http://cern.ch/ReplicaCatalog-1.0"
FOR $tuple in /tupleset/tuple[@type="service"]
LET $s := $tuple/content/service
WHERE SOME $op IN $s/interface[@type = $repcat]/operation
SATISFIES
$op/name="XML getPFNs(String LFN)" AND $op/bindhttp/@verb="GET"
AND contains($op/allow, "cms.cern.ch")
RETURN $tuple
```

- *Rasti visus kopijos katalogus ir grąžinti jų fizinio failo vardus (PFNs), duodant loginį failo vardą (LFN); neįrašant PFNs nepaleidžiant su ftp://*

```
LET $repcat := "http://cern.ch/ReplicaCatalog-1.0"
LET $s :=
/tupleset/tuple[@type="service"]/content/service[interface@type =
$repcat]
RETURN
FOR $pfn IN invoke($s, $repcat, "XML getPFNs(String LFN)",
"http://myhost.cern.ch/myFile")/tupleset/PFN
WHERE starts-with($pfn, "ftp://")
RETURN $pfn
```

- *Grąžinti kopijos katalogo paslaugų skaičių.*

```
RETURN
count(/tupleset/tuple/content/service[interface/@type="http://cern
.ch/ReplicaCatalog-1.0"])
```

- *Rasti visas (vykdymo paslauga, rezervavimo paslauga) poras, kur abi poros paslaugos yra tame pačiame domene. (Užduotis nori skaityti ir rašyti vietoje).*

```
LET $executorType := "http://cern.ch/executor-1.0"
LET $storageType := "http://cern.ch/storage-1.0"
FOR $executor IN
/tupleset/tuple[content/service/interface/@type=$executorType],
$storage IN
/tupleset/tuple[content/service/interface/@type=$storageType
AND domainName(@link) = domainName($executor/@link)]
RETURN <pair> {$executor} {$storage} </pair>
```

16 Pav. XQuery kalbos užklausų pavyzdys.

Tos pačios taisyklės naudojamos užklausoms taip pat naudojamos ir XQuery atlikimui. Bet kuris realizacija gali naudoti modulinį ar paprastą XQuery procesorių, tokį kaip *Quip* XML užklausos operacijai (XQuery užklausa). Bet ne tik turinys, bet taip pat nuorodos turinys, kontekstas, tipas, laiko žymos, metaduomenys ir t.t., yra sekos dalis, užklausa taip pat gali būti atrinka šioje informacijoje.

3.4 Kešavimas

Atminties turinio kešavimas yra svarbus besikreipiančios programos efektyvumui. Registras gali ne tik palaikyti nuorodų turinį, bet taip pat ęsamo turinio kopiją, rodytos nuorodos. Su kešavimo atmintimi, programoms (klientams) nereikalinga atnaujinti tinklo prijungimo kiekvienam nuorodos turiniui užklausimo rezultatų rinkinyje, turiniui gauti. Ši vengia draudžiamo vėlavimo, ypatingai esant platiems rezultato rinkiniams. Registras (MS WINDOW duomenų bazė) gali (bet nereikalauja) prižiūrėti kešą. Registras, kuris neprižiūri atminties, ignoruoja bet kokį perduotą turinį ir turinio tiekėjo. Tai išsaugo tik nuorodų turinį. Vietoje atminties turinio gražina tuščias eilutes (žiūrėkite antrą seką kaip pavyzdį 5 paveiksle). Atminties išdavimų nuoseklumas auga. Atminties registro užklausos operacijos gali grąžinti sekas su pasenusiu turiniu, turiniu, kuris nebegalioja, atsižvelgiant į pagrindinės tiekėjo bazinės kopijos turinį.

Atminties registras gali įdiegti stiprią ar silpną atminties nuoseklumo būseną. Stipri atsargos atminties nuoseklumo būsena yra serverio sugadinimas [29]. Čia turinio tiekėjas informuoja registrą su pranešimo seka, kai tik turi vietoje modifikuotą turinį. Mes naudojame šį metodą pritaikytoje versijoje, kur atminties registras gali veikti atsižvelgiant į kliento (programos) stumimo raštą (stumimo registras) ar serverio skleidimo raštą (skleidimo registras) ar hibridą. Atitinkama sąveika yra tokia:

3.4.1 Traukimo registras

Turinio tiekėjas skelbia turinio nuorodą. Registras tada skleidžia esamą turinį per nuorodos turinio paiešką į atsargos atmintį. Kai tik turinio tiekėjas sumodifikuoja turinį, praneša registrui su pranešimo seka palaikant laiką kada turinys buvo sumodifikuotas. Registras gali tada nuspręsti skleisti esamą turinį vėl, tam kad atnaujinti atmintį. Tai registrui spręsti jei ir kada išskleisti turinį. Registras gali išskleisti turinį bet kada. Pavyzdžiui, gali dinamiškai išskleisti atsiųstą sekos turinį paveiktas užklausos. Tai svarbu dažnai išsaugant atmintyje dinamiškus duomenis tokius kaip įkeltą tinklą.

3.4.2 Stūmimo registras

Skelbimo seka išstumta iš turinio tiekėjo į registrą turi ne tik turinio nuorodą, bet taip pat esamą turinį. Kai tik turinio tiekėjas modifikuoja turinį, stumia seką su nauju turiniu į registrą, kuris gali atitinkamai atnaujinti atsargos atmintį.

3.4.3 Hybridinis registras

Hybridinis registras įdiegia abi skleidimo ir stumimo sąveikas. Jei turinio tiekėjas tiktai praneša, kad turinys buvo pakeistas, registras gali pasirinkti išskleisti esamą turinį į atsargos atmintį. Jei turinio tiekėjas išstumia turinį, atsargos atmintis gali būti atnaujinta su išstumtu turiniu. Tai yra registro tipas vėliau prisiimantis, kai tik atminties registras yra svarstomas.

Neatmintinės registras ignoruoja turinio elementus, jei yra. Pranešimas, sakoma bus be turinio, jei turinys nėra teikiamas sekoje. Šiaip pranešama: bus su turiniu. Pranešimas be turinio nori pasakyti, kad nėra įrašo apie kuriamą išsaugotą turinį (neutralus). Tai neduoda suprasti, kad turinys neturi būti išsaugomas ar sugadintas.

3.5 Dinaminė būsena

Patikimai, nuspėjamai ir paprastai paplitusiai priežiūros būsenai, registro seka yra prižiūrima kaip ir *dinaminė būsena.* Sekos pagaliau gali būti pašalintos, jei ne iš naujo parsiųstos pranešimų patvirtinimo savalaike eilute iš turinio tiekėjo. Pabaigai, sekos palaiko laiko žymes. Seka nebegaliojanti ir pašalinta, jei ne aiškiai atnaujinta per periodišką pranešimą, nuo to laiko vadinama *atsiųsta iš naujo*. Kitais žodžiais, atsiuntimas iš naujo leidžia turinio tiekėjui įtakoti nuorodos turinį ir/arba kešo atminties turinį likti tolimesniam laikui.

Stiprios atminties nuoseklios būsenos serverio anuliavimas yra plečiamas. Lankstumui ir išraiškingumui, yra pritaikyta Tinklelio Pranešimo Struktūros idėja [30]. Pranešimo operacijos ima absoliutaus laiko žymes TS1, TS2, TS3, TC per sekas. Semantikai yra sekantys. Turinio tiekėjas teigia, kad jo turinys buvo modifikuotas TS1 laiku ir tas jo esamas turinys tikėtinas būti galiojančiu nuo TS1 laiko iki mažiausiai TS2. Tikimasi, kad turinio nuoroda egzistuoja tarp TS1 ir mažiausiai TS3 laiko. Laiko žymės gali paklusti neligybei $TS1 \leq TS2 \leq TS3$. TS2 sukelia paslėpto turinio atminyje pasibaigimą, tada TS3 sukelia nuorodų turinio pasibaigimą. Paprastai, TS1 lygus paskutinės modifikacijos ar pirmos publikacijos laikui, TS2 lygus TS1 plius keletą minučių ar valandų, ir TS3 lygus TS2 plius keletas valandų ar dienų. Pavyzdžiui, TS1, TS2 ir TS3 gali atspindėti laiką, 10 minučių ir 2 valandas atitinkamai.

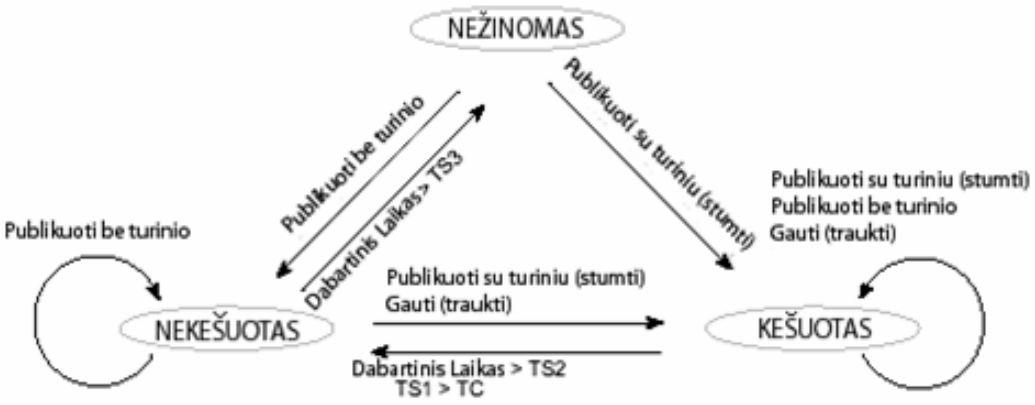
Sekos taip pat palaiko TC laiko žymę, kuri indikuoja laiką, kada sekoje įdėtas turinys (ne turinio tiekėjo turinio kopija), buvo paskutinis sumodifikuotas, typiškai tarpininkavimu kelyje

tarp programos (kliento) ir turinio tiekėjo (pvz. Registras). Jei turinio tiekėjas publikuoja turinį, tada mes paprastai turime TS1= TC. TC gali būti nulio reikšmė, jei sekos neturi turinio. Iš čia, registras neatliekantis išsaugojimo visada turi TC rinkinį į nulį. Pavyzdžiui, aukšto dinamiškumo tinklo įkėlimo tiekėjas gali skelbti savo nuorodą be turinio ir TS1=TS2 patarimui, kad yra netinkamas saugoti atmintyje jo turiniui. Konstantos yra skelbiamos su turiniu ir TS2=TS3=begalybė, TS1=TC=*esamasLaikas.* Laiko žymių semantikai gali būti reziumuoti :

```
TS1= Turinio tiekėjo laiko paskutinis sumodifikuotas turinys
TC= Sekos įdėto turinio laikas buvo sumodifikuotas
TS2= Tikėtinas laikas, kai esamas turinys tiekėjuje yra mažiausiai
galiojantis
TS3= Tikėtinas laikas, kai nuorodos turinio tiekėjuje, yra mažiausiai
galiojantis (ęsantis)
```

Ląstelių įterpimas, atnaujinimas ir ištrinimas įvyksta laiko žymės būsenos keitimuose, reziumuojama 7 paveiksle. Ląstelių rinkinio viduje, jos yra identifikuojamos esmine ląstele, kuri yra pora (nuorodos turinys, kontekstas). Ląstelės gali būti vienoje iš trijų būsenų: neatpažįstama, nepaslėpta ar paslėpta. Ląstelė yra neatpažįstama, jei nėra registre (raktas neegzistuoja). Kitaip, yra žinoma. Kai seka paskirta nepaslėptos būsenai, besitesiantis vidinis modifikavimo laikas TC yra nulis ir atsargos atmintis yra ištrinta, jei yra. Nepaslėptai sekai mes turime TC < TS1. Kada laąstelė paskirta paslėptai būsenai, turinys yra atnaujintas ir TC yra esamo laiko rinkinys. Kesuotoms ląstelėms mes turime TC ≥TS1.

Ląstelės pereina iš nežinomos į kešuotą ar nekešuotą būseną, jei tiekėjas publikuoja atitinkamai su ar be turinio. Ląstelės tampa nežinomos, jei nuorodos turinys negalioja (esamasLaikas >TS3); Ląstelės tada ištrinamos. Tiekėjas gali įtakoti ląstelių ištrinimą publokuodamas su esamuLaiku>TS3.



17 pav. Dinaminės būsenos pasikeitimai

Ląstelės keičia būseną iš nekešuotos į kešuotą, jei tiekėjas publikuoja nuorodą kartu su turiniu, arba jei tiekėjas paskelbia turinį ar jei registras pats pasiima turinį iš šaltinio. Traukiant turinį, registras gali palikti TS2 nepakeistą, bet taip pat gali sekti būseną, kuri pratęsia lastelės gyvavimo laiką (ar bet kokią kitą suderintą būseną). Ląstelės būsena pasikeičia iskešuotos į nekešuotą, jei turinys nebegalioja. Toks negaliojimas įvyksta, kai atnaujinimas negaunamas laiku (*esamasLaikas*>TS2), ar jei atnajinimas naujo rodo, kad tiekėjas atnajino turinį (TC<TS1).

3.6 Lankstus atsinaujinimas

Nuorodos turinys, turinio atsargos atmintis, hybridinis išskleistas bendravimo modelis ir XQuery efektyvumas leidžia platų rėžį dinamiško turinio siuntimų, kurie gali būti valdomi visų trijų sistemos sudedamųjų dalių: turinio tiekėjo, registro ir programos (kliento). Visos trys sudėtinės dalys gali rodyti, kaip valdyti turinį, atsižvelgiant į jų siuntimo supratimą. Pavyzdžiui, turinio tiekėjas gali modeliuoti savo turinio siuntimą per stumiamas tinkamas laiko žymes ir turinį. Registras gali modeliuoti savo turinio siuntimą per kontroliuojamą tiekėjo publikacijų priėmimą ir aktyvų turinio siuntimo iš tiekėjo išskleidimą. Jei rezultatas (tinklo statistika) yra aukštyn į datą atsižvelgiant į registrą, bet iš datos atsižvelgiant į programą, programa gali išskleisti atsiųstą turinį iš tiekėjo. Tačiau, tai yra neefektyvu plataus rezultato rinkiniams. Vis dėlto, tai svarbu programoms (klientams), kad turinio rezultatai yra grąžinami, atsižvelgiant į jų siuntimų supratimą, ypatingai esant dažnam, dinamiškam turinio keitimuisi.

Atkurti, tai yra registrui spręsti, koks jo atminties dydis yra pasenęs, ir jei ir kada iškleisti turinio siuntimą. Pavyzdžiui, registras gali įdiegti būseną, kuri dinamiškai išskleidžia turinio siuntimą sekai, kai tik užklausa veikia sekas. Pavyzdžiui, jei užklausa interpretuoja nuorodos turinį kaip identifikatorių vardo tarpe (kaip LDAP) ir pažymi tik sekų vardo tarpe, tik šios sekos gali būti laikomos iš naujo parsiųsti.

Iš naujo parsiųsti, programai (klientui) prašant**.** Iki šiol, registras gali spėti, koks programos siuntimo supratimas gali būti, kai tuo pačiu metu prižiūrint jo lemiamą įtaką. Programa dar neturi kelio nurodyti savo reikšmę ekrano vaizde registrui. Mes siūlome adresuoti šią problemą su paprasta *siuntimas iš naujo programai klausiant* strategija po registro įtakos kontrole. Strategija išnaudoja XQuery kalbos raiškų ir dinamišką duomenų papildymą. Programos užklausa gali pati tikrinti laiko žymės reikšmes ir sekų rinkinį. Jei užklausa nusprendžia, kad duotos sekos yra pasenusios (if type=”networkLoad” AND TC<current Time() –10), tai kuria XQuery dokumentą (URL nuorodos turinį) funkciją su

atitinkančiu nuorodos turiniu tam, kad išskleisti ir gauti turinio perduotą siuntimą, kuris tada vekia, bet kokiu norimu keliu.

Šis mechanizmas yra reikalingas registrui atspėti koks programos siuntimo supratimas gali būti. Tai taip pat suprantama, kad registras nereikalauja kompleksinės logikos užklausos nagrinėjimui, analizei, suskaidymui, sujungimui it t.t. Be to išskleisti užklausos siuntimo rezultatai gali būti pakartotinai panaudoti užklausų sekai. Kadangi užklausa vykdoma registre, registras gali įdiegti dokumento funkciją, kuri ne tik išskleis ir grąžins esamą turinį, bet kaip veiksmas taip pat atnaujina sekų atsargos atmintį savo duomenų bazėje. Registras išlaiko savo įtaką, tam tikra prasme, kad gali naudoti įtakos būseną, galbūt visiškai ignoruoti užklausos persiuntimo atkūrimus ir grąžinti seno turinio vietoje. Persiuntimo programai prašant strategija yra paprasta ir kontroliuojama.

4. TINKLO PASLAUGOS SURADIMO ARCHITEKTŪRA

Turint detaliai apibrėžtus visus registro aspektus, mes pereiname prie tinklo paslaugos lygmens apibrėžimo, kuris padeda Interneto programinės įrangos sąveikavimui. Toks lygmens Interneto vaizdas ekrane kaip platus paslaugų rinkinys su papildomu, gerai apibrėžtu sąsajų rinkiniu. Tinklo paslauga susideda iš sąsajų rinkinio su susijusiomis operacijomis. Tokios operacijos gali būti susietos su vienu ar keliais tinklo protokolais ir pabaigos taškais. Sąsajų, operacijų ir susiejimų su tinklo protokolais ir pabaigos taškais apibrėžimas yra duotas kaip paslaugos aprašymas. Atradimo struktūra apibrėžia tinkamas paslaugas, sąsajas, operacijas ir protokolo susiejimą su suradimu. Pagrindinė problema yra:

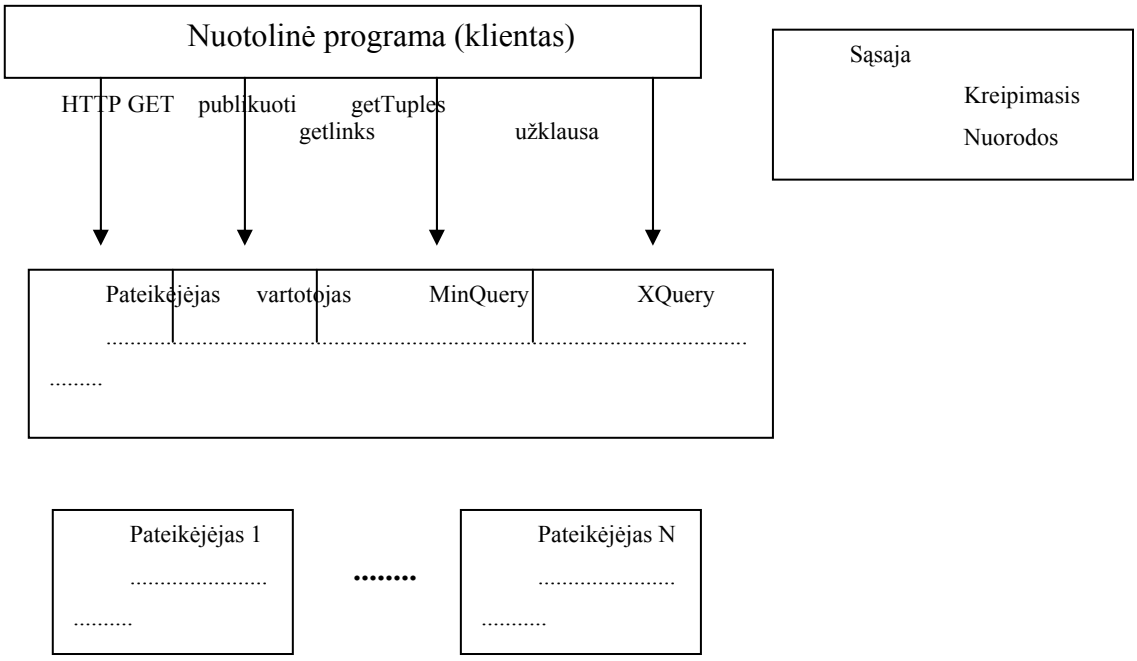
- *Ar galime mes apibrėžti atradimo struktūrą, kuri padeda sąveikavimui, pripažįsta pramonės standartus, ir atidaro, modulinį, lankstų, unifikuotą ir paprastą dar vis efektyvų atradimą?*

Mes siūlome ir specifikuojame tokią struktūrą, vadinamą Paslaugos Atradimo Archiektūra (WSDA). WSDA palaiko dingusias sąvokas, sąsajas ir protokolus po vienu skėčiu. Tai specifikuoja bendravimo primityvų mažą rinkinį (blokus) atradimui. Šie primityvai atstato paslaugos identifikavimą, paslaugos paieškos aprašymą, duomenų publikavimą taip pat kaip minimalų ir efektyvų užklausos prižiūrėjimą. Individualūs primityvai gali būti kombinuojami ir priderinami kartu, specialių programų (klienčių) ir paslaugų, sukurti platų rėžį elgsenų ir sinergijų.

Sąsaja	**Operacija**	**Atsakomybė**
Pakeitėjas	XML getServiceDescription ()	Leidžia programai ieškoti esamos paslaugos aprašymo ir iš čia įsikelti visas paslaugos galimymes.
Vartotojas	(TS4, TS5) publikuoti (XML tupleset)	Turinio tiekėjas gali publikuoti dinamišką žymeklį vadinamą turinio nuoroda, kuris igalina vartotoją (registrą) ieškoti esamo turinio. Nebūtinai, turinio tiekėjas gali taip pat įtraukti esamo turinio kopiją kaip publikacijos dalį. Kiekviena įvesties seka turi turinio nuorodą, tipą, kontekstą, keturias laiko žymes ir metaduomenis ir turinį
MinQuery	XML getTuples() XML getLinks()	Teikia paprasčiausią, galimą užklausos priežiūrą (“*žymėti*

		visus – stilius"). Get tuples operacija grąžina visų galimų tuples pilną rinkinį. Minimali getLinks operacija yra identiška, bet pakeičia tuščią eilutę paslėpto atmintyje turinio.
XQuery	XML užklausa (XQuery užklausa)	Teikia efektyvią XQuery priežiūrą. Vykdo XQuery per prieinamą sekų rinkinį. Kadangi ne vien turinys, bet taip pat nuorodos turinys, kontekstas, tipas, laiko žymės, metaduomenys ir t.t. yra dalis sekų, užklausa gali taip pat žymėti šią informaciją.

1. Lentelė WSDA sąsajos ir jų atitinkamos operacijos.



18. Paveikslas. Programos sąveika su WSDA sąsajomis

4.1 Sąsajos

Keturios WSDA sąsajos ir jų atitinkamos operacijos yra reziumuotos 1. Lentelėje. 8 Paveikslas vaizduoja sąveiką programos su šių sąsajų įdiegimais. Leiskite aptarti sąsajas kiek plačiau.

4.1.1 Pateikėjėjas

Pateikėjėjo sąsaja leidžia programai ieškoti esamos (atnaujinti) paslaugos aprašymo. Aiškiau, programos iš niekur įgalina ieškoti paslaugos esamo aprašymo (objektas į vietinės apsaugos būseną). Vadinasi, paslauga reikalauja pristatyti programai (klientui) (padaryti prieinamą), priemonė ieškoti paslaugos aprašymo. Įgalinti programas užklausti visuotinio tinklo kontekste, reikalinga keletas identifikatorių paslaugai. Toliau, paieškos aprašymo mechanizmas reikalauja būti susietas su kiekvienu tokiu identifikatoriumi. Kartu šie yra savikėlos šifras (ar programėlė) visoms paslaugos galimybėms.

Svarbiausia, kad identifikatorius ir paieškos mechanizmai galėtų sekti priimtiną konvenciją, pasiūlydami naudoti atitinkamą URI. Tačiau praktikoje pagrindinis mechanizmas toks kaip paslaugos atradimas gali tik tikėtis džiaugtis plačiu priėmimu, pritaikymu ir visur būvimu, jei paslaugų palikimo papildymas yra lengvai sudaromas. Paslaugos atradimo įžanga kaip nauja ir pridedama, papildoma paslaugos galimybė reikalauja kaip mažo šanso, galimybės į plačios bazės vertingo paslaugų palikimo, verčiau nekeisti ir viskas. Gali būti įmanoma įdiegti su atradimu susijusį funkcionalumą nekeičiant core paslaugos. Toliau, paieškos mechanizmui rekėtų turėti labai ribotą sąsają, kad padėti lengvam įdiegimui ir būti tiek paprastas kiek įmanomas.

Tuo būdu, daugumai, mes apibrėžiame, kad identifikatorius gali būti bet koks URI. Tačiau, priežiūrai daugiau reikalavimų, identifikatorius turi būti bendrai pasirenkamas URL [24], ir pasirenkamas paieškos mechanizmas HTTP(S). Taigi, mes apibrėžiame, kad HTTP(S) užklausa identifikatoriui turi grąžinti esamos paslaugos aprašymą (objektas į vietinės apsaugos būseną). Kitais žodžiais, naudojama paprasta hypernuoroda. Likusiame skyriuje mes naudosime sąvoką *paslaugos nuoroda* tokiam identifikatoriui įgalinančiam paslaugos paieškos aprašymą. Kaip ir WWW pasauliniame tinkle, paslaugos nuorodos (ir turinio nuorodos, žiūrėti žemiau) gali laisvai būti pasirinktos kaip ilgai jos derinamos į URI ir HTTP URL specifikacijai [24].

Kadangi paslaugos aprašymai turi aprašyti paslaugos esminius dalykus, rekomenduojama, kad paslaugos nuorodos sąvoka būtų papildoma aprašymo dalis. Rezultate, paslaugos aprašymai gali būti paieškomi per Pateikėjėjo sąsają, kuri apibrėžia operaciją

getServiceDescription() šiam tikslui. Operacija yra identiška paslaugos paieškos aprašymui ir iš čia susieta į (kreiptasi per) HTTP(S) GET užklausą į duotą paslaugos nuorodą. Pridedamo protokolo susiejimai gali būti apibrėžti kaip reikalinga.

4.1.2 Vartotojas

Vartotojo sąsaja leidžia turinio tiekėjui publikuoti sekų rinkinį vartotojui. Publikavimo operacija turi parašą (TS4, TS5) publikacija (XML tupleset). Detaliau, žiūrėti 2.2 skyrių.

MinQuery. MinQuery sąsaja teikia paprasčiausią, įmanomą užklausos atlikimą ("žymėti visus"-stilius). GetTuples() ir getLink() operacijos grąžina sekas įtraukiant ir išmetant atitinkamai paslėptos atminties turinį. Detaliau, žiūrėkite 2.3 skyrių.

Pažengęs užklausos atlikimas gali būti išreikštas į minimalios užklausos galimybių viršūnę. Tokios aukšto lygio galimybės suvokiamai nepriklauso vartotojui ir minimaliai užklausos sąsajai, kuri yra liečiama su turinio nuorodos pagrindinės galimybės sudarymu (paslaugos nuoroda) pasiekiamu programoms (klientams).

Galimybė	**XQuery**	**XPath**	**SQL**	**LDAP**
Paprasta, vidutinė, kompleksinė užklausos apie sekų rinkinį	taip	ne	taip	Ne
Užklausa apie struktūrizuotus ir pusiau struktūrizuotus duomenis	taip	taip	ne	taip
Užklausa apie nevienalyčius duomenis	taip	taip	ne	taip
Užklausa apie XML duomenų modelį	taip	taip	ne	ne
Navigacija per hierarhinias duomenų struktūras (kelio reikšmės)	taip	taip	ne	Tikslus tinka vienas
Jungimai (kombinuojami sudėtiniai duomenų šaltiniai į vienintelį rezultatą)	taip	ne	taip	ne
Dinamiškas duomenų papildymas iš sudėtinių, nevienalyčių šaltinių tokių kaip duomenų bazės, dokumentai ir nuotolinės paslaugos	taip	taip	ne	ne
Duomenų restruktūrizavimo raštai (PAŽYMĖTI IŠ KUR SQL)	taip	ne	taip	ne
Kartojimas per rinkinius (uždaram)	taip	ne	taip	ne
Įdėtis keletos reikšmių rūšių su pilna dauguma	taip	ne	ne	ne
Kintamųjų susiejimas ir naujų struktūrų kūrimas iš susietų kintamųjų (LET uždaras)	taip	ne	taip	ne
Kurybinės užklausos	taip	ne	ne	ne
Sąliginės reikšmės (IF .... THEN ... ELSE)	taip	ne	taip	ne
Aritmetika, palyginimai, loginis ir reikšmių rinkinys	Taip, visi	taip	Taip, visi	Log. Eilutė
Operacijos duomenų tipuose iš rinkimo sistemos	taip	ne	taip	ne
Kiekybinės reikšmės (KELETAS, KIEKVIENAS uždaras)	taip	ne	taip	ne
Standartinės funkcijos rūšiavimui, eiliavimui, matematinė, sujungimo	taip	ne	taip	ne

Vartotojo atpažinimo funkcijos	taip	ne	taip	ne
Reguliarus reikšmės atitikimas	taip	taip	ne	ne
Trumpos ir lengvos suprasti užklausos	taip	taip	taip	taip

2. Lentelė. XQuery, XPath, SQL ir LDAP kalbų galimybių palyginimas.

Kaip analogija, aptarkime susijusias, bet aiškias žiniatinklio hyper-nuorodos ir žiniatinklio paieškos sąvokas: žiniatinklio hyper-nuoroda yra pagrindinė galimybė be kurios niekas žiniatinklyje nedirba. Keletas skirtingų žiniatinklio paieškos variklių naudojančių įvairovę paieškos sąsajų ir strategijų gali ir yra padėti ant žiniatinklio nuorodos. XQuery atlikimo rūšis, kurią siūlome žemiau yra, žinoma, ne viena galima ir naudojama. Atrodo neprotinga priimti, kad vienintelis visuotinio standarto užklausos mechanizmas patenkins esamus ir būsimus plačios srities bendrijų poreikius. Sudėtinis toks mechanizmas turėtų galėti koegzistuoti. Vadinasi, vartotojas ir užklausos sąsajos yra apgalvotai atskirtos ir saugomos kaip įmanoma minimaliai, ir yra pateiktas papildomas sąsajos tipas XQuery atsakymui.

XQuery. Didesnis skaičius ir turinio nevienalytiškumas ir programos tampa svarbesnėmis pagrindinio tikslo užklausa galimybėmis. Realistinė, visur esanti paslauga ir šaltinio atradimas laikosi ir žlunga su sugebėjimu reikšti užklausas turtinga pagrindinio tikslo - užklausa kalba [6]. Užklausos kalba naudojama paslaugai ir šaltinio atradimui turi patenkinti reikalavimus, esančius II Lentelėje (reikšmės mažėjančia tvarka). Kaip matome iš lentelės, LDAP, SQL ir XPath nepatenkina esminių reikalavimų, tada XQuery kalba patenkina visus reikalavimus. XQuery sąsajos operacija XML užklausa (XQuery užklausa) plačiau aptarta 2.3 skyriuje.

4.2 Tinklo protokolų susiejimai ir paslaugos

WSDA sąsajų operacijos susietos į numatytą transportavimo protokolą. XQuery sąsaja susieta į Dumenų bazės protokolą (PDP) (žiūr.5 Skyrių). PDP atlieka duomenų bazės užklausas plačiai sričiai duomenų bazių struktūrų ir reakcijų modelius tokius, kad tikslūs visur esančio Interneto atradimo infrastruktūros išreiškiant mąsteliškumu, efektyvumu, sąveika, praplėtimu ir patikimumu gali būti sutiktos. Ypatingai, tai leidžia gerą sutapimą, retą vėlavimą, kanalų sudarymą, taip pat kaip ankstyvią ir/arba dalinę rezultato rinkinio paiešką, tiek išskleidimo ir tiek stūmimo būsenoje. Visoms kitoms operacijoms ir argumentams mes pasisaviname paprastumui HTTP(S) GET ir POST kaip transportą, ir XML esančius parametrus. Papildomas protokolo susiejimas gali būti apibrėžtas kaip reikalingas.

Mes apibrėžiame dvi registro paslaugų pavyzdžio rūšis: Vadinamasis hypermin registras gali (mažiausiai) prižiūrėti tris sąsajas *Pateikėjėjas*, *Vartotojas* ir *MinQuery* (neįtraukiant XQuery priežiūros). Hyper registras gali prižiūrėti šias sąsajas plius XQuery sąsają. Vedant kitą kelią, bet kokiai paslaugai įvyksta priežiūra, tarp kitų, atitinkamos sąsajos apibrėžia kaip hypermin registrą ar hyper registrą. Kaip paprastai, sąsajos gali turėti pabaigos taškus, kurie yra pavadinti atskiru sudėtiniu rodiniu, ar jie gali būti skleisti per sudėtinius hostingus ar valdymo domenus.

Tai yra, jokiu būdu, reikalavimas, kuris tik skirtas hyper registro paslaugų ir hypermin registro paslaugų gali įdiegti WSDA sąsajas. Bet kokia savavališka paslauga gali nuspręsti pasiūlyti ir įdiegti nieko, keletą ar visas iš šių keturių sąsajų. Pavyzdžiui, užduoties tvarkyklė gali nuspręsti įdiegti, tarp kitų, MinQuery sąsają, nurodyti paprastą būdą atrasti metaduomenų sekas susietas į esamą užduoties eilės būseną ir prižiūrimą Paslaugos Kokybę. Tvarkyklė gali nenorėti įdiegti vartotojo sąsajos, kadangi jo metaduomenų sekos yra griežtai tik skaitomos. Toliau, ji gali nenorėti įdiegti XQuery sąsajos, kadangi laikoma virš sunaikinimo jos tikslams. Net priešingai tokia tvarkyklės paslauga neapibrėžia kaip hypermin ar hyper registras, aiškiai siūlomas naudingas, pridėtas kintamasis. Kiti paslaugų įdiegimo pavyzdžiai WSDA sąsajų dalis naudojama tokia kaip staigi, nauja paslauga ar monitoriaus blokinys. Šios paslaugos gali nuspręsti įdiegti Vartotojo sąsają, iškviesti išorinius šaltinius duomenų papildymui, bet jie gali nerasti naudingos pasiūlyti ar įdiegti bet kokios užklausos sąsajos.

Labiau pretenzingame scenarijuje užduoties tvarkyklės pavyzdys gali nuspręsti publikuoti sekų rinkinį taip pat į jau egzistuojančią nuotolinio hyper registro paslaugą. Nurodyti klientams (programoms) kaip gauti XQuery valdymo galimybę, tvarkyklė gali paprastai nukopijuoti nuotolinio hyper registro paslaugos XQuery sąsajos aprašymą ir skelbti tai kaip jos nuosavą sąsają įtraukiant tai į jos nuosavos paslaugos aprašymą. Ši virtualizacijos rūšis nėra apgaulė, bet savybė su svarbiu, praktišku kintamuoju, kadangi tai leidžia minimalias pastangas tvarkyklės dalies įdiegimui ir priežiūrai.

Pasirenkamai, tvarkyklė gali įtraukti į savo vietą sekų rinkinį (gaunamą per getlinks() operaciją), kuris kreipiasi į nuotolinės hyper registro paslaugos aprašymą. Sąsajos nurodytas kintamasis sekų konteksto savybei naudojamas, kaip žemiau išdėstyta :


```
<tuple link="https://registry.cern.ch/getServiceDescription"
    type="service" ctx="x-ireferral://cern.ch/XQuery-1.0"
    TS1="30" TC="0" TS2="40" TS3="50">
</tuple>
```

4.3 Tinklo paslaugų suradimo architektūros savybės

WSDA svarbiausios savybės :

4.3.1 Standartų integracija

WSDA apima ir papildo vientisus ir plačiai pripažįstamus pramonės standartus tokius kaip XML, XML Schemą [28], Paprasto Objekto Prieinamumo Protokolas (SOAP) [5], Žiniatinklio Paslaugos Aprašymo Kalba (WSDL) [4] ir XQuery [23]. Tai leidžia papildymą pagalbinių standartų tokių kaip Žiniatinklio Paslaugos Tikrinimo Kalba (WSIL) [26].

4.3.2 Sąveikavimas

WSDA paaukština žiniatinklio paslaugos sąveikos lygį iki Interneto programinės įrangos viršaus, kadangi tai apibrėžia tinkamas paslaugas, sąsajas, operacijas ir protokolo susiejimą. WSDA neįveda naujų Interneti standartų. Šiek tiek, tai protingai kombinuoja esančią sąveiką įrodytą atidaryto Interneto standartų tokių kaip HTTP(S), URI [24], MIME [27], XML, XML Schema [28] ir BEEP [31].

4.3.3 Moduliškumas

WSDA yra modulinė, kadangi ji apibrėžia sudėtinio-tikslo bendravimo primityvų (blokų) atradimui rinkinį. Šie primityvai atstato paslaugos identifikaciją, paslaugos paieškos aprašymą, taip pat kaip monimalią ir efektyvią užkausos atlikimą. Atsakomybė, apibrėžimas ir evoliucija bet kurio duoto primityvo yra ypatingas ir nepriklausomas visiems kitiems primityvams.

4.3.4 Įdiegimo ir naudojimo patogumas

Kiekvienas bendravimo primityvas yra apgalvotai sukurtas išvengti bet kokio nereikalingo sudėtingumo. Kūrimo pricipas yra *padaryti paprastus ir bendrus dalykus lengvai, ir galimus efektyvius dalykus*. Kitais žodžiais, sprendimai yra atmesti tai dar reiškia aprūpinimą efektyviomis galimybėmis, kad net paprastos problemos yra komplikuotai sprendžiamos. Pavyzdžiui paslaugos aprašymo paieška numatyta paprastame HTTP(S) GET. Dar, mes neišbraukiame ir leidžiame identifikavimo ir paieškos mechanizmus tokius kaip pasiūlyti UDDI (Universalus Aprašymas, Atradimas ir Papildymas) [8], RDBMS ar Java RMI registrai (per sekos metaduomenis, nurodytus WSIL [26]). Toliau, turinio seka duota XML”e, bet yra įmanomas taip pat papildomas MIME [27] turinio naudojimas (dvejatainiai vaizdai, failai, MS-Word dokumentai. Kitas pavyzdys, minimalios užklausos sąsaja faktiškai

nereikalauja idiegimo pastangų programos ir serverio dalyje. Kur reikalinga, Xquery atlikimas gali, bet nereikalauja, būti įdiegtas ar naudojamas.

4.3.5 Aiškumas ir lankstumas

WSDA yra aiški ir lanksti, nes kiekvienas primityvas gali būti naudojamas, įdiegtas, papildytas ir padidintas įvairias būdais. Pavyzdžiui, paslaugos sąsajos gali turėti pabaigos taškus, paplitusius aplink sudėtinius hostus ir valdymo domenus. Tačiau, nieko nėra, kas priverčia visas sąsajas būti tame pačiame hoste ar įdiegtas programoje. Iš tikrųjų, tai yra dažnai naturalaus išskleidimo scenarijus. Toliau, net yra duoti numatyto tinklo protokolo susiejimai, papildomi susiejimai gali būti apibrėžti kaip rekalingi. Pavyzdžiui, vartotojo sąsajos įdiegimas gali prisirišti prie HTTP(S), SOAP/BEEP [32], FTP ar Java RMI. Grąžintas užklausos sekos rinkinys gali būti prižiūrimas atsižvelgiant į atminties ryšio būsenos įvarovę, randantis statinėje elgsenoje. Vartotojas gali naudoti papildymo veiksmą ant sekos publikavimo. Pavyzdžiui, galima interpretuoti seką iš schemos kaip komandą ar aktyvų pranešimą, sukeliant sekos transformavimą ir/ar siuntimą kitiems vartotojams**.**

Lankstumui, paslaugą prižiūrintis WSDA sekos rinkinys išskleidžiamas bet kokiuo keliu. Pavyzdžiui, duomenų bazė gali būti laikome XML faile, tame pačiame formate kaip getTuples užklausos operacijos grąžinta. Tačiau, seka gali būti perskaičiuota ar laikoma reliacinėje duomenų bazėje.

4.3.6 Galingumas

WSDA yra galinga, nes jos primityvai gali buti programų ir paslaugų sukombinuoti ir sudėti kartu, kad duoti platų elgsenų rėžį. Kiekvienas primityvas yra riboto kintamojo. Teisingas paprasto bendravimo primityvų kintamasis slypi sugebėjime generuoti galingas, pasirodančias sinergijas. Pavyzdžiui, WSDA primitivų kombinacija įgalina kurti paslaugas kopijai, vardo raiškai, paplitusioms varžytinėms, greitoms naujienoms ir pranešinėjimams, programinei įrangai ir blokinio konfiguravimo valdymui, sertifikatui ir saugykloms, taip pat kaip Tinklelio stebėjimo įrankiams. Kitas pavyzdys, vartotojas ir užklausos sąsajos gali būti sukombinuotos įdiegti Peer-to-Peer (P2P) duomenų bazės tinklą paslaugos atradimui (žiūr.4 skyrių). Čia, tinklo nodas yra paslauga, kuri mažiausiai demonstruoja sąsajas publikacijai ir P2P užklausoms.

4.3.7 Vienodumas

WSDA yra sujungta, kadangi ji įtraukia masyvą skirtingų sąvokų, sąsajų ir protokolų. Tai leidžia koegzistuoti ir būti papildytoms sudėtinių, paplitusių sistemų sąvokoms ir įdiegimams. Programos (klientai) gali dinamiškai adaptuoti savo elgseną paslaugos savistabos galimybėse. Aiškiai, nėra sprendimo, kuris yra optimalus esant nevienalytėse sistemose, tokiose kaip Duomenų Tinkleliai, elektroninės prekybos vietos ir greitos interneto naujienos ir pranešimų paslaugos. Savistabos ir adaptacijos galimybės vis daugiau daro nereikalingu perduoti duotai problemai vienintelį sprendimą, tuo būdu įgalinant bendradarbiaujančių sistemų sujungimą.

4.3.8 Nesuardomumas

WSDA yra nesuardoma, nes siūlo sąsajas, bet neperduoda, kad kiekviena paslauga pasaulyje privalo laikytis prie standartinių sąsajų rinkinio.

5. TINKLINĖS DUOMENŲ BAZĖS

Didelėje paskirstytoj sistemoje informacijos sekų rinkinys yra padalintas per daugelį nodų, dėl priežasties įtraukiant autonomiją, mąstelio keičiamumą, prieinamumą, įvykdymą ir apsaugą. Nėra aišku kaip įgalinti galingą užklausos priežiūrą ir bendrą funkcionalumą tai vykdyti sistemoje kaip vieną visumą, verčiau duotoje jos dalyje. Toliau, nėra aišku kaip leisti ieškoti atsiųstų rezultatų, sudarant sąlygas dinamiškam turiniui. Pasirodo, kad Peer-to-Peer (P2P) duomenų bazės tinklas gali būti tinkamas prižiūrėti dinamišką duomenų bazės paiešką, pavyzdžiui paslaugos atradimui.

Pagrindinės problemos tada yra:

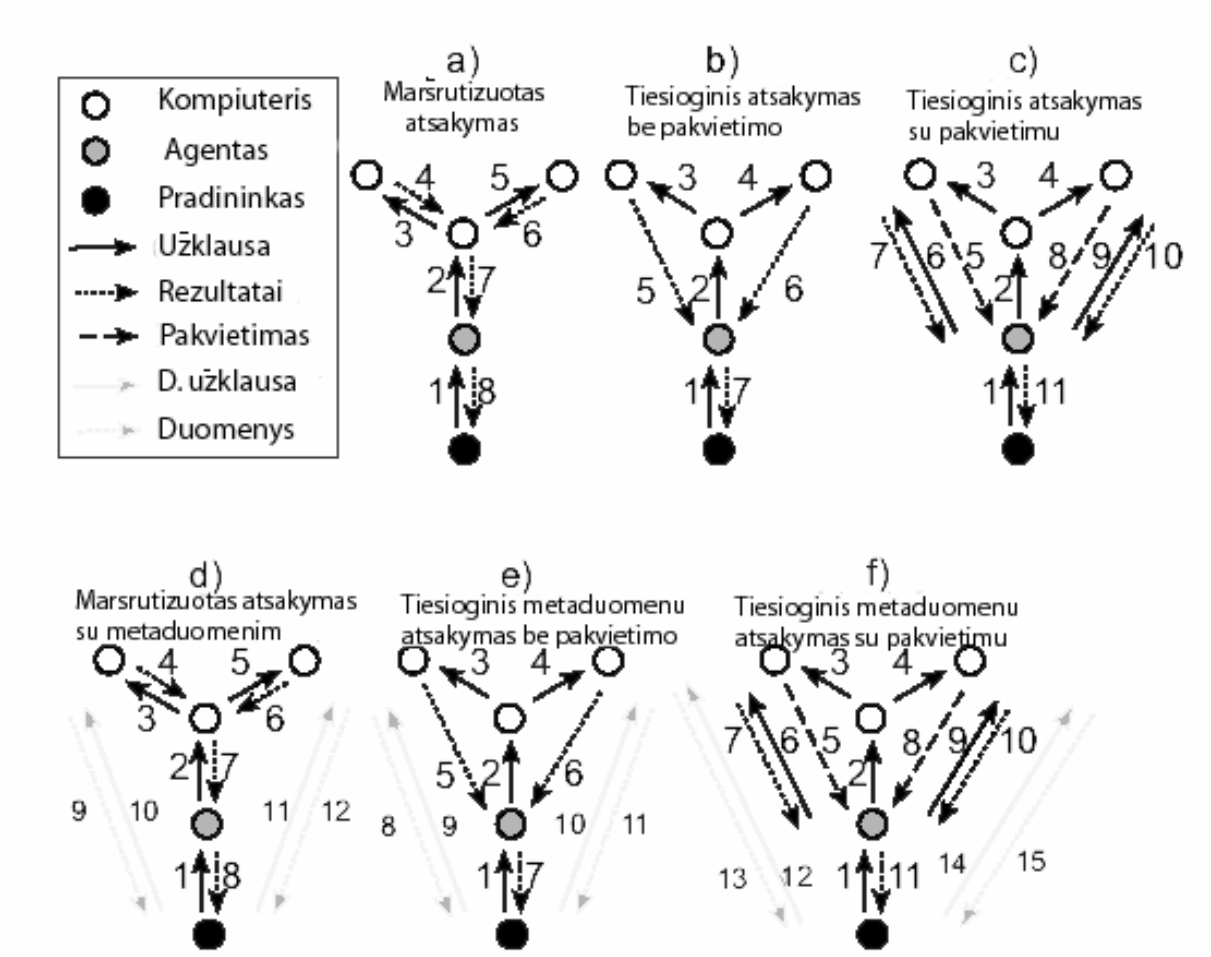
Kokia yra detalesnė struktūra ir sukūrimo pasirinkimai P2P duomenų bazės paieškai paslaugos atradimo kontekste? Kokie atsakymo modeliai gali būti naudojami grąžinti priderintus užklausos rezultatus? Kaip turi būti susistemintas P2P užklausos procesorius? Kokie užklausos tipai gali būti atsakyti (tinkamai) P2P tinklo? Kokie užklausos tipai turi galimybę nedelsiant siųsti rezultatus? Kaip rezultatų maksimumas gali būti pateiktas patikimai be vartotojo norimų laiko rėmų, net jei užklausos tipas neatlieka siuntimo? Kaip ciklai gali būti aptikti patikimai naudojant pertraukas? Kaip užklausos apimtis gali būti naudojama topologijos charakteristikoms atsakant į užklausą?

Ar galime mes surasti sujungtą P2P duomenų bazės struktūrą užklausos priežiūrai plačioje sistemoje, jungiančioje daug valdymo domenų? Aiškiau, ar mes galime surasti struktūrą, kuri yra sujungta, tam tikra prasme, leidžia išreikšti atradimo programas plačiai grupei duomenų tipų, nodų topologijų, užklausos kalboms, užklausos atsakymo būdams, žymėjimo būsenoms, siuntimo charakteristikoms, pertraukai ir kitoms pasirinkimo apimtims?

Šiame skyriuje žengiame žingsnį suvienodinant duomenų bazės valdymo sistemos ir P2P laukus, kurie ligi šiol gavo žymų, bet atskirą dėmesį. Mes praplėtėme duomenų bazės sąvokas ir praktiką – atstatyti P2P paieškai. Panašiai taip pat, mes praplėtėme P2P sąvokas ir praktiką – prižiūrėti galingas užklausos kalbas tokias kaip XQuery [23] ir SQL [33]. Todėl mes siūlome (UPDF) *Unified Peer-to-Peer Database Framework* ir atitinkantį (PDP) *Peer Database Protocol* užklausos priežiūrai plačiai paplitusiose sistemose jungiančiose daug valdymo domenų. Jie yra suvienodinti, ta prasme, kad leidžia išreikšti atradimo programas plačioms grupėms duomenų tipų, nodų topologijoms, užklausos kalboms, užklausos atsakymo būdams, žymėjimui, siuntimo charakteristikoms, pertraukoms ir kitoms pasirinkimo apimtims.

5.1 Nukreiptas, tiesioginis, metaduomenų atsakymai

Kai bet koks sukūrėjas tikisi ieškoti P2P tinklo su keleta užklausų, jis siunčia užklausą nodui tarpininkui. Nodas pritaiko užklausą savo vietinei duomenų bazei ir grąžina atitinkamus rezultatus; jis taip pat siunčia užklausą žymėti nodus. Šie grąžina savo vietinės užklausos rezultatus; jie taip pat siunčia užklausą žymėti nodus. Mes siūlome išskirti keturias technikas grąžinti atitinkamos užklausos rezultatus sukūrėjui: Nukreiptas atsakymas, tiesioginis



19 pav. P2P atsakymo būdai

atsakymas, nukreiptas metaduomenų atsakymas ir tiesioginis metaduomenų atsakymas, kaip parodyta 9 Pav. Leiskite panagrinėti pagridines prasmes Gnutell'os naudojimo atveju. Tipiška Gnutella užklausa tokia kaip *"Like a virgin"* surastų keletus šimtų failų, dauguma iš jų nurodo į labai panašų muzikinį failą. Ne visi atitinkantys failai yra identiški, nes egzistuoja susijusios dainos (remiksai, gyvi įrašai) ir dainos versijos (su skirtingomis pavyzdžių atrinkimo kategirijomis). Muzikinis failas turi mažiausiai keletos megabaitų dydį. Tūkstantis konkuruojančių vartotojų pateikia užklausas į Gnutella tinklą.

5.1.1 Nukreiptas atsakymas

(9-a Paveikslas). Resultatai yra perduodami atgal į sukūrėją tais pačiais keliais, kuriais užklausa ėjo tolyn. Kiekvienas (pasyvus) nodas gražina savo (aktyviai) programai (klientui) ne tik jo vietinius rezultatus, bet taip pat visus nuotolinius rezultatus. Nukreipiami pranešimai per loginę P2P nodų tinklo perdangą yra mažiau efektyvūs, nei nukreipiami per fizinį IP ruterio tinklą [34]. Nukreipiant atgal net vienintelis failas kiekvienai užklausai per sudėtinius nodus suvartos didelius mastus visos sistemos dažnių juostos pločio. Kadangi P2P tinklas auga, jis yra fragmentuotas, nes nodai su žemos dažnių juostos jungimais negali išsilaikyti su duomenų srautu [35]. Todėl, nukreipti atsakymai yra netinkmi failo parsiuntimo sistemoms, tokioms kaip Gnutella. Bendrai, ekonomika diktuoja, kad nukreipti atsakymai yra netinkami sistemoms, kurios grąžina daug ir/arba plačių radinių.

5.1.2 Tiesioginis atsakymas su ir be kvietimo

Kad geriau suprasti idėją, mes iš pradžių pristatysime paprasčiausią variantą, kuris yra *Tiesioginis atsakymas Be Kvietimo* (9-b Paveikslas). Radiniai negrąžinami nukreipimu atgal per tarpinius nodus. Kiekvienas (aktyvus) nodas, kuris turi vietinius rezultatus, siunčia juos tiesiogiai į (pasyvų) tarpininką, kuris sukombinuoja ir neša atgal sukūrėjui. Atsakymo duomenų srautas nekeliauja per P2P sistemą.

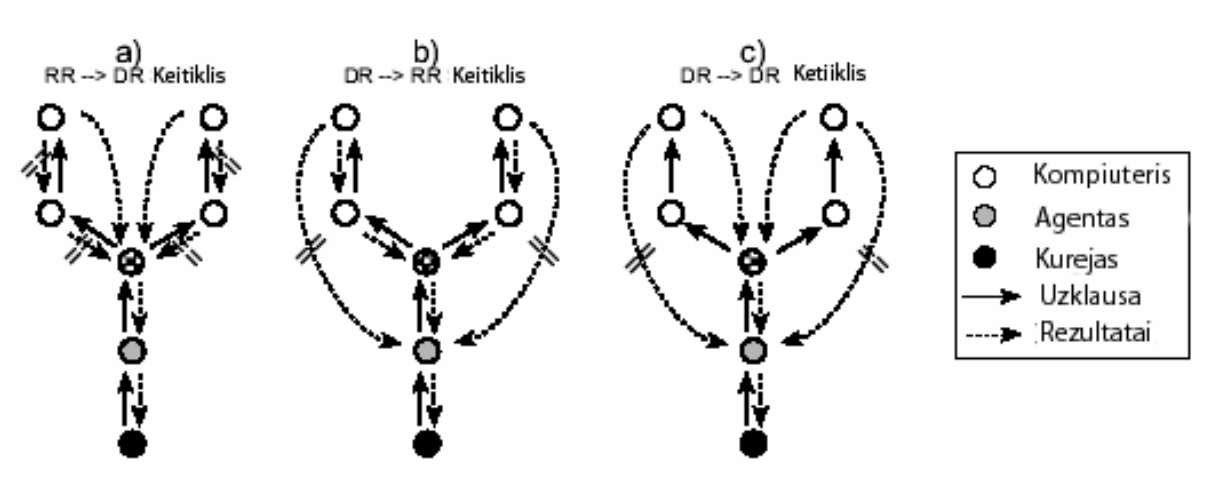
Leiskite papaiškinti šio atvejo esmę. Kaip jau buvo minėta, tipiška Gnutella užklausa tokia kaip *"Like a virgin"* atitinkanti keletus šimtų failų, dauguma iš jų nurodantys į kopijas labai panašios muzikos failo. Gnutell'os vartotojams pakaktų gauti tik mažą dalelę atitinkančių failų. Siunčiant atgal visus šiuos failus nereikėtų suvartoti daug tiesioginio dažnių juostos pločio, labiau ribota Gnutella yra vartotojams su per didelia dažnių juosta. Tačiau pažymėkime, kad Gnutell'os sistema būtų tik minimaliai veikiama vieninteliam vartotojui parsisiunčiant, sakoma, milijoną muzikos failų, nes plačiausia pranešimų srauto dalelė nekeliauja per P2P sistemą pati.

Apskritai, individuali ekonomika diktuoja, kad tiesioginiai atsakymai be kvietimo yra netinkami sistemoms, kurios grąžina daug vienodų ir/arba plačių radinių, kol maža dalelė bus pakankama. Kvietimo variantas (9-c Paveikslas) sušvelnina problemą perstatant kontrolės srautą. Nodai su atitinkamais failais aklai nestumia failų į tarpininką. Vietoj to, jie kviečia tarpininkus inicijuoti parsisiuntimus. Pavyzdžiui, galima filtruoti ir pažymėti dalelę duomenų šaltinių ir failų ir atmesti likusius iš pakvietimų. Primename, kad šiame skyriuje, mes naudojame sąvoką Tiesioginis Atsakymas kaip sinonimą Tiesioginiam Atsakymui Be Kvietimo**.**

5.1.3 Nukreiptas metaduomenų atsakymas ir tiesioginis metaduomenų atsakymas

Čia, sąveika susideda iš dviejų fazių. Pirmoje fazėje, nukreipti ar tirsioginiai atsakymai (9-d,e,f Paveikslai) yra naudojami. Tačiau, nodai negrąžina duomenų radinių atsakant užklausoms, bet tik mažiems duomenų radiniams. Metaduomenys turi pakankamai informacijos įgalinti sukūrėją ieškoti duomenų radinių ir galbūt panaudoti filtrus prieš paiešką. Antroje fazėje, sukūrėjas pažymi metaduomenyse, kuris duomenų radinys yra tinkamas. (aktyvus) kūrėjas tirsiogiai prisijungia prie tinkamo (pasyvaus) duomenų šaltinio ir klausia duomenų radinių. Dar, atsakymo pranešimų srauto dalelė nekaliauja per P2P sistemą.

Nukreiptas Metaduomenų atsakymas yra naudojamas failų parsiuntimo sistemoms yokioms kaip Gnutella. Gnutella negrąžina failų; ji tik grąžina paaiškintą HTTP URLs rinkinį. Kitas pavyzdys yra paslaugos atradimo sistema, kur pirma fazė grąžina paslaugos nuorodų rinkinį vietoj pilnų paslaugos aprašymų. Antroje fazėje, sukūrėjas prisijungia prie dalelės šių paslaugos nuorodų parsiųsti paslaugos aprašymus. Kitas



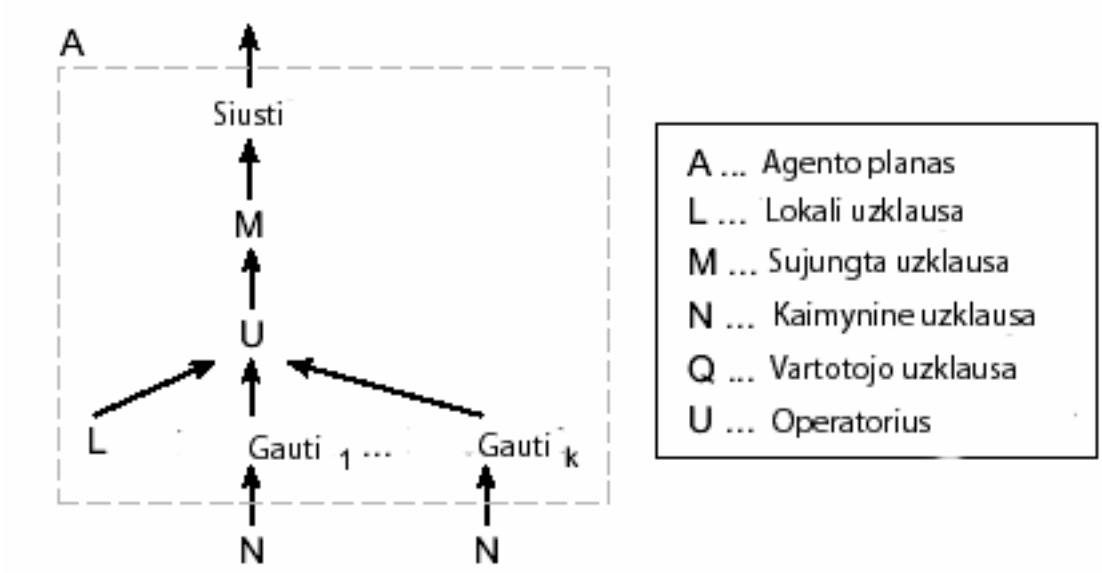
10. Pav. Atsakymo perkėlimų ir prijungimų metodai (nukreiptas atsakymas, tiesioginis atsakymas).

Pavyzdys yra persiuntimo sistema, kur pirma fazė naudoja nukreiptą metaduomenų atsakymą grąžinti nodų rinkinio paslaugos nuorodas turint vietinius, atitinkamus rezultatus (*"'Paklausk šių nodų atsakymo"*). Sekančioje fazėje, sukūrėjas ar agentas prisijungia tiesiogiai prie dalelės šių nodų užklausti ir ieškoti radinių rinkinio. Šis variantas išvengia "kvietimų audros", galimos t*iesioginio atsakymo* atveju. Persiuntimai taip pat yra žinomi kaip *nukreipimai*. Metaduomenų atsakymo metodas su apimties spinduliu nulis gali būti naudojamas įdiegti Domeno Vardo Sistemos (DNS) persiuntimo elgseną.

Smulkesniam palyginimui įvairių atsakymo modelių savybių, žiūrėkite ankstesnius mokymus [36]. Nors iš funkcionalios perspektyvos visi atsakymų būdai yra vienodi, būdas nėra optimalus prie visų aplinkybių. Kyla toks klausimas: į kokį išplėtimą duotas P2P tinklas privalo perduoti atsakymo būdo naudojimą per sistemą. Stebėkite, kad nodai yra savaravkiški ir apibrėžti tik jų sąsaja. Todėl mes siūlome, kad atsakymo būdai gali būti sumaišyti prijungimais ir perstatymais pakeitimuose kaip parodyta 10 Paveiksle.

5.2 Užklausos apdorojimas

Duomenų bazės sistemoje egzistuoja vienintelė, vietinė duomenų bazė ir nulis ar daugiau kaimyninių. Iš užklausos apdorojimo perpektyvos, P2P duomenų bazės sistema turi tas pačias savybes kaip duomenų bazęs sistema pasikartojančioje struktūroje. Iš čia, mes siūlome kurti P2P užklausos variklį kaip pagrindinį užklausos variklį [37,12]. Duota užklausa sieja operatorių skaičių (SELECT, UNION, CONCAT, SORT, JOIN, SEND, RECEIVE, SUM, MAX, IDENTITY), kurie gali ar negali būti neapsaugoti užklausos kalbos lygyje. Pavyzdžiui, operatorius SELECT paima rinkinį ir grąžina naują rinkinį su sekomis . Operatorius UNION skaičiuoja prisijungimus dviejų ar daugiau rinkinių.



11 Pav. Vykdymo plano šablonas

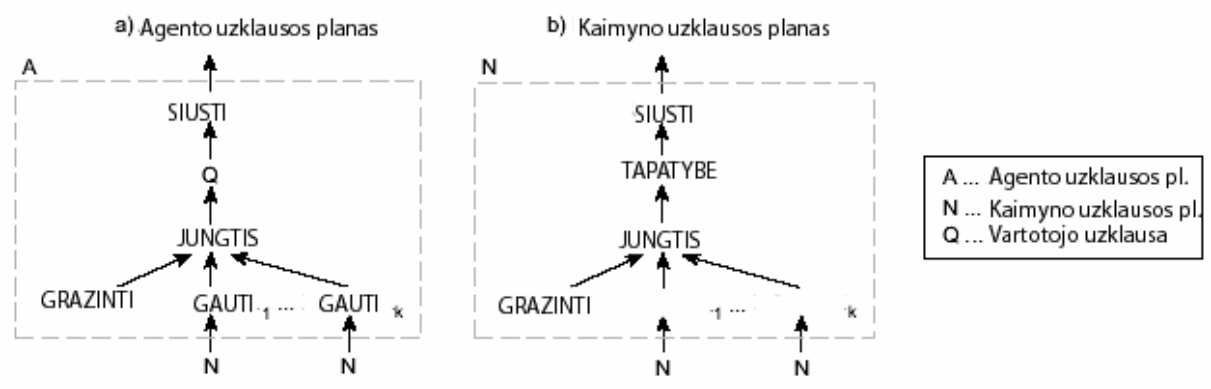
Operatorius CONCAT sujungia dviejų ar daugiau rinkinių elementus į užsakymo sudėtinį rinkinį (nešalinant dublikatų). Operatorius IDENTITY grąžina savo nepakeistą įvesties rinkinį. Operatoriaus semantikai gali būti patenkinti keletos operatorių įdiegimų, naudojant įvairovę algoritmų, kiekvienas su skirtingu šaltinio vartojimu , vykdymo

charakteristikomis.Užklausos optimizatorius pasirenka efektyvų užklausos vykdymo planą, kuris yra sujungtas iš trijų operatorių. Vykdymo plane, pagrindinis operatorius vartoja radinius iš šalutinio operatoriaus.

Užklausos vykdymo plano šablonas. Bet kokia užklausa Q mūsų užklausos modelyje gali būti atsakyta tarpininko su vykdymo plano šablonu, pavaizduotu 11 Paveiksle. Planas pritaiko vietinę užklausą L vietinės duomenų bazės sekos rinkinio fone. Kiekvienas kaimyninis klausiamas grąžinti radinio rinkinį kaimyninei užklausai N. Vietinis ir kaimyninis radinių rinkiniai yra suvienyti į vienintelį radinio rinkinį sujungimo operatoriaus U, kuris turi UNION ar CONCAT formą. Sujungta užklausa M pritaikoma, paima kaip įvestį radinio rinkinį ir grąžina naują radinio rinkinį. Galutinis radinio rinkinys yra siunčiamas programai (klientui), kitam nodui ar sukūrėjui.

5.3 Centralizuotas vykdymo planas

Žiūrėti, kad vietoje bet kokios užklausos bet kokios duomenų bazės rūšies fone, sistema gali būti šioje struktūroje, mes gauname paprastą centralizuotą vykdymo planą, kuris visada atitinka bet kokios užklausos Q semantikus. Planas turi subplanus plano šablone A, svarbiausia atskirti planus tarpininko nodui (12-a Paveikslas) ir kaimyniniams nodams (12-b Paveikslas). XQuery ir SQL atveju parametrai yra sekantys :



12.Paveikslas. Centralizuotas vykdymo planas

Kitais žodžiais, tarpininko planas A duoda visas, neapdorotas sekas iš vietinio ir visų nuotolinių duomenų bazių, sujungia radinio rinkinius ir pritaiko užklausą Q. Kaimyninės laikė perrašytą kaimyninė užklausą N, kuri duoda visas neapdorotas sekas ir grąžina jų junginius. Kaimyninė uklausa N yra suskirstoma (žiūr.žemiau).

Toks pat centraliuotas planas dirba nukreiptam ir tiesioginiam atsakymui, abu su ar be metaduomenų. Prie tiesioginio atsakymo, nodas siunčia užklausą N, bet nemėgina gauti nuotolinių radinio rinkinių (išsiunčiami tušti radinio rinkiniai). Nodas nesiunčia radinio rinkinio savo pirmtakui, bet tiesiogiai grąžina tarpininkui.

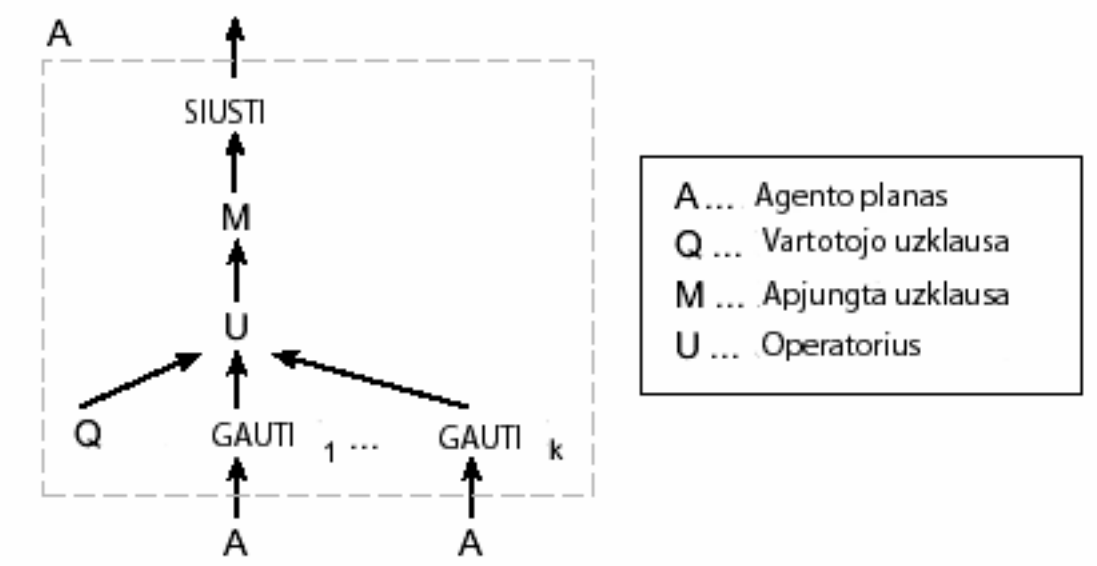
Centralizuotas vykdymo planas gali būti neefektyvus, nes duomenų bazės dideli kiekiai turėtų būti persiųsti tarpininkui prieš pritaikant vartotojo užklausą. Tačiau, kartais tai tik vienas planas, kuris atitinka užklausos semantikus. Tai visada kompleksinės užklausos atvejis.

5.3.1 Pastoviai padalijama užklausa

P2P tinklas gali būti efektyvus atsakant užklausoms, kurios yra pastoviai padalijamos.Užklausa Q yra pastoviai padalijama jei, plano A šablonui egzistuoja užklausa M ir sujungėjas operatorius U atitikti Q užklausos semantikus įgtjant, kad L ir M yra išrinktos kaip L = Q ir N = A. Kitais žodžiais, užklausa yra pastoviai padalinama, jei labai panašus vykdymo planas gali būti pritaikytas kiekvienam nodui P2P topologijoje. Atitinkamas vykdymo planas pavaizduotas 13 Paveiksle.

Užklausos įvestis ir išvestis turi tą pačią formą kaip vietinės užklausos L išvestis. Užklausos apdorojimas gali būti lygiagretizuotas ir paskleistas per visus esančius nodus,

Dabar mes paaiškinsime paprastos, vidutinės ir kompleksinės užklausų apibrėžimą.



13 Pav. Pastoviai padalijamos užklausos vykdymo planas

Paprasta užklausa. Užklausa yra paprasta, jei ji pastoviai padalijama naudojant M = IDENTITY, U = UNION. Pavyzdys yra rasti visas (prieinamas paslaugas.

Vidutinė užklausa. Užklausa yra vidutinė, jei nepaprasta, bet pastoviai padalinama. Pavyzdys yra grąžinti paslaugų katalogo kopijos skaičių.

Kompleksinė užklausa. Užklausa yra kompleksinė, jei nėra pastoviai padalinama. Pavyzdys yra rasti visas (vykdymo paslaugą, rezervavimo paslaugą) poras, kur abi paslaugos būna be to pačio domeno.

Supaprastinimui, šiame skyriuje mes manome, kad vartotojo aiškumas teikia M ir U tolyn su Q užklausa. Jei M ir U nėra teikiamos kaip dalis užklausos bet kokiam duotam nodui, nodas veikia stabiliai manydamas, kad užklausa nepadalinta. Panagrinėkime pavyzdžiui sekančias vidutines Xquery užklausas.

Paslaugų katalogo kopijos skaičiaus grąžinimas. Užklausa skaičiuoja sumą skaičių rinkinio. Sujungėjas CONCAT, rasti paslaugą su viršlaikiu

5.3.2 Kanalai

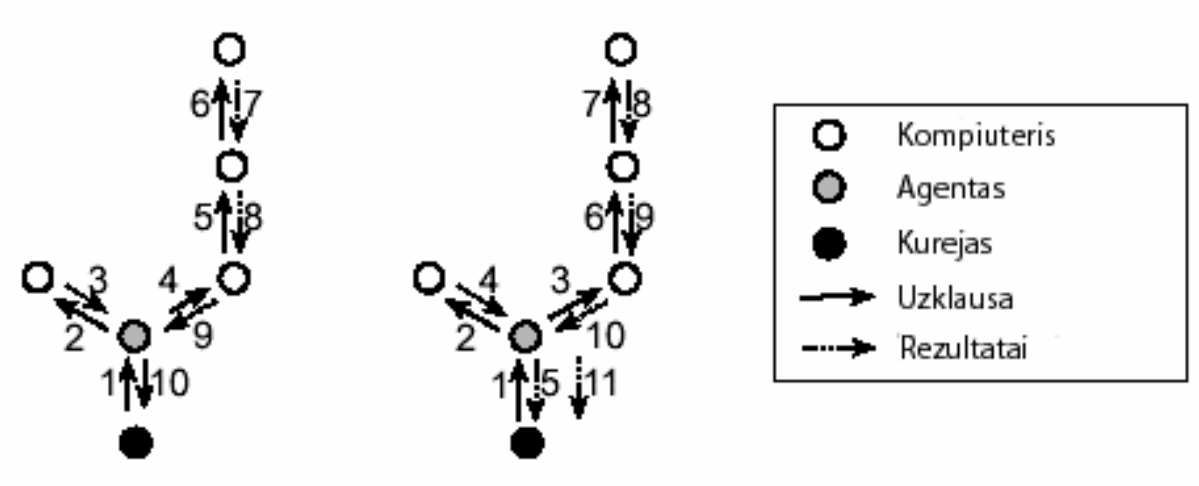
Daugelio programų sėkmė priklauso nuo to kaip greita jos gali pradėti grąžinti rezultatų porcijas, o ne nuo to kaip greitai gali grąžinti visą rezultatą. Dažnai sukūrėjas pageidauja pradėti darbą su mažais pradiniais gražinamų rezultų kiekiais, jei pastarieji ateina patikimai ir greitai. Rezultatai, ateinantys veliau, gali būti apdorojami vėliau arba tiesiog ignoruojami. Toks atvejis dažnai sutinkamas paskirstyto tipo sistemose, kur daug kompiuterių vykdo užklausas, ir vykdymas gali sustoti dėl įvairių priežasčių. Tokia situacija ypač būdinga sistemoms, kurių mazgai yra nutolę vienas nuo kito.

Bet kurio tipo operatoriai turi vientisą pasikartojančią sąsajas, vadinamas open(), next() ir close(). Efektyvumui padidinti, operatorius close() gali grąžinti kelis rezultatus vienu paketu. Semantika yra tokia: „Duok man minimum N ir maksimum M rezultatų“ (Mažiau nei N rezultatų grąžinama, kai paieška būna išsekusi). Pavyzdžiui, SEND ir RECEIVE tinklo komunikacijų operatoriai veikia paketų principu.

Monotoninė tam tikrų operatorių (SELECT, UNION, CONCAT, SEND, RECEIVE) semantika priima tik vieną iš kelių rezultatų. Priešingai, operatoriai SORT, GROUP, MAX ir kai kurie JOIN metodai ir kt. reikalauja, kad operatoriai priimtu visus rezultatus vykdant open() operatorių tam kad galėtų pateikti rezultatus, kai bus iškviestas sekantis next() operatorius. Šitie operatoriai turi matyti visus įeinančius parametrus prieš pateikdami galutinį rezultatą. Tai sukelia svarbias gaišties laiko ir vykdymo implikacijas. Nepriklausomai nuo to, ar šakninis operatorius rodo didelį ar mažą gaišties laiką, kad pristatytų savo užsakovui pirmąjį rezultatą iš rezultatų rinkinio, priklauso nuo to, kokie naudojami operatoriai, kurie savo ruožtu priklauso nuo vykdomos užklausos. Kitaip tariant, vienų užklausų atveju tiekėjas gali iškart tiekti rezultatus kanalais, kitu atveju jis turi palaukti, kol įvykdys užklausą ir galės pateikti pirmąjį rezultatą.

Sakoma, kad užklausa *vykdoma kanalais* kai ji gali grąžinti nors vieną rezultatą, nesulaukius visų parametrų. Priešingu atveju sakoma, kad užkalusa *kanalais nevykdoma*. Paprasto užklauso (pvz. Gnutella tinkle) vykdomos kanalais. Vidutinio sudėtingumo užklausos gali būti vykdomos ir nevykdomos kanalais. Tuo tarpu sudėingos užklausos

paprastai kanalais nevykdomos. 14 pav. Pavaizduotos kanalais vykdomos ir nevykdomos užklausos.



14 pav. Rezultatų grąžinimas paprastai (kairėje) ir kanalais (dešinėje).

5.4 Statinės kilpos pertrauka ir dinaminio nutraukimo pertrauka

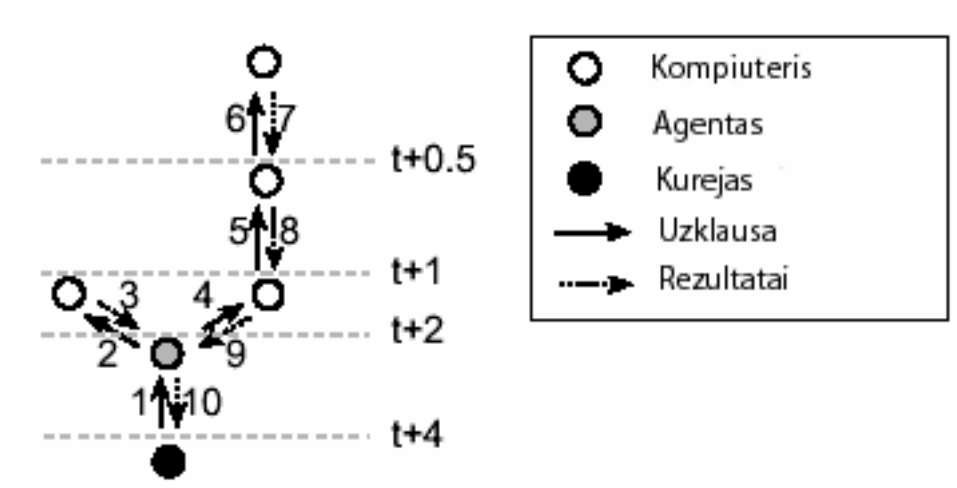
Akivaizdu, kad ateina laikas, kai vartotojas jau būna nesuinteresuotas užklausos rezultatais, nepriklausomai nuo to, ar dar yra naujų rezultatų. Užklausos ir atsakymo tinklo pranešimų srautas po kurio laiko išnyksta. Be to, P2P sistemos skatinamos stengtis apriboti resursų išnaudojimą apsiginant nuo užklasų, kurios tyčia ar ne vykdomos amžinai arba pagamina didžiulius rezultatų kiekius. Kad išspręsti šitas problemas, prie užklausos pridedama dinaminio nutraukimo pertrauka. Nutraukimo pertrauka yra tarsi riba, kurios negalima peržengti. Kartu su užklausa vienas mazgas praneša kitam: „Aš ignoruosiu visus tavo rezultatus, jeigu negausiu jų šiandien iki 12:00:00.“ Tuomet problema tampa užtikrinti, kad grąžinami rezultatai tilptų i šitą laiko tarpą. Statinės pertraukos reikšmė lieka nepakitusi kilpos vykdymo metu, nebent amžinos užklausos aptikimo atveju, kai pertrauka tampa begalinė. Priešingai, siekiam, kad dinaminės pertraukos reikšmė būtų mažinama su kiekvienu šuoliuku. Mazgai, kurie yra labiau nutolę nuo pradininko, gali būti pertraukti anksčiau nei mazgai, kurie yra arčiau pradininko.

5.4.1 Dinaminio nutraukimo pertrauka

Statinio nutraukimo pertrauka yra visiškai netinkama ne-kanalinio rezultatų gavimo atveju, nes tuomet iškyla rimta patikimumo problema, kurią vadinsime sinchroninio nutraukimo pertrauka. Jeigu nors vienas užklausos mazgas nebeveikia, visi kiti mazgai tuo metu laukia, galiausiai visi pertraukiami ir bando grąžinti nors dalinį rezultatą. Tačiau nė vienas iš dalinių rezultatų nepasieks pradininko, nes visi mazgai pertraukiami vienu metu.

Kad išspręsti sinchroninio nutraukimo problemą, siūlome naudoti dinamines nutraukimo pertraukas. Kai naudojami dinaminės nutraukimo pertraukos, mazgai, esantys toliau nuo pradininko, pertraukiami anksčiau nei mazgai, esantys arčiau pradininko. Tai padaro nedidelį saugumo langą daliniams rezultatams sugrįžti atgal pas pradininką. Tarpiniai mazgai turėtų sumažinti pertraukos reikšmę kaip priklauso, tam kad užtikrintų pakankamai didelį laiko langą ateinantiems ir išeinantiems daliniams rezultatams.

Pastebėkite, kad kuo arčiau yra mazgas nuo pradininko, tuo jis yra svarbesnis. Kuo mazgas yra arčiau pradininko, tuo daugiau jis sunaudoja tinklo resursų. Todėl sugumo laiko langas skaičiuojamas pagal tokią formulę:



15 pav. Dinaminio nutraukimo pertrauka

$Timeout_N = currenttime_N + (timeout_{N-1} - currenttime_N)/2$

Pavyzdys 15 pav. Laiko momentu t paradininkas paleidžia užklausą su dinamine nutraukimo pertrauka t+4 sekundės. Kitaip tariant, jis perspėja agentą ignoruoti rezultatus po laiko tarpo t+4. Savo ruožtu agentas siekia tilpti į saugumo ribas ir apskaičiuoja, kad jis turi išlaikyti 2 sekundžių saugumo langą, tuo pačiu pradedamas grąžinti dalinius rezultatus laiko momentu t+2. Agentas perspėja savo kaimynus ignoruoti rezultatus po laiko tarpo t+2.

Kaimynai taip pat siekia tilpti į saugumo ribas. Iš 2 turimų sekundžių, jie nusprendžia paskirti 1 sekundę, o visa kas likę palikti aukščiau esančiai šakai. Galų gale saugumo langas tampa toks mažas, kad mazgas netelpa į pertraukos ribas. Nesėkmingo mazgo rezultatai yra ignoruojami ir atmetami jo daliniai rezultatai. Tačiau esantys žemiau mazgai ir mazgai, esanty kitose šakose yra nepaliečiami. Jų rezultatai išlieka ir turi pakankamai laiko sugrįžti pas pradininką per laiko tarpą t+4.

5.4.2 Statinė kilpos pertrauka

Ta pati užklausa gali pasiekti mazgą kelis kartus, skirtingais maršrutais, sudėtinga struktūra. Patikimam užklausos atpažinimui, ji turi identifikatorių ir tam tikrą gyvavimo laiką. Prie kekvienos užklausos pradininkas prikabina *kilpos pertrauką* ir skirtingą *tranzakcijos identifikatorių*, kuris yra universalus unikalus identifikatorius (UUI). Mazgas turi tranzakcijų identifikatorių būsnų lentelę ir grąžina klaidą, jei gauta užklausa jau buvo matyta ir dar nebuvo pertraukta. Kai nutraukiama kilpa, mazgas gali "pamiršti" apie užklausą ištrindamas ją iš savo būsenų lentelės. Kad patikimai atpažinti kilpą, mazgas turi nepamiršti tranzakcijos identifikatoriaus kol dar nebuvo pasiekta kilpos pertrauka. Statinės kilpos pertrauka yra privaloma, norint pilnai išlaikyti užklausos semantiką. Kitaip atsiranda problema, kurią mes siulome vadinti ne-sinchroninė kilpos pertrauka. Nesinchroninė kilpos pertrauka atsiranda dėl to, kad kai kurie mazgai vis tiek persiunčia užklausą kitiems mazgams, kai paskirties punktai jau pamiršta apie ją. Kitaip tariant, problema tame, kad kilpos pertrauka įvyksta ne tuo pačiu momentu visur. Todel kilpos pertrauka turi būti pastovi, kad kilpos galėtų būti patikimai atpažintos. Kartu su užklausa, abi reikšmės turi būti identiškos (pvz. t+4). Po pirmo šuoliuko, abi reikšmės tampa nesusijusios.

Taip pat abort timeout <= loop timeout. Kad užtikrinti patikimą kilpų atpažinimą, kilpos pertrauka turi buti pastovi, o nutraukimo pertrauka gali būti statinė arba dinaminė. Kai rezultatai grąžinami ne kanalais, dinaminio nutraukimo pertraukos apskaičiavimas panadojant pusėtiną eksponentinį mažėjimą užtikrina, kad maksimalus rezultatų kiekis bus grąžintas per uždtuotą laiko tarpą. Dinaminės pertraukos galėtų būti apskaičiuojamos pagal sudėtingas formules, kurios atsižvelgtų į tinklo būsenos parametrus ir/arba ekonominius modelius.

6. IŠVADOS

- Šiame darbe aprašomos ir apibendrinamos esminės paieškos problemos savybės ir išvystomas sprendimas, kuris gali būti pritaikytas daugeliui paskirstytų sistemų internete.
- Parodoma kaip įgyvendinti abstrakčias bendros paskirties užklausas per duombazes, susidedančias iš autonominių dinaminų nuotolinių mazgų, apjungtų į įvairių tipų paskirstytas topologijas.
- Aprašyti pirmi žingsniai link tinklinio skačiavimo, P2P skaičiavimo, paskirstytų duomenų bazių ir *web servisų* apjungimo.
- Pasiūlyta ir smulkiai apibūdinta atvira atradimo struktūra - *Tinklo Paslaugų Atradimo Architektūra (WSDA*). WSDA parodo internetą kaip platų paslaugų rinkinį su išplečiamu gerai apibrėžtų sąsajų rinkiniu.
- Išplėsta duomenų bazės sąvoką ir realizuota P2P paiešką. Išplėstos P2P sąvokos, paieškai pritaikant tokias galingas užklausos kalbas kaip XQuery ir SQL.
- Pasiūlyta *Unifikuota Peer-to-Peer duomenų bazių struktūra* (UPDBS) ir atitinkantis *Peer duomenų bazių protokolas* (PDBP) bendros paskirties užklausoms vykdyti plačiose sistemose, apjungiančiose daug valdymo domenų. Tai leidžia vykdyti ivairių duomenų tipų paiešką, čia gali gali būti naudojamos ivairios nodų topologijos (žiedas, medis, diagrama), užklausų kalbos (XQuery, SQL), naudojami keli užklausų atsakymo būdai (nukreiptas, tiesioginis ir maršrutizuojamas atsakymas), vyksta kaimynų pasirinkimas (XQuery forma), rezultatų perdavimas kanalais, pertrauko ir kitos užklausos aprėpties reguliavimo savybės.

Šio darbo rezultatai atveria tokias tolimesnių tyrimų krypis kryptis:

- *Pirma*, būtų įdomu toliau išplėsti Duomenų Bazių Valdymo ir P2P skaičiavimo sąvokas ir pritakymą. Pavydžiui, P2P užklausų optimizacija. Taip pat duomenų bazių resursų valdymą ir įdomu būtų panagrinėti autorizacijos mechanizmus kiekvienam vartotojui ar užklausai.
- *Antra*, galima panagrinėti ir patikslinti kešo atsinaujinimo sąlygas ir konkretizuoti sąveiką tarp dumenų tiekėjo, registratūros ir kliento.
- *Trečia*, būtų naudinga paanalizuoti ir palyginti WSDA ir Open Grid Services Architecture (OGSA) pagal dizainą ir specifikacijas. Tikslas būtų apjungti abi technologijas panaudojant abiejų geriausias savybes.

7. PADĖKOS

Norėčiau padėkoti savo darbo vadovui doc. dr. Broniui Paradauskui už visapusę pagalbą ir suteiktą informaciją atliekant šį darbą.

8. LITERATŪROS SĄRAŠAS

1. Ian Foster, Carl Kesselman, and Steve Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int'l. Journal of Supercomputer Applications*, 15(3), 2001.

2. Wolfgang Hoschek, Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. Data Management in an International Data Grid Project. In *1st IEEE/ACM Int'l. Workshop on Grid Computing (Grid'2000)*, Bangalore, India, December 2000.

3. Large Hadron Collider Committee. Report of the LHC Computing Review. Technical report, CERN/LHCC/2001-004, April 2001. http://cern.ch/lhc-computing-reviewpublic/ Public/Report final.PDF.

4. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. *W3C Note 15*, 2001. http://www.w3.org/TR/wsdl.

5. World Wide Web Consortium. Simple Object Access Protocol (SOAP) 1.1. *W3C Note 8*, 2000.

6. Wolfgang Hoschek. *A Unified Peer-to-Peer Database Framework for XQueries over Dynamic Distributed Content and its Application for Scalable Service Discovery*. PhD Thesis, Technical University of Vienna, March 2002.

7. P. Cauldwell, R. Chawla, Vivek Chopra, Gary Damschen, Chris Dix, Tony Hong, Francis Norton, Uche Ogbuji, Glenn Olander, Mark A. Richman, Kristy Saunders, and Zoran Zaev. *Professional XML Web Services*. Wrox Press, 2001.

8. UDDI Consortium. UDDI: Universal Description, Discovery and Integration. http://www.uddi.org.

9. J.D. Ullman. Information integration using logical views. In *Int'l. Conf. on Database Theory (ICDT)*, Delphi, Greece, 1997.

10. Daniela Florescu, Ioana Manolescu, Donald Kossmann, and Florian Xhumari. Agora: Living with XML and Relational. In *Int'l. Conf. on Very Large Data Bases (VLDB)*, Cairo, Egypt, February 2000.

11. A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to heterogeneous data sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):808–823, 1998.

12. M. Tamer ¨ Ozsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1999.

13. Gnutella Community. Gnutella Protocol Specification v0.4. dss.clip2.com/GnutellaProtocol04.pdf.

14. I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

15. B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical report, U.C. Berkeley UCB//CSD-01-1141, 2001.

16. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.

17. M. van Steen, P. Homburg, and A. Tanenbaum. A wide-area distributed system. *IEEE Concurrency*, 1999.

18. Nelson Minar. Peer-to-Peer is Not Always Decentralized. In *The O'Reilly Peer-to-Peer and Web Services Conference*, Washington, D.C., November 2001.

19. Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Bob Schwartzkopf, Heinz Stockinger, Kurt Stockinger, and Brian Tierney. Giggle: A Framework for Constructing Scalable Replica Location Services. In *Proc. of the Int'l. IEEE/ACM Supercomputing Conference (SC 2002)*, Baltimore, USA, November 2002. IEEE Computer Society Press.

20. Leanne Guy, Peter Kunszt, Erwin Laure, Heinz Stockinger, and Kurt Stockinger. Replica Management in Data Grids. Technical report, Global Grid Forum Informational Document, GGF5, Edinburgh, Scotland, July 2002.

21. Heinz Stockinger, Asad Samar, Shahzad Mufza®ar, and Flavia Donno. Grid Data Mirroring Package (GDMP). *Journal of Scientific Programming*, 2002.

22. William Bell, Diana Bosio, Wolfgang Hoschek, Peter Kunszt, Gavin McCance, and Mika Silander. Project Spitfire - Towards Grid Web Service Databases. Technical report, Global Grid Forum Informational Document, GGF5, Edinburgh, Scotland, July 2002.

23. World Wide Web Consortium. XQuery 1.0: An XML Query Language. *W3C Working Draft*, December 2001.

24. T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. *IETF RFC 2396*.

25. World Wide Web Consortium. XML-Signature Syntax and Processing. *W3C Recommendation*, February 2002.

26. P. Brittenham. An Overview of the Web Services Inspection Language, 2001. www.ibm.com/developerworks/webservices/library/ws-wsilover.

27. N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. *IETF RFC 2045*, November 1996.

28. World Wide Web Consortium. XML Schema Part 0: Primer. *W3C Recommendation*, May 2001.

29. J. Wang. A survey of web caching schemes for the Internet. *ACM Computer Communication Reviews*, 29(5), October 1999.

30. S. Gullapalli, K. Czajkowski, C. Kesselman, and S. Fitzgerald. The grid notification framework. Technical report, Grid Forum Working Draft GWD-GIS-019, June 2001. http://www.gridforum.org.

31. Marshall Rose. The Blocks Extensible Exchange Protocol Core. *IETF RFC 3080*, March 2001.

32. E. O'Tuathail and M. Rose. Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP). *IETF RFC 3288*, June 2002.

33. International Organization for Standardization (ISO). Information Technology-Database Language SQL. *Standard No. ISO/IEC 9075:1999*, 1999.

34. Matei Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Int'l. Conf. on Peer-to-Peer Computing (P2P2001)*, Linkoping, Sweden, August 2001.

35. Clip2Report. Gnutella: To the Bandwidth Barrier and Beyond. http://www.clip2.com/gnutella.html.

36. Wolfgang Hoschek. A Comparison of Peer-to-Peer Query Response Modes. In *Proc. of the Int'l. Conf. on Parallel and Distributed Computing and Systems (PDCS 2002)*, Cambridge, USA, November 2002.

37. Donald Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, September 2000.

38. T. Urhan and M. Franklin. Dynamic Pipeline Scheduling for Improving Interactive Query Performance. *The Very Large Database (VLDB) Journal*, 2001.

39. Jordan Ritter. Why Gnutella Can't Scale. No, Really. http://www.tch.org/gnutella.html.

40. A. Puniyani B. Huberman L. Adamic, R. Lukose. Search in power-law networks. *Phys. Rev*, E(64), 2001.

41. S.E. Deering. *Multicast Routing in a Datagram Internetwork*. PhD Thesis, Stanford University, 1991.

42. Wolfgang Hoschek. A Unified Peer-to-Peer Database Protocol. Technical report, DataGrid-02-TED-0407, April 2002.

43. W. Yeong, T. Howes, and S. Kille. Lightweight Directory Access Protocol. *IETF RFC 1777*, March 1995.

44. Mary Fernandez, Morishima Atsuyuki, Dan Suciu, and Tan Wang-Chiew. Publishing Relational Data in XML: the SilkRoute Approach. *IEEE Data Engineering Bulletin*, 24(2), 2001.

45. Daniela Florescu, Ioana Manolescu, and Donald Kossmann. Answering XML Queries over Heterogeneous Data Sources. In *Int'l. Conf. on Very Large Data Bases (VLDB)*, Roma, Italy, September 2001.

46. Michael Stonebraker, Paul M. Aoki, Witold Litwin, Avi Pfe®er, Adam Sah, Je® Sidell, Carl Staelin, and Andrew Yu. Mariposa: a wide-area distributed database system. *The Very Large Database (VLDB) Journal*, 5(1), 1996.

47. Ashley Beitz, Mirion Bearman, and Andreas Vogel. Service Location in an Open Distributed Environment. In *Proc. of the Int'l. Workshop on Services in Distributed and Networked Environements*, Whistler, Canada, June 1995.

48. Object Management Group. Trading Object Service. *OMG RPF5 Submission*, May 1996.

49. J. Waldo. The Jini architecture for network-centric computing. *Communications of the ACM*, 42(7), July 1999.

50. Erik Guttman. Service Location Protocol: Automatic Discovery of IP Network Services. *IEEE Internet Computing Journal*, 3(4), 1999.

51. Weibin Zhao, Henning Schulzrinne, and Erik Guttman. mSLP - Mesh Enhanced Service Location Protocol. In *Proc. of the IEEE Int'l. Conf. on Computer Communications and Networks (ICCCN'00)*, Las Vegas, USA, October 2000.

52. Steven E. Czerwinski, Ben Y. Zhao, Todd Hodes, Anthony D. Joseph, and Randy Katz. An Architecture for a Secure Service Discovery Service. In *Fifth Annual Int'l. Conf. on Mobile Computing and Networks (MobiCOM '99)*, Seattle, WA, August 1999.

53. William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *Proc. of the Symposium on Operating Systems Principles*, Kiawah Island, USA, December 1999.

54. Bernard Traversat, Mohamed Abdelaziz, Mike Duigou, Jean-Christophe Hugly, Eric Pouyoul, and Bill Yeager. Project JXTA Virtual Network, 2002. White Paper, http://www.jxta.org.

55. Steven Waterhouse. JXTA Search: Distributed Search for Distributed Networks, 2001. White Paper, http://www.jxta.org.

56. Project JXTA. JXTA v1.0 Protocols Specification, 2002. http://spec.jxta.org.

57. Brian Tierney, Ruth Aydt, Dan Gunter, Warren Smith, Valerie Taylor, Rich Wolski, and Martin Swany. A Grid Monitoring Architecture. Technical report, Global Grid Forum Informational Document, January 2002. http://www.gridforum.org.

58. Jason Lee, Dan Gunter, Martin Stoufer, and Brian Tierney. Monitoring Data Archives for Grid Environments. In *Proc. of the Int'l. IEEE/ACM Supercomputing Conference (SC 2002)*, Baltimore, USA, November 2002. IEEE Computer Society Press.

59. Ian Foster, Carl Kesselman, Je®rey Nick, and Steve Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, January 2002. http://www.globus.org.

60. Steven Tuecke, Karl Czajkowski, Ian Foster, Je®rey Frey, Steve Graham, and Carl Kesselman. Grid Service Specification, February 2002. http://www.globus.org.

61. Wolfgang Hoschek. The Web Service Discovery Architecture. In *Proc. of the Int'l. IEEE/ACM Supercomputing Conference (SC 2002)*, Baltimore, USA, November 2002. IEEE Computer Society Press.

62. P. Mockapetris. Domain Names - Implementation and Specification. *IETF RFC 1035*, November 1987.

63. Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid Information Services for Distributed Resource Sharing. In *Tenth IEEE Int'l. Symposium on High-Performance Distributed Computing (HPDC-10)*, San Francisco, California, August 2001.

64. IEEE Computer Society. *Data Engineering Bulletin*, 23(2), June 2000.

65. Jayavel Shanmugasundaram, Kristin Tufte, David J. DeWitt, Je®rey F. Naughton, and David Maier. Architecting a Network Query Engine for Producing Partial Results. In *WebDB 2000*, 2000.

66. Zachary G. Ives, Alon Y. Halevy, and Daniel S. Weld. Integrating Network-Bound XML Data. *IEEE Data Engineering Bulletin*, 24(2), 2001.

67. J. F. Naughton, D. J. DeWitt, D. Maier, A. Aboulnaga, J. Chen, L. Galanis, J. Kang, R. Krishnamurthy, Q. Luo, N. Prakash, R. Ramamurthy, J. Shanmugasundaram, F. Tian, K. Tufte, S. Viglas, Y. Wang, C. Zhang, B. Jackson, A. Gupta, and R. Chen. The Niagara Internet Query System. *IEEE Data Engineering Bulletin*, 24(2), 2001.

68. Annita N. Wilschut and Peter M. G. Apers. Dataflow query execution in a parallel main-memory environment. In *First Int'l. Conf. on Parallel and Distributed Information Systems*, December 1991.

69. Zachary G. Ives, Daniela Florescu, Marc T. Friedman, Alon Y. Levy, and Daniel S. Weld. An adaptive query execution system for data integration. In *ACM SIGMOD Conf. On Management of Data*, 1999.

70. Tolga Urhan and Michael J. Franklin. Xjoin, A reactively-scheduled pipelined join operator. *IEEE Data Engineering Bulletin*, 23(2), June 2000.

71. Beverly Yang and Hector Garcia-Molina. E±cient Search in Peer-to-Peer Networks. In *22nd Int'l. Conf. on Distributed Computing Systems*, Vienna, Austria, July 2002.

72. Adriana Iamnitchi and Ian Foster. On Fully Decentralized Resource Discovery in Grid Environments. In *Int'l. IEEE Workshop on Grid Computing*, Denver, Colorado, November 2001.

73. Y. Papakonstantinou and V. Vassalos. Query rewriting for semistructured data. In *ACM SIGMOD Conf. On Management of Data*, 1999.

74. Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD Thesis, University of California, Irvine, 2000.