TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Tanel Tomson

Töö pealkiri täpsustada

Võrgukommutaatorite visualiseerimine

Bakalaureusetöö (9 EAP)

Juhendaja: Meelis Roos, MSc

Tartu 2018

Võrgukommutaatorite visualiseerimine

Lühikokkuvõte:

One or two sentences providing a basic introduction to the field, comprehensible to a scientist in any discipline.

Two to three sentences of more detailed background, comprehensible to scientists in related disciplines.

One sentence clearly stating the general problem being addressed by this particular study.

One sentence summarising the main result (with the words “here we show´´ or their equivalent).

Two or three sentences explaining what the main result reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

One or two sentences to put the results into a more general context.

Two or three sentences to provide a broader perspective, readily comprehensible to a scientist in any discipline, may be included in the first paragraph if the editor considers that the accessibility of the paper is significantly enhanced by their inclusion.

Võtmesõnad:

List of keywords

CERCS:

CERCS kood ja nimetus: `https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e`

Visualizing network switches

Abstract:

Abstract

Keywords:

List of keywords

CERCS:

CERCS code and name: https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e

Sisukord

1 Sissejuhatus **8**

2 Rakenduse nõuded **9**
2.1 Kasutamise eeldused 9
2.2 Rakenduse nõuded 9

3 Sarnased lahendused **11**
3.1 Netcrawl 11
3.2 Netdisco 11
3.3 Foo 11

4 Kasutatud tehnoloogiad **12**
4.1 Keeled 12
4.2 CDP ja LLDP 12
4.3 SNMP 12
4.4 easySNMP 13
4.5 Cytoscape.js 13

5 Valminud rakenduse kirjeldus **14**
5.1 Tagasüsteem 14
5.1.1 Rakenduse seadistamine 14
5.1.2 Rakenduse põhivoog 15
5.1.3 Päritavad andmed 15
5.1.4 Andmete faili salvestamine 15
5.1.5 Veahaldus 16
5.1.6 Logimine 16
5.2 Kasutajaliides 16
5.2.1 Välised teegid 16
5.2.2 Lokaalse faili laadimine brauseris 16
5.2.3 Tagasüsteemi loodud sisendi lugemine 17
5.2.4 Graafi kuvamine 17
5.2.5 Hüpikvihjed 18
5.2.6 Kasutaja muudatuste salvestamine 18

6 Edasised tegevused **19**

7 Kokkuvõte **20**

Viidatud kirjandus **21**

Lisad **22**
I. Glossary 22
II. Litsents 23

Unsolved issues

Töö pealkiri täpsustada . . . 1
One or two sentences providing a basic introduction to the field, comprehensible to a scientist in any discipline. . . . 2
Two to three sentences of more detailed background, comprehensible to scientists in related disciplines. . . . 2
One sentence clearly stating the general problem being addressed by this particular study. . . . 2
One sentence summarising the main result (with the words "here we show´´ or their equivalent). . . . 2
Two or three sentences explaining what the main result reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge. . . . 2
One or two sentences to put the results into a more general context. . . . 2
Two or three sentences to provide a broader perspective, readily comprehensible to a scientist in any discipline, may be included in the first paragraph if the editor considers that the accessibility of the paper is significantly enhanced by their inclusion. . . . 2
List of keywords . . . 2
CERCS kood ja nimetus: https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e . . . 2
Abstract . . . 2
List of keywords . . . 2
CERCS code and name: https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e . . . 2
Eemalda lõplikust versioonist . . . 7
What you are doing in each section (a sentence or two per section) . . . 8
Riistvara ja tarkvara jm nõuded . . . 9
Netdisco ülevaade . . . 11
kolmas sarnane lahendus? . . . 11
Tabel OID-dest ja kus neid andmeid kasutatakse? . . . 15
Iga OID kohta: oid, kas seadme kohta või naabri?, kus kasutajaliideses kuvatakse? 15
sysName kasutatakse id-na – kas tahan rääkida sellest, et miks? indeksid erinevate mib-de vahel erinevad . . . 15
Pilt graafist . . . 18
Mis andmed kuvatakse . . . 18
Hüpikaknast näidispilt, kui koodis kõik asjad sinna lisatud on . . . 18
Sissejuhatav tekst . . . 19
räsi arvutamine ja teate kuvamine, kui andmefail on taustal muutunud, kui see implementeeritud saab . . . 19

puudus graafi servade labelitega 19
what did you do? 20
What are the results? 20
future work? 20
kuupäev 23
Eemalda lõplikust versioonist

1 Sissejuhatus

Võrgukommutaator (inglise keeles *switch*, edaspidi ka lihtsalt kommutaator) on võrguseade, mis ühendab seadmeid arvutivõrku. Näiteks kontorites, kus on reeglina hulgaliselt võrguseadmeid (arvutid, printerid jne), kasutatakse kommutaatoreid, et suur arv seadmeid saaksid arvutivõrguga ühenduda.

Võrgukommutaatoritest võib mõelda kui sorteerimisekeskustest postiteenuses. Igal sorteerimiskeskusesse jõudval kirjal on sihtkoht, selleks on ümbrikul addressaat ja aadress. Sorteerimiskeskuses otsustatakse, kuhu iga kiri edasi tuleb saata. Sama ülesanne on ka võrgukommutaatoril - otsustada, millisele ühendatud võrguseadmele iga pakett (kiri) edasi saata.

Kommutaatoreid - mida reeglina haldavad süsteemiadministraatorid - võib arvutivõrgus olla mitu. On protokolle, mille abil saab võrgukommutaatoritelt haldusinfot küsida. Kommutaatorilt infot küsides on tihti vastuseks saadud andmete hulk suur ja vajab lisatöötlust. Ülevaate saamine, kuidas kommutaatorid üksteisega ühendatud on, võib muutuda tülikaks ja aeganõudvaks. Töö autor kirjutas programmi, mis abistab võrgukommutaatorite haldamist, visualiseerides kommutaatorite paiknemist üksteise suhtes. Fookus on just kommutaatorite endi paiknemise visualiseerimisel, jättes lisainformatsiooni (muud ühendatud seadmed) tagaplaanile.

Järnevalt on toodud töö struktuur. Peatükis "Rakenduse nõuded", kirjeldatakse rakenduse jooksutamiseks vajalikud tingimused ning enne rakenduse arendamist paika pandud nõuded. Peatükis "Sarnased lahendused"tuuakse ülevaade sarnast funktsionaalsust pakkuvatest rakendustest. "Kasutatud tehnoloogiad"kirjeldab tehnoloogiaid, teeke ja rakendusi, mida programm kasutab või mida kasutati arendusprotsessis. Peatükis "Valminud rakenduse kirjeldus"tehakse detailne ja tehniline ülevaade rakenduse arhitektuurist. Samuti põhjendatakse tehtud otsuseid ning näidatakse rakenduse nõuete täitmist.

What you are doing in each section (a sentence or two per section)

2 Rakenduse nõuded

Enne rakenduse programmeerimist fikseeriti rakenduse nõuded. Järgnev peatükk kirjeldab eeldused rakenduse käivitamiseks ning selle nõuded.

Riistvara ja tarkvara jm nõuded

2.1 Kasutamise eeldused

Rakenduse kasutamiseks on tehtud eeldus, et kasutajal on kontroll kaardistatavate võrgukommutaatorite üle. Võrgukommutaatorid peavad olema programmi jaoks kättesaadavad ja vajalikku informatsiooni edastama. See tähendab, et võrgukommutaatoritel peab töötama SNMP teenus ja see peab olema seadistatud jagama CDP või LLDP andmeid.

2.2 Rakenduse nõuded

1. Rakenduse käivitamine toimub käsurealt, vajalik python (koos vähemalt ühe SNMP teegiga) ja SNMP
2. Rakenduse sisendiks on ühe või mitme võrgukommutaatori aadress ja seadmetes seadistatud SNMP kommuuni nimi
3. Kui antakse ette mitu seadet, on võimalus määrata erinevatele seadmetele erinevad SNMP kommuunide nimed
4. Rakendus küsib igalt sisendiks saadud seadmelt LLDP/CDP andmed (nimi, ip, mac; ühenduste füüsiline port, nimi, ip, mac)
5. Rakendus töötleb ja vormindab saadud andmed
6. Tekkinud vead kuvatakse kasutajale mõistlikult (mis juhtus, mida teha)
7. Rakendus loob antud andmete põhjal staatilise veebilehe (.html), kus kuvatakse graaf võrgukommutaatorite kohta
8. Staatilise veebilehe vaatamiseks pole vaja internetiühendust (CSS ja JS lokaalsed)
9. Graafi tipud on võrgukommutaatorid
10. Graafi servad tippude vahel on võrgukommutaatorite ühendused, erinevad ühenduste kiirused on tähistatud erinevalt
11. Graafis on vaikimisi paigutus mõistlik, see tähendab, et tipud ei kattu ja kui tekib mitu graafi, kuvatakse need üksteisest pisut eemal

12. Graafis on võimalik seadmeid enda suva järgi lohistada
13. Graafis kuvatakse tippudes seadme nimi (aadress), tipule peale vajutades on võimalik näha lisainfot seadme kohta
14. Tipule peale vajutades on võimalik näha nimekirja kommutaatoriga ühendatud teistest seadmetest
15. Kasutaja graafile tehtud muudatused salvestatakse kasutaja brauseriga lokaalselt (küpsised või localstorage) ja taastatakse järgmisel avamisel

3 Sarnased lahendused

Käesolevas peatüki eesmärk on tutvustada ja anda ülevaade sarnastest lahendustest, tuues välja nende tugevused ja puudused.

3.1 Netcrawl

Netcrawl on lihtne programm, mille sisendiks on ühe võrguseadme aadress ja SNMP kogukonnasõne. Rakendus küsib seejärel seadmelt naabrite info ja kordab tegevust naabrite puhul. Väljundfail on DOT-keeles[1], mida kasutab visualiseerimiseks Graphviz[2] - graafide visualiseerimise tarkvara. Tulemuseks on pildifail graafiga, kus on leitud võrgukommutaatorid ja nende naabrid. [ky]

Netcrawl on üsna sarnane töö käigus loodava rakendusega. Rakenduse kasutamine on lihtne ja mõne üksiku mööndusega rahuldab programm meie nõuded. Küll aga on mõned olulised puudused. Suurim erinevus on programmi väljundi formaat - Netcrawli kasutamisel tekib pildifail. Nõue 12 täpsustab aga võimaluse kasutajal graafi tippe interaktiivselt ümber paigutada. Samuti paneb programm kõik leitud seadmed graafi (sh kommutaatoriga ühendatud teised seadmed) - see tähendab, et kui seadmeid on palju, muutub graaf segaseks. Loodud programmis tehti otsus, et kommutaatoritega ühendatud seadmeid graafis ei kuvata, vaid need kuvatakse graafi tippudele (kommutaatoritele) vajutades hüpikvihjetega (nõue 14). Lisaks ei saa Netcrawliga anda erinevatele kommutaatoritele erinevaid kogukonnasõnesid (nõue 3).

3.2 Netdisco

Netdisco ülevaade

3.3 Foo

kolmas sarnane lahendus?

[1] `https://graphviz.gitlab.io/_pages/doc/info/lang.html`
[2] urlhttps://www.graphviz.org/

4 Kasutatud tehnoloogiad

Antud peatükis tutvustatakse tehniliselt võrguprotokolle, teeke ning muid vahendeid, mida valminud rakendus kasutab.

4.1 Keeled

Rakenduse tagasüsteem on kirjutatud programmeerimiskeeles Python, kasutati versiooni 3. Peamine põhjus oli, et autoril oli varasem kogemus Pythoniga juba olemas. Samuti kontrolliti enne valiku tegemist, et leiduks sobiv SNMP teek, mis on rakenduse loomiseks vajalik.

Kasutajaliides on loodud kasutades HTML-i ja CSS-i. Lisaks on graafi kuvamiseks kasutatud programmeerimiskeelt Javascript.

4.2 CDP ja LLDP

CDP (*Cisco Discovery Protocol*) ja LLDP (*Link Layer Discovery Protocol*) on mõlemad võrguprotokollid ühendatud võrguseadmetele haldusinformatsiooni jagamiseks. LLDP ja CDP on funktsionaalsuse poolest küllaltki sarnased. Peamine erinevus kahe protokolli vahel on, et CDP on Cisco (võrguseadmete tootja) loodud ja hallatav omandprotokoll. [Bha15a] LLDP seevastu on avatud ja seadmetootjate ülene protokoll, LLDP standardit haldab IEEE. [Bha15b]

4.3 SNMP

SNMP (*Simple network management protocol*) on laialdaselt kasutatav võrguhaldusprotokoll, mis kuulub OSI mudelis rakenduskihti. Protokoll lihtsustab ja ühtlustab võrguseadmetelt haldusinfo küsimist. [Laa08, 151]

Hallatavad seadmed hoiavad haldusinfot MIB-i struktuuris, mis on hierarhiline andmebaas. Igal objektil on oma unikaalne identifikaator (OID). [Laa08, 152] Valminud rakendus kasutab konkreetseid OID-sid, et kommutaatoritelt vajalikke andmeid küsida.

SNMP protokoll defineerib päringute päises ka parameetri kogukonnasõne (*community string*). Kogukonnasõne võimaldab teha päringutele triviaalset autentimist (nt lubada andmete pärimist vaid kindlate kogukonnasõnedega). [Laa08, 154-155]

SNMP-st on 3 põhilist versiooni: 1, 2 ja 3. Valminud rakendus kasutab versiooni 2 (täpsemalt 2c).

Valminud rakendus kasutab SNMP protokolli, et sisendiks saadud võrguseadmetelt CDP ja LLDP andmeid küsida.

4.4 easySNMP

Kommutaatoritelt üle SNMP protokolli andmete küsimiseks kasutab rakendus Pythoni teeki easySNMP. Töö algfaasis katsetati alternatiivina ka teeki PySNMP[3]. Valik tehti aga easySNMP kasuks, kuna PySNMP on kommutaatoritega suhtlemisel tuntavalt aeglasem. [eas]

Seadmetelt andmete küsimiseks kasutatakse session.walk() [4] funktsiooni, mis kasutab SNMP GetNextRequest operatsiooni.

4.5 Cytoscape.js

Järgmiseks oli vaja Javascripti teeki graafide brauseris esitamiseks. Andmeid visualiseerivaid (ja muuhulgas graafe toetavaid) teeke leidub arvukalt, ent autor kitsendas otsingu lihtsuse huvides vaid graafidele spetsialiseerunud teekidele. Samuti pidi teek olema avatud lähtekoodiga ning toetama graafi tippudele hüpikvihjete lisamist.

Kasutusele võeti Cytoscape.js[5], mis vastas kirjeldatud nõuetele. [cyt] Pärast esmaseid katsetusi demodega tõusis esile hea dokumentatsioon, mis tegi teegi kasutamise lihtsaks, lai valik erinevaid kujundusi ning graafide kuvamisel mõistlik ekraanipinna kasutamine.

Kuigi teek vaikimisi hüpikvihjeid kuvada ei oska, on selleks loodud laiendeid. Hüpikvihjete kuvamiseks kasutusele võetud teegid on kirjeldatud peatükis 5.2.1.

[3] `http://snmplabs.com/pysnmp`
[4] `https://easysnmp.readthedocs.io/en/latest/session_api.html`
[5] `http://js.cytoscape.org/`

5 Valminud rakenduse kirjeldus

Rakenduse võib funktsionaalsuse järgi jagada kaheks: tagasüsteem ja kasutajaliides. Tagasüsteemi ülesanne on küsida ja töödelda sisendiks saadud võrguseadmetelt info ühendatud seadmete kohta. Kasutajaliides koostab tagasüsteemi poolt saadud andmetest veebilehe kommutaatorite graafiga. Peatükis kirjeldatakse detailselt mõlema osa arhitektuuri.

5.1 Tagasüsteem

Tagasüsteem on kirjutatud programmeerimiskeeles Python. Välistest teekidest on kasutusel vaid easySNMP (4.4). Tagasüsteemi lähtekood asub `src/` kaustas (va `src/web/`, mis on kasutajaliides).

5.1.1 Rakenduse seadistamine

Rakenduse sätted loeb programm failist `config.ini`. Rakenduse lähtekoodis on näidisfail `config.ini.sample`, mille rakenduse käivitaja saab kopeerida ja vastavalt soovidele muuta.

```
[Application Config]

# Default community string is used when switch has no specific
    community string set
defaultCommunityString = public

# Switches to be looked up.
switches = with-specific-community.example.com specificcommunity
           with-default-community.example.com
           another-with-default-community.example.com

# Enable or disable application logging (true or false)
debug = false
```

Joonis 1. Näidis sättefail config.ini.sample

Kasutaja saab määrata kommutaatorid, millelt naabrite info küsitakse. Seejuures saab igale kommutaatorile vajadusel määrata eraldi kogukonnasõne (vt 4.3 ja samuti nõue 3). Lisaks on võimalik seadistada rakenduse logimist silumise tasemele(täpsemalt vt 5.1.6).

Rakendus kasutab sätetefaili parsimiseks Pythoni moodulit configparser[6] ja sellest lähtuvalt on sätetefail INI-struktuuriga. Sätetefaili puudumisel või kui sätete lugemine

[6] https://docs.python.org/3.6/library/configparser.html

ebaõnnestub, kuvatakse kasutajale vastav veateade ja programm lõpetab töö. Sätete lugemise implementatsioon on failis `src/config_helper.py`.

5.1.2 Rakenduse põhivoog

Kui sätetefailist on seadistatud kommutaatorid loetud, asutakse neilt andmeid küsima. Seda tehakse ükshaaval, üle kommutaatorite itereerides. Vastuseks saadud andmed hoiustatakse sisemiselt sõnastikus, seejuures jagatakse iga kommutaatori käest saadud andmed kolmeks:

1. kommutaatori enda andmed (graafis tipud)
2. kommutaatoriga ühendatud teiste kommutaatorite andmed (graafis servad)
3. ülejäänud kommutaatoriga ühendatud seadmete info (graafi tippude hüpikvihjetes)

Iga saadud naabri korral kontrollitakse, kas see on seadistatud kommutaator (ehk graafis teine tipp). Kui jah, siis talletatakse andmed, mis on vajalikud graafi serva kuvamiseks (nt pesade nimed, ühenduse kiirus). Samas sel juhul ei kasutata naabri kohta saadud andmeid. Teisisõnu - eelistatakse kommutaatori andmeid, mida kommutaator ise enda kohta andis, mitte andmeid, mida temaga ühenduses olev kommutaator tema kohta andis. Kui naaber aga ei ole teine sisendiks saadud kommutaator (vaid kommutaatoriga ühendatud muu seade), siis on tema kohta saadud info meile oluline ja naabri info salvestatakse.

Oluline on, et rakendus kasutab mõlemat protokolli: CDP ja LLDP. Kui kommutaator kasutab vaid üht, kasutatakse seda. Kui kommutaator toetab mõlemat, siis lõpuks kuvatakse LLDP andmeid (küsitakse mõlemad, salvestatakse LLDP).

5.1.3 Päritavad andmed

Kommutaatorite käest küsitavad andmed võib jagada kaheks: info seadme enese kohta ja info kommutaatoriga ühenduses olevate seadmete (naabrite) kohta. Järgnevalt toome välja kõik andmed, mida kommutaatoritelt päritakse, sealhulgas ka SNMP identifikaatorid (OID-d, vt 4.3), mida päringutes kasutati.

Tabel OID-dest ja kus neid andmeid kasutatakse?

Iga OID kohta: oid, kas seadme kohta või naabri?, kus kasutajaliideses kuvatakse?

sysName kasutatakse id-na – kas tahan rääkida sellest, et miks? indeksid erinevate mib-de vahel erinevad

5.1.4 Andmete faili salvestamine

Et andmed jõuaksid tagasüsteemilt kasutajaliidesesse, kirjutatakse need Javascripti lähtekoodi faili. Põhjus, miks kasutatakse Javascripti faili (laiendiga `.js`) ja mitte näiteks

JSON-formaati, on kirjeldatud peatükis 5.2.3. Andmed itereeritakse ja vormindatakse vastavalt Cytoscape.js teegi sisendformaadile. Andmete faili kirjutamise implementatsioon on failis `src/output_helper.py`.

5.1.5 Veahaldus

Vigadele kõige altim on kommutaatoritelt andmete küsimine - juhul kui kommutaatoriga ei saada ühendust (vale seadistus, võrguprobleem vms). Veahalduse roll on sel juhul tagada, et programmi töö jätkuks teistelt kommutaatoritelt andmete küsimisega. Sel juhul logitakse kasutajale olemasolevate andmetega võimalikult informatiivne viga ja minnakse programmi vooga edasi.

5.1.6 Logimine

Logimiseks kasutatakse Pythoni moodulit logging[7]. Rakenduse sättefailis on võimalik määrata logimise taset (`debug`, kui `debug = true` või `info`, kui `debug = false`). `Debug` tasemel logikirjed on mõeldud vaid arendamiseks ja vigade silumiseks.

5.2 Kasutajaliides

Kasutajaliides on staatiline veebileht. Kasutajaliidese töö on kuvada tagasüsteemi poolt loodud andmed graafina kasutaja brauseris. Kasutajaliidese lähtekood asub `src/web/` kataloogis.

5.2.1 Välised teegid

Graafide kuvamiseks kasutatakse Javascripti teeki cytoscape.js (4.5). Lisaks kasutatakse laiendit cytoscape-popper[8], mis lisab cytoscape.js-le Popper.js[9] (teek muuhulgas hüpikvihjete kuvamiseks) toe. Lisaks võeti kasutusele Tippy.js[10], mis teeb hüpikvihjete loomise ja kohandamise Popper.js-iga lihtsamaks.

Välised teegid asuvad kataloogis `src/web/lib/`. Kõik kasutatud välised failid on lähtekoodiga kaasas (ei laeta internetist). Põhjuseks turvalisus ja võimalus rakendust ilma võrguühenduseta kasutada.

5.2.2 Lokaalse faili laadimine brauseris

Lokaalsel veebilehel lokaalse faili Javascriptiga lugemine on keerulisem kui võiks arvata. Brauserid kaitsevad kasutajaid pahatahtliku koodi laadimise ja jooksutamise eest, aga

[7] `https://docs.python.org/3.6/library/logging.html`
[8] `https://github.com/cytoscape/cytoscape.js-popper`
[9] `https://popper.js.org`
[10] `https://atomiks.github.io/tippyjs/`

samamoodi piiratakse ka pääsu kasutaja lokaalsetele failidele.

Brauserid rakendavad reeglina doomenisisese ressursikasutuse (*same-origin policy*) printsiipi. See tähendab, et veebileht ei saa jooksutada koodi ning ei pääse ligi failidele, mis asuvad teistes domeenides. Et võimaldada ka doomenivälist ressursikaasutust on W3C loonud spetsifikatsiooni CORS (*Cross-origin resource sharing*). [w3C14]

Antud spetsifikatsiooni implementeerivad brauserid ise ning sellele on erinevad arendajad lähenenud erinevalt [Bar08]. Loodud programmi kontekstis tuli esile probleem, et kasutades `file://` URI-skeemi, on faili päritolu (*origin*) definitsioon CORS-i spetsifikatsioonis lahtine [Bar11, peatükk 4, punkt 4]. See tähendab, et brauserid käituvad antud olukorras erinevalt ja ei ole garantiid, et igas brauseris tagasüsteemi poolt loodud faili lugemine õnnestub.

5.2.3 Tagasüsteemi loodud sisendi lugemine

Töö autor lootis algselt kommutaatorite andmed tagasüsteemilt kasutajaliidesele edastada JSON-formaadis, seejuures kasutada Pythoni ja Javascripti standardteeke. Katsetuste käigus tuli eelnevalt kirjeldatud erinev brauserite käitumine aga välja. Chromium (ja Google Chrome) ei lubanud `file://` URI-skeemiga lokaalset faili üldse laadida. Firefox lubaks faili laadida, aga (alates küljendusmootori Gecko versioonist 1.9) seab ette nõuded faili asukohale [ffC].

Kasutaja brauserite kohta eeldusi teha ei tahetud - programm peaks töötama sõltumata brauserist. Üks lahendus oleks veebirakenduse kuvamiseks kasutada lokaalset veebiserverit ja küsida andmefail üle `http://` URI-skeemi [11] [12] [13]. See on aga töö raames loodava rakenduse kontekstis üleliigne keerukus ja probleemi lahenduseks ei sobinud.

Selle asemel otsustati andmed tagasüsteemi poolt vormindada Javascripti koodifailiks. Fail defineerib ühe muutuja, milles on kommutaatorite andmed sõnastikus. Seejuures vormindatakse andmed sõnastikus nõnda, et selle saab otse cytoscape.js (vt 4.5) teegile ette anda. See tähendab, et sõnastikus on elemendid `nodes` ja `edges`, vastavalt graafi tipud ja servad. Antud fail laetakse kasutajaliideses Javascripti lähtekoodina ja seejärel on kommutaatorite andmed kasutajaliideses olemas ja neid saab kasutajale kuvada.

5.2.4 Graafi kuvamine

Kuna sisendandmed on juba tagasüsteemi poolt teegile cytoscape.js loetavaks vormindatud, on graafi kuvamiseks vajalik vaid cytoscape.js seadistada. Peamine otsus, mis tuli

[11] `https://stackoverflow.com/questions/46258449/cors-error-requests-are-only-supported-for-protocol-schemes-http-etc`

[12] `https://stackoverflow.com/questions/20041656/xmlhttprequest-cannot-load-file-cross-origin-requests-are-only-supported-for-ht`

[13] `https://stackoverflow.com/questions/35335047/how-to-get-rid-of-cross-origin-request-block-in-chrome`

teha, oli küljenduse valimine[14]. Hea küljendus kasutab kogu ekraanipinna ära (jaotab tipud laiali) ja samuti asetab tipud nõnda, et servad ei kattuks (vt nõue 11). Autori katsetustest tuli välja, et küljendus `breadthfirst` kasutas ekraanipinda kõige mõistlikumalt - see võeti ka kasutusele. Välja tasub tuua ka `concentric` küljendus, mis toimis samuti hästi, ent mitme graafi kuvamisel jäi viimane hätta kogu ekraanipinna ära kasutamisega.

Graafi kujunduses võeti aluseks cytoscape.js-i demo *linkout-example* [15]. Samas lisati tippudes tekstile kontrasti (must taust), et lugemist lihtsustada. Graafi kuvamine on implementeeritud failis `src/web/code.js`.

Graafi tippudeks on kommutaatorid, millelt andmeid küsiti (nõue 9). Servadeks on ühendused kommutaatorite vahel (nõue 10). Seejuures kuvatakse servadel ka mõlema kommutaatori pesa, mille kaudu nad ühendatud on. Algselt oli plaanis servadel kuvada ühenduste kiirused, kuid otsustati pesa nimede kasuks, sest nimi sisaldab lisaks pesa numbrile infot ka kiiruse kohta (Gi on 1000Mbps, Fa on 100Mbps jne).

Pilt graafist

5.2.5 Hüpikvihjed

Vastavalt nõuetele 13 ja 14, peavad tippudele vajutades avanema hüpikvihjed. Nagu peatükis 5.2.1 juba kirjeldati, implementeeriti hüpikvihjed teegiga Tippy.js, mis sisemiselt kasutab Popper.js teeki. Popper.js teegi cytoscape.js teegiga integreerimiseks kasutati cytoscape-popper laiendit.

Autori roll hüpikvihjete tööle saamiseks oli Tippy.js implementatsioon seadistada. Tippy pakub mitmeid kujundusi[16]. Autor valis läbipaistva kujunduse, et hüpikaknad ei kataks ära hüpikakna all olevaid graafi servasid. Hüpikvihjete avamine ja sulgemine (mõlemad tipule vajutades) implementeeeriti Tippy.js teegi väliselt eraldi, eesmärgiga, et korraga oleks võimalik avada mitu vihjet.

Vihjete sisu koostatakse Javascriptiga, andmed saadakse samast tagasüsteemi poolt loodud sisendfailist, mille põhjal ka graaf tehakse.

Mis andmed kuvatakse

Hüpikvihjete implementatsioon on samuti failis `src/web/code.js`.

Hüpikaknast näidispilt, kui koodis kõik asjad sinna lisatud on

5.2.6 Kasutaja muudatuste salvestamine

Vastavalt nõudele 15 pidi kasutajaliides salvestama kasutaja muudatused graafile (tippude asukohtade muudatused). Antud funktsionaalsust teek cytoscape.js otse ei paku, küll aga

[14] `http://js.cytoscape.org/#layouts`
[15] `http://js.cytoscape.org/demos/linkout-example`
[16] `https://atomiks.github.io/tippyjs/themes`

defineeritakse funktsioon `cy.json()`[17], mis võimaldab graafi paigutust JSON-formaadis importida ja eksportida.

Implementeeriti graafi asetuse salvestamine, kui kasutaja veebilehelt lahkub või seda värskendab. JSON andmed salvestatakse sõnena brauseri lokaalsesse andmesalvestisse (localStorage[18]). Kui kasutaja veebilehe avab, laetakse ja taastatakse graafi paigutus andmesalvestist. Kui seda varasemalt salvestatud ei ole (esimene külastus), siis laetakse graaf andmefailist.

Samuti lisati veebilehele nupp graafi paigutuse taastamiseks - graafi andmefailist uuesti laadimiseks.

6 Edasised tegevused

- Sissejuhatav tekst
- räsi arvutamine ja teate kuvamine, kui andmefail on taustal muutunud, kui see implementeeritud saab
- puudus graafi servade labelitega

[17] `http://js.cytoscape.org/#cy.json`
[18] `https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage`

7 Kokkuvõte

what did you do?

What are the results?

future work?

Viidatud kirjandus

[Bar08] Adam Barth. Security in depth: Local web pages. Chromium Blog, https://blog.chromium.org/2008/12/security-in-depth-local-web-pages.html, 2008.

[Bar11] Adam Barth. The web origin concept. https://tools.ietf.org/html/rfc6454, 2011.

[Bha15a] Deben Bhattarai. Cisco discovery protocol (cdp). The Cisco Learning Network, 2015. https://learningnetwork.cisco.com/docs/DOC-26872.

[Bha15b] Deben Bhattarai. Link layer discovery protocol (lldp). The Cisco Learning Network, 2015. https://learningnetwork.cisco.com/docs/DOC-26851.

[cyt] Cytoscape.js. http://js.cytoscape.org/#introduction. Vaadatud: 25.03.2019.

[eas] Easy snmp documentation. https://easysnmp.readthedocs.io. Vaadatud: 25.03.2019.

[ffC] Same-origin policy for file: Uris. https://developer.mozilla.org/en-US/docs/Archive/Misc_top_level/Same-origin_policy_for_file:_URIs. Vaadatud: 26.03.2019.

[ky] Github kasutaja ytti. Netcrawl - lldp/cdp crawler. https://github.com/ytti/netcrawl. Vaadatud: 08.04.2019.

[Laa08] Erkki Laaneoks. *Sissejuhatus võrgutehnoloogiasse*. Tartu Ülikooli Kirjastus, 2008.

[w3C14] Cross-origin resource sharing. W3C Recommendation, https://www.w3.org/TR/cors/, 2014.

Lisad

I. Glossary

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Tanel Tomson**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Võrgukommutaatorite visualiseerimine

   mille juhendaja on Meelis Roos

   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

kuupäev

Tartus, pp.kk.aaaa