Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

LAAJAKAISTAVERKON TOPOLOGIAN VISUALISOINTI

Diplomityö

Diplomityön aihe on hyväksytty Tietotekniikan osaston osastoneuvostossa 11.2.2004.

Työn 1. tarkastaja: Prof., Heikki Kälviäinen
Työn 2. tarkastaja ja ohjaaja: DI Timo Kujala

Mikko Kärkkäinen
Kimpisenkatu 17 a 3
53100 Lappeenranta
Puh. 040 5636580
Email. mikko.karkkainen@lut.fi

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

Mikko Kärkkäinen

Laajakaistaverkon topologian visualisointi
Diplomityö, 90 sivua, 22 kuvaa, 1 taulukko ja 3 liitettä.
2004

Tarkastajat: Prof., Heikki Kälviäinen, DI Timo Kujala

Hakusanat: laajakaistaverkko, visualisointi, graafin piirtäminen, voimiin perustuvat algoritmit
Keywords: broadband network, visualization, graph drawing, force-directed algorithms

Viimeisten vuosien aikana laajakaistaoperaattoreiden laajakaistaverkot ovat nopeiden ja kiinteähintaisten laajakaistaliittymien johdosta kasvaneet suuriksi kokonaisuuksiksi. Kokonaisuuksia hallitaan erilaisilla verkonhallintatyökaluilla. Verkonhallintatyökalut sisältävät suuren määrän eri tasoista tietoa laitteista ja laitteiden välisistä suhteista. Kokonaisuuksien hahmottaminen ilman tiedoista rakennettua kuvaa on vaikeaa ja hidasta.

Laajakaistaverkon topologian visualisoinnissa muodostetaan kuva laitteista ja niiden välisistä suhteista. Visualisoitua kuvaa voidaan käyttää osana verkonhallintatyökalua, jolloin käyttäjälle muodostuu nopeasti näkymä verkon laitteista ja rakenteesta eli topologiasta.

Visualisoinnissa kuvan piirto-ongelma täytyy muuttaa graafin piirto-ongelmaksi. Graafin piirto-ongelmassa verkon rakennetta käsitellään graafina, joka mahdollistaa kuvan muodostamisen automaattisia piirtomenetelmiä hyväksikäyttäen.

Halutunlainen ulkoasu kuvalle muodostetaan automaattisilla piirtomenetelmillä, joilla laitteiden ja laitteiden välisten suhteiden esitystapoja voidaan muuttaa. Esitystavoilla voidaan muuttaa esimerkiksi laitteiden muotoa, väriä ja kokoa. Esitystapojen lisäksi piirtomenetelmien tärkein tehtävä on laskea laitteiden sijaintien koordinaattien arvot, jotka loppujen lopuksi määräävät koko kuvan rakenteen. Koordinaattien arvot lasketaan piirtoalgoritmeilla, joista voimiin perustuvat algoritmit sopivat parhaiten laajakaistaverkkojen laitteiden sijaintien laskemiseen.

Tämän diplomityön käytännön työssä toteutettiin laajakaistaverkon topologian visualisointityökalu.

ABSTRACT

Lappeenranta University of Technology
Department of Information Technology

Mikko Kärkkäinen

Visualization of the broadband network topology
Master's thesis, 90 pages, 22 figures, 1 table, 3 appendices
2004

Supervisors: Prof., Heikki Kälviäinen, M.Sc. Timo Kujala

Keywords: broadband network, visualization, graph drawing, force-directed algorithms

During recent years the broadband networks of network operators have become massive entities due to high speed and fixed rate charging of the network access capabilities. Entities are being managed using various network management tools. These tools contain a lot of information about the devices and respective relationships which are often hard to understand without a corresponding visual image.

When visualizing the broadband network topology, an image of the devices and respective relationships is being created. This visualized image can then be used as a part of the network management tool, thus giving the user a quick view of the network devices and the structure, or the network topology.

Visualization process includes converting a picture drawing problem into a graph drawing problem. The graph drawing problem approaches the network topology as a graph utilizing automated drawing methods in creating the visual image.

Creating a layout for the image is utilized by automated drawing methods. Automated drawing methods can be manipulated representation of the devices and respective relationships. For example shape, color, or size can be modified by representation of the devices and respective relationships. The most important task in utilizing drawing methods is to calculate the device location coordinates which ultimately form the overall structure of the image. The coordinates are calculated using drawing algorithms, the best being the force-directed algorithms in calculating the coordinates of the broadband network devices.

Visualization tool of the broadband network topology was implemented in the practical work of this master's thesis.

ALKUSANAT

Tämä diplomityö on tehty TeliaSoneran Lappeenrannan yksikössä.

Työn tarkastajana Lappeenrannan teknillisessä yliopistossa on toiminut professori Heikki Kälviäinen. Hänelle esitän kiitokseni työn ohjauksesta.

Työn ohjaajana TeliaSoneran puolesta on toiminut Timo Kujala, jolle haluan esittää lämpimät kiitokseni. Kiitokset myös Pasi Sirenille, Jyri Syväojalle, Mikko Tynkkyselle käytännön työn ohjauksesta sekä Jarkko Sikiölle ja Matti Siitoselle, jotka lukivat työni ja antoivat rakentavaa palautetta.

Haluan kiittää myös esimiestäni Harri Tumeliusta, joka on antanut mahdollisuuden tehdä tämän diplomityön.

Kiitokset myös perheelleni ja Eijalle tuesta ja vinkeistä tätä diplomityötä tehdessäni.

Lappeenrannassa 2. elokuuta 2004

SISÄLLYSLUETTELO

1 JOHDANTO 4

1.1 Tausta 4
1.2 Tavoitteet ja rajaukset 5
1.3 Työn rakenne 6

2 LAAJAKAISTAVERKKO 7

2.1 Laajakaistaverkon käsitteet ja laitteet 7
2.1.1 Käsitteet 8
2.1.2 Laitteet 9
2.2 Laajakaistaverkon arkkitehtuuri 10
2.3 Arkkitehtuurin fyysiset verkot 12
2.3.1 Frame Relay 13
2.3.2 ATM 14
2.3.3 xDSL 16
2.4 Arkkitehtuurin loogiset verkot 17
2.4.1 MPLS 17
2.4.2 VLAN 20
2.4.3 VPN 21

3 TOPOLOGIAN TIEDONKERÄYSTEKNIIKAT 23

3.1 SNMP 24
3.2 Hallintatietokanta 26
3.3 Topologian rakennus 29

4 VISUALISOINTI 31

4.1 Tausta 31
4.2 Peruskäsitteet 32
4.3 Graafin piirtoparametrit 34
4.3.1 Piirtämiskäytäntö 35
4.3.2 Esteettisyys 36
4.3.3 Rajoitukset 38
4.3.4 Tehokkuus 39
4.4 Piirtomallit 40
4.5 Voimiin perustuvien mallien algoritmit 43
4.5.1 Eadesin algoritmi 44
4.5.2 Fruchtermanin ja Reingoldin algoritmi. 46
4.5.3 GEM 48
4.5.4 Kamadan ja Kawain algoritmi 50
4.5.5 Simuloitu jäähdytysalgoritmi 55
4.6 Algoritmien vertailu 58
4.7 Algoritmien kompleksisuudet 59

5 SURFVISUAL-TYÖKALUN TOTEUTUS 63

5.1 SurfManager-ympäristö 63
5.2 SurfVisual toteutuksen työkalut 63
5.2.1 Perl 64
5.2.2 SVG 65
5.3 SurfVisual-työkalun rakenne 66
5.4 SurfVisual-työkalun elementin rakenne 68
5.5 SurfVisual-työkalun näkymiä 70

6 YHTEENVETO 74

VIITTEET 75

LIITTEET 82

LYHENNELUETTELO

ADSL	Asynchronous Digital Subscriber Line
AT	Address Translation
ATM	Asynchronous Transfer Mode
ATM-MIB	ATM-Management Information Protocol
CGI	Common Gateway Interface
CMIP	Common Management Information Protocol
CPAN	Comprehensive Perl Archive Network
DH-algoritmi	Davidsonin ja Harelin algoritmi
DLCI	Data Link Connection Identifier
DNS	Domain Name Server
DSLAM	Digital Subscriber Line Access Multiplexer
EGP	Exterior Gateway Protocol
Ethernet-CSMACD	Ethernet-Carrier Sense Multiple Access with Collision Detection
Etherlike-MIB	Etherlike-Management Information Base
FR-algoritmi	Fruchtermanin ja Reingoldin algoritmi
GEM	Graph embedder
HDSL	High-bit-rate Digital Subscriber Line
HTML	HyperText Markup Language
ICMP	Internet Control Message Protocol
Interface-MIB	Interface-Management Information Base
IP	Internet Protocol
IP-Sec	Internet Protocol Security
KK-algoritmi	Kamadan ja Kawain algoritmi
L2	Layer 2
L2TP	Layer 2 Tunneling Protocol
L3	Layer 3
LAN	Local Area Network
MAC	Media Access Control
MAN	Metropolitan Area Network

MIB	Management Information Base
MIB-II	Management Information Base version 2
MPLS	Multiprotocol Label Switch
NP	Nondeterministic Polynomial Time
OSI	Open System Interconnection
PERL	Practical Extraction and Report Language
PN	Private Network
PVC	Permanent Virtual Channel
QoS	Quality of Service
SDSL	Symmetric Digital Subscriber Line
SHDSL	Symmetric High-Bitrate Digital Subscriber Loop
SNMP	Simple Network Management Protocol
SNMPv2	Simple Network Management Protocol version 2
SNMPv3	Simple Network Management Protocol version 3
SVG	Scalable Vector Graphic
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
VCI	Virtual Channel Identifier
VLAN	Virtual Local Area Network
VLSI	Very Large Scale Integrated
VPI	Virtual Path Identifier
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
xDSL	Digital Subscriber Line
XML	Extensible Markup Language

1 JOHDANTO

1.1 Tausta

Viimeisten vuosien aikana laajakaistaoperaattoreiden laajakaistaverkot ovat nopeiden ja kiinteähintaisten laajakaistaliittymien johdosta kasvaneet suuriksi kokonaisuuksiksi. Laajakaistaverkkojen kokonaisuudet koostuvat runkoverkoista ja niihin kytkeytyneistä laajakaistaliittymistä, jotka sisältävät laitteita, laitteiden välisiä linkkejä ja tietoliikenneprotokollia. Suuret verkot sisältävät näin ollen suuren määrän eri tasoista tietoa laitteista ja laitteiden välisistä suhteista. Laitteiden välisistä linkeistä muodostuu myös verkon rakenne eli topologia. Ilman visualisointia suurten verkkojen topologioita on lähes mahdotonta hahmottaa.

Laajakaistaverkon topologian visualisoinnissa muodostetaan kuva laitteista ja niiden välisistä suhteista. Visualisoinnissa verkkojen laitteet ja laitteiden väliset suhteet muutetaan solmuiksi ja kaariksi, jolloin visualisointiongelma muuttuu graafin piirto-ongelmaksi. Perinteisesti graafin piirto-ongelma on ratkaistu piirtämällä graafi manuaalisesti piirto-ohjelmaa hyväksi käyttäen. Piirtämällä käyttäjä on saanut luotua graafin, mikä vastaa haluttua lopputulosta. Haluttu lopputulos koostuu selkeästä ja helposti luettavasta kuvasta. Vastaavanlaisen kuvan piirtäminen suuren verkon graafille on hankalaa ja aikaa vievää. Tällaisia piirto-ongelmia varten on kehitetty automaattiset piirtomenetelmät.

Automaattisilla piirtomenetelmillä kuvalle muodostetaan halutunlainen ulkoasu, mikä määräytyy käytettävän piirtomallin mukaan. Piirtomallia muuttamalla ulkoasu voidaan esittää mm. suuntaamattomana suorajanaisena, suunnattuna hierarkisena tai orthogonaalisena näkymänä. Voimiin perustuvat piirtomallit sopivat parhaiten laajakaistaverkkojen ulkoasujen mallintamiseen. Yksinkertaisen rakenteensa johdosta voimiin perustuva piirtomalli on helppo ymmärtää ja toteuttaa.

Voimiin perustuvien piirtomallien algoritmit voidaan toteuttaa jousiin tai optimointiin perustuvilla malleilla. Jousiin perustuvissa malleissa laitteiden väliset linkit korvataan kuvitteellisilla jousilla, joiden vetävien ja vastustavien voimien

minimoinnilla saavutetaan graafille symmetrinen ulkoasu. Kamadan ja Kawain algoritmi on yksi tunnetuimmista jousiin perustuvista algoritmeista. Optimointiin perustuvissa malleissa ensimmäisenä rakennetaan graafin ulkoasulle kustannusfunktio. Optimoiduilla funktionparametreilla kustannusfunktio tuottaa graafille halutun ulkoasun. Oikeiden funktionparametrien löytäminen vaatii vaikean optimointiongelman ratkaisemista, joka yleensä ratkaistaan fysikaaliseen analogiaan perustuvilla menetelmillä. Davidsonin ja Harelin algoritmi on ensimmäinen ja tunnetuin optimointiin perustuva algoritmi.

Tämä diplomityö on tehty TeliaSoneran Lappeenrannan yksikössä. Diplomityössä toteutettiin laajakaistaverkon topologian visualisointityökalu, joka on tarkoitus liittää osaksi TeliaSoneran SurfManager-verkonhallintajärjestelmää.

SurfVisual-visualisointityökalu täydentää jo olemassa olevaa IP Topology Manager -työkalua, jonka keräämien topologiatietojen pohjalta SurfVisual-työkalulla voidaan visualisoida eritasoisia topologiakuvia.

1.2 Tavoitteet ja rajaukset

Tämän diplomityön tarkoituksena on käsitellä laajakaistaverkon topologian visualisoiminen. Diplomityön pääpaino on keskittynyt enemmän topologian visualisointiin, kuin laajakaistaverkon rakenteen käsittelyyn. Laajakaistaverkon arkkitehtuurin läpikäyminen on kuitenkin välttämätöntä, jotta laajakaistaverkkojen sisältämien verkkojen topologiat tulevat tutuiksi. Laajakaistaverkosta ei ole tarkoitus käsitellä kaikkia mahdollisia käsitteitä, laitteita ja tekniikoita, joita laajakaistaverkot pitävät sisällään, vaan antaa yleiskuvaus ADSL-laajakaistaverkon (Asynchronous Digital Subscriber Line) arkkitehtuurin sisältämistä käsitteistä, laitteista ja tekniikoista.

Visualisoinnissa keskitytään graafin piirron peruskäsitteisiin, piirtomalleihin ja piirtoalgoritmeihin. Laajakaistaverkot koostuvat suurimmaksi osaksi suuntaamattomista verkoista, joiden topologioiden ulkoasujen piirtämiseen liittyvät piirtome-

netelmät käydään tarkemmin läpi. Suunnattujen verkkojen piirtomenetelmiin liittyvät asiat jätetään vähemmälle huomiolle. Suuntaamattomien verkkojen topologioiden ulkoasujen mallintamiseen soveltuvat parhaiten voimiin perustuvat piirtomallit, joihin perustuvat piirtoalgoritmit muodostavat diplomityön tärkeimmän osion.

1.3 Työn rakenne

Johdantoa seuraavassa toisessa luvussa käsitellään laajakaistaverkko yleisesti. Laajakaistaverkon arkkitehtuurista käsitellään ADSL-verkon osuus. Kolmannessa luvussa käsitellään laajakaistaverkon topologian tiedonkeräystekniikat. Neljäs luku sisältää visualisointiosuuden, jossa edetään graafin piirron peruskäsitteistä voimiin perustuviin algoritmeihin. Viidennessä luvussa kerrotaan käytännön työn toteutuksesta. Viimeisessä luvussa esitetään yhteenveto diplomityöstä.

2 LAAJAKAISTAVERKKO

Laajakaistaverkolla tarkoitetaan runkoverkon ja siihen kytkeytyneiden laajakaistaliittymien kokonaisuutta. Runkoverkolla tarkoitetaan tiedonsiirroltaan nopeaa, yleensä ATM-kytkentäistä (Asynchronous Transfer Mode) verkkoa, jonka avulla kaupunkien ja maiden väliset verkot yhdistetään toisiinsa kiinni. Laajakaistaliittymällä on alunperin tarkoitettu kiinteää 2 Mbps siirtonopeuteen kykenevää liittymää [Riu03], mutta operaattoreiden mainostamisen johdosta laajakaistaliittymällä tarkoitetaan nykyään kaikkia 64 kbps siirtonopeuteen kykeneviä kiinteitä liittymiä. Laajakaistaliittymä voidaan toteuttaa puhelinverkossa, kaapelitelevisioverkossa, valokaapelissa, langattomasti, satelliittiyhteyksin, matkapuhelinverkoissa tai sähköjohdoilla [Raj02]. Suosituin laajakaistaliittymä on puhelinverkossa toteutettu ADSL-liittymä [Laa04], jonka arkkitehtuuri käydään kappaleessa tarkemmin läpi.

Käsite laajakaistaverkko sisältää suuren määrän erilaisia laitteita, laitteiden välisiä linkkejä ja tietoliikenneprotokollia, jotka yhdistävät laitteet ja laitteiden väliset linkit saumattomaksi kokonaisuudeksi. Kappaleen tarkoituksena ei ole käydä seikkaperäisesti läpi kaikkia näitä laitteita ja tekniikoita, vaan tutustua keskeisimpiin laajakaistaverkon laitteisiin, tekniikoihin ja käsitteisiin, mitkä ovat esiintyneet käytännön työtä tehdessäni. Kappale aloitetaan käymällä laitteet ja käsitteet läpi, mikä helpottaa seuraavana käsiteltävien arkkitehtuurin tekniikoiden lukemista.

Työn aiheena on visualisoida laajakaistaverkon topologia, joten arkkitehtuurin tekniikoita pyritään ajattelemaan myös visualisoinnin näkökulmasta. Visualisoinnin näkökulmia ovat esimerkiksi tekniikoiden sisäiset topologiat ja topologiaparametrit.

2.1 Laajakaistaverkon käsitteet ja laitteet

Laajakaistaverkon arkkitehtuuri sisältää suuren määrän erilaisia käsitteitä ja laitteita, jotka on hyvä käydä läpi ennen kuin siirrytään tutkimaan teknologioita sy-

vällisemmin. Käsitteisiin kuuluvat verkot, laitteet, portit, linkit, liittymät ja yhteydet. ADSL-laajakaistaverkon kannalta tärkeimmät laitteet ovat keskitin, reititin, kytkin ja DSLAM (Digital Subscriber Line Access Multiplexer).

2.1.1 Käsitteet

Joskus samat laajakaistaverkon käsitteet voidaan ymmärtää monella eri tavalla, kirjoittajasta tai lukijasta riippuen. Tässä työssä käytetään seuraavia käsitteitä:

- Verkko: Kaikki laajakaistaverkon verkkoelementit (laitteet, portit, linkit, liittymät ja yhteydet) sisältyvät verkkoon. Laitteet ja niiden väliset linkit ja yhteydet muodostavat yhdessä verkkotopologian. Verkon topologia voidaan jakaa loogiseksi ja fyysiseksi topologiaksi. Fyysinen topologia tarkoittaa perinteistä verkkomallia, missä laitteet yhdistyvät toisiinsa fyysisten yhteyksien avulla. Loogisessa topologiassa laitteiden väliset yhteydet eivät vaadi fyysistä mediaa välilleen, vaan kaikki laitteet voivat olla enemmän tai vähemmän loogisesti yhteydessä toisiinsa [Tan03].

- Laitteet: Kytkin, reititin, keskitin ja työasema ovat keskeisimpiä laajakaistaverkon laitteita. Laitteiden ominaisuuksiin voidaan laskea sen fyysinen sijainti ja teknologian sanelemat ominaisuudet. Verkon topologiasta voi löytyä myös laitteita, mitkä eivät välttämättä ole fyysisiä laitteita vaan niiden ainoana tehtävänä on mallintaa verkon loogisuutta. Tällaisia laitteita kutsutaan metalaitteiksi. Fyysinen laite voi kuulua vain yhteen verkkoon, mutta metalaite voidaan jakaa useammaksi loogiseksi laitteeksi eri verkkoihin.

- Portti: Laitteet sisältävät portteja, jotka toimivat fyysisinä linkkeinä ulkomaailmaan [Ant98]. Portit voivat olla erityyppisiä esimerkiksi ethernet-, sarja- ja ATM-portteja.

- Linkki: Linkki on fyysinen yhteys kahden laitteen välillä. Linkki alkaa laitteen portista ja päättyy toisen laitteen porttiin. Portilla voi olla vain yksi linkki. Käytännössä tämä linkki voi olla kuparijohto, optinen kuitu tai ilmatie. Linkki kuvaa fyysistä yhteyttä laitteiden välillä, joten sen ominaisuudet riippuvat käytettävästä teknologiasta. Esimerkiksi ADSL-linkin ominaisuuksia ovat konstellaatiokuvio ja symbolinopeus [Lah98]. Linkin kummassakin päässä on käytettävä samaa tekniikkaa, mikä takaa laitteiden yhteensopivuuden. Linkit voivat yhdistää eri verkkoja keskenään.

- Liittymä: Liittymää voidaan pitää linkin erikoistapauksena. Liittymä on linkki runkoverkkoon. Liittymän ja linkin tekniikat ovat samanlaisia.

- Yhteys: Yhteys on kahden pisteen välinen looginen yhteys. Looginen yhteys ei välttämättä vaadi fyysistä yhteyttä laitteiden välillä. Looginen yhteys ei ole välttämättä edes samassa verkossa, missä päätelaitteet sijaitsevat. Looginen yhteys voi olla esimerkiksi ATM-verkon virtuaalipolku, missä yhteys ATM-verkossa näyttää yksinkertaiselta polulta, mutta käytännössä polku voi kulkea jokaisen ATM-kytkimen kautta päästäkseen loppupisteeseen. Kolmiosainen yhteys on yhteyden erikoistapaus, jossa yhteyden muodostus jaetaan kahteen osaan. Kolmiosaista yhteyttä joudutaan käyttämään tilanteissa, joissa lähdekone ei osaa yksin muodostaa yhteyttä kohdekoneeseen vaan tarvitsee avukseen ulkopuolisen laitteen. Esimerkiksi ADSL-laitteesta muodostetaan aluksi yhteys ulkopuoliseen DSLAM-laitteeseen (Digital Subscriber Line Access Multiplexer), josta DSLAM ottaa yhteyttä kohdekoneeseen eli reitittimeen [Lah98].

2.1.2 Laitteet

Laajakaistaverkot sisältävät paljon erilaisia laitteita, joista ADSL-laajakaistaverkon kannalta tärkeimpiä ovat keskitin, reititin, kytkin ja DSLAM.

- Keskitin: Keskitin on yksinkertainen laite, joka toistaa sisääntuloporttiin tulevan liikenteen kaikille ulostuloporteille. Keskitin toimii OSI-mallin (Open System Interconnection) fyysisellä kerroksella, jolloin paketin sisällöllä ei ole vaikutusta reititykseen.

- Kytkin: Kytkin toimii OSI-mallin toisella kerroksella, jolloin kehyksen kohdeosoite saadaan eroteltua paketin sisällöstä. Sisääntuloporttiin tulevat kehykset reititetään ainoastaan kohdekoneelle, jolloin keskittimien tapaista turhaa tiedonsiirtoa ei tapahdu. Kytkimet ovat piirikytkentäisien verkkojen tukipilareita, jolloin kehykset reititetään lähdekoneelta kohdekoneelle yhden polun kautta.

- Reititin: Reitittimiä käytetään pakettikytkentäisissä verkoissa pakettien siirtämiseen aliverkkojen välillä. Reititin toimii OSI-mallin kolmannella kerroksella, jolloin reitityksessä kohdeosoitteena käytetään verkko-osoitetta eli IP-osoitetta (Internet Protocol).

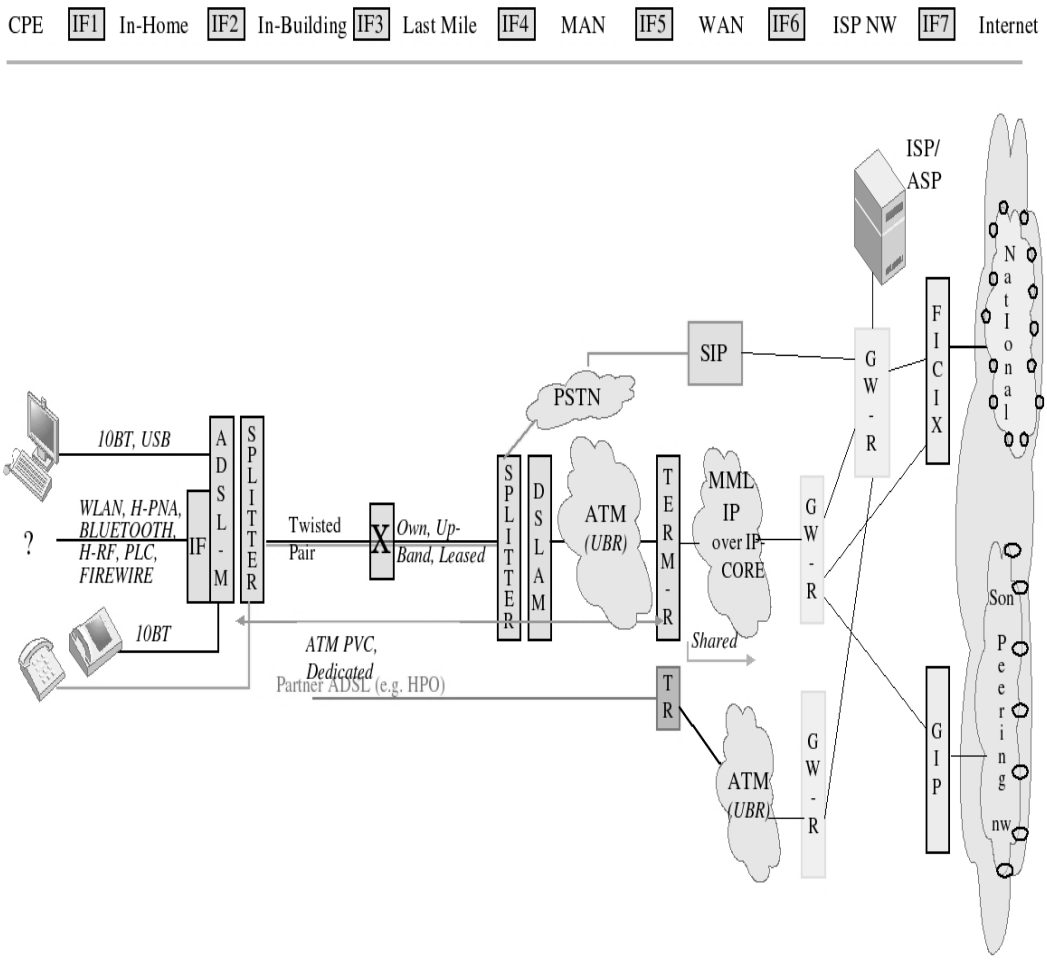
- DSLAM: Laajakaistaverkoissa DSLAM-komponentti sijaitsee palveluntarjoajan päässä. DSLAM-komponentin tärkeimpiin tehtäviin kuuluu laajakaistakäyttäjien yhteyksien liittäminen nopeaan runkoverkkoon (ATM-verkko). Lisäksi DSLAM erottelee puhelinlinjassa kulkevan datan ja puheen erilleen. DSLAM:in vastuulla on myös liikenteenhallinta käyttäjältä DSLAM:lle ja toisinpäin. [Ads97]

2.2 Laajakaistaverkon arkkitehtuuri

Edellä käydyn kappaleen perusteella todettiin laajakaistaverkon sisältävän suuren määrän erilaisia käsitteitä ja laitteita. Laajakaistaverkon arkkitehtuurissa käsitteet, laitteet ja tiedonsiirtotekniikat muodostavat lopullisen arkkitehtuurikokonaisuuden.

Laajakaistaverkon arkkitehtuuri on erittäin laaja käsite, minkä takia arkkitehtuurista käydään läpi ainoastaan ADSL-laajakaistaverkon arkkitehtuuri. ADSL-laajakaistaverkon arkkitehtuuri antaa hyvän kuvan myös muiden laajakaistaverkkojen arkkitehtuureista, koska usein laajakaistaliittymän jälkeinen osuus arkkitehtuurista pysyy samana, vaikka laajakaistaliittymä muuttuisikin.

Arkkitehtuuri voidaan jakaa osakokonaisuuksiksi, jotka muodostuvat kotikoneen ja Internetin välille (kuva 1). Osakokonaisuudet erotellaan toisistaan rajapinnoilla, kuvan yläreunan mukaisesti. Ensimmäiset rajapinnat voidaan yleistää LAN-(Local Area Network) käsitteeksi. Käsitteellä tarkoitettaan lähiverkon alueen laitteita ja tiedonsiirtotekniikoita.



Kuva 1. ADSL-laajakaistaverkon arkkitehtuuri.

Laajakaistaliittymien kannalta tärkeimpänä rajapintana voidaan pitää seuraava rajapintaa eli viimeistä mailia (Last Mile), mikä yhdistää käyttäjän laitteen operaattorin keskittimeen [Ker02]. xDSL-teknologiat (Digital Subscriber Line) lasketaan kuuluvaksi tähän osakokonaisuuteen.

Seuraava osakokonaisuus kattaa huomattavasti suuremman alueen, jolloin puhutaan MAN-verkosta (Metropolitan Area Network). MAN-verkko koostuu taajaman tai kaupungin kokoisesta alueesta, joka muodostuu ATM- ja Frame Relay -verkoista. [Tan03]

Laajin osakokonaisuus muodostuu WAN-verkosta (Wide Area Network), joka kattaa laajoja maantieteellisiä alueita. WAN-verkoissa käytetään samoja tekniikoita kuin MAN-verkoissa, mutta tiedonsiirtonopeudet ovat huomattavasti nopeampia. [Tan03]

Laajakaistaverkon arkkitehtuurin verkot voidaan jakaa fyysisiin ja loogisiin verkkoihin. Fyysisistä verkoista käsitellään xDSL-, Frame Relay - ja ATM-verkot ja loogisista verkoista VLAN (Virtual Local Area Network)-, MPLS (Multiprotocol Label Switch)- ja VPN-verkot (Virtual Private Network).

2.3 Arkkitehtuurin fyysiset verkot

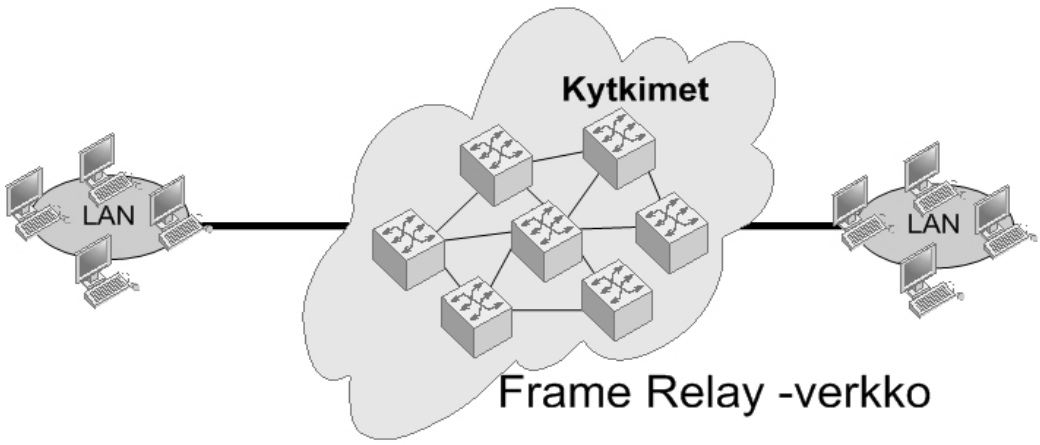
Arkkitehtuurin fyysisiin verkkoihin lasketaan laitteista ja tiedonsiirtoprotokollista muodostuvat verkot. Fyysiset verkot voidaan jakaa runkoverkkoihin ja laajakaistaliittymiin. Runkoverkoista käsitellään ATM- ja Frame Relay -verkot ja laajakaistaliittymistä xDSL-verkko.

2.3.1 Frame Relay

Frame Relay -tiedonsiirtoprotokolla toimii monien pienempien yritysten verkkojen yhdistävänä tiedonsiirtoprotokollana [Tan03], vaikkakin ATM-tekniikka on syrjäyttämässä tätä vanhaa tekniikkaa vähitellen pois. Suurimpana syynä Frame Relay -tekniikan olemassa ololle voidaan pitää tekniikan edullisempaa hintaa verrattuna esimerkiksi kilpailevaan ATM-tekniikkaan. Frame Relay -verkkojen yhteensopivuus ATM-verkkojen kanssa on myös suuri tekijä Frame Relay -tekniikan suosioon [Atm04, Mpl94].

Frame Relayn siirtonopeus 50 Mbps ei ole enää kilpailukykyinen nykyisten vastaavien gigabittisten tekniikoiden rinnalla, mutta yhdistämällä Frame Relay-verkon päätelaitteet ATM-verkon yhdyskäytävällä saadaan tekniikasta huomattavasti suorituskykyisempi [Mpl94].

Frame Relay -verkon topologia (kuva 2 [Spr01]) koostuu Frame Relay -kytkimistä ja kytkimien välisistä suhteista. Kytkimien välillä kulkevat fyysiset linkit voidaan ajatella putkiksi, joiden sisällä kulkee kanavia. Kanavia kutsutaan virtuaalisiksi kanaviksi (Permanent Virtual Channel), jotka voivat muodostaa yhden tietoyhteyden kahden portin välille. Virtuaaliset kanavat erotetaan toisistaan DLCI- (Data Link Connection Identifier) parametrin avulla, mikä vastaa virtuaalisenkanavan id-numeroa. [Cis04a]



Kuva 2. Frame Relay -verkon arkkitehtuuri.

Frame Relay -verkon topologian visualisoinnissa jokainen erillinen kytkin piirretään ja nimetään omalla nimellään. Kytkimet yhdistävät linkit piirretään viivoina kytkinten välille. Linkkien alku- ja loppupäähän lisätään virtuaalisten kanavien tunnukset eli DLCI-parametrien arvot. Kytkimien välisten linkkien lukumäärä kasvaa usein suureksi, jolloin kytkimien välille piirretään yksi paksu viiva. Paksu viiva toimii hyperlinkkinä uudelle sivulle, jossa visualisoidaan ainoastaan kyseisten kytkimien väliset linkit.

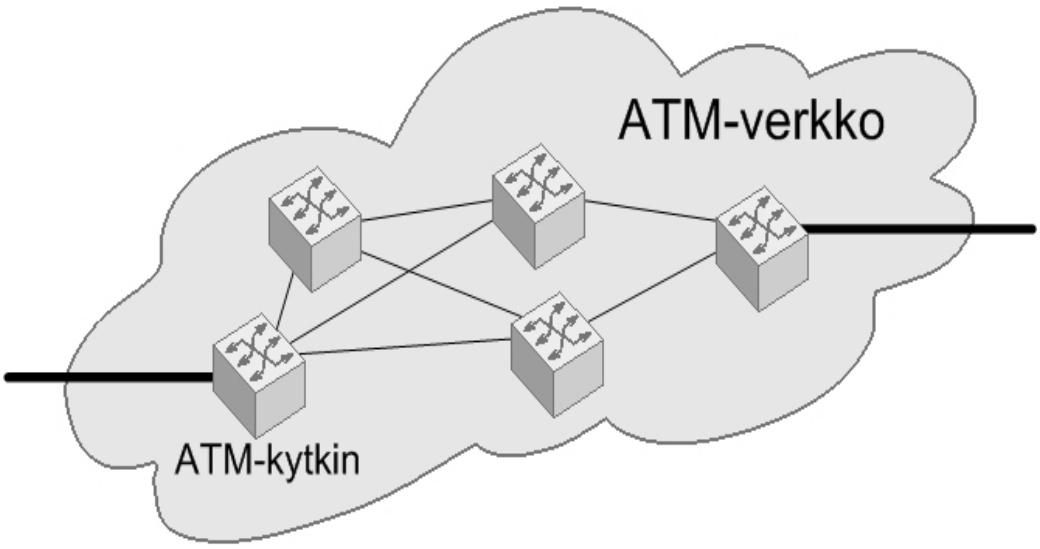
2.3.2 ATM

ATM-tiedonsiirtotekniikkaa voidaan perustellusti sanoa internetin tukipilariksi, koska noin 80 prosenttia internetin runkoverkoista käyttää ATM-tiedonsiirtotekniikkaa hyväkseen [Xil04a].

ATM-tiedonsiirtotekniikan suosio perustuu tekniikan skaalautumiseen eritasoisiin verkkoihin. Skaalattavista ominaisuuksista siirtomedian tyyppi, tiedonsiirtonopeus ja laitteiden väliset välimatkat voidaan muuttaa verkon vaatimusten mukaisiksi. Verkossa voidaan siirtää ääntä, dataa tai videokuvaa samanaikaisesti ilman ongelmia [Tan03]. Erityyppinen media voidaan lisäksi lähettää joko broadcast- tai unicast-lähetyksinä [Cis04b], joten ATM soveltuu hyvin myös kaapeli-tv-verkkoihin. Ongelmattoman tiedonsiirron takaa suuri tiedonsiirtonopeus, joka skaalautuu parista megabitistä aina kymmeniin gigabitteihin/sekunti. Lisäksi ATM-verkon laitteiden välimatkoja voidaan kasvattaa useisiin satoihin kilometreihin siirtonopeuden tästä kumminkaan kärsimättä.

Siirtotekniikan vahvuus perustuu datavirran pilkkomiseen kiinteän kokoisiksi soluiksi (54 tavua). Kiinteän kokoinen ja itsestään reitittyvä solu mahdollistaa prosessoinnin kytkimissä, kytkentätauluissa ja väylissä suoraan fyysisillä piireillä, jolloin systeemi voidaan rakentaa ilman hitaampaa ohjelmallista prosessointia. [Pry95]

ATM-verkon topologia (kuva 3) koostuu ATM-kytkimistä ja kytkimien välisistä suhteista. Jokainen ATM-kytkin sisältää usein lukuisia portteja, joista muodostetaan linkit toisten kytkimien porttien välille. Kytkimien välillä kulkevat viivat voidaan ajatella putkiksi, joiden sisällä kulkee lukuisia pienempiä putkia. Paksumpaa putkea kutsutaan virtuaaliseksi poluksi (Virtual Path), jonka sisällä kulkee pienempiä virtuaalisia kanavia (Virtual Channel) [Cis04b]. Virtuaalisten polkujen ja -kanavien avulla voidaan porttien välinen linkki limittää (multiplexing) useammaksi linkiksi.



Kuva 3. ATM-verkon arkkitehtuuri.

Yhteyden rakentamisessa käyttäjä muodostaa yhteyden reunakytkimeen, jossa kytkentätaulun perusteella valitaan käytettävä virtuaalipolku ja -kanava. Virtuaalipolun ja kanavan perusteella määräytyy myös seuraava ATM-kytkin, jossa suoritetaan seuraava kytkentätaulun prosessointi. Näin edetään kunnes saavutetaan ATM-verkon toinen reunakytkin, jolloin yhteydelle on muodostunut putki ATM-verkon läpi.

ATM-verkon topologian visualisointia voidaan pitää lähes identtisenä Frame Relay -verkon kanssa (kappale 2.3.1). Verkkojen visualisoinneissa poikkeavat ainoastaan linkkeihin lisättävien tunnuksien parametrit. ATM-verkossa linkkeihin lisä-

tään virtuaalipolkujen ja virtuaalikanavien parametrien arvot. Linkkeihin lisätään myös linkin pään tyyppi (lähde/kohde), jolloin linkin suunta on nähtävissä.

2.3.3 xDSL

xDSL on yleisnimitys kaikille xDSL-perheen (ADSL, HDSL (High-bit-rate Digital Subscriber Line), SDSL (Symmetric Digital Subscriber Line), SHDSL (Symmetric High-Bitrate Digital Subscriber Loop), jne.) laajakaistaliittymille. ADSL on toistaiseksi xDSL-perheen suosituin laajakaistaliittymä [Lig03, Laa04], jonka tekniikka ja arkkitehtuuri käydään xDSL-perheestä tarkemmin läpi.

ADSL-laajakaistaliittymällä muodostetaan yhteys käyttäjältä operaattorin puhelinkeskukseen, josta yhteyden muodostamista internetiin jatketaan muita tekniikoita hyväksikäyttämällä (kuva 1). ADSL on digitaalinen tiedonsiirtotekniikka, missä siirtomediana käytetään puhelinverkon kuparikaapelia. Puhelinverkon kuparikaapelissa ääni ja data voi liikkua samanaikaisesti, koska jakajalla (splitter) äänen ja datan taajuusalueet saadaan erotettua toisistaan (kuva 1). Koska tekniikka on rakentunut jo olemassa olevan puhelinverkon päälle, ovat ADSL-laajakaistaliittymät tulleet suosituksi niiden kohtuullisten hintojensa puolesta. Erityisesti ADSL on saavuttanut suosiota sen asymmetrisen luonteen johdosta. Asymmetrinen tiedonsiirto soveltuu hyvin verkko-selailuun, jossa sisään tulevan datan osuus on huomattavasti suurempi kuin ulospäin lähtevän. ADSL-tekniikassa sisään tulevan datan maksimi siirtonopeus on 8Mbps ja ulos lähtevän 1Mbps. [Ker02]

ADSL-yhteyttä muodostettaessa käyttäjän ADSL-modeemi ottaa yhteyden DSLAM-keskittimelle, joka kokoaa kaikkien lähiverkon asiakkaiden modeemiyhteydet yhteen paikkaan ja reitittää yhteydet ATM-verkolle tai muulle vastaavalle siirtoyhteydelle (kuva 1) [Xil04].

Laajakaistaverkon visualisoinnissa ADSL-verkon topologiasta piirretään yleensä vain DSLAM-keskitin. Laajakaistaverkon kokonaisuudessa käyttäjien ADSL-

modeemit jätetään yleensä piirtämättä, koska yksittäisten modeemien tiedot eivät ole oleellisia verkon rakenteen kannalta. Modeemien tarkemmat tiedot puuttuvat usein myös tietokannasta, koska modeemeista puuttuu SNMP-tuki (Simple Network Management Protocol). Näin ollen ADSL-modeemit esitetään tietokannassa metalaitteina, joiden perusteella loogisen topologian rakentaminen on tarvittaessa mahdollista.

2.4 Arkkitehtuurin loogiset verkot

Arkkitehtuurin loogisiin verkkoihin lasketaan tiedonsiirtoprotokollan avulla toteutetut verkot. Loogiset verkot sisältyvät jo olemassa olevan verkon päälle, jonka ominaisuuksia laajennetaan ja tehostetaan uuden tiedonsiirtotekniikan avulla. Loogisista verkoista käsitellään MPLS- ja VLAN- ja VPN-verkot.

2.4.1 MPLS

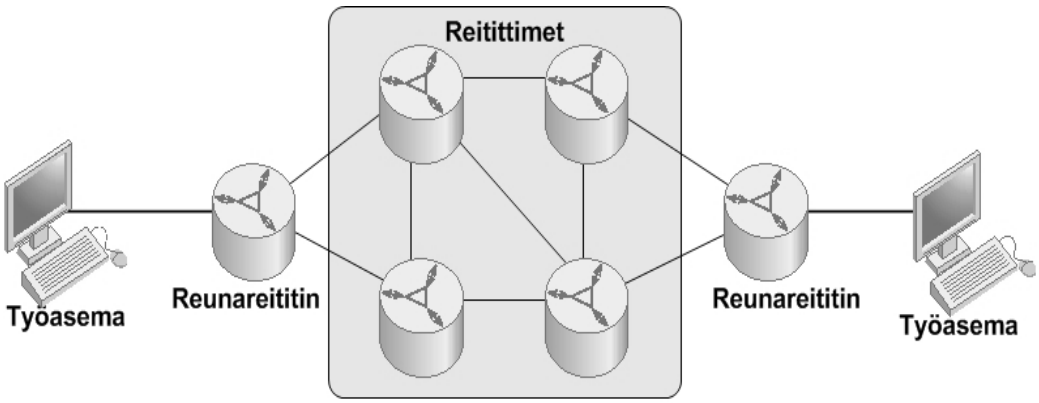
Perinteisissä yhteydettömissä IP-verkoissa paketti kulkee reitittimeltä reitittimelle ja jokaisessa reitittimessä paketille lasketaan seuraavan reitittimen osoite. Seuraavan reitittimen osoite lasketaan kohdeosoitteen perusteella, jonka laskeminen vaati monia työvaiheita ja huomattavasti reitittimen resursseja. [Tan03]

MPLS-tekniikka tarjoaa uuden näkökannan perinteisien tekniikoiden rinnalle. MPLS-tekniikan avulla voidaan yhteydettömästä verkosta tehdä yhteydellinen, kuten ATM- ja Frame Relay-verkot.

MPLS-verkossa jokainen yhteys merkitään tunnuksella (label), jonka perusteella paketti kulkee MPLS-verkon läpi seuraavaan verkkoon. Reititys MPLS-verkon sisällä on näin ollen nopeaa, koska mitään varsinaista reitityksen laskemista ei tarvitse suorittaa. Yhteyksien merkitseminen listaa samalla MPLS-verkon resurssien tilan, joten yhteyksille voidaan taata haluttu palvelutaso (QoS) [Cis04c]. Reititys ei rajoitu pelkästään IP-verkkoon, vaan paketteja voidaan ohjata myös ATM-

ja Frame Relay-verkkojen sisällä. L2-tason verkkoihin MLPS-tekniikka tuo loogisuutta, joka kasvattaa verkon hallittavuutta.

MPLS-verkon arkkitehtuuri (kuva 4 [Cis04c]) koostuu reunareitittimistä (Edge Label Switch Router) ja reitittimistä (Label Switch Router). Reunareitittimet linkittyvät toisiinsa reitittimien välityksellä ja käyttäjän laitteet yhdistyvät MPLS-verkkoon reunareitittimen rajapinnasta.

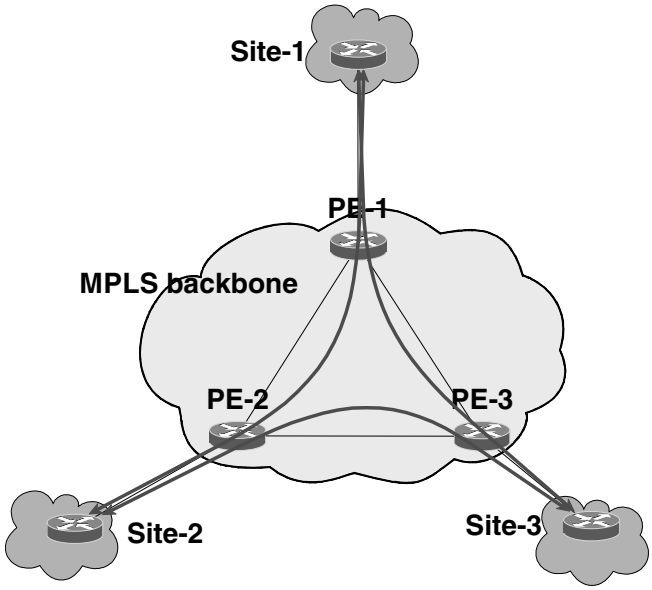


Kuva 4. MPLS-verkon arkkitehtuuri.

Käyttäjän laitteen kannalta MPLS-verkko on tavallinen IP-runkoverkko, joka ei osallistu MPLS:n toimintaan. Vastaavasti reunareitittimillä on suurin vastuu reitityksen kannalta, koska reunareitittimessä päätetään koko polku, miten paketti tullaan reitittimään MPLS-verkon toisella reunalla sijaitsevalle vastinreitittimelle.

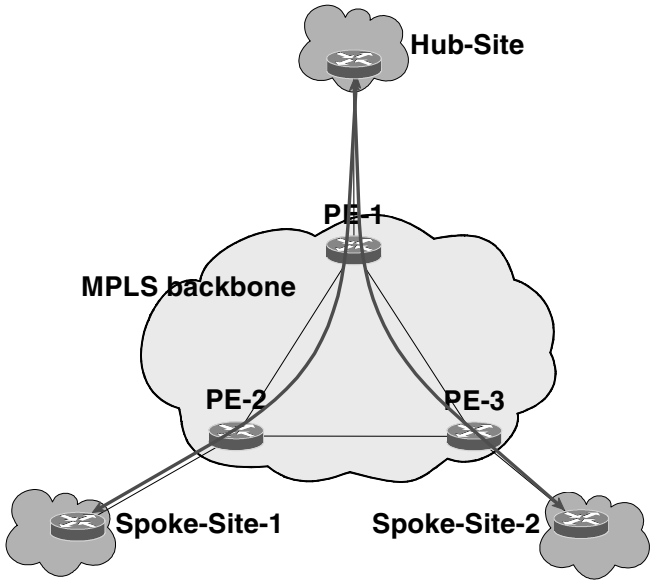
MPLS-verkon sisälle voidaan lisäksi rakentaa topologioiltaan eri tyyppisiä virtuaaliverkkoja, joilla voidaan rajata verkon liikennettä sisään- ja ulospäin.

Fullmesh-virtuaaliverkossa (kuva 5) kaikki laitteen näkevät toisensa. Tummat viivat osoittavat, kuinka tieto kulkee vapaasti MPLS-verkon sisällä käyttäjältä toiselle.



Kuva 5. Fullmesh-virtuaaliverkko.

Hub and Spoke-virtuaaliverkossa (kuva 6) Spoke-site-laitteet näkevät vain hub-site-laitteen. Vastaavasti Hub-site-laite näkee kaikki Spoke-site-laitteet. Viivojen perusteella nähdään, kuinka Spoke-Site-1 on tietämätön Spoke-Site-2-laitteen olemassa olosta.



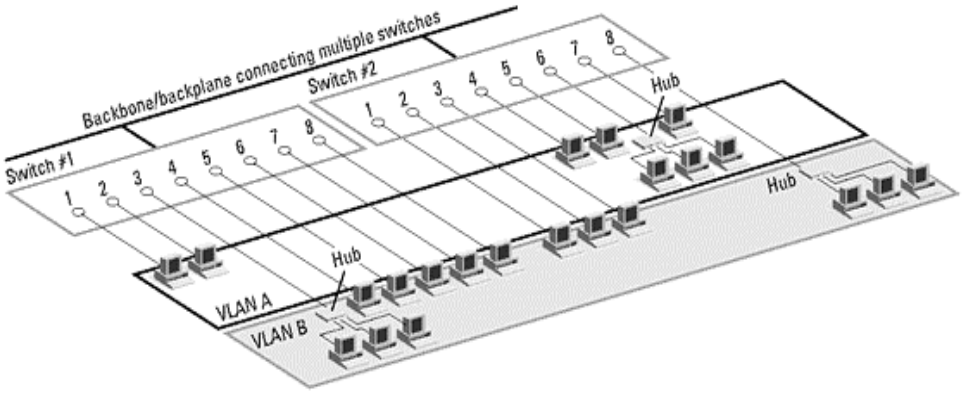
Kuva 6. Hub and Spoke –virtuaaliverkko.

Visualisoinnin näkökulmasta katsottuna MPLS-verkon topologian piirtäminen poikkeaa aiemmin käsitellyistä fyysisten verkkojen visualisoinnista, koska MPLS-verkko visualisoidaan jo olemassa olevan verkon päälle. Kahden päällekkäisen verkon tapauksessa MPLS-verkon rakenne mallinnetaan fyysisen verkon linkkeihin nuolilla ja eri väreillä. MPLS-verkon virtuaaliverkot voidaan mallintaa samalla tavalla.

2.4.2 VLAN

Ennen virtuaali-lähiverkkoja (VLAN) suurten organisaatioiden lähiverkkojen aliverkot jaettiin suoraan fyysisen sijainnin mukaan. Fyysiseen sijaintiin perustuvat aliverkot olivat selkeitä kokonaisuuksia, mutta asettivat organisaation verkolle rajoituksia ja heikkouksia [Tan03]. Perinteisen verkon heikkouksista onkin muodostunut VLAN-tekniikan vahvuuksia.

VLAN-tekniikka mahdollistaa aliverkkojen muodostamisen loogisen jaottelun mukaisesti (kuva 7), jolloin laitteiden fyysisillä sijainneilla ei ole merkitystä [Pas96]. Looginen jaottelu parantaa aliverkkojen tietoturvaa, jolloin ainoastaan saman loogisen ryhmän koneilla on mahdollisuus nähdä oman verkkonsa verkkoliikennettä [Tun03].



Kuva 7. VLAN-arkkitehtuuri. [Pas96]

Lähiverkon kuorman jakamista eri segmentteihin voidaan pitää myös VLAN-tekniikan vahvuutena. Esimerkiksi yritysten kehityspuolen osastoissa verkon kapasiteettiä rasitetaan huomattavasti toisia osastoja enemmän, jolloin kehittäjät voidaan rajata omaksi virtuaaliseksi segmentikseen. [Tun03]

VLAN-tekniikan vahvuuksia on lisäksi broadcast-viestien lähetysten hallinta. Virtuaalisegmenttien broadcast-viestit lähetetään ainoastaan oman verkon laitteille, jolloin erityisesti suuremmissa verkoissa säästetään huomattavasti verkon resursseja. [Tun03]

VLAN-arkkitehtuuri (kuva 7) koostuu keskittimistä ja kytkimistä. Keskittimillä kootaan fyysisesti lähekkäin olevat koneet omaksi kokonaisuudeksi, joka liitetään virtuaaliverkon jäseneksi kytkinten avulla. Varsinaiset VLAN-topologiat muodostetaan portti-, MAC (Media Access Control)- tai IP-osoitteen perustella [Pas96].

Visualisoinnin näkökulmasta katsottuna VLAN-verkon topologian mallintaminen tapahtuu samalla tavalla kuin MPLS-verkon, koska VLAN-verkko visualisoidaan jo olemassa olevan verkon päälle. VLAN-verkkojen virtuaaliverkot voidaan erottaa toisistaan eri väreillä, jolloin saman virtuaaliverkon kytkimet ja keskittimet ovat saman värisiä.

2.4.3 VPN

Ennen julkisen verkon (internet) syntymistä organisaatiot rakensivat verkkonsa puhelinoperaattoreilta vuokrattujen kaistojen päälle. Vuokratun kaistan päälle rakennettu sisäverkko (Private network) takasi organisaatiolle luotettavan ja turvallisen verkkoratkaisun. Verkkoratkaisun hyvien ominaisuuksien takia jotkut organisaatiot rakentavat vieläkin osan verkoistaan operaattoreilta vuokratun kaistan päälle. Sisäverkon ainoa ongelma on sen kohtuuttoman kallis hinta, minkä johdosta korvaava VPN-tekniikka on kehitetty. [Tan03]

VPN-tekniikalla pyritään saavuttamaan alkuperäisen sisäverkon edut muodostamalla virtuaalinen sisäverkko julkisen verkon päälle. Julkisen verkon päälle rakennettu virtuaalinen sisäverkko on huomattavasti kaistan vuokrausta halvempi ja näin ollen yleisin organisaatioiden sisäverkkojen toteutustekniikka.

VPN voidaan jakaa kolmeen kategoriaan:

- Salattu-VPN
- Luotettu-VPN
- Hypridi-VPN

Salatun VPN-kategorian VPN-yhteydet muodostetaan tunneloimalla suojaamaton liikenne salaavan VPN-protokollan sisään. Yleisimpiä VPN-protokollia ovat IP-Sec (Internet Protocol Security) ja L2TP (Layer 2 Tunneling Protocol). [Vpn03] Luotetun VPN-kategorian yhteydet rakennetaan ATM- ja Frame Relay -verkkojen päälle virtuaalisilla piireillä. Usein VPN-yhteyden virtuaalinen piiri rakennetaan MPLS-tekniikkaa hyväksikäyttämällä. [Vpn03] Hypridin VPN-kategorian yhteydet rakennetaan salattujen VPN- ja luotettujen VPN-yhteyksien yhdistelmällä.[Vpn03]

Visualisoinnin näkökulmasta katsottuna luotetun VPN-kategorian yhteydet ovat visualisoitavissa. VPN-verkon topologia muodostuu MPLS-verkkojen virtuaalipiireistä. VPN-verkon topologiat visualisoidaan näin ollen samalla tavalla kuin MPLS-verkot.

3 TOPOLOGIAN TIEDONKERÄYSTEKNIIKAT

Laajakaistaverkon visualisoimiseen tarvitaan aina myös verkon rakenteen tiedot, minkä pohjalta visualisoiminen suoritetaan. Usein laajakaistaverkon topologian tiedot on saatavilla keskitetystä tietokannasta, mistä visualisointityökalu hakee tiedot ja mallintaa verkon näiden tietojen pohjalta. Topologiatiedot voidaan kysyä ajonaikaisesti myös suoraan laitteilta, jolloin mallinnettava tieto on varmasti reaaliaikaista. Ajonaikainen tiedonkeräys olisikin optimaalisen vaihtoehto, ellei tiedonkeräys johtaisi suuremmilla verkoilla pitkiin viiveisiin, jotka tekevät interaktiivisen selailun mahdottomaksi. Paras ratkaisu saadaan aikaiseksi yhdistämällä ajonaikainen haku ja keskitetty tietokanta yhdeksi kokonaisuudeksi. Yhdistetyssä kokonaisuudessa tietokannan topologiatietoja päivitetään ajastetusti ajonaikaisen topologiatiedonkeräjään avulla.

Laajakaistaverkot sisältävät suuren määrän laitteita ja laitteita yhdistäviä linkkejä, joten olemassa olevaa tietoa on paljon tarjolla. Suurin osa tarjolla olevasta tiedosta on epäoleellista topologian kannalta, joten oleellisen tiedon löytäminen voi olla haastavaa, ellei läpikäytävässä tiedossa ole selkeää rakennetta.

Laajakaistaverkon topologiatiedot sijaitsevat joko siirtoyhteys- tai verkkokerroksella. Siirtoyhteyskerros sijaitsee OSI-mallin toisella kerroksella, jota kutsutaan myös L2-kerrokseksi (Layer 2). Verkkokerros sijaitsee OSI-mallin kolmannella kerroksella, jota vastaavasti kutsutaan L3-kerrokseksi (Layer 3). [Tan03]

Topologian ajonaikaisista tiedonkeräystekniikoista SNMP-verkonhallintatyökalu kykenee L2- ja L3-kerroksien tasoiseen tiedonkeräykseen. SNMP:n korvaajaksi tarkoitettu CMIP (Common Management Information Protocol) pystyy vastaavaan tiedonkeräykseen, mutta tekniikka ei ole laajasti käytössä [Sta93]. Topologian ajonaikaisista tiedonkeräystekniikoista L3-kerroksen tasoiseen tiedonkeräykseen pystyvät monet työkalut, jotka perustuvat perinteisiin Ping- ja Traceroute-ohjelmiin. L3-kerroksen tasoisessa tiedonkeräyksessä voidaan nimipalvelun DNS (Domain Name Server) tietoja käyttää hyväksi [Kes98].

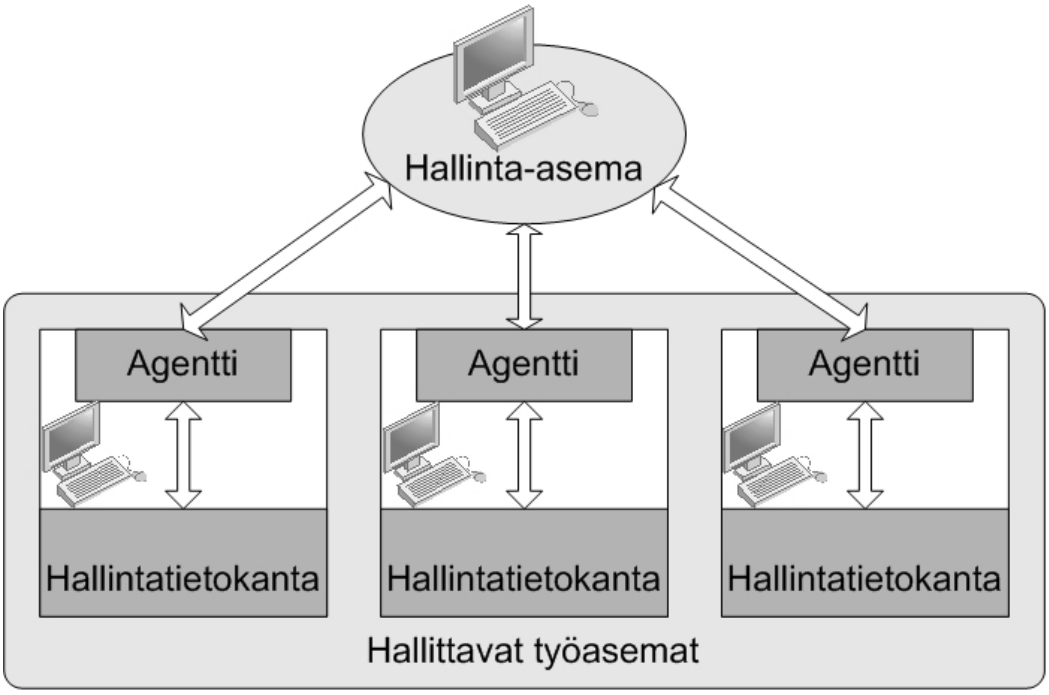
Laajakaistaverkko sisältää L2- ja L3-kerroksen tasoista topologiatietoa, joten kappaleessa käsitellään tarkemmin SNMP-verkonhallintatyökalun tekniikka. Kappaleessa käydään läpi myös SNMP-verkonhallintatyökalun MIB-II (Management Information Base) hallintatietokanta ja sen yhteydet laajakaistaverkkoon. Lopuksi käydään esimerkin mukaisesti läpi, kuinka MIB-II hallintatietokannan tietojen perusteella rakennetaan laajakaistaverkon L3-kerroksen tasoinen topologiarakenne.

3.1 SNMP

SNMP oli ensimmäinen verkonhallintaprotokolla, joka ilmestyi 1980-luvun puolivälissä. SNMP:n kehitys eteni hyvin nopeasti ja sen oli tarkoitus olla väliaikainen ratkaisu verkonhallintaan, kunnes parempi ja korvaava protokolla saataisiin määriteltyä. SNMP osoittautui käytännössä hyvin toimivaksi ja on yhä laajassa käytössä maailmalla [Tur00, Sta93]. Seuraavissa SNMP-versioissa (SNMPv2, SNMPv3) on keskitytty lähinnä alkuperäisen protokollan tietoturvapuutteellisuuksien korjaamiseen [Int04].

SNMP-malli sisältää seuraavat komponentit (kuva 8 [Cis04]):

- Hallinta-asema (Management station).
- Hallinta-agentti (Management agent).
- Hallintatietokanta (Management information base).
- Hallintaprotokolla (Network management protocol).



Kuva 8. SNMP-malli.

Hallinta-asemasta käsin hallitaan koko verkon toimintaa. Hallinta-asema muuttaa käyttäjän toiminnot käskyksi, joiden perusteella hallinta-agentit toimivat. Hallinta-asema sijaitsee tyypillisesti erillisessä koneessa. Hallinta-agentit ovat ohjelmamoduuleita, jotka sijaitsevat hallittavissa laitteissa. Hallinta-agentit toimivat hallinta-aseman käskyjen mukaan, mutta voivat toimia myös itsenäisesti. Jokainen hallittava laite sisältää objekteja sisältävän hallintatietokannan. Jokainen objekti sisältää jonkun laitteen ominaisuuden, joka on hierarkkisen hallintatietokannan mukainen. Hallintaprotokolla toimii TCP/IP-verkon (Transmission Control Protocol/Internet Protocol) päällä ja toimittaa hallinta-aseman käskyt yleensä UDP-paketteina (User Datagram Protocol) kohdekoneille, jolloin pakettien uudelleen lähetyksestä vastaavat hallinta-asema ja hallinta-agentit. [Int04]

Varsinainen SNMP-protokolla mahdollistaa seuraavat perusoperaatiot;

- Get: Laitteilta haetaan hallintatietokannan mukaisia muuttujien arvoja.
- Set: Laitteille asetetaan hallintatietokannan mukaisia muuttujien arvoja.

- Trap: Laitteet voivat lähettää itsenäisesti sanomia hallinta-asemalle. Yleensä lähetettävä sanoma kertoo laitteen kytkeytymisestä tai sulkeutumisesta.

Laitteen on tuettava SNMP-protokollaa, jotta muuttujien hakeminen ja asettaminen on mahdollista.

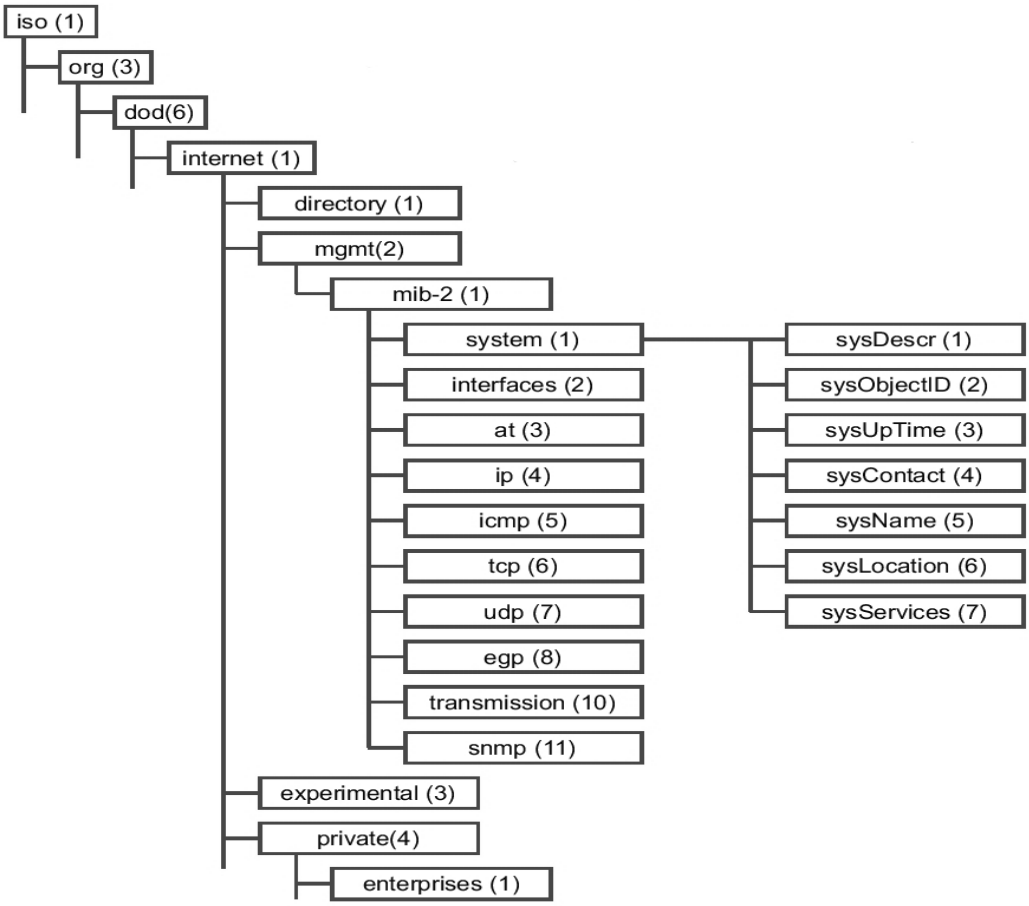
3.2 Hallintatietokanta

SNMP-verkonhallintaprotokollassa laitekohtaiset tiedot on järjestetty hierarkkiseen hallintatietokantaan (MIB). Hallintatietokannoista yleisemmin käytössä on standardi MIB-II, jossa määritellään monia yleisiä laite-, portti- ja protokollakohtaisia muuttujia. Laitevalmistajilla on myös omia hallinnointitietokantoja, koska yleinen MIB-II ei kata kaikkia valmistajan laitteen ominaisuuksia. Muita yleisiä hallintietokantoja ovat ATM-MIB, Interface-MIB ja EtherLike-MIB. Erityisesti Interface-MIB:iä tuetaan laajasti, koska se sisältää lisätietoa MIB-II interface-ryhmän porteista [Lah98].

Hallinnointitietokannan muuttujiin viitataan hierarkkisen tunnuksen mukaisesti (kuva 9). Esimerkiksi MIB-2 objektiin viitattaan iso (1), org (3), dod (6), internet (1), directory (2), mgmt (1) ja MIB-II (1) polun mukaisesti eli 1.3.6.1.2.1 [Sta93]. MIB-II hallintatietokanta jaetaan kymmeneen erilliseen ryhmään, joista jokainen sisältää lukuisia ryhmäkohtaisia muuttujia [RFC91]:

- system: yleistä tietoa järjestelmästä.
- interface: fyysisten porttien tilatietoja.
- at (address translation) : osoitteen muunnostietoa.
- ip: IP-protokollatietoa.
- icmp: ICMP-protokolla (Internet Control Message Protocol) tietoa.
- tcp: TCP-protokolla (Transmission Control Protocol) tietoa.
- udp: UDP-protokollatietoa.
- egp: EGP-reititysprotokolla (Exterior Gateway Protocol) tietoa.

- transmission: siirtokerroksen toiminnan tietoa.
- snmp: SNMP-protokollatietoa.



Kuva 9. MIB-II hierarkkinen rakenne. [Int04]

System-, interface- sekä ip-ryhmän muuttujat sisältävät visualisoinnissa hyväksikäytettävää tietoa. System ryhmän sysDescr, sysName ja sysLocation muuttujissa määritellään visualisointi- ja topologiakohtaisia tietoja. Seuraavassa kyseisten muuttujien polkujen tunnukset:

- 1.3.6.1.2.1.1 - SNMP MIB-2 System
  - 1.3.6.1.2.1.1.1 – sysDescr
  - 1.3.6.1.2.1.1.5 – sysName
  - 1.3.6.1.2.1.1.6 – sysLocation

SysDescr-muuttujassa määritellään laitteen tyyppi, käyttöjärjestelmä ja muita laitekohtaisia tietoja. Visualisoinnin kannalta laitteen tyyppiä voidaan pitää oleellisena tietona, koska laitteen tyypin perusteella valitaan graafissa esitettävä ikoni. SysName-muuttujassa määritellään laitteen nimi. SysLocation-muuttujassa määritellään laitteen fyysinen sijainti. SysLocation-muuttujan arvoa voidaan käyttää hyväksi mallinnettaessa suurempia maantieteellisiä alueita (Suomen kartta). [Sta93, RFC91]

Interface-ryhmän muuttujien arvot käsittelevät laitteiden porttien ominaisuuksia ja tiloja, joten ryhmän muuttujat sisältävät topologiakohtaista tietoa. Seuraavassa interface-ryhmän muuttujien polkujen tunnukset:

- 1.3.6.1.2.1.2 - SNMP MIB-2 Interfaces
  - 1.3.6.1.2.1.2.2 – ifTable
    - 1.3.6.1.2.1.2.2.1 - ifEntry
    - 1.3.6.1.2.1.2.2.1.3 - ifType
    - 1.3.6.1.2.1.2.2.1.6 – ifPhysAddress
    - 1.3.6.1.2.1.2.2.1.7 - ifAdminStatus

IfTable-objekti sisältää taulukon kaikista laitteen porteista, joiden ominaisuudet löytyvät ifEntry-objektin muuttujista. IfType-muuttujassa määritellään portin tyyppi. Portin tyyppejä voivat olla mm. ethernet-csmacd (6), dsl (18) ja Frame Relay (32). IfPhysAddress-muuttujassa määritellään portin laiteosoite eli MAC-osoite. IfAdminStatus-muuttujassa määritellään portin tila. Portti voi olla käytössä, suljettuna tai testaustilassa. [Sta93, RFC91]

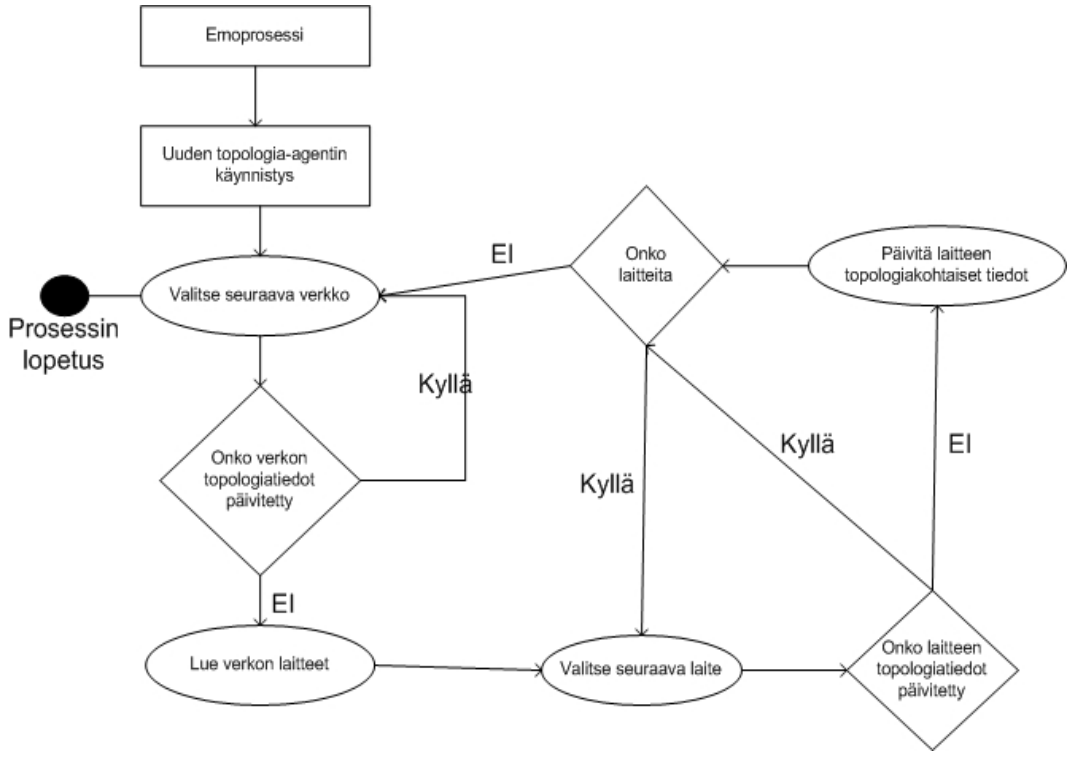
IP-ryhmän muuttujia voidaan pitää tärkeimpinä IP-topologian kannalta, koska ryhmässä määritellään laitteiden porttien IP-osoitteet. Seuraavassa tärkeimpien ip-ryhmän muuttujien polkujen tunnukset:

- 1.3.6.1.2.1.4 – ip
  - 1.3.6.1.2.1.4.20 – ipAddrTable
    - 1.3.6.1.2.1.4.20.1 – ipAddrEntry
    - 1.3.6.1.2.1.4.20.1.1 - ipAdEntAddr
    - 1.3.6.1.2.1.4.20.1.3 - ipAdEntNetMask

IpAddrTable-taulukko sisältää jokaisen portin IP-osoitekohtaiset ominaisuudet, jotka löytyvät ipAddrEntry-objektin muuttujista. ipAdEntAddr-muuttuja sisältää portin IP-osoitteen. ipAdEntNetMask-muuttuja sisältää IP-osoittetta vastaavan maskin osoitteen.

3.3 Topologian rakennus

Hallinta-agenttien lisäksi verkossa liikkuu topologia-agentteja, jotka keräävät verkon topologiatietoa keskitettyyn tietokantaan. Topologia-agentit keräävät topologiatietoa suorittamalla peräkkäisiä SNMP-kutsuja.



Kuva 10. Topologia-agentin prosessi kaavio.

Topologia-agentien toimintaa valvoo yleensä emoprosessi (kuva 10), joka myös päivittää keskitettyä tietokantaa topologia-agenttien tiedonkeräyksien perusteella. Topologia-agentin prosessi alkaa satunnaisesti valitusta organisaation verkosta. Jos verkosta ei ole aiemmin luotu IP-topologiarakennetta edetään verkon ensimmäiseen laitteeseen. Seuraavaksi käydään läpi laitteen portit ja porttien topologiatiedot, jotka päivitetään laitekohtaisiin topologiatietoihin. Päivityksen jälkeen siirrytään seuraavaan laitteeseen. Verkon topologiatietojen päivitysprosessi lopetetaan, kunnes kaikki verkon laitteet on käyty läpi. Koko prosessi lopetetaan, kunnes kaikki organisaation verkot on käyty läpi.

4 VISUALISOINTI

Tietokonegrafiikka on yleisnimitys kaikelle monitorilla esitettävälle informaatiolle. Informaation voidaan ajatella koostuvan olioista ja olioiden välisistä suhteista. Nykypäivän informaatiossa olioiden lukumäärät ja olioiden väliset suhteet kasvavat ja muuttuvat entistä monimutkaisemmiksi. Tietokonegrafiikassa tällaisen informaation muuttamista ymmärrettävään muotoon kutsutaan visualisoimiseksi. Tietojenkäsittelyssä yleisimpiä visualisoinnin kohteita ovat tietokannat, tietorakenteet, tietoverkot ja ohjelmistokehitys [Dav96].

4.1 Tausta

Visualisoinnissa oliot ja olioiden suhteet muutetaan solmuiksi ja kaariksi, jolloin visualisointiongelma muuttuu graafin piirto-ongelmaksi. Perinteisesti graafin piirto-ongelma on ratkaistu piirtämällä graafi manuaalisesti, piirto-ohjelmaa hyväksi käyttäen. Piirtämällä käyttäjä on saanut luotua graafin, joka vastaa haluttua lopputulosta. Haluttu lopputulos koostuu selkeästä ja symmetrisestä kuvasta, joka lukijan on helppo hahmottaa. Suuresta graafista, selkeän ja symmetrisen kuvan luonti piirtämällä on hankalaa ja aikaa vievää. Tällaisia piirto-ongelmia varten on kehitetty automaattiset piirtomenetelmät eli piirtoalgoritmit.

Piirtoalgoritmien avulla selkeän ja symmetrisen kuvan luonti suuremmastakin graafista on mahdollista, mutta optimaalisen lopputuloksen saavuttaminen on mahdotonta [Chr04, Gar83]. Viimeisten vuosien aikana piirtoalgoritmien tarve on kasvanut, mikä on kasvattanut myös piirtoalgoritmien suosiota. Suosiota ovat lisänneet myös käyttöliittymäkirjastojen tarjonta ja tietokoneiden tehokkuuksien kehittyminen, jotka ovat poistaneet viimeisetkin esteet piirtoalgoritmien leviämiselle [Beh99].

Myös automaattisissa piirtomenetelmissä piirtäjällä on valta päättää, minkä näköinen lopputuloksesta tulee. Käyttäjä voi valita, mitä piirtämiskäytäntöä graafin visualisoinnissa käytetään. Käytettävän piirtämiskäytännön mukaan sama graafi

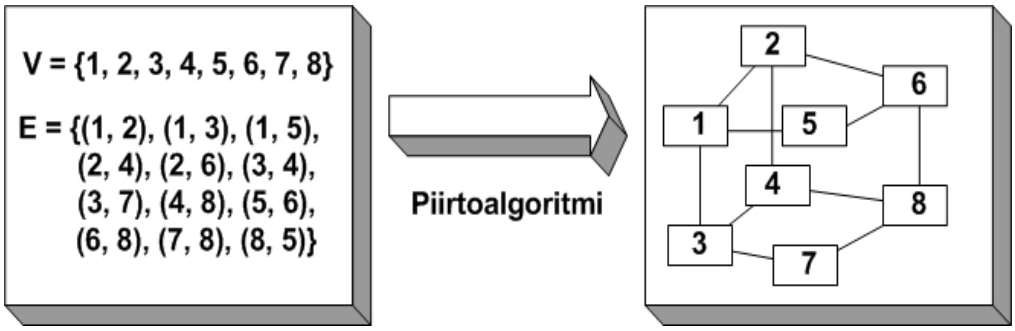
voidaan esittää mm. suuntaamattomana suorajanaisena, suunnattuna hierarkkisena tai ortogonaalisena näkymänä. Näkymän perusteella piirtäjä valitsee piirtomallin, joka käyttää haluttua piirtämiskäytäntöä. Lopuksi valitaan piirtomallin mukainen algoritmi. Algoritmien valinnassa esteettisyyskriteerit ja tehokkuus ovat tärkeimpiä valintaperusteita. Käyttäjän halutessa graafin ulkoasusta mahdollisimman miellyttävän valitaan mahdollisimman monta esteettisyyskriteeriä täyttävä algoritmi [Him95]. Vastaavasti jos käyttäjä haluaa algoritmin toimivan myös interaktiivisessa käytössä, valitaan algoritmi tehokkuuden perusteella.

Koska tietoverkot on totuttu näkemään suuntaamattomien suorajanaisten viivojen täyttämistä kuvista, jotka yhdistävät objektit toisiinsa kiinni, keskitytään tässä kappaleessa suuntaamattomien suorajanaisten graafien piirtämiseen. Tietoverkot voidaan laskea yleisiksi suuntaamattomiksi graafeiksi, jotka ovat helpoiten visualisoitavissa voimiin perustuvilla piirtomalleilla. Voimiin perustuvia piirtomalleja ovat jousiin ja optimointiin perustuvat mallit. Jousiin perustuvista malleista Eadesin, Fruchtermanin ja Reingoldin, GEM (Graph embedder) ja Kamadan ja Kawain algoritmit käydään läpi. Kamandan ja Kawain algoritmi käydään tarkemmin läpi, koska algoritmia käytetään käytännön työssä hyväksi. Optimointiin perustuva simuloitu jäähdytysalgoritmi käsitellään myös. Lopuksi vertaillaan kaikkia viittä algoritmia keskenään.

4.2 Peruskäsitteet

Olioista ja olioiden välisistä suhteista koostuvaa kuvaa kutsutaan visualisoinnissa verkoksi tai graafiksi. Graafi voidaan esittää matemaattisen mallin mukaisesti [Rad92] tai visuaalisessa muodossa.

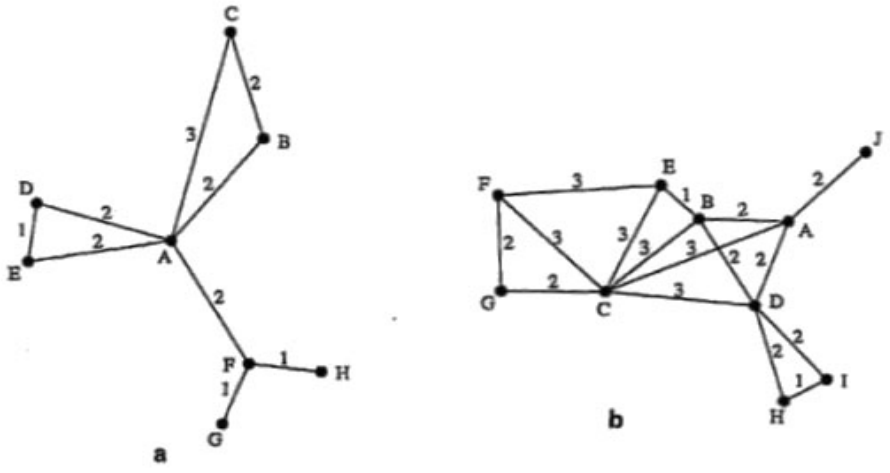
Kuvassa (kuva 11 [Tun99]) vasemmalla sijaitseva matemaattinen malli muutetaan piirtoalgoritmin avulla visuaaliseksi esitykseksi. Graafin visualisoinnissa ensimmäinen tehtävä on muuttaa kuva matemaattiseksi malliksi, jolloin kuvan oliot ja olioiden suhteet muutetaan solmuiksi (vertices) ja kaariksi (edge).



Kuva 11. Graafin piirtoprosessi.

Matemaattisessa mallissa $G = (V, E)$ G viittaa graafin (graph) abstraktiin esitykseen [Web98], jossa V sisältää kaikki graafin solmualkiot V={1,2,3...} ja E kaarialkiot E={(1,2),(1,4),...)}. Kaarialkiossa $e = (u, v)$, u ja v ovat kaaren päätepisteitä eli solmuja, jolloin u ja v ovat toistensa vierekkäisiä solmuja. Niinpä solmun naapuristoksi kutsutaan kaikkia solmun vierekkäisiä solmuja [Bat99].

Painotettu graafi koostuu normaalisti solmuista ja kaarista, mutta jokaiselle kaarelle määritellään painoarvo (kuva 12). Kuvan perusteella nähdään, kuinka painotuksilla voidaan vaikuttaa kuvan luettavuuteen.



Kuva 12. Painotettu graafi. [Kam89]

Suunnatuissa graafeissa (digraph) kaaret ovat aina suunnattuja, joten kahden solmun välillä oleva kaari mahdollistaa vain toiseen suuntaan kulkevan liikenteen. Toisin sanoen päätepisteiden *i* ja *j* järjestys on merkityksellinen $(e_{ij} != e_{ji})$.

Suuntaamattomassa graafeissa kaarilla ei ole suuntaa (kuva 11), jolloin solmuja yhdistävät kaaret mahdollistavat solmujen välisen liikenteen kumpaakin suuntaan. Niinpä solmujen *i* ja *j* järjestyksellä ei ole väliä $(e_{ij} == e_{ji})$.

Planaariseksi graafiksi kutsutaan graafia, jossa graafi esitetään ilman päällekkäisiä kaaria (kuva 12.a), mutta jos yksikin kaari kulkee toisen kaaren päältä, ei graafi ole enää tämän jälkeen planaarinen (kuva 12.b).

Syklisessä graafissa solmujen kaaret muodostavat ympyrän, jolloin graafin viimeinen solmualkio yhdistyy ensimmäiseen solmualkioon. Syklittömästä graafista (Directed Asyclic Graph) tällainen kaari puuttuu.

4.3 Graafin piirtoparametrit

Jokaisella piirtoalgoritmilla on ainakin yksi piirtoparametri, joka määräytyy graafin ominaisuuksien perusteella. Ominaisuuksien perusteella graafit luokitellaan eri luokkiin kuuluviksi. Luokkaparametrin avulla esimerkiksi suunnattu ja suuntaamaton graafi saadaan erotettua toisistaan. Ennen piirtämistä kyseisen luokkaparametrin arvon tietäminen edesauttaa ainakin seuraavissa tapauksissa:

- Jotkut algoritmit toimivat vain tietyn luokan graafeilla. Algoritmit voivat toimia myös paremmin tietyn luokan graafeilla kuin toisilla [Bat99].

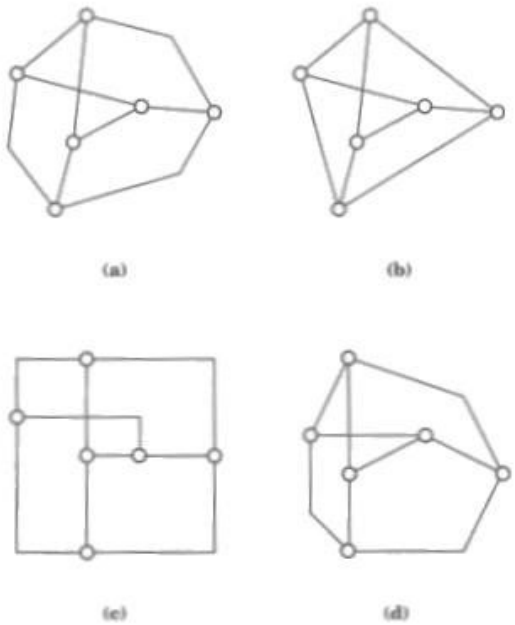
- Käyttäjä voi halutessaan tuoda esiin jonkun graafin ominaisuuden, mikä onnistuu vain tietyllä algoritmilla ja siihen soveltuvalla luokkaparametrilla. Esimerkiksi suunnatun syklittömän graafin kaaret voidaan suunnata samaan suuntaan, jolloin kuvan syklittömyyden poissaoloa saadaan painotettua [Bat99].

Graafin luokkaparametria voidaankin hyvällä syyllä pitää yhtenä graafin piirron tärkeimmistä parametreista. Luokkaparametrille soveltuvalla algoritmilla harvoin päästään suoraan haluttuun lopputulokseen, joten käyttäjäkohtaisia piirtoparametreja tarvitaan myös.

Käyttäjäkohtaisia piirtoparametreja ovat piirtämiskäytäntö-, esteettisyys-, rajoitus- ja tehokkuusparametri.

4.3.1 Piirtämiskäytäntö

Piirtämiskäytännöllä määritellään päälinjat graafin piirrolle. Päälinjoissa määritellään solmujen muodot ja kaarien janojen piirtokäytännöt. Esimerkiksi ohjelman sisäistä toimintaa kuvaava datavuodiagrammi piirretään piirtämiskäytännöllä, jossa kaikki solmut kuvataan laatikkoina ja kaaret monijanaisina suorina, jotka piirretään vaaka- ja pysty-akseleiden suuntaisesti.



Kuva 13. Sama kuva eri piirtämiskäytännöillä piirrettynä: a) Monijana; b) Suorajana; c) Ortogonaali; d) Ruudukko. [Bat99]

Automaattisten piirtomenetelmien yleisimmät piirtämiskäytännöt [Bat99]:

- Monijana-piirtämiskäytäntö: Monijana-piirtämiskäytännössä (kuva 13.a) jokainen kaari piirretään kuten monijanainen viiva, jolloin kaari voi sisältää kulmia solmujen välillä. Suurten tietoverkkojen mallintamisessa ongelmaksi monijana-piirtämiskäytännössä muodostuu kulmien paljous, jolloin solmujen välisiä suoria yhteyksiä on vaikea huomioida.

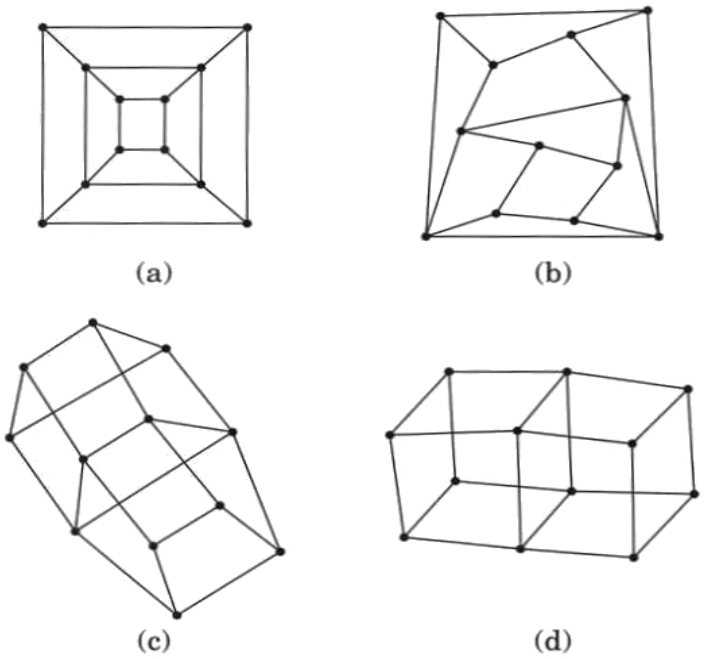
- Suorajana-piirtämiskäytäntö: Suorajanaisessa piirtämiskäytännössä (kuva 13.b) jokainen kaari piirretään suoraan solmusta solmuun. Suorajana-piirtämiskäytäntöä käytetään paljon tietoverkkojen mallintamisessa [Kam89, Dav96, Fru91], koska solmujen välisiä yhteyksiä on helppo seurata. Voimiin perustuvissa piirtomalleissa käytetään ainoastaan suorajana piirtämiskäytäntöä.

- Ortogonaali-piirtämiskäytäntö: Ortogonaalisessa piirtämiskäytännössä (kuva 13.c) kaaret piirretään monijanaisina suorina, jotka kulkevat pitkin vaaka- tai pysty-akseleita. Ortogonaalista piirtämiskäytäntöä käytetään paljon VLSI (Very Large Scale Integrated) piirisuunnittelussa ja sovelluskehityksessä [Bat99].

- Ruudukko-piirtämiskäytäntö: Ruudukko-piirtämiskäytännössä (kuva 13.d) solmut, kaaret ja kaarien kulmat sijaitsevat kokonaislukukoordinaatistossa.

4.3.2 Esteettisyys

Erityyppisille graafeille on vaikea määritellä yhteistä kriteeriä, jonka perusteella graafien ulkoasujen paremmuutta voitaisiin mitata [Kam89]. Yksittäisen graafin ulkoasua voidaan sitä vastoin mitata monella erilaisella esteettisyyskriteerillä [Bat94]. Voimiin perustuvien piirtomallien yleisimmät esteettisyyskriteeterit ovat päällekkäisten kaarien minimointi ja symmetrisyys.



Kuva 14. Graafi eri esteettisyysparametreja painottaen piirrettynä: a) Käsin piirretty; b) Päällekkäisten kaarien minimointi; c-d) Symmetrisyyden painotus [Dav96].

Seuraavassa on esitetty voimiin perustuvien mallien yleisimmät esteettisyysparametrit:

- Päällekkäisten kaarien minimointi: Päällekkäisten kaarien minimoinnilla pyritään poistamaan kaikki toisensa ylittävät kaaret pois (kuva 14.b). Tämä kasvattaa usein luettavuutta, mutta jossain tapauksissa voi myös sekoittaa ulkoasua [Kam89].

- Symmetrisyys: Symmetrisyysparametrilla pyritään rakentamaan graafista mahdollisimman symmetrinen (kuva 14.a,c,d). Ensimmäisessä jousialgoritmissa toisena esteettisyyskriteerinä käytettiin symmetrisyyttä [Ead84].

- Alueenrajoitus: Graafin alueenrajoitusparametrilla voidaan piirtoalueen kokoa rajoittaa, jolloin kaikki graafin objektit sijoitetaan rajatulle alueelle.

Jos esitettävä graafi ei mahdu kokonaan ruudulle, kärsii graafin kokonaisuuden hahmottaminen. Kokonaisuuden hahmottamista voidaan parantaa rajaamalla graafin koko ruudun kokoiseksi. Algoritmeissa harvinainen ominaisuus, mutta FR-algoritmissa (Fruchtermanin ja Reingoldin) valittavissa [Fru91].

- Kaaren vakiopituus: Kaaren vakiopituusparametrilla määritellään nimensä mukaisesti vakiopituus kaarelle. Kamadan ja Kawain algoritmissa kaaren vakiopituus esteettisyyttä käytetään aina, koska algoritmin toiminta perustuu vakiopituisiin kaariin [Kam89].

- Sijaintien rajoitukset: Sijaintien rajoitukset pitää sisällään monia erilaisia rajoituksia, joilla rajoitetaan solmujen ja kaarien sijainteja ruudulla. DH-algoritmissa (Davidsonin ja Harelin) voidaan esimerkiksi rajoittaa solmujen sijoittamista reunojen läheisyyteen tai solmujen sijoittamista toisiensa viereen [Dav96].

- Kaaren maksimipituus: Kaaren maksimipituusparametrilla pyritään pitämään kaikkien kaarien maksimipituus tarpeeksi lyhyenä, jolloin graafeissa esiintyvät pitkät kaaret saadaan karsittua pois.

4.3.3 Rajoitukset

Piirtämiskäytännöllä ja esteettisyysparametreilla määritellääään graafin piirron säännöt ja kriteerit, jotka vaikuttavat graafin solmuihin ja kaariin. Kyseisillä parametreilla ei voida kuitenkaan rajata sääntöjen ja kriteereiden vaikutusta vain tiettyyn osajoukkoon graafista. Rajoitusparametreilla voidaan vastaavasti rajata parametrien ominaisuuksien vaikutus vain tiettyyn osajoukkoon kohdistuvaksi. Esimerkiksi tietoverkkojen visualisoinnissa on luettavuuden kannalta loogista sijoittaa ulkoverkkoon yhdistyvät laitteet graafin reunoille, jolloin ulkoverkon ja sisäverkon raja on helpommin havaittavissa. Seuraavaksi käsitellään muutamia yleisimpiä rajoitusparametreja:

- Keskitys: Keskitysrajauksella voidaan osa solmuista sijoittaa keskustaan. Organisaation verkon visualisoinnissa on loogista sijoittaa verkon sisäverkko graafin keskelle, johon myös lukijan katse ensimmäiseksi kohdistuu.

- Ulkoistus: Ulkoistus on päinvastainen toiminta keskittämisrajaukselle, jolloin osa solmuista sijoitetaan graafin reunoille. Organisaation verkon visualisoinnissa voidaan esimerkiksi ulkoverkkoon yhteydessä olevat laitteet sijoittaa graafin ulkoreunoille, jolloin graafin näkymä vastaa myös verkon fyysistä olemusta.

- Klusterointi: Klusterointi-rajauksella voidaan osa solmuista sijoittaa lähelle toisiaan. Esimerkiksi verkoissa samaan keskittimeen yhdistyvät laitteet voidaan klusteroida omaksi kokonaisuudeksi.

- Vasemmalta oikealle (ylhäältä alas) järjestys: Vasemmalta oikealle rajauksessa graafin solmut järjestellään kulkevaksi haluttuun suuntaan. Voimiin perustuvissa malleissa graafin kulku voidaan määritellä voimien suuntaiseksi [Bat99].

- Muoto: Muotorajauksella rajataan osa graafin solmuista haluttuun muotoon. Verkkojen visualisoinnissa verkon topologia (tähti, rengas, väylä) on muotorajauksen avulla mahdollista rakentaa.

4.3.4 Tehokkuus

Aiemmin läpikäydyt piirtoparametrit ovat vaikuttaneet graafin toteutuksessa lähinnä luettavuuteen. Nykyajan interaktiiviset ohjelmat vaativat luettavuuden lisäksi ohjelmalta myös tehokkuutta. Tehokkuusparametrin ja luettavuusparametrien välillä vallitsee suhteellinen yhteys, jolloin toisen painotus heikentää toista ja päinvastoin. Käytännössä samanaikaisesti tehokkaimman ja luettavimman graafin

toteutus ei ole mahdollista. Interaktiivisen ohjelman suunnittelijan suurin ongelma onkin löytää sopiva suhde tehokkuuden ja luettavuuden välille.

Esimerkiksi voimiin perustuvassa Kamadan ja Kawain algoritmissa graafin luettavuus/tehokkuus-suhde määritellään yhden parametrin perusteella. Algoritmin alustuksessa määritellään luettavuusparametri, jonka perusteella graafia iteroidaan ulkoasultaan miellyttävämmäksi. Luettavuusparametria pienentämällä saadaan algoritmin ajoaikaa lyhennettyä.

Suuren organisaatioverkon interaktiivinen selailu voi vaatia luettavuusparametrin arvon pienentämistä, jolloin ohjelman visualisoimisesta aiheutuvat viiveet pyritään minimoimaan alle kahteen sekuntiin [Fri95]. Vastaavasti, jos organisaation verkosta halutaan tulostaa paperiversio, pyritään maksimaaliseen graafin luettavuuten ja näin ollen kasvatetaan parametrin arvoa.

4.4 Piirtomallit

Edellisessä kappaleessa käytiin läpi graafin piirtoparametreja, joita piirtomallien käyttämät algoritmit käyttävät hyväkseen. Piirtomalleilla kuvataan malli, kuinka graafia lähdetään rakentamaan. Piirtomalleissa on yleensä muutamia selkeästi eroteltavia työvaiheita, jotka mallin mukaisesti käydään järjestyksessä läpi.

Suurin osa piirtomalleista soveltuu suunnattujen tai monijanaisten graafien suunnitteluun, jotka eivät varsinaisesti sovellu tietoverkkojen mallintamiseen. Yleisimmistä piirtomalleista topologia-muoto-koko (Topology-Shape-Metric), hierarkkinen (Hierarchical), hajoita ja hallitse (Divide and Conquer), näkyvyys (Visibility) sekä täydentämispiirtomallit (Augmentation) voidaan laskea juuri tähän ryhmään [Bat99].

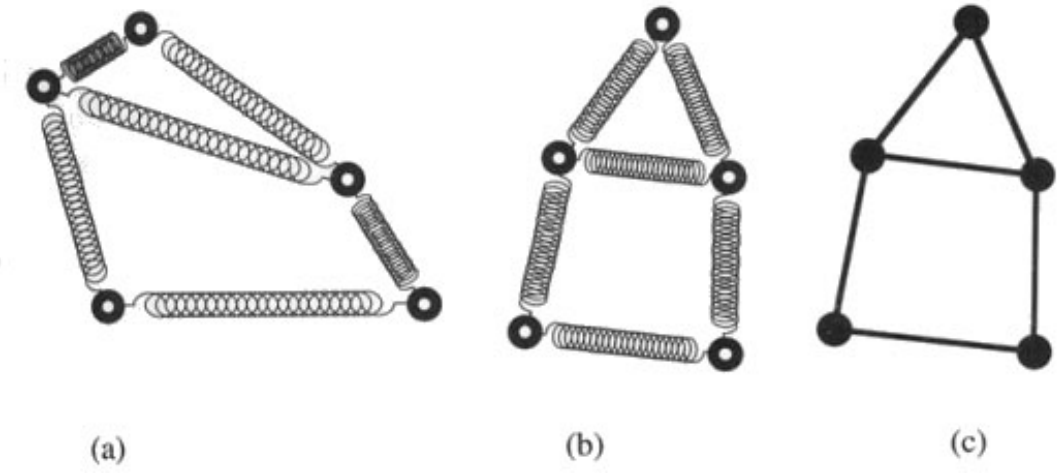
Edellä mainituista piirtomalleista suurimpaan osaan voidaan yhdistää seuraava rakenne. Ensimmäisenä graafista muodostetaan pelkistetty malli, johon graafin solmut ja kaaret on sisällytetty. Pelkistetystä mallista on poistettu lisäksi päällek-

käiset kaaret pois, mikä nopeuttaa seuraavien työvaiheiden suoritusta. Pelkistetyn kuvan muodostamista kutsutaan myös planarisoimiseksi [Bat99]. Planarisoinnin jälkeen piirtomallit sisältävät yleensä kolmesta neljään erillistä työvaihetta, minkä jälkeen ulkoasu on valmis.

Tietoverkot on totuttu näkemään suuntaamattomien ja suorajanaisten kaarien täyttämistä kuvista, jotka yhdistävät erinäköiset ja kokoiset objektit toisiinsa. Tietoverkot voidaan ajatella yleisiksi suuntaamattomiksi graafeiksi, joiden visualisointiin käytetään käytännössä vain voimiin perustuvaa piirtomallia. Tietoverkkojen visualisoinnissa suunnittelijan tärkeimmäksi tehtäväksi jääkin sopivan algoritmin valinta, joka pohjautuu voimiin perustuvaan piirtomalliin.

Voimiin perustuva piirtomalli poikkeaa huomattavasti muista yleisistä piirtomalleista. Piirtomallissa ei ole montaa erillistä työvaihetta, vaan käytännössä yhtä työvaihetta (optimointia) toistetaan alusta loppuun asti. Yksinkertaisen rakenteensa johdosta voimiin perustuva piirtomalli on helppo ymmärtää ja toteuttaa [Bat99].

Optimaalisen ulkoasun tuottaminen yleiselle suuntaamattomalle graafille on NP-vaikea (Nondeterministic Polynomial Time) ongelma [Gar83, Tun94, Chr04]. NP-vaikea ongelma ei ole determisesti ratkaistavissa. Tästä johtuen voimiin perustuvat piirtomallit ratkaisevat aina optimointiongelman, jota lähdetään ratkaisemaan heurastisia menetelmiä hyväksikäyttäen. Karkeasti sanoen voimiin perustuva piirtomalli saa syötteenä graafin ja palauttaa saman graafin muokatulla ulkoasulla. Fysiikan lakeihin ja analogiaan perustuvien heurastisten menetelmien on kokemusten perusteella huomattu tuottavan ongelmille optimaalisia ratkaisuja [Bat99]. Voimiin perustuvissa piirtomalleissa heurastiset menetelmät voidaan jakaa kolmeen erilliseen luokkaan [Fri95].



Kuva 15. Jousiin perustuva malli: a) Aloitustila; b) Energialause minimoitu; c) Jousi mallin muutto graafiksi. [Bat99]

Jousiin perustuvassa mallissa graafin jokainen solmu korvataan renkaalla tai partikkelilla, jotka sisältävät positiivisen ja saman suuruisen energiavarauksen. Vastaavasti graafin kaikki kaaret korvataan jousilla, jotka käyttäytyvät normaalien jousien tavoin. Kaikilla renkailla on positiivinen varaus, joten vierekkäiset renkaat vastustavat toisiaan, eli renkailla on vastustava voima. Jousen ollessa lepopituudessa ei jouseen kohdistu voimia, mutta jousta venytettäessä alkaa jousi vetämään itseään takaisin lepopituuteensa, jolloin jouseen vaikuttaa vetävä voima. Kuvan 15.a piirtomalli on saanut graafin syötteenä sisään, jossa renkaiden sijainnit ovat sattumanvaraisesti valittu. Kuvan jousien vetovoimat ovat lähes ”käsinkosketeltavissa”. Seuraavaksi mallissa iteroidaan renkaille uudet sijainnit, jotka johtavat jossain vaiheessa vastustavien ja vetävien voimien tasapainoon. Kuvan 15.b jouset ovat saavuttaneet lepopituutensa ja renkaiden varaukset pitävät graafin kokonaisuuden symmetrisenä.

Optimointiin perustuvassa mallissa ensimmäisenä tehtävänä on rakentaa kustannusfunktio. Kustannusfunktio sisältää erillisiä esteettisyyskriteerifunktioita, mitkä kaikki vaikuttavat omalla esteettisyyskriterillään tuotettavan kuvan kokonaisesteettisyyteen. Usein kustannusfunktiot sisältävät monia esteettisyysfunktioita, jotka kaikki kasvattavat funktion kompleksisuutta. Seuraavaksi optimointimallissa

pyritään löytämään kustannusfunktiolle minimienergiakonfiguraatio, eli kustannusfunktion arvoa pyritään optimoimaan pienemmäksi. Kustannusfunktion optimoinnissa voidaan käyttää mitä tahansa optimointitekniikkaa, mutta kokemuksen perusteella staattinen mekaniikka on osoittautunut hyväksi heurastiseksi optimointitekniikaksi [Dav96]. Staattisesta mekaniikasta tuttua simuloitua jäähdytysmenetelmää hyväksikäyttämällä saadaan kustannusfunktio optimoitua lähelle optimiratkaisua [Fri95]. Simuloitu jäähdytysmenetelmä perustuu lämpötilaan, mitä jäähdytetään jokaisella iteraatiokierroksella pienemmäksi. Renkaiden siirtojen suuruudet riippuvat vallitsevasta lämpötilasta. Korkeassa lämpötilassa renkaiden siirtymät ovat suuria ja matalassa lämpötilassa pieniä.

Graafin esikäsittelyyn perustuvassa mallissa syötteeksi annettavan graafin renkaat sijoitetaan esikäsittelyalgoritmin mukaiseen järjestykseen. Seuraavaksi esikäsitelty graafi annetaan syötteenä voimiin perustuville piirtomalleille

4.5 Voimiin perustuvien mallien algoritmit

Voimiin perustuvissa piirtomalleissa määritellään päälinjat algoritmien suunnittelulle. Algoritmien suunnittelu on helppoa, koska voimiin perustuva piirtomalli on yksinkertainen ja helppo toteuttaa [Bat99]. Voimiin perustuvat algoritmit ovat helposti laajennettavissa rajoitusparametrien avulla. Esimerkiksi sijainti- ja clusterointirajoitusparametrien avulla voidaan graafin ulkoasuun tuoda lisää selkeyttä ja luettavuutta.

Helpon toteutuksensa lisäksi algoritmeilla on kyky tuottaa kauniita ja symmetrisiä kuvia sellaisistakin kuvista, jotka muilta piirtoalgoritmeilta on jäänyt toteuttamatta. Tehokkuutensa ansiosta voimiin perustuvat algoritmit muistetaan myös paljon koneen resursseja kuluttavista ominaisuuksistaan. Esimerkiksi DH- ja KK-algoritmit (Kamadan ja Kawain) käyttävät hyväkseen paljon koneen muistia, joka suurilla graafeilla saattaa loppua kesken [Tun99].

Algoritmeista käsitellään viisi yleisintä algoritmia. Ensimmäiset neljä algoritmia perustuvat jousimalliin, joista ensimmäisenä esitellään Eadesin algoritmi, jota voidaan pitää myös ensimmäisenä voimiin perustuvana algoritmina [Bat99]. Kappaleessa 4.5.2 esitellään Fruchtermanin ja Reingoldin algoritmi, joka pohjautuu Eadesin malliin. Seuraavaksi esitellään GEM-algoritmi, joka vastaavasti perustuu osaksi FR-algoritmiin. Kappaleessa 4.5.4 käsitellään Kamadan ja Kawain algoritmi, jossa jousien voimat perustuvat graafin teoreettisiin pituuksiin. KK-algoritmia käytetään myös diplomityön käytännön työssä hyväksi, joten algoritmin lokaali minimointiprosessi käydään muita algoritmeja tarkemmin läpi. Viimeisenä algoritmina esitellään simuloitu jäähdytysalgoritmi, joka pohjautuu optimointimalliin. Lopuksi vertaillaan esiteltyjen algoritmien ominaisuuksia ja kompleksisuuksia keskenään.

4.5.1 Eadesin algoritmi

Jousialgoritmeja on käytetty hyväksi jo 1960-luvulta [Tut60] lähtien ratkaistaessa matemaattisia ongelmia, mutta Eadesin jousialgoritmia käytettiin 1980-luvulla ensimmäisenä tietokonegraafiikassa [Bat99]. Eades nimitti algoritmiaan jouseen perustuvaksi (Spring Embedder), minkä vuoksi kaikkia Eadesin algoritmiin perustuvia piirtoalgoritmeja (KK, FR, GEM) nimitetään jouseen perustuviksi tai jousialgoritmeiksi. Eadesin malli perustuu fysiikan analogiaan, joten solmut korvataan metallirenkailla ja kaaret jousilla. Metallirenkaat toimivat vastustavina voimina ja jouset vetävinä voimina. Eadesin malli perustuu Hookin yleiseen jousikaavaan (1) [Bat99]:

$$\sum_{(u,v)\in E} k_{uv}^{(1)} \left(d\left(p_u, p_v\right) - l_{uv} \right) \frac{x_v - x_u}{d\left(p_u, p_v\right)} + \sum_{(u,v)\in VxV} \frac{k_{uv}^{(2)}}{\left(d\left(p_{u,} p_v\right) \right)^2} \frac{x_v - x_u}{d\left(p_u, p_v\right)}, \qquad (1)$$

, missä $k_{uv}^{(1)}$ on jousen vetävä voima, $k_{uv}^{(2)}$ renkaan vastustava voima, $d\left(p_u, p_v\right)$ u ja v renkaiden välinen deukalinen pituus ja l_{uv} jousen lepopituus. x_v ja x_u sisältävät renkaiden x-akselin koordinaatti arvoja.

Eadesin kokemusten perusteella Hookien yleisessä jousikaavassa (1) lineaarisesti kasvava jousen vetävä voima kasvoi liian suureksi pitkillä välimatkoilla [Ead84]. Eades ratkaisi ongelman seuraavalla logaritmisella kaavalla (2) [Ead84]:

$$k_{uv}^{(1)} = C_1 * \log(d / C_2), \tag{2}$$

missä d on jousen pituus ja C_1 ja C_2 ovat vakioita. C_2 parametrilla voidaan määritellä jousen vakiopituus, joka on myös algoritmin toinen esteettisyysparametri. Jos kaavan sijoitetaan $d = C_2$, jouseen vaikuttavat voimat ovat 0 ja näin ollen jousi pysyy vakiopituudessaan. Symmetrisyys on algoritmin toinen esteettisyysparametri, joten renkaiden vastustavat voimat lasketaan kaavan (3) [Ead84] mukaisesti:

$$k_{uv}^{(2)} = \frac{C_3}{d^2}, \tag{3}$$

missä C_3 on vakio ja d on renkaiden välimatka. Kaava perustuu käänteisen neliön lakiin, minkä perusteella renkaan vastustava voima vähenee mitä kauempana toinen rengas sijaitsee. Eadesin vastustavien voimien kaavassa kaikkien renkaiden vastustavat voimat lasketaan yhteen pois lukien viereisten renkaiden voimat.

Eadesin pseudo-algoritmi [Ead84]:

```
Algoritmi Eades(G:graafi)
Sijoita graafin G renkaat sattumanvaraisiin paikkoihin
Toista 100 kertaa
        Laske voimat jokaiselle renkaalle;
        Siirrä jokaista rengasta, renkaaseen kohdistuvan voiman mukaisesti;
        Palauta renkaiden sijainnit;
End //Eades
```

Eadesin algoritmi toimii alle 50 renkaan syötteillä, mutta yli 50 renkaan määrä rikkoo graafin osagraafeiksi tilan ahtauden johdosta. Ongelmallisia graafeja algo-

ritmille ovat myös tiheät graafit. Tiheällä graafilla tarkoitetaan graafia, jossa jousia on huomattavasti enemmän suhteessa renkaisiin $(|E| > 3|V|)$ [Fri95]. Ongelma johtuu algoritmin ominaisuudesta jättää vierekkäisten renkaiden vastavoimat laskematta laskettaessa kaikkien renkaiden vastavoimia yhteen. Ongelmasta johtuen tiheisiin graafeihin voi muodostua keskittymiä [Beh99]. Graafin alustuksessa suoritettava renkaiden sattumanvarainen sijoittelu voi myös osoittautua ongelmaksi tapauksissa, joissa lopputulos muuttuu hyvästä huonoksi alustuksen perusteella. Eadesin algoritmin ongelmiin voidaan laskea myös sen vaatima laskentateho, erityisesti vastustavien ja vetävien voimien laskeminen on paljon laskentatehoa vaativaa. [Beh99].

Eades tiedosti algoritmin ongelmat ja suosittelikin algoritmiaan lähinnä graafin ensimmäiseksi approksimointityökaluksi [Ead84]. Eadesin algoritmissa ilmenneet ongelmat poistettiin Eadesin jälkeen ilmestyneissä algoritmeissa.

4.5.2 Fruchtermanin ja Reingoldin algoritmi.

Fruchtermanin ja Reingoldin (FR) algoritmi pohjautuu samaan kaavaan (1), Eadesin algoritmin kanssa. Niinpä Frunchtermanin ja Reingoldin algoritmia kutsutaan myös jousialgoritmiksi tai lyhemmin FR-algoritmiksi [Fru91].

Eadesin mallia tehostettiin muuttamalla vetävien ja vastustavien voimien kertoimien laskentaa tehokkaammaksi. Laskentaa saatiin tehostettua yksinkertaistamalla kertoimien laskenta kaavoja sekä vähentämällä käyttäjän määrittelemien muuttujien lukumäärä yhteen. Seuraavassa yksinkertaistetut vetävien f_a ja vastustavien f_r voimien kertoimien laskenta kaavat [Fru91]:

$$f_a(d_{uv}) = \frac{d_{uv}^2}{k}, \tag{4}$$

$$fr(d_{uv}) = \frac{-k^2}{d_{uv}}, \tag{5}$$

missä vakiomuuttuja *k* on optimaalinen etäisyys solmujen välillä ja d_{uv} *u* ja *v* pisteiden välinen deukalinen pituus [Beh99]. Käyttäjä voi määrittää optimaalisen etäisyyden solmujen välille, muuten etäisyys lasketaan kaavasta piirtoalustan koko/graafin renkaiden lukumäärä [Dav96].

Eadesin mallia laajennettiin lisäksi lämpötila- ja jäähdytysominaisuuksilla. Fruchterman ja Reingold kopioivat ominaisuudet suoraan simuloidusta jäähdytysalgoritmista [Dav96]. Lämpötilan avulla renkaiden siirtomatkojen suuruutta pystytään säätelemään. Aloituslämpötilassa renkaiden siirtomatkat ovat suurimmat. Jokaisen iteraatiokierroksen jälkeen lasketaan renkaiden voimat yhteen, minkä perusteella lasketaan seuraava jäähdytyskerroin. Ensimmäisten kierroksien aikana yhteenlaskettujen voimien summa on suuri, jolloin jäähdytyskerroin on myös suuri. Jäähdytyskertoimella kerrotaan senhetkinen lämpötila, jolloin lämpötila alenee ja renkaiden siirtomatkat pienenevät. Jäähdytyksen johdosta lämpötila alenee lineaarisesti alaspäin [Beh99]. Iteraatiokierrosten loppuvaiheessa lämpötilaa ei enää alenneta, jolloin renkaiden siirtely muuttuu hienosäädöksi [Dav96].

FR-algoritmin hyviin puoliin kuuluu lisäksi piirtokehysominaisuus, jonka avulla tuotettava kuva voidaan mahduttaa halutun kehyksen sisään. Tuotettu kuva on tämän jälkeen helppo siirtää kiinteän kokoiseen tilaan. Grafiikkaohjelmissa monitorin koko voisi olla esimerkiksi tällainen kiinteänkokoinen tila.

Fruchterman ja Reingold havaitsivat paremman jäähdytystekniikan muuttavan dramaattisesti kuvan laatua ja iteraatiokierrosten lukumäärää etsittäessä miellyttävää ulkoasua. Paremmalla jäähdytystekniikalla toteutettua versiota kutsutaan ruudukkomuunnelmaksi (grid variant). Muunnelmassa piirtoalusta jaetaan ruudukoksi. Renkaaseen kohdistuvat vetävät voimat lasketaan kuten alkuperäisessä versiossa, mutta vastustavat voimat lasketaan vain ruudun renkaiden kesken, missä myös laskettava rengas sijaitsee. Ruudukkomuunnelman avulla laskentatehoa saatiin huomattavasti kasvatettua, mutta ruudukkokeskeisyys ulkoasussa ei ollut halutun-

lainen [Beh99, Tun99]. Epämiellyttävän ulkoasun johdosta malli ei saavuttanut laajaa kannatusta.

4.5.3 GEM

Kuvaan upotettava eli lyhemmin GEM-algoritmi on yksi voimiin perustuvien mallien monimutkaisimmista algoritmeista, mutta samalla myös yksi tehokkaimmista. GEM-algoritmiin on otettu ominaisuuksia FR- ja DH-algoritmeista [Bra96]. Lisäksi algoritmiin on lisätty muutamia uusia heurastisia ominaisuuksia (mm. rotaatio, gravitaatio), joiden avulla ulkoasun hyvyyttä voidaan mitata entistä tarkemmin.

GEM-algoritmia lähdetiin kehittämään interaktiiviseen käyttöön sopivaksi, jolloin algoritmin on pystyttävä tuottamaan ulkoasu suurille graafeille noin kahdessa sekunnissa [Fri95]. Tavoitteissa onnistuttiin, koska algoritmilla voidaan tuottaa yli 100 solmun graafien ulkoasuja tarpeeksi nopeasti [Bat99, Bra96, Beh99].

GEM-algoritmin kehittäjät huomasivat lämpötilaan perustuvan optimoinnin tuottavan parempia tuloksia kuin perinteinen KK-algoritmi [Fri95]. Lämpötilateema oli vain äärettömän hidas, joten lämpötilateemaa piti kehittää nopeammaksi. Aiemmin kehitetyissä DH- ja FR-algoritmeissa käytettiin pelkästään gloobaalia lämpötilaa, jolla mitattiin graafin keskimääräistä lämpötilaa. Globaalin lämpötilan lisäksi GEM:issä lasketaan jokaiselle renkaalle myös oma lämpötila (paikallinen lämpötila). Lisäksi GEM oli ensimmäinen algoritmi, missä käytettiin aiempaa laskuhistoriaa hyväksi, minkä johdosta lämpötilateema nopeutui huomattavasti[Beh99].

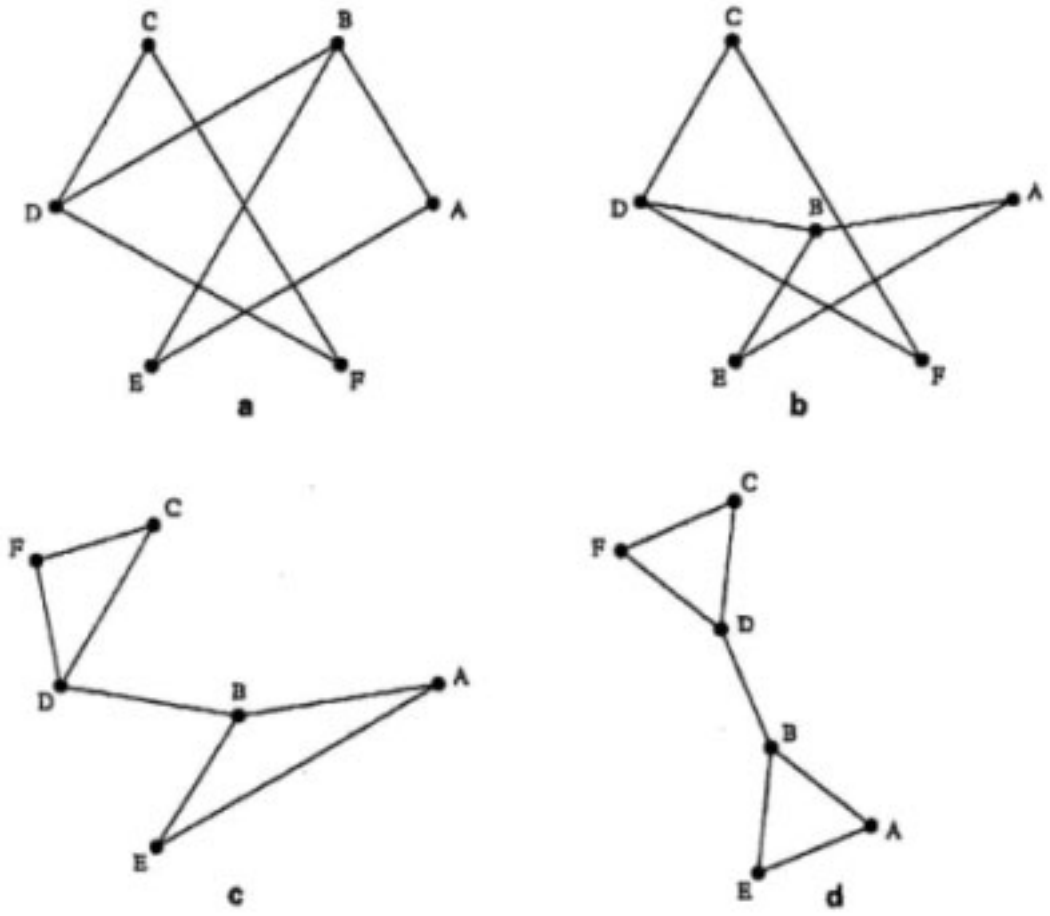
GEM pseudo-algoritmi [Beh99]:

```
Algoritmi GEM(G:graafi)
renkaiden sijaintien alustus;
while T_global > T_min ja kierros < R_max do
        valitse sattumanvaraisesti yksi rengas v;
        renkaan v impulssi = graavitaatio voima + sattumanvarainen häiriö +
        vastustavat voimat + vetävät voimat;
        päivitä v:n sijainti, rajoita liikettä paikallisella lämpötilalla;
        päivitä v:n  paikallinen lämpötila;
End // while
End // GEM
```

GEM-algoritmissa suoritetaan aluksi renkaiden alustus, jolloin jokaiselle renkaalle arvotaan sijainnit sattumanvaraisesti tai järjestellään sijainnit toisen algoritmin (insert) avulla paikoilleen [Beh99]. Lisäksi renkaiden impulssiarvot ja paikalliset lämpötilat alustetaan lähtöarvoilla. Algoritmi lopettaa toimintansa, kunnes globaali lämpötila eli kaikkien renkaiden lämpötilojen keskiarvo laskee alle käyttäjän määrittelemän raja-arvon tai jos iteraatiokierrokset saavuttavat maksimiarvon. Jokaisen iteraatiokierroksen aikana siirretään vain yhtä rengasta, mikä valitaan sattumanvaraisesti. Rengas valitaan sattumanvaraisesti, koska sattumanvarainen valinta on determinististä [Kam89] valintaa huomattavasti nopeampi [Fri95]. Sattumanvaraisesta valinnasta huolimatta algoritmi tuottaa lähes aina onnistuneen ulkoasun graafille. Valitulle renkaalle suoritetaan renkaan impulssin laskenta, joka muodostuu renkaaseen vaikuttavasta gravitaatiovoimasta, sattumanvaraisesta häiriöstä sekä vastustavista ja vetävistä voimista. Seuraavaksi renkaan sijaintia päivitetään impulssin perusteella, jota rajoitetaan renkaan paikallisella lämpötilalla. Renkaan paikallinen lämpötila lasketaan edellisen lämpötilan ja viimeisimmän siirron perusteella. Lopuksi päivitetään renkaan paikallinen lämpötila ja aloitetaan uusi iteraatiokierros.

4.5.4 Kamadan ja Kawain algoritmi

Kamadan ja Kawain algoritmi perustuu malliin [Kru80], jossa voimat perustuvat graafin teoreettisiin pituuksiin. Nykyään tähän malliin viitataan KK-lyhenteellä. KK-jousialgoritmissa solmut ajatellaan varauksellisiksi partikkeleiksi, jotka ovat yhdistetty toisiinsa jousien välityksellä. Partikkeleiden välillä vallitsevien varauksien suuruudet lasketaan euklidisen pituuden ja teoreettisen pituuden suhteesta. Kahden erillisen partikkelin teoreettinen välimatka saadaan kertomalla partikkeleiden lyhimmän polun pituus halutulla optimaalisella jousen pituudella. Tämän johdosta painotettujen graafien ulkoasujen mallinnus on KK-algoritmilla mahdollista [Dav96]. Algoritmin energia lauseessa lasketaan kaikkien partikkeleiden varaukset yhteen ja pyritään saavuttamaan energialauseelle lokaali minimi [Kam89].



Kuva 16. Energialauseen minimointi prosessi. [Kam89]

Energialausetta minimoidaan determisesti, eli jokaisen iteraatiokierroksen lopuksi suurimman varauksen sisältävä partikkeli siirretään oikealle paikalleen (kuva 16). Energialause perustuu fysiikan analogiaan ja tarkemmin seuraaviin Hookin jousi kaavoihin:

$$F = KX, \tag{6}$$

$$P.E = \frac{1}{2} KX^2, \tag{7}$$

missä *K* on jousivakio ja *X* jousen poikeama. Hookin kaavasta (6) saadaan johdettua jousen potentiaalienergian kaava (7) [Kum96]. Kun jousta poikkeutetaan tasapainoasemastaan joudutaan tekemään kaavan (7) suuruinen työ. Työ varastoituu jouseen potentiaalienergian muodossa. Energialauseen minimoimisessa juuri näitä potentiaalienergioita pyritään minimoimaan, mikä käytännössä tarkoittaa jousien saattamista lepopituuteen.

KK-algoritmin energialauseesta *E* kaava (8) nähdään yhtäläisyys potentiaalienergiakaavaan (7) [Kam89]:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(| p_i - p_j | - l_{ij} \right)^2, \tag{8}$$

missä k_{ij} jousivakio, l_{ij} teoreettinen jousen pituus ja p_i, p_j partikkeleiden koordinaattiarvot. Energialauseessa jousen tasapainopoikkeama määritellään euklidisen ja teoreettisen pituuden l_{ij} erotuksena. Graafin kokonaisenergia saadaan määritettyä, kunnes kaikkien partikkeleiden ($p_1, p_2, \ldots, p_n$) varaukset on laskettu yhteen. Energialauseessa l_{ij} tarkoittaa teoreettista jousen pituutta p_i ja p_j partikkeleiden välillä. Teoreettinen jousen pituus saadaan laskettua seuraavan kaavan (9) [Kam89] perusteella:

$$l_{ij} = L * d_{ij}, \tag{9}$$

missä L vastaa jousen haluttua pituutta, jonka käyttäjä voi itse määrittää ja d_{ij} muuttuja tarkoittaa lyhintä polkua partikkeleiden välillä. Lyhimmät polut partikkelien välille lasketaan algoritmin alustusvaiheessa ja laskemiseen voidaan käyttää esimerkiksi Floydin algoritmia. Energialauseessa jousien voimakkuuksia säädellään lisäksi seuraavan jousivakion k_{ij} kaavan (10) [Kam89] avulla:

$$k_{ij} = \frac{K}{{d_{ij}}^2}. \tag{10}$$

Jousivakion avulla korostetaan vierekkäisien partikkeleiden vetovoimia toisiinsa ja pitkien välimatkojen vetovoimia vähennetään.

Korvaamalla energialauseeseen partikkeleiden $p_1, p_2, \ldots, p_n$ arvot partikkelien koordinaattien arvoilla voidaan energiakaava kirjoittaa kaavan (11) [Kam89] mukaisesti:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left\{ \left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + {l_{ij}}^2 - 2 l_{ij} \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \right\}, \tag{11}$$

missä x_i, x_j, y_i ja y_j sisältävät vertailtavien partikkelien koordinaattiarvoja. Energialause sisältää näin ollen 2*n* erillistä muuttujaa, joiden sopivilla arvoilla energialause saadaan minimoitua.

Heurastisissa menetelmissä on äärimmäisen vaikea löytää energialauseelle pienintä optimiarvoa, mutta lokaalin minimin löytäminen on huomattavasti nopeampaa ja on usein lähellä varsinaista optimaalista ratkaisua [Sil04]. Newton-Raphson-metodin [Rob95] avulla voidaan ratkaista kaavan (12) lokaali minimointi ongelma, jossa energialause on derivoitu *x*:n ja *y*:n suhteen. Seuraavassa energialauseen kaava (12) ja energialauseen osoittais derivaattojen kaavat (13) ja (14) [Kam89]:

$$\frac{\partial E}{\partial x_m} = \frac{\partial E}{\partial y_m} = 0 \text{ arvoilla } 1 \le m \le n, \tag{12}$$

$$\frac{\partial E}{\partial x_m} = \sum_{i \neq m} k_{mi} \left\{ (x_m - x_i) - \frac{l_{mi}(x_m - x_i)}{\left\{ (x_m - x_i)^2 + (y_m - y_i)^2 \right\}^{\frac{1}{2}}} \right\}, \tag{13}$$

$$\frac{\partial E}{\partial y_m} = \sum_{i \neq m} k_{mi} \left\{ (y_m - y_i) - \frac{l_{mi}(y_m - y_i)}{\left\{ (x_m - x_i)^2 + (y_m - y_i)^2 \right\}^{\frac{1}{2}}} \right\}, \tag{14}$$

kummankin derivaatan saavuttaessa lokaalin minimin saadaan jokaiselle partikkelille lokaali minimi ja näin ollen lokaali minimi myös koko energialauseelle. Energiakaavan derivaattojen muuttujia ei voida laskea kaavasta suoraan, koska derivaatat eivät ole itsenäisiä yhtälöitä vaan vaikuttavat toinen toisiinsa. Ongelma ratkaistaan liikuttamalla vain yhtä partikkelia $p_m(x_m, y_m)$ kerrallaan, jonka aikana muut partikkelit jäädytetään paikoilleen. Algoritmissa korkeimman energiavarauksen sisältävä partikkeli siirretään ensimmäisenä omalle paikalleen. Yksittäisen partikkelin energia taso Δ_m lasketaan kaavan (15) [Kam89] mukaisesti:

$$\Delta_m = \sqrt{\left\{ \frac{\partial E}{\partial x_m} \right\}^2 + \left\{ \frac{\partial E}{\partial y_m} \right\}^2}. \tag{15}$$

Newton-Raphson-metodi perustuu iterointiin, joka lopetetaan kunnes tarpeeksi hyvä tulos on saatu aikaiseksi. Newton-Raphson-metodi aloitetaan arvoilla $\left(x_m^{(0)}, y_m^{(0)}\right)$, jotka vastaavat korkeimman energiatason omaavan partikkelin (x_m, y_m) pistettä. Seuraavan iteroimiskierroksen arvot lasketaan kaavan (16) [Kam89] mukaisesti.

$$\partial x = x_m^{(t+1)} - x_m^{(t)}, \partial y = y_m^{(t+1)} - y_m^{(t)}, \text{ arvoilla } t = 1, 2, \ldots \tag{16}$$

Kaavojen (13) ja (14) funktioiden liittymäkohdat ovat avainasemassa tutkittaessa kaavojen yhteisten muuttujien lokaaleja minimiarvoja. Kahden erillisen energiakaavan kaarien yhteiset pisteet ratkaistaan seuraavista energiakaavojen lineaarisista approksimointikaavoista [Rob95, Kam89]:

$$\frac{\partial^2 E}{\partial x_m^2}\left(x_m^{(t)}, y_m^{(t)}\right)\delta x + \frac{\partial^2 E}{\partial x_m \partial y_m}\left(x_m^{(t)}, y_m^{(t)}\right)\delta y + \frac{\partial E}{\partial x_m}\left(x_m^{(t)}, y_m^{(t)}\right) = 0. \tag{17}$$

$$\frac{\partial^2 E}{\partial y_m^2}\left(x_m^{(t)}, y_m^{(t)}\right)\delta x + \frac{\partial^2 E}{\partial y_m \partial x_m}\left(x_m^{(t)}, y_m^{(t)}\right)\delta y + \frac{\partial E}{\partial y_m}\left(x_m^{(t)}, y_m^{(t)}\right) = 0. \tag{18}$$

$$\delta x = -\frac{\dfrac{\partial E}{\partial x_m}\dfrac{\partial^2 E}{\partial y_m^2} - \dfrac{\partial^2 E}{\partial x_m \partial y_m}\dfrac{\partial E}{\partial y_m}}{\dfrac{\partial^2 E}{\partial x_m^2}\dfrac{\partial^2 E}{\partial y_m^2} - \dfrac{\partial^2 E}{\partial x_m \partial y_m}\dfrac{\partial^2 E}{\partial y_m \partial x_m}}. \tag{19}$$

$$\delta y = -\frac{\dfrac{\partial^2 E}{\partial x_m^2}\dfrac{\partial E}{\partial y_m} - \dfrac{\partial E}{\partial x_m}\dfrac{\partial^2 E}{\partial y_m \partial x_m}}{\dfrac{\partial^2 E}{\partial x_m^2}\dfrac{\partial^2 E}{\partial y_m^2} - \dfrac{\partial^2 E}{\partial x_m \partial y_m}\dfrac{\partial^2 E}{\partial y_m \partial x_m}}. \tag{20}$$

Lineaarisista approksimointikaavoista (17) ja (18) ratkaistaan tuntemattomat muuttujat δx ja δy, joita käytetään hyväksi valittaessa seuraavaan iteroimiskierroksen arvoja kaavassa (16). Tuntemattomien muuttujien kaavat (19) ja (20) sisältävät Jaakobin matriisin osittaisderivaattoja, jotka lasketaan kaavoista (13) ja (14).

KK pseudo-algoritmi [Kam89]:

Algoritmi KK(Graph:G)
Laske d_{ij} arvoilla $1 \le i \ne j \le n$;
Laske l_{ij} arvoilla $1 \le i \ne j \le n$;
Laske k_{ij} arvoilla $1 \le i \ne j \le n$;
Sijoita partikkelit $p_1, p_2, \ldots, p_n$; paikoilleen;
while $(\max_{i\Delta i} > \varepsilon)$
    Sijoita korkeimman energian omaavan partikkelin arvo muuttujaan
    while$(\Delta_m > \varepsilon)$
        laske muuttujat δx ja δy kaavoista (19)(20);
        $x_m := x_m + \delta x$;
        $y_m := y_m + \delta y$;
    End // while
End // while
End // KK

KK-algoritmi aloitetaan alustamalla lyhimmän polun d_{ij}, teoreettisen pituuden l_{ij} ja jousivakion k_{ij} muuttujien arvot. Seuraavaksi syötteeksi annetun graafin partikkelit sijoitetaan sattumanvaraisesti paikoilleen. Ensimmäisessä while-silmukassa testataan korkeimman partikkelin varausta käyttäjän määrittelemään kynnysarvoon ε. Jos kynnysarvo alitetaan, algoritmi lopetetaan, koska graafi on saavuttanut lokaalin miniminsä. Vastaavasti jos kynnysarvoa ei aliteta, siirrytään sisempään while-silmukkaan, jossa partikkelin sijaintia siirrellään Newton-Raphson-metodin palauttamien arvojen perusteella. Sisemmästä while-silmukasta päästään pois, kun siirreltävän partikkelin varaus alittaa kynnysarvon. Algoritmia jatketaan niin pitkään, kunnes kaikkien partikkeleiden varaukset alittavat käyttäjän määrittelemän kynnysarvon.

4.5.5 Simuloitu jäähdytysalgoritmi

Simuloituja jäähdytysmalleja (Simulated Annealing) on käytetty menestyksekkäästi matemaattisissa optimointialgoritmeissa jo pitkään, koska mallin avulla on pystytty löytämään optimaalisia ratkaisuja monimutkaisille kustannusfunktioille.

Mallin tehokkuus perustuu fysikaaliseen analogiaan, joka johdetaan metallien lämpökäsittelystä. Kun sulaa, hehkuvaa metallia jäähdytetään sopivan hitaasti, sen kiderakenne päätyy tavoiteltuun minimienergiatilaan. Liian nopealla jäähdytyksellä aineen hiukkaset löytävät lokaalin minimiarvonsa ja pysähtyvät siihen, eikä tavoiteltua globaalia minimiä saavuteta [Sil04, Goe01].

Davidson ja Harel [Dav96] käyttivät ensimmäisenä simuloitua jäähdystysmenetelmää apuna graafien piirrossa. Simulointimenetelmällä pyritään löytämään kustannusfunktion minimienergia eli optimaaliratkaisu tai lähes optimiratkaisu. Kustannusfunktion tarkoituksena on löytää graafille mahdollisimman hyvä ulkoasu. Hyvä ulkoasu saadaan kokoamalla esteettisyysparametrit samaan lauseeseen eli kustannusfunktioon. Seuraavassa kustannusfunktion kaava (21) [Bat99].

$$f(L) = \sum_{i=1}^{5} \lambda_i f_i(L), \tag{21}$$

kustannusfunktioon $f(L)$ kuuluu viisi erillistä komponenttia $f_i(L)$, jotka kaikki vastaavat jostain kuvan esteettisyysparametrista. Käyttäjällä on mahdollisuus vaikuttaa jokaisen esteettisyysparametrin painotuksiin λ_i. Seuraavassa lista kustannusfunktion esteettisyysparametri komponenteista [Bat99]:

$f_1(L) = \sum_{(u,v)\in V} \left(1/d_{uv}^2\right)$, missä *d* on solmujen *u* ja *v* välinen etäisyys. Funktion avulla vierekkäisten solmujen etäisyys pidetään halutun suuruisena.

$f_2(L) = \sum_{u\in V} \left(\left(1/r_u^2\right) + \left(1/l_u^2\right) + \left(1/t_u^2\right) + \left(1/b_u^2\right)\right)$, missä muuttujat r_u, l_u, t_u ja b_u ovat solmun *u* etäisyydet vasemmasta ja oikeasta sekä ylä- ja alareunasta. Tämä estää solmujen sijoittelun reunojen läheisyyteen.

$f_3(L) = \sum_{(u,v)\in E} d_{uv}^2$, komponentin tarkoituksena on pitää kaarien pituudet sallituissa rajoissa.

$f_4(L)$, Funktion tarkoituksena on vähentää päällekkäisten kaarien olemassaoloa.

$f_5(L) = \sum_{(u \in V),(e \in E)} \left(1 / g(u,e)^2 \right)$, missä $g(u,e)$ on minimietäisyys solmun *u* ja kaaren *e* välillä. Komponentin tarkoituksena on lisätä vierekkäisten solmujen ja kaarien välimatkaa toisiinsa.

Simuloitun jäähdytysmetodin pseudo-algoritmi [Dav96]:

Algoritmi DH(Graph:G)
1) Valitse aloituskonfiguraatio σ ja aloitus lämpötila T;
2) Toista seuraavaa (vakio interointi määrä)
   (a) Valitse uusi konfiguraatio σ', σ konfiguraation naapuristosta;
   (b) Laske kustannusfuntion arvot E ja E', parametreille σ' ja σ;
      Jos ($E' < E$ tai random < $e^{(E-E')/T}$) $\sigma := \sigma'$;
3) Vähennä lämpötilaa ja pienennä naapuriston sädettä;
4) Jos lopetus sääntö on tosi, lopeta algoritmi; Muuten siirry takaisin kohtaan 2;

DH-algoritmissa vaiheessa 1 systeemille arvotaan sattumanvaraisesti lähtökonfiguraation arvo σ, joka sisältää solmujen sijainnit ja vertailupisteen. Alussa määritellään myös aloitus lämpötila T sekä naapuriston säteen suuruus. Seuraavaksi vaiheessa 2 siirrytään silmukkaan, missä vertailupisteen naapuriston säteen sisältä valitaan sattumanvaraisesti uuden konfiguraation vertailupiste. Jos uuden konfiguraation vertailupisteen arvo kustannusfunktiossa on pienempi kuin nykyisen konfiguraation arvo, valitaan uusi konfiguraatio vanhan tilalle. Ellei kustannusfunktio parane uudella arvolla, voidaan konfiguraatio silti hyväksyä todennäköisyydella $P(T,d) = e^{-(d/T)}$, missä d on kustannusfunktioiden erotus ja T on lämpötila. Todennäköisyysfunktio perustuu Boltzmannin jakaumaan, joten alussa lämpötilan *T* ollessa suuri, todennäköisyys huonojen siirtojen hyväksymiselle on suuri, mutta lämpötilan laskiessa siirtojen hyväksymisen todennäköisyys muuttuu pienemmäksi. Iteroimiskierroksien loputtua siirrytään vaiheeseen 3 missä lämpötilaa ja naapuriston sädettä pienennetään jäähdytys- ja sädekertoimien avulla. Algoritmin ajo lopetetaan, jos lopetussääntö vaiheessa 4 on tosi.

Simuloidun jäähdytysmenetelmän hyviä puolia on sen monipuolinen säätämismahdollisuus. Jokaista esteettisyysparametria voidaan painoittaa halutulla tavalla, joilloin ulkoasu voidaan personoida käyttäjäkohtaisesti. DH-algoritmin huonoja puolia on sen vaikea ohjattavuus. Tasapainoisten säätöjen löytäminen esteettisyysparametrien välille voi olla aikaa vievää. Esimerkiksi päällekkäisten kaarien minimoinnin painottaminen voi johtaa symmetrisyyden rikkomiseen. Lisäksi DH-algoritmi on äärimmäisen paljon laskenta-aikaa kuluttava algoritmi, joten algoritmia ei voi käyttää interaktiivisissa ohjelmissa [Bra96]. Alkuperäisestä algoritmista on tehty variaatio, joka on hieman alkuperäistä nopeampi [Har95], mutta variaationkin vahvuus perustuu sen kykyyn piirtää esteettisesti hyviä ulkoasuja.

4.6 Algoritmien vertailu

Algoritmien tutkimusraporteista löytyy tarkkoja suorituskyky- ja esteettisyysvertailuja muiden algoritmien kesken, mutta näiden tuloksien uudelleen suorittaminen testioloissa on usein vaikeaa [Bra96]. Vaikeaksi tehtävässä tekee algoritmien satunnaiset ominaisuudet, jotka vaihtelevat annettavan syötteen ja parametrien mukaan. Algoritmien toteutuksella on myös suuri vaikutus suorituskykyyn.

Seuraavat vertailut perustuvat suurimmaksi osaksi Himsoltin tutkimusraporttiin [Bra96], jonka tuloksia löytyy liitteistä I ja II. Liitteistä löytyvät käyrät ovat suuntaa antavia tuloksia, jotka saattavat vaihdella olosuhteiden mukaan, mutta käyristä voidaan hahmotella yleiskuva algoritmien ominaisuuksista. Tutkimusraportissa vertaillaan algoritmien suoritusnopeuksia ja ulkoasujen paremmuuksia. Ulkoasun vertailuista päällekkäisten kaarien vertailu on jätetty huomioimatta, mitä voidaan pitää testin heikkoutena.

Testin perusteella GEM- ja KK-algoritmit olivat huomattavasti muita nopeampia. Testissä käytettyä KK-algoritmia oli muokattu hieman alkuperäisestä versiosta, mikä nopeutti huomattavasti KK-algoritmin suorituskykyä. Alkuperäinen KK-algoritmi ei nimittäin pärjännyt GEM:in tutkimusraportin (liite II) mukaan GEM- ja FR-algoritmien suorituskyvylle. GEM-raportin mukaan GEM-algoritmi olisi

vertailtavien algoritmien nopein suurilla graafeilla [Fri95], mutta uusimman tutkimusraportin [Beh99] mukaan (liite III) FR-algoritmi olisi vertailtavien algoritmien nopein vaihtoehto suurilla graafeille. Vertailtavista algoritmeista selvästi hitain oli DH-algoritmi, jonka toteutus poikkeaa perinteisistä jousialgoritmeista.

Vertailtaessa algoritmien ulkoasujen paremmuutta voidaan kaaren vakiomitalla ja symmetrisyydellä tehdä erot ulkoasujen välille. Vertailtavat algoritmit pyrkivät samankaltaisiin esteettisyyskriteereihin, joten samankaltaiset lopputulokset eivät tule yllätyksenä. Voimiin perustuvien algoritmien tuottamien ulkoasujen välillä ei näin ollen ole suuria eroja [Dav96]. Vertailtavista algoritmeista KK-algoritmi tuottaa keskimääräisesti parhaimman ulkoasun satunnaisesti valittavalle graafille [Bra96]. DH-algoritmi on vertailun säädettävin eli mahdollisia esteettisyyskriteereitä voidaan painottaa käyttäjän halujen mukaisesti. Tosin algoritmia on vaikea säätää, mikä hidastaa oikean ulkoasun löytämistä, mutta oikeilla parametrin arvoilla algoritmilla voidaan tuottaa graafille paras ulkoasu [Bra96].

Vertailun jokainen algoritmi soveltuu hyvin yleisten suuntaamattomien graafien piirtämiseen. Algoritmien yleisenä ongelmana voidaan pitää irrallisten solmujen aiheuttamaa kuvan vääntymistä, mutta kytketyllä graafilla ongelmaa ei ilmene [Beh99]. Mikään algoritmi ei nouse vertailussa ylivoimaisesti parhaaksi, koska jokaisessa algoritmissa on ominaisuuksia, jotka mahdollistavat parhaimman ulkoasun rakentamisen tietyn tyyppisille graafille. Himsoltin tutkimusraportissa suositellaan käytettäväksi GEM- ja KK-algoritmeja ensimmäiseksi. Alle 60 solmun graafeilla FR-algoritmia voidaan pitää hyvänä valintana. Jos nopeus ei ole algoritmin kriteeri, suositellaan käytettäväksi DH-algoritmia [Bra96].

4.7 Algoritmien kompleksisuudet

Voimiin perustuvien algoritmien tarkkoja kompleksisuusarvoja on vaikea arvioida, koska heurastisessa optimoinnissa iteraatiokierrosten määrä voi vaihdella graafien välillä melkoisesti [Kam89]. Piirrettävän graafin koko ja kompleksisuus voivat myös vaikuttaa algoritmin kokonaiskompleksisuuteen. Lisäksi algoritmien

parametreilla voidaan vaikuttaa huomattavasti algoritmin kompleksisuuteen. Sisään tulevan graafin solmujen satunnainen järjestys voi myös vaikuttaa lopputulokseen ja näin ollen vaikuttaa algoritmin kompleksisuuteen [Fri95].

Algoritmien kompleksisuuksia voidaan kuitenkin vertailla yleisellä tasolla, jolloin algoritmeissa käytetään oletusarvoisia parametreja. Lisäksi algoritmien kompleksisuus voidaan laskea kiinteänpituisten iteraatiokierrosten lukumäärän mukaan, poikkeuksena alkuperäinen KK-algoritmi, jonka iteraatiokierroksia ei ole rajattu [Tun99]. Yleisellä tasolla arvioidut kompleksisuudet ovat vain suuntaa antavia ja usein algoritmit toimivat huomattavasti nopeammin ainakin pienemmillä graafeilla.

Kaikki vertailtavat algoritmit perustuvat iterointiin, joten tämän vuoksi algoritmien kompleksisuuksia vertaillaan globaalin iteraatiokierroksen perusteella [Tun99, Beh99]. Algoritmin iteraatiokierroksen kulkemista alusta loppuun kutsutaan globaaliksi iteraatiokierrokseksi [Beh99]. Vertailussa kannattaa ottaa huomioon myös globaalien iteraatiokierrosten aikana tapahtuvat graafin muutokset. Osa algoritmeista liikuttaa kaikkia renkaita kerralla, mutta osa liikuttaa vain yhtä rengasta kerrallaan.

Taulukko 1. Voimiin perustuvien algoritmien laskennalliset kompleksisuudet

	Jouset	**Rengas-Rengas vastustavat voimat**	**Rengas-Jousi vastustavat voimat**	**Globaali iteraatio kompleksisuus**
Eades	$O(m)$	$O(n^2)$		$O(n^2)$
FR	$O(m)$	$O(n^2)$ $O(n)$ ”Ristikko”		$O(n^2)$ $O(m)$
GEM	$O(m)$	$O(n^2)$		$O(n^2)$
KK	$O(Tn)$ $O(n^2)$ Rajattu T			$O(Tn)$
DH	$O(m)$	$O(n^2)$	$O(nm)$	$O(nm)$

Taulukossa 1 m vastaa graafin jousien lukumäärää, n renkaiden lukumäärää ja T KK-algoritmin sisemmän solmun iteraatioiden lukumäärää.

Taulukon 1 perusteella Eadesin, FR- ja GEM-algoritmien globaalien iteraatioiden kompleksisuudet ovat luokkaa $O(n^2)$. Kyseiset algoritmit tarvitsevat empiirisen estimoinnin perusteella $O(n)$ globaalia iterointikierrosta saavuttaakseen halutun lopputuloksen [Beh99], joten kokonaiskompleksisuudeksi saadaan $O(n^3)$. FR-algoritmin ristikkomuunnelman globaalin iteraatiokierroksen kompleksisuus on taulukon nopein, mutta muunnelman nopeus hajottaa graafin esteettisyyden [Tut99].

Alkuperäisen KK-algoritmin globaalin iteraatiokierroksen kompleksisuus $O(Tn)$ on muista poikkeava, koska sisäisen solmun iteraatioiden lukumäärä T voi vaihdella graafien mukaan [Kam89]. KK-algoritmin variaatiossa [Bra96, Beh99] sisäisen solmun iteraatiot on rajattu n*10 kertaan. Variaation kompleksisuudeksi saadaan näin ollen $O(n^2)$. KK-algoritmin kompleksisuudessa on otettava huomioon myös alustuksen kompleksisuus, joka loppujen lopuksi määrää koko algoritmin

kompleksisuuden. Kamadan ja Kawain ehdotuksessa lyhyimpien polkujen laskentaan käytetään Floydin algoritmia, jonka kompleksisuus on $O(n^3)$. Lyhyimpien polkujen laskentaan voidaan käyttää myös Dijkstran algoritmia, jolloin kompleksisuus $O(nm \log n)$ alenee huomattavasti [Cor90, Tut99]. Lyhyimpien polkujen tallennus on $O(n^2)$ tilakompleksinen, joten suurien graafien vaatima muistimäärä voi osoittautua pullonkaulaksi.

DH-algoritmin globaalin iteraariokierroksen kompleksisuus on $O(nm)$, joka muodostuu renkaan siirron aiheuttamista laskentakuluista. Renkaan siirrossa renkaan ja jousen välimatkan ja päällekkäisten jousien minimoiminen ovat laskennallisesti yhtä raskaita toimenpiteitä, joista muodostuu kyseinen $O(nm)$ kompleksisuus [Dav96]. Globaalien iteroimiskierroksien iterointi on rajoitettu 30*n kertaan, jolloin DH-algoritmin kokonaiskompleksisuudeksi saadaan $O(n^2 m)$. DH-algoritmin tilakompleksisuus $O(\max(n^2, m^2))$ on vielä KK-algoritmiakin suurempi.

Suljettujen lähdekoodien ja optimoitujen voimiin perustuvien algoritmien aikakompleksisuudet ovat usein huomattavasti pienempiä kuin julkisilla ja alkuperäisillä versioilla. Esimerkiksi FADE- ja JIGGLE-algoritmien aikakompleksisuudet ovat noin $O(m + n \log n)$ luokkaa [Qui00, Qui03, Tun98].

5 SURFVISUAL-TYÖKALUN TOTEUTUS

Käytännön työn tarkoituksena oli toteuttaa verkon topologian visualisointityökalu. Aiemman nimeämiskäytännön mukaisesti työ nimettiin SurfVisual-visualisointityökaluksi.

SurfVisual on tarkoitus liittää osaksi TeliaSoneran SurfManager-verkonhallintajärjestelmää. SurfManagerin tietokanta sisältää mm. suuren määrän eri tekniikoiden topologiatietoja, joiden pohjalta SurfVisual mallintaa eritasoisia topologiakarttoja.

SurfVisual-visualisointityökalun olen suunnitellut ja toteuttanut itsenäisesti.

5.1 SurfManager-ympäristö

SurfManager-verkonhallintajärjestelmä kokoaa TeliaSoneran erilaiset verkonhallintakomponentit yhtenäisen hallinta-arkkitehtuurin piiriin. SurfManager on konsepti, joka koostuu useasta eri palvelusta. SurfManagerilla on oma teknologiariippumaton laitetietokantansa, jossa säilytetään laitteiden topologia ja laitteiden konfiguraatiotietoja. SurfManagerin oleellisin ero TeliaSoneran tietoverkon muihin teknologiariippumattomiin tietojärjestelmiin on sen sisältämä topologiatieto.

SurfManager-sovelluksista SurfTeam on verkonhallintakeskukselle tarkoitettu verkonhallintatyökalu, jolla valvotaan ja konfiguroidaan monia eri teknologioita. SurfTeam-verkonhallintatyökalun hallinnointi-ikkunasta käynnistetään myös SurfVisual-työkalu, jolla mallinnetaan syötteenä saatujen parametrien mukainen verkkotopologia.

5.2 SurfVisual toteutuksen työkalut

SurfVisual toteutuksessa käytettiin Perl-ohjelmointikieltä (Practical Extraction and Report Language). Perl-ohjelmointikieleen päädyttiin, koska Perlin Graphviz-

moduuli sisältää tehokkaita piirtoalgoritmeja. Piirtoalgoritmin tuottamat ulkoasut mallinnetaan SVG-vektorigraafikkaa (Scalable Vector Graphic) hyväksikäyttämällä. SVG-vektorigrafiikan interaktiivisuus on toteutettu JavaScriptillä.

5.2.1 Perl

Hieman jo vanhaksi käynyt Perl-ohjelmointikieli soveltui hyvin käytännön työn ohjelmointikieleksi. Perl on optimoitu käymään läpi tekstitiedostoja, poimimaan niistä tietoa ja tulostamaan tämän tiedon pohjalta raportteja. Optimoitu tekstitiedoston läpikäyminen osoittautui topologiatietojen läpikäynnissä tarpeelliseksi ominaisuudeksi. Perl osoittautui myös moduuleilla helposti laajennettavaksi kokonaisuudeksi.

Perlin omista moduuleista hyväksikäytettiin CGI-moduulia, mikä helpotti CGI-kutsujen käsittelyä. Kolmannen osapuolen moduuleista koostuvasta CPAN-moduulikirjastosta käytettiin OraPerl-, Graphviz- ja IPv4Addr-moduuleita.

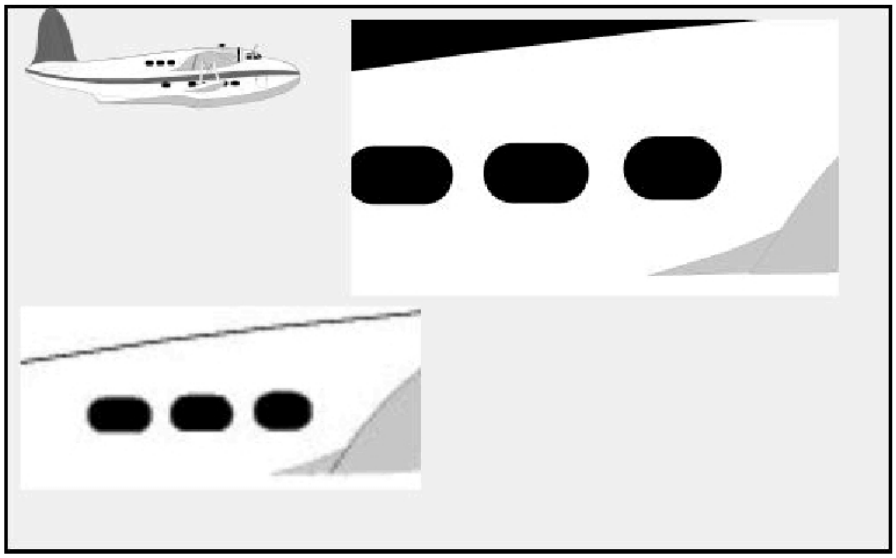
Topologiatiedon kerääminen suoritettiin Oracle-tietokannasta. Perlissä Oracle-tietokantaa käsitellään OraPerl-tietokantarajapintaa hyväksikäyttämällä. OraPerl-rajapinnalla tietokanta hakujen suorittaminen oli helppoa ja tehokasta myös suuremmilla tietomäärillä.

Graphviz-moduulin avulla toteutettiin verkkojen ulkoasu. Graphviz-moduulin avulla rakennetusta ulkoasusta poimitaan laitteiden sijaintien koordinaatit talteen myöhempää SVG-kuvan rakennusta varten. Verkkotopologioiden ulkoasujen tuottamiseen Graphviz-moduulista löytyvät Neato- ja Dot-ohjelmat. Neato soveltuu suuntaamattomien graafien ja Dot suunnattujen graafien ulkoasujen rakentamiseen. Suuntaamattomien graafien ulkoasujen rakentamiseen Neatossa käytetään Kamadan ja Kawain algoritmia.

IPv4Addr-moduulia käytettiin IP-osoitteiden käsittelyssä hyväksi.

5.2.2 SVG

SVG on W3C:n (World Wide Web Consortium) SVG-työryhmän kehittämä kaksiulotteisen vektorigrafiikan esityskieli. Työryhmään kuuluu suuria yrityksiä kuten Adobe, Agfa, Apple, Canon, Corel, Ericsson, HP, IBM, Kodak, Macromedia, Microsoft, Nokia, Sharp ja Sun Microsystems. Avoimena standardina SVG on täysin riippumaton laitteisto- ja ohjelmistovalmistajista. SVG 1.1 versio on viimeisin W3C:n hyväksymä SVG-esityskielistandardi. [W3c04, Mik01]



Kuva 17. Rasteri vs. SVG. [Duc02]

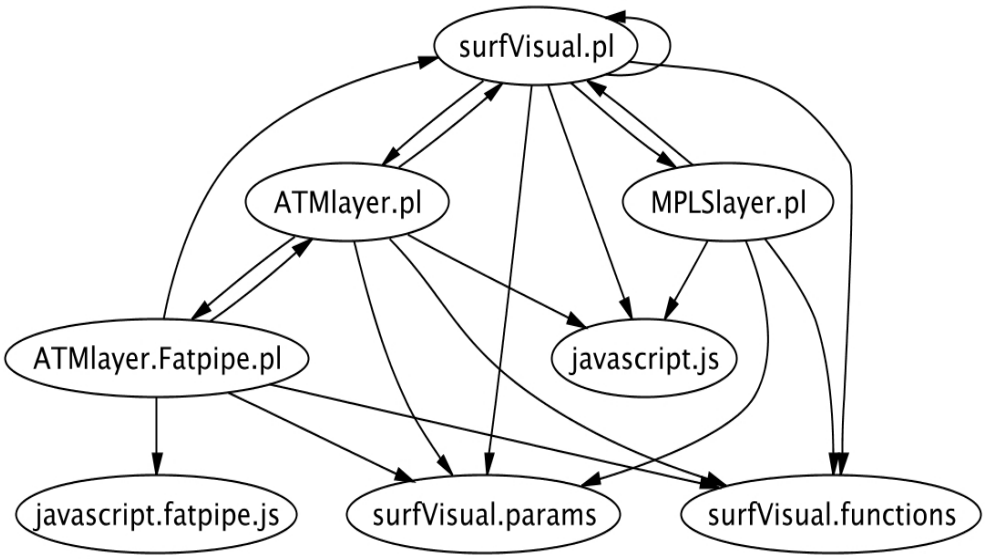
Käytännön työssä SVG-formaatti osoittautui oikeaksi valinnaksi. SVG:een tarjoamat ominaisuudet ovat huomattavasti monipuolisemmat perinteiseen rasterigrafiikkaan verrattuna. Perinteisessä rasterigrafiikassa kuvat esitetään yleensä GIF- tai JPEG-formaateissa. Rasterikuvan jokainen esitettävä pikseli tallennetaan ominaisuuksineen, kun vastaavasti vektorigrafiikassa kuvan jokaista pikseliä ei tarvitse kuvata erikseen, vaan kuva muodostetaan muotojen ja polkujen avulla [Mik01]. Käytännön työssä objektien väliset suhteet esitetään viivalla. SVG-formaatissa viiva muodostetaan yhdellä line-tagilla, jolle annetaan parametreina viivan aloitus- ja päätepisteet. Rasterigrafiikassa viivat toteutetaan viiva-algoritmeja hyväksikäyttämällä, jolloin jokainen viivan pikseli piirretään erikseen.

Virheetöntä zoomausta voidaan pitää yhtenä SVG-formaatin suurimmista vahvuuksista. SVG-kuva määritellään objekteina, jolloin kuva pysyy zoomauksen aikana virheettömänä ja tarkkana. Virheetön ja tarkka zoomaus mahdollistaa näin ollen suuren verkon esityksen rajoitetussa näkymässä. Koko verkon kattavasta aloituskuvasta voidaan rajata haluttu verkonosa, joka zoomauksen jälkeen on edelleen virheetön ja tarkka. Kuvasta 17 nähdään SVG- ja rasteri-zoomauksen ero, jossa SVG-zoomaus (ylempi) on pysynyt tarkkana, mutta rasteri-zoomaus (alempi) on muuttunut epäselväksi.

SVG-formaatti perustuu XML-formaatiin (Extensible Markup Language), joten SVG-kuva on luettavissa oleva teksitiedosto. SVG-formaattia on mahdollista laajentaa kaikilla XML-formaattiin perustuvilla tekniikoilla [Sch04]. Dynaaminen sisältö ja interaktiivisuus ovat näin ollen osa SVG-formaattia. Käytännön työssä käyttöliittymä toteutettiin yhdellä dynaamisella SVG-kuvalla. SVG-kuva muuttuu käyttäjän interaktiivisen toiminnan perusteella. Yksi käytännön työn interaktiivisista vaatimuksista oli verkkotopologian laitteiden liikuttaminen. JavaScriptillä toteutettu liikuttaminen testattiin SVG- ja rasterikuvilla. SVG-kuvan laitteiden liikuttaminen oli jouhevaa, koska käytännössä vain viivan päätepisteen ja laitteen koordinaatti parametrit muuttuivat. Rasterikuvan laitteiden liikuttamisessa viivan jokaiselle pikselille lasketaan uusi arvo, minkä johdosta liikuttaminen muuttuu “tökkiväksi”.

5.3 SurfVisual-työkalun rakenne

SurfVisual-työkalun rakenne muodostuu kolmesta pääelementistä ja näiden apuelementeistä. Pääelementtien avulla rakennetaan eri tasoisia verkkotopologioita. Rakenteen keskeisin elementti on surfVisual.pl. SurfVisual.pl pääelementin avulla muodostetaan verkkotopologian ensimmäinen näkymä, josta voidaan siirtyä verkkotopologian eri tasoille. Verkkotopologian eri tasot muodostetaan ATMlayer.pl ja MPLSlayer.pl pääelementtejä hyväksikäyttämällä.



Kuva 18. SurfVisual-työkalun rakenne.

SurfVisual-työkalussa käytettävät elementit (kuva 18):

- surfVisual.pl: SurfVisual on rakenteen keskeisin elementti. SurfVisual-elementin avulla rakennetaan käyttäjälle ensimmäinen näkymä, josta voidaan siirtyä verkkotopologian eri tasoille. Elementin avulla muodostetaan IP-kerroksen mukainen verkkotopologia.

- ATMlayer.pl: ATMlayer.pl-elementin avulla rakennetaan ATM-kerroksen mukainen verkkotopologia. Jos kahden laitteen välillä kulkee useampia linkkejä, yhdistetään linkit putkeksi, joka toimii hyperlinkkinä ATMlayer.Fatpipe.pl apuelementille (kuva 18).

- ATMlayer.Fatpipe.pl: ATMlayer.Fatpipe.pl-apuelementin avulla mallinnetaan kaikki kahden ATM-kytkimen väliset linkit.

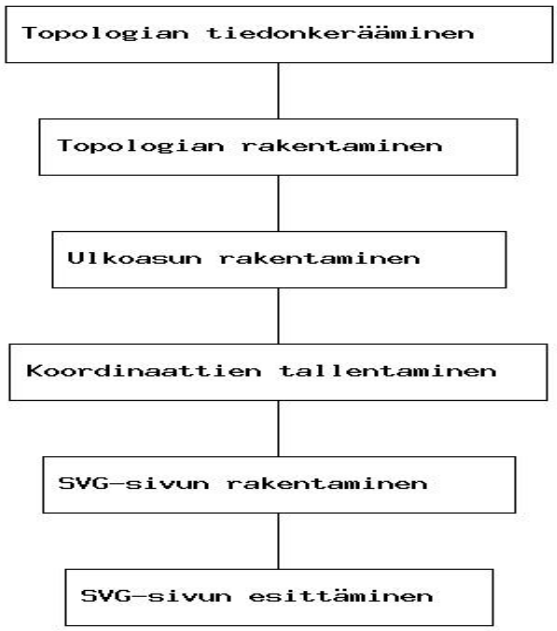
- MPLSlayer.pl: MPLSlayer.pl-elementin avulla mallinnetaan kaikki verkkoon kuuluvat MPLS-verkot.

- javascript.js & javascript.fatpipe.js: JavaScript-apuelementtien avulla muodostetaan SVG-sivuille dynaamisuus ja interaktiiviset toiminnot.

- surfVisual.params & surfVisaul.functions: surfVisual-apuelementeihin on koottu yleisiä funktioita ja parametreja, joita surfVisual-työkalun elementit käyttävät hyväkseen.

5.4 SurfVisual-työkalun elementin rakenne

SurfVisual-työkalun pääelementtien tarkoituksena on mallintaa eri tasoisia verkontopologioita. Verkontopologian mallintaminen eri tasoilla on lähes samanlaista, joten pääelementtien rakennetta voidaan pitää pääpiirteittäin samanlaisena.



Kuva 19. Elementin rakenne

Pääelementtien toiminta alkaa topologian tiedonkeräämisellä (kuva 19). Verkon topologiatiedot löytyvät tietokannasta, joten elementti suorittaa lukuisia tietokantakutsuja topologiarakenteen selvittämiseksi. Topologiarakenteen selvittyä tallen-

netaan objektit ja objektien väliset suhteet taulukoihin tulevien funktioiden hyväksikäytettäväksi.

Topologiataulukoista löytyvät muuttujat verkon topologian rakenteen muodostamiseen, jotka seuraavaksi yhdistetään yhdeksi kokonaisuudeksi Perlin Graphviz-moduulin avulla. Graphviz-moduulin funktioille syötetään topologiataulukoiden muuttujien arvot, jolloin Graphviz-moduulin verkkomuuttujaan generoituu verkon ajonaikainen topologian rakenne.

Graphviz-moduulin verkkomuuttuja sisältää verkon topologian rakenteen, joka annetaan syötteenä ulkoasualgoritmille. Ulkoasualgoritmilla rakennetaan topologian rakenteelle ulkoasu. Ulkoasualgoritmeina käytetään Graphviz-moduulin omaa algoritmia suunnatuille verkoille sekä KK-algoritmia suuntaamattomille verkoille.

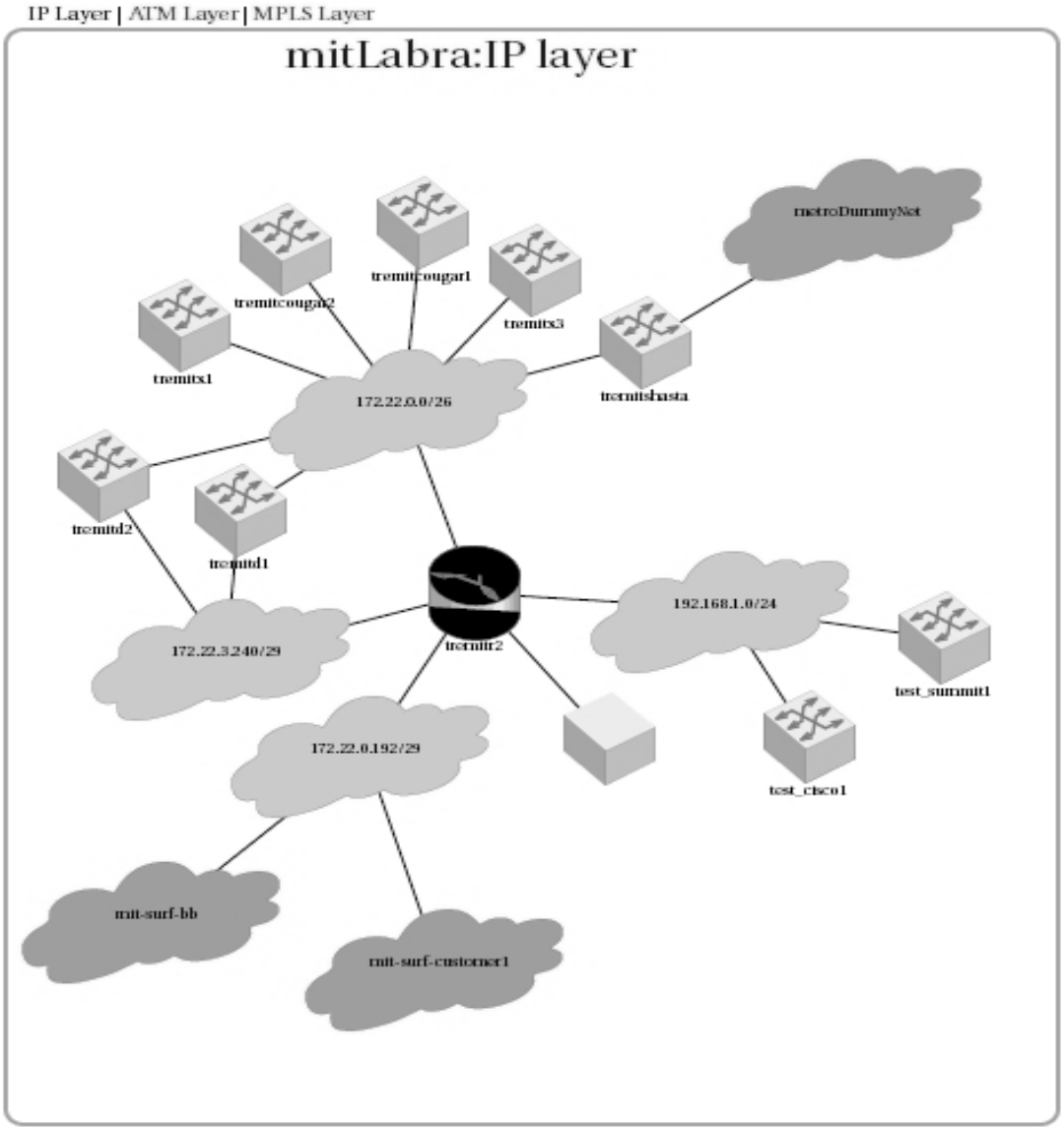
Ulkoasualgoritmin rakentama ulkoasu on päivittynyt Graphviz-moduulin verkkomuuttujaan. Käytännössä verkkomuuttujan objektimuuttujien koordinaattiarvot päivittyvät. Verkkomuuttujan sisältämiin objektimuuttujiin ei voida viitata suoraan, joten koko verkon koordinaattiarvot on tulostettava tiedostoon. Tiedostosta objektien koordinaattiarvot erotellaan toisistaan ja tallennetaan koordinaattitaulukkoon.

Laitteiden koordinaattitietojen perusteella voidaan rakentaa varsinainen SVG-sivu. SVG-sivu koostuu eri tyyppisistä laitteista, joten SVG-sivun rakennus vaiheessa tarkastetaan laitteen tyyppi ja piirretään laitteen mukainen ikoni. Laitteen ikoni sijoitetaan piirtoalustalle koordinaattitaulukon koordinaattien arvojen mukaisesti. Laitteiden väliset suhteet piirretään viivoina laitteiden välille.

Viimeisenä suoritetaan SVG-sivun esitys. SVG-sivu esitetään osana HTML-sivu kokonaisuutta.

5.5 SurfVisual-työkalun näkymiä

Edellisissä kappaleissa on puhuttu paljon pääelementeistä ja pääelementtien rakenteista. Tämän kappaleen tarkoituksena on esitellä muutamia SurfVisual-työkalun elementtien tuottamia topologiakuvia.

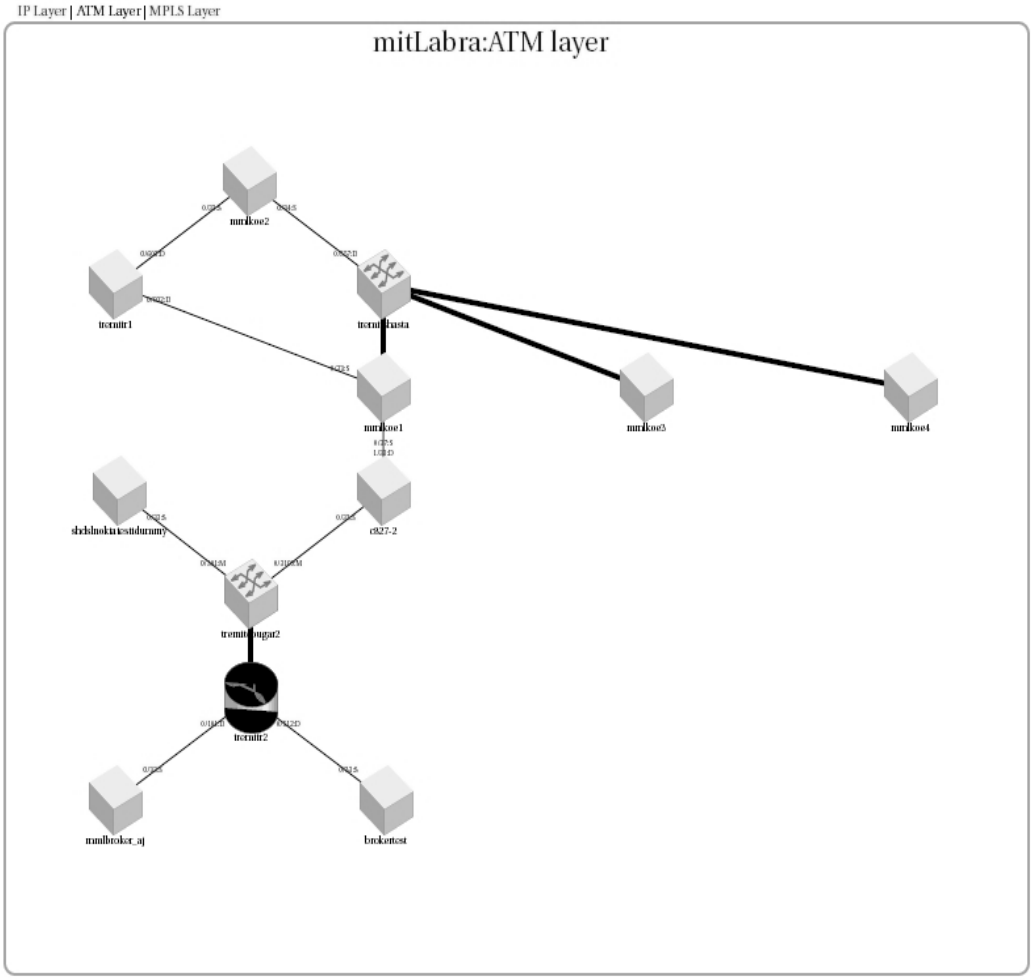


Kuva 20. surfVisual.pl-elementin tuottama kuva.

SurfVisual.pl-elementin tuottamassa kuvassa (kuva 20) esitetään IP-kerroksen tasoinen verkkotopologia. Vaaleat pilvet esittävät aliverkkoja, jotka toimivat laitteita yhdistävinä linkkeinä. Pilviä hyväksikäyttämällä laitteiden välisiä linkkejä ei

tarvitse piirtää suoraan laitteelta laitteelle, jolloin kuvaan saadaan huomattavasti enemmän selkeyttä. Tummilla pilvillä mallinnetaan naapuriverkkoja, jotka toimivat myös hyperlinkkeinä naapuriverkkoon.

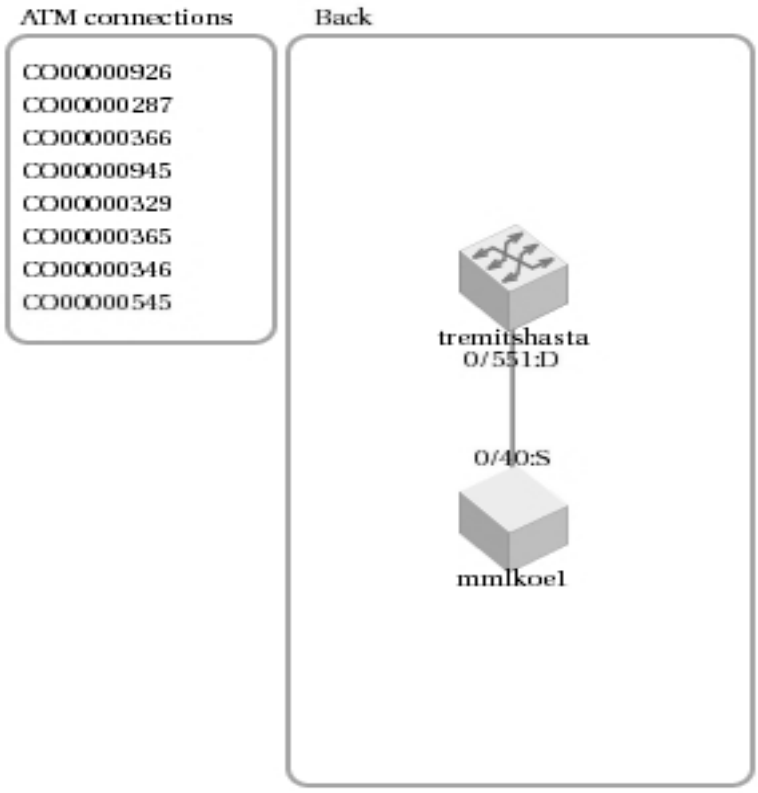
Kuvasta nähdään KK-algoritmin tuottaman kuvan esteettisyysparametrien painotukset. KK-algoritmissa pyritään päällekkäisten kaarien minimoimiseen ja symmetrisyyteen. Tuotetusta kuvasta ei löydy yhtään päällekkäistä kaarta ja myös kuva on symmetrinen, joten algoritmi on tuottanut halutunlaisen ulkoasun. Kuvasta nähdään myös algoritmin suorajanainen piirtämiskäytöntö, joka helpottaa laitteiden yhteyksien havainnointia.



Kuva 21. ATMlayer.pl-elementin tuottama kuva.

ATMlayer.pl-elementin tuottamassa kuvassa (kuva 21) esitetään ATM-kerroksen tasoinen verkkotopologia. Kahden laitteen väliseen linkkiin lisätään VPI/VCI-parametrien arvot sekä linkin pään tyyppi. ATM-linkissä parametrien arvot ja linkin pään tyyppi (lähde/kohde) merkitään linkin kumpaankin päähän, jolloin linkin suunta on nähtävissä. Jos kahden laitteen välillä kulkee useampia linkkejä, yhdistetään linkit yhdeksi linkiksi (paksu viiva), joka toimii hyperlinkkinä ATM-layer.Fatpipe.pl-elementille.

ATM-verkko on suunnattu verkko, joten kuvaa ei toteuteta KK-algoritmilla vaan Graphvizin omalla suunnatuille graafeille tarkoitetulla algoritmilla. Kuvasta voidaan havaita hierarkkisen piirtomallin hyväksikäyttö. Hierarkkinen piirtomalli näyttäisi sopivan hyvin ATM-verkon tapaisille suunnatuille verkoille. Hierarkkinen ja suorajanainen piirtämiskäytäntö tuottaa kuvalle loogisen ja helposti seurattavan kuvan.



Kuva 22. ATMlayer.Fatpipe.pl -elementin tuottama kuva.

ATMlayer.Fatpipe.pl-elementissä mallinnetaan kahden ATM-kytkimen väliset yhteydet. Kuvan (kuva 22) näkymä on saatu aikaiseksi painamalla yhtä kuvan (kuva 21) hyperlinkeistä (paksu viivaa). Listan ATM-yhteyksistä käyttäjä voi valita ATM-yhteyden, jonka VPI/VCI-parametrien arvot ja linkin pään tyyppi esitetään.

6 YHTEENVETO

SurfVisual-visualisointityökalulla verkonhallintatyökalun sisältämät tiedot laitteista ja laitteiden välisistä suhteista on mahdollista visualisoida helposti hahmotettavaksi kuvaksi. SurfVisual-työkalun avulla IP-, ATM-, MPLS- ja VPN-verkot voidaan visualisoida topologiakartoiksi, joista käyttäjälle muodostuu nopeasti näkymä verkon laitteista ja topologiasta.

Verkon topologian visualisoiminen vaatii graafin piirron perusteiden tuntemista. Käytännön työn kokemusten perusteella oikeita piirtomalleja ja piirtoalgoritmeja hyväksikäyttämällä saadaan verkon topologian ulkoasusta toivotunlainen. Jos topologian mallinnukseen löydetään hyvä algoritmi, kannattaa algoritmin tarjoamat parametrit käydä huolellisesti läpi. Parametrien arvoilla voidaan vaikuttaa huomattavasti joidenkin algoritmien suoritusaikoihin. Tässä työssä käsiteltiin laajasti voimiin perustuvia algoritmeja ja niiden parametreja, joita voidaan hyödyntää tulevaisuuden visualisointiprojekteissa.

Käytännön työn kokemusten perusteella laajakaistaverkon topologian visualisoiminen on mahdollista, jos käytössä on tarvittavat topologiatiedot. IP Topology Manager -työkalu osoittautui toimivaksi toteutukseksi, joka keräsi ja päivitti topologiatietoja tietokantaan.

SurfVisual-visualisointityökalun jatkokehityksen tavoitteena on käytettävyyden parantaminen ja uusien ominaisuuksien lisääminen. Uusia ominaisuuksia voisivat olla hälytysjärjestelmän tietojen hyväksikäyttäminen sekä työkalun näkymän laajentaminen. Hälytysjärjestelmän tietojen perusteella vikatilassa oleva laite voitaisiin visualisoida esimerkiksi punaisella värillä, jolloin ongelmallinen laite olisi nopeasti paikallistettavissa. Visualisointityökalun aloitus näkymää voitaisiin laajentaa Suomen kartan laajuiseksi, jolloin kaikki Suomen verkot olisivat nopeasti valittavissa ja visualisoitavissa.

VIITTEET

[Ads97] ADSL Forum Technical Report TR-002, Network Migration, 1997.

[Ant98] Antikainen, Jarno. Runkoverkonhallinnan soveltaminen lähiverkkoihin. Diplomityö, TTKK, Ohjelmistotekniikka, 1998.

[Atm04] Frame Relay Interworks, The ATM Forum, 2004.
Saatavissa: http://www.atmforum.com/aboutatm/frame.html
Viitattu: 30.7.2004

[Bat94] Di Battista, G., Eades, P., Tamassia, R. ja Tollis, I. G. Annotated Bibliography on Graph Drawing Algorithms. Computational Geometry: Theory and Applications, 1994.

[Bat99] Di Battista, G., Eades, P., Tamassia, R. ja Tollis, I. Graph Drawing: Algorithms for the Visualization of Graphs, Prentice Hall, Upper Saddle River, 1999.

[Beh99] Behzadi, Lila. An improved spring-based graph embedding algorithm and Layout-Show: a Java environment for graph drawing. Master's thesis, York University, North York, Ontario, Canada, 1999.

[Bra96] Brandenburg, F. J., Himsolt, M. ja Rohrer C. An experimental comparison of force-directed and randomized graph drawing algorithms. Proceedings of the 3rd International Symposium on Graph Drawing, Springer-Verlag, 1027, 76-87, 1996.

[Chr04] Christov, Ivan. The Crossing Number of a Graph, Massachusetts Institute of Technology, Cambridge, Boston, 2004.

[Cis04] Simple Network Management Protocol. Cisco Systems, Inc., 2004.

[Cis04a] Asynchronous Transfer Mode. Cisco Systems, Inc., 2004.

[Cis04b] Frame Relay. Cisco Systems, Inc., 2004.

[Cis04c] MPLS/Tag Switching. Cisco Systems, Inc., 2004.

[Cor90] Cormen T., Leiserson C. ja Rivest R. Introduction to Algorithms, MIT Press, London, 1990.

[Dav96] Davidson, R ja Harel, D. Drawing Graphs Nicely Using Simulated Annealing, ACM Transactions on Graphics, 15, no. 4, 301-331, 1996.

[Duc02] Duce, D., Herman, I. ja Hopgood, B. SVG Tutorial, World Wide Web Consortium, 2002.
Saatavissa: http://www.w3.org/2002/Talks/www2002-svgtut-ih/hwtut.pdf
Viitattu: 30.7.2004

[Ead84] Eades, P. A heuristic for graph drawing. Congressus Numerantium, no. 42, 149-160, 1984.

[Fri95] Frick, A., Ludwig, A. ja Mehldau, H. A Fast Adaptive Layout Algorithm for Undirected Graphs, In Proceedings of Graph Drawings ’94, 1995.

[Fru91] Fruchterman, T ja Reingold E. Graph drawing by force-directed placement, Software-Practice and Experience, 1129-1164, 1991.

[Gar83] Garey, M. R. ja Johnson D. S. Crossing Number is NP-Complete, Society for Industrial and Applied Mathematics Journal, Algebraic and Discrete Methods, 4, no. 3, 312-316, 1983.

[Goe01] Goel A. CS 6660 Intelligent Agents, Intelligent Agents at the College of Computing of Georgia Institute of Technology, 2001.
Saatavissa: http://www.cc.gatech.edu/classes/AY2002/cs6660_fall/notes4/
Viitattu: 30.7.2004

[Har95] Harel, D ja Sardas, M. Randomized graph drawing with heavy-duty preprocessing. Journal of Visual Languages and Computing, 6, no. 3, 233-253, 1995.

[Him95] Himsolt, M. Comparing and evaluating layout algorithms within GraphEd. Journal of Visual Languages and Computing, 6, no. 3, 93-100, 1995.

[Int04] Simple Network Management Protocol, Interspeak AB, 2004.
Saatavissa: www.interpeak.com/files/snmp.pdf
Viitattu: 30.7.2004

[Kam89] Kamada, T. ja Kawai, S. An algorithm for drawing general undirected graphs. Information Processing Letters, 31, no. 1, 7-15, 1989.

[Ker02] Kerttula, Esa. 010635000 Telematiikan erikoiskurssi, Lappeenrannan teknillinen yliopisto, 2002.

[Kes98] Keshav, S. Domain Topology Generation Project Octopus, CNRG Research Group, 1998.
Saatavissa:
http://www.cs.cornell.edu/cnrg/topology_aware/topology/topology.html
Viitattu: 30.7.2004

[Kru80] Kruskal J. B. ja Seery J. Designing Network Diagrams. Proceedings of the First General Conference on Social Graphics, U. S. Department of the Census, Washington, D.C., no. 49, 22-50, 1980.

[Kum96] Kumar, A. ja Fowler, R.H. A Spring Modeling Algorithm to Position Nodes of an Undirected Graph in Three Dimensions. Technical Report Department of Computer Science, University of Texas, America, 1996.

[Laa04] Eri teknologiat, Laajakaista.fi, Liikenne- ja viestintäministeriö, 2004.
Saatavissa: http://www.laajakaistainfo.fi/teknologiat/index.php
Viitattu: 30.7.2004

[Lah98] Lahdensivu, Kimmo. Corban soveltaminen verkonhallintajärjestelmässä. Diplomityö, TTKK, Ohjelmistotekniikka 1998.

[Lig03] TeliaSonera Leads Finnish Broadband, Light Reading, Inc., 2003.
Saatavissa:
http://www.lightreading.com/document.asp?doc_id=41106&site=lightreading
Viitattu: 30.7.2004

[Mik01] Mikkonen, Antti. Scalable Vector Graphics, Tietojenkäsittelytieteen laitos, Helsingin Yliopisto, 2001.
Saatavissa: http://www.cs.helsinki.fi/u/ajmikkon/svg/svg_pruju.pdf
Viitattu: 30.7.2004

[Mpl94] FRF.5, Frame Relay/ATM PVC Network Interworking Implementation(FRF.5), Multi Protocol Label Switching Forum, 1994.

[Pas96] Passmore D. ja Freeman J. The Virtual LAN Technology Report, Decisys, Inc., 1996.

[Pry95] de Prycker, M. ja Horwood, E. Asynchronous Transfer Mode: Solution for Broadband ISDN, Prentice Hall, 1995.

[Qui00] Quigley A. Large Scale 3D Clustering and Abstraction, Department of Computer Science and Software Engineering, University of Newcastle, Australia, 2000.

[Qui03] Quigley A. Large Scale Force Directed Layout, University of Sydney, 2003.
Saatavissa: http://www.cs.usyd.edu.au/~aquigley/3dfade/
Viitattu: 30.7.2004

[Rad92] Rade, Lennart. Beta Mathematics Handbook: Concepts, Theorems, Methods, Algorithms, Formulas, Graphs, Tables. 1992.

[Raj02] Rajiv, J. Building the New Broadband Access Network, CommVerge Solution, 2002.
Saatavissa: http://cnscenter.future.co.kr/resource/rsc-enter/presentation/APRICOT02/C0102.pdf
Viitattu: 30.7.2004

[RFC91] RFC 1213, Management Information Base for Network Management of TCP/IP-based internets:MIB-II, 1991.

[Riu03] Riukulehto, T. Verkonkutojan käsikirja, Point it Ky, 2003.

[Rob95] Robert A. Calculus, A Complete Course, Addison-Wesley Ltd, 1995.

[Sch04] Schaller, C. SVG and its Path into the Linux Desktop, OSNews.com, 2004.
Saatavissa: http://www.osnews.com/story.php?news_id=6460
Viitattu: 30.7.2004

[Sil04] Silvennoinnen, Risto. Matemaattinen optimointiteoria 2, Tampere university of Technology, Department of Mathematics, 2004.
Saatavissa: http://matriisi.ee.tut.fi/courses/73125/Luento7.pdf
Viitattu: 30.7.2004

[Spr01] Frame Relay, Product Details, Sprint, 2001.
Saatavissa:
http://www.sprintbiz.com/bizpark/products_services/frame_relay/details.html
Viitattu: 30.7.2004

[Sta93] Stallings, S. SNMP, SNMPv2, and CMIP, Addison-Wesley, ISBN 0-201-63331-0, 1993.

[Tan03] Tanenbaum, A. Computer Networks. Prentice Hall PTR, 2003.

[Tun94] Tunkelang, D. A Practical Approach to Drawing Undirected Graphs, Technical Report CMU-CS-94-161, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994.

[Tun98] Tunkelang D. JIGGLE: Java Interactive General Graph Layout Environment, S.H.Whitesides (Eds.) Proc. 6th International Symposium on Graph Drawing (GD98), LNCS 1547, p. 413-422, Springer, 1998.

[Tun99] Tunkelang, D. A Numerical Optimization Approach to General Graph Drawing, Ph.D. Thesis, Carnegie Mellon University, 1999.

[Tur00] Turunen, J. ja Leppälahti, J. Verkonhallinta, Teknillinen korkeakoulu, 2000.
Saatavissa: http://keskus.hut.fi/opetus/s38118/s00/tyot/35/protokollat.shtml
Viitattu: 30.7.2004

[Tut60] Tutte, W. T. Convex representations of graphs, Proceedings of the London Mathematical Society, England, no. 10, 304-320, 1960.

[Vpn03] VPN Consortium, VPN Technologies: Definitions and Requirements, 2003.
Saatavissa: http://www.vpnc.org/vpn-technologies.html
Viitattu: 30.7.2004

[W3c04] About SVG, World Wide Web Consortium , 2004.
Saatavissa: http://www.w3.org/Graphics/SVG/About
Viitattu: 30.7.2004

[Web98] Webber, R. Finding the Best Viewpoint for Three-Dimensional Graph Drawings. PhD thesis, University of Newcastle, Australia, 1998.

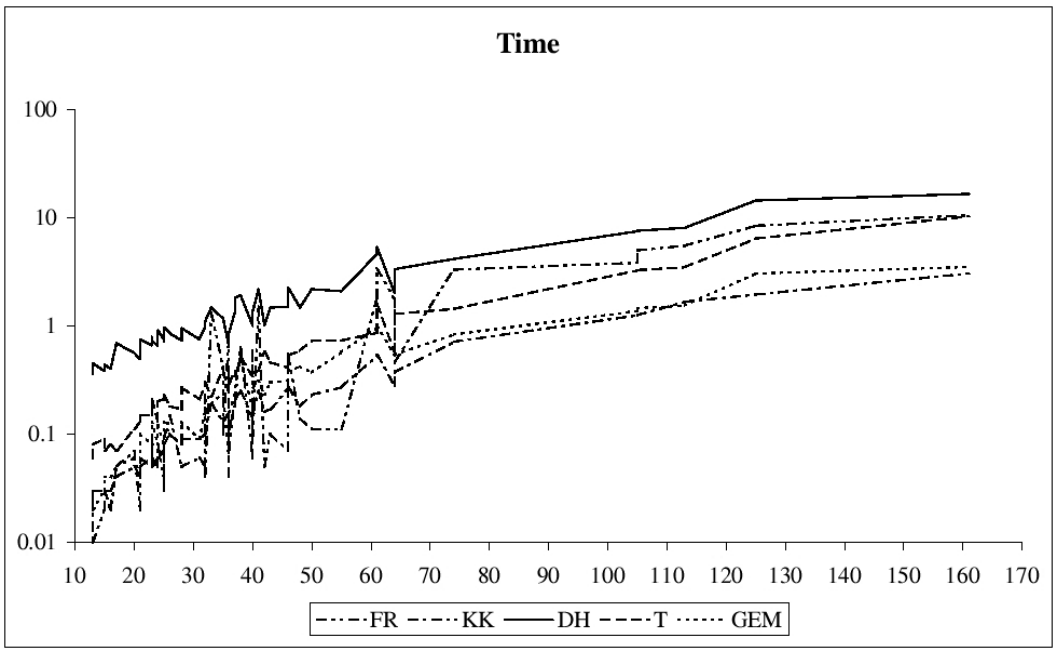
[Xil04] Digital Subscriber Line Access Multiplexer (DSLAM), Xilinx, Inc., 2004.
Saatavissa:
http://www.xilinx.com/esp/networks_telecom/optical/net_equip/dslam.htm
Viitattu: 30.7.2004

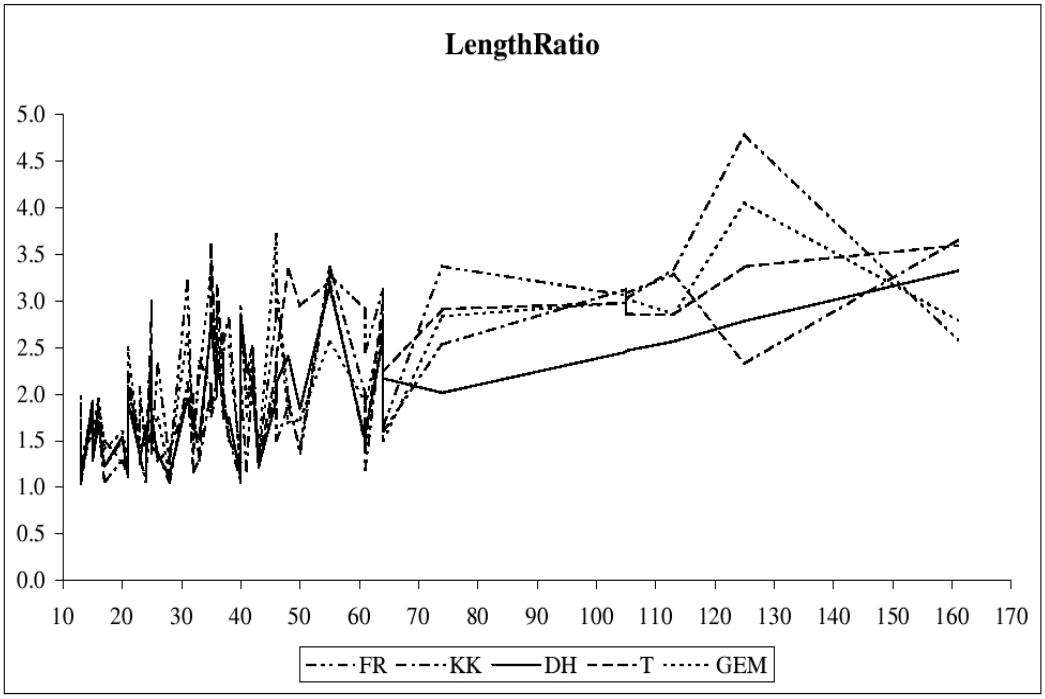
[Xil04a] ATM and Xilinx Solutions, Xilinx, Inc., 2004.
Saatavissa:
http://www.xilinx.com/esp/networks_telecom/optical/collateral/atm.pdf
Viitattu: 30.7.2004

LIITTEET

LIITE I. Voimiin perustuvien algoritmien vertailu.

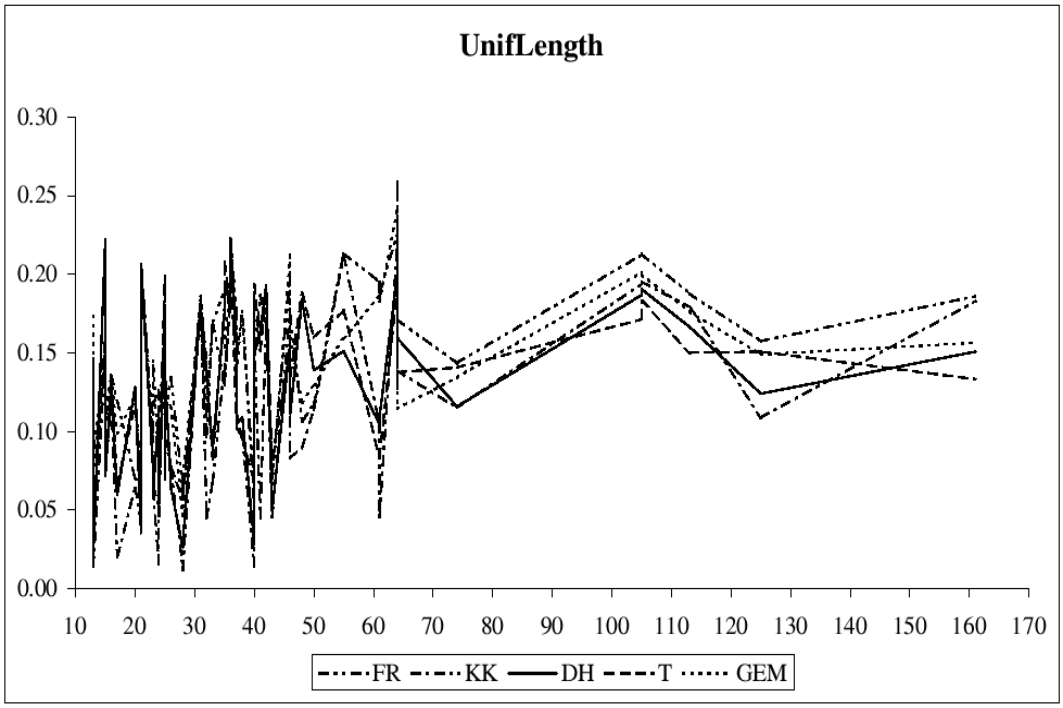
Materiaali perustuu Brandenburgin [Bra96] tutkimukseen.

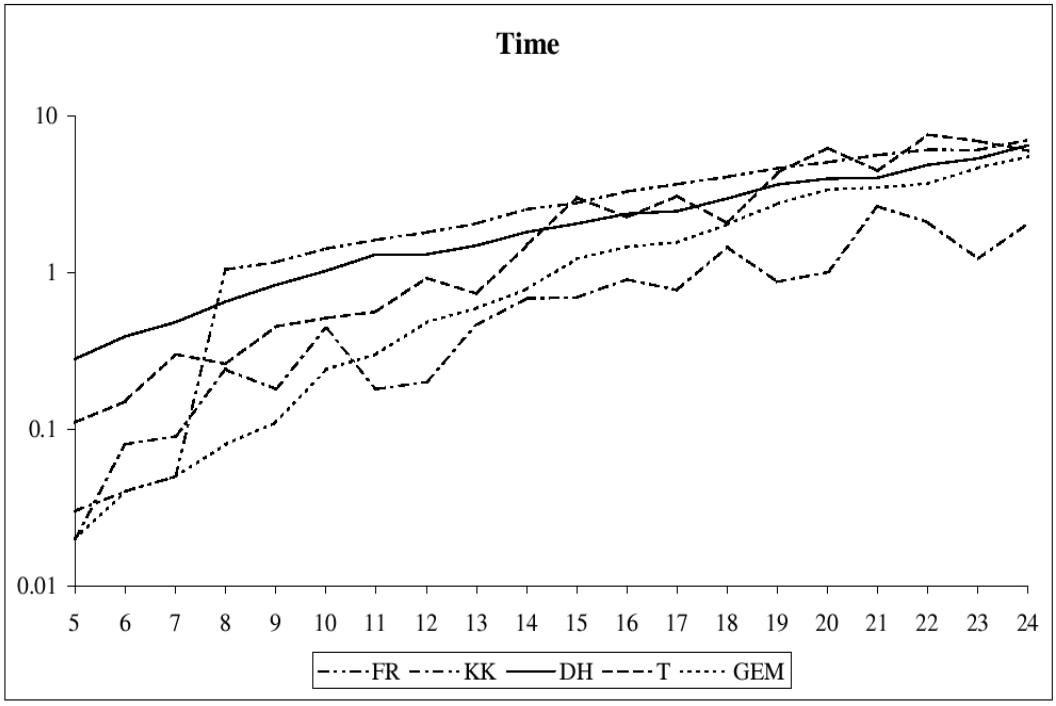




(jatkuu)

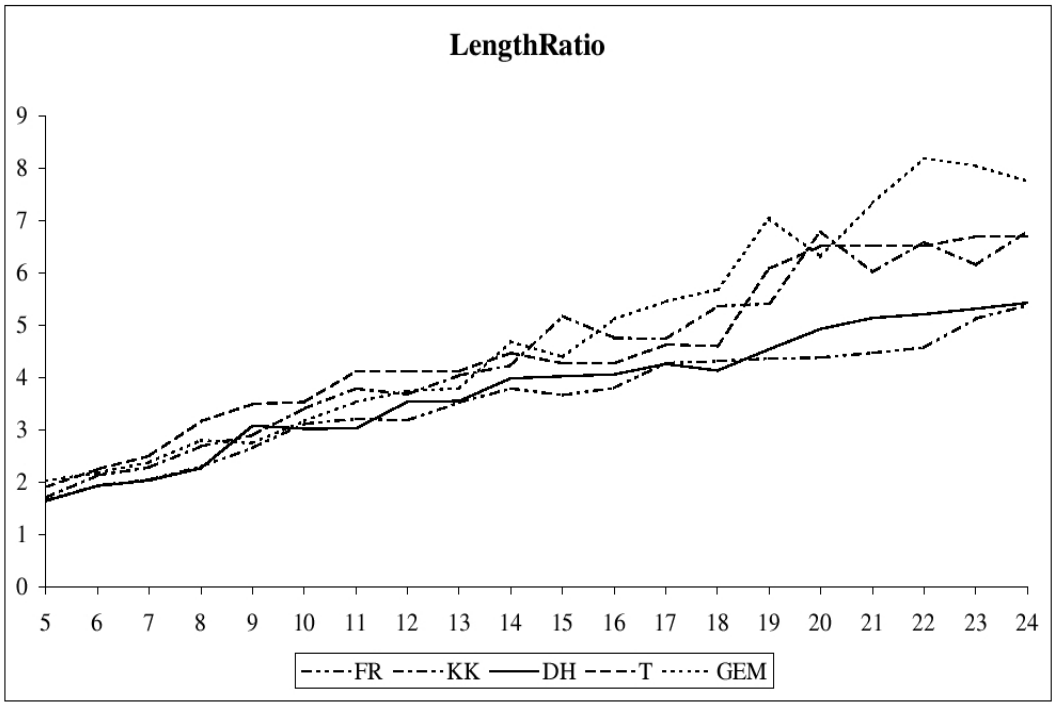
(jatkoa)

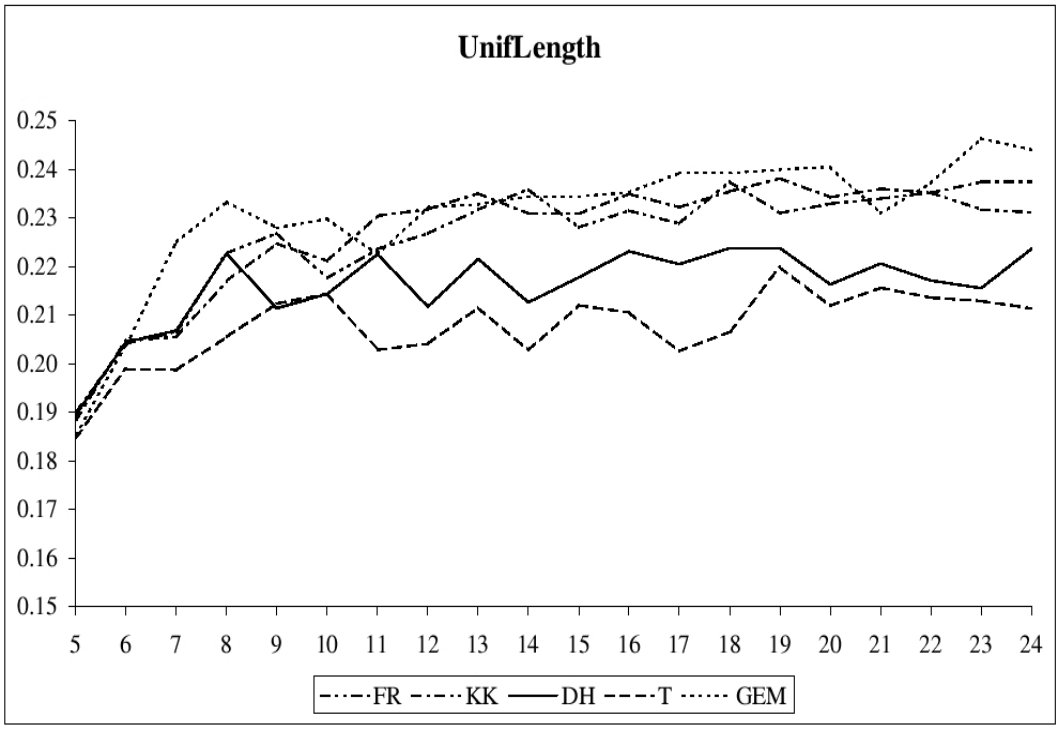




(jatkuu)

(jatkoa)


LengthRatio
9
8
7
6
5
4
3
2
1
0
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
FR KK DH T GEM


UnifLength
0.25
0.24
0.23
0.22
0.21
0.20
0.19
0.18
0.17
0.16
0.15
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
FR KK DH T GEM

LIITE II. Gem-, KK-, ja FR-algoritmien suoritusaikojen vertailu**.**

Materiaali perustuu Frickin [Fri95] tutkimukseen.

Table 1. Test suite: Graphs 1–8 are tiny, 9–14 small, 15–22 medium, 23–26 large, 27–30 huge.

#	graph Name	$\|V\|$	$\|E\|$	Density	GEM time[s]	GEM χ	GEM $\tilde{E}$	GEM $D_{\min}$	GEM $D_{\max}$	KK time[s]	KK χ	KK $\tilde{E}$	KK $D_{\min}$	KK $D_{\max}$	FR time[s]	FR χ	FR $\tilde{E}$	FR $D_{\min}$	FR $D_{\max}$
1	Binary Tree	15	14	sparse	0.13	0	0.145	0.851	6.117	0.75	0	0.030	0.952	5.550	0.54	0	0.216	0.771	6.353
2	Path	16	15	sparse	0.19	0	0.082	0.821	10.961	0.97	0	0.048	0.903	13.057	0.95	0	0.130	0.735	12.148
3	Cycle	16	16	normal	0.19	0	0.018	0.955	5.361	0.99	0	0.037	0.949	5.483	0.70	0	0.003	0.995	5.171
4	Square Grid	16	24	normal	0.17	0	0.048	0.943	4.300	1.15	0	0.019	0.962	4.231	0.59	0	0.063	0.935	4.184
5	Wheel	13	24	normal	0.12	0	0.319	0.643	2.681	0.62	4	0.253	0.522	2.647	0.10	0	0.318	0.675	2.643
6	Hypercube 4D	16	32	normal	0.15	24	0.413	0.337	2.004	1.08	22	0.181	0.323	2.738	0.06	24	0.064	0.478	2.685
7	$K_{8,8}$	16	64	dense	0.23	596	0.257	0.298	1.911	1.57	686	0.236	0.214	1.670	1.75	624	0.691	0.782	4.235
8	K_{12}	12	66	dense	0.15	406	0.371	0.431	1.619	0.73	402	0.371	0.427	1.577	1.05	407	0.371	0.443	1.608
9	Star	24	23	sparse	0.28	0	0.132	0.326	2.213	2.88	0	0.187	0.139	2.346	1.83	0	0.185	0.412	2.263
10	Binary Tree	31	30	sparse	0.52	0	0.205	0.611	8.677	6.44	1	0.131	0.510	7.947	2.92	0	0.315	0.535	9.554
11	Dodecahedron	20	30	normal	0.22	6	0.149	0.403	3.603	1.84	10	0.130	0.452	3.539	0.69	10	0.137	0.567	3.511
12	Hypercube 5D	32	80	normal	0.42	177	0.049	0.221	3.189	7.42	168	0.119	0.237	3.215	1.43	177	0.062	0.000	3.150
13	Triangular Grid	28	63	normal	0.37	0	0.099	0.809	6.109	5.14	0	0.040	0.902	6.079	1.31	0	0.144	0.659	6.419
14	K_{24}	24	276	dense	0.59	8129	0.417	0.305	1.761	6.15	8347	0.416	0.266	1.75	4.47	7962	0.418	0.344	1.765
15	Path	48	47	sparse	1.63	0	0.054	0.803	17.83	24.48	3	0.103	0.135	13.276	6.51	2	0.180	0.423	14.964
16	Binary Tree	63	62	sparse	1.90	0	0.261	0.437	12.00	49.23	1	0.115	0.286	9.960	10.58	1	0.418	0.302	14.142
17	Fibonacci Tree	54	53	sparse	1.77	0	0.267	0.543	13.84	31.90	3	0.131	0.283	10.657	8.12	1	0.407	0.352	15.365
18	Cycle	48	48	normal	1.73	0	0.034	0.911	15.28	22.54	1	0.116	0.364	14.408	6.49	0	0.107	0.666	16.667
19	Square Grid	49	84	normal	1.11	0	0.095	0.832	8.257	26.88	0	0.056	0.886	8.705	6.21	0	0.126	0.808	8.326
20	Torus	64	128	normal	1.98	46	0.311	0.473	7.894	63.67	54	0.326	0.220	7.880	11.7	44	0.454	0.481	8.341
21	Triangular Grid	55	135	normal	1.55	0	0.115	0.718	9.227	35.32	3	0.094	0.552	9.645	9.18	0	0.173	0.547	9.654
22	Hypercube 6D	64	192	dense	1.20	1004	0.062	0.211	3.871	55.86	977	0.201	0.119	3.777	12.27	1000	0.069	0.216	3.806
23	Binary Tree	127	126	sparse	9.19	0	0.311	0.298	16.125	1.26	2044	0.497	0.000	2.324	41.28	2	0.521	0.228	20.311
24	Hexagonal Grid	96	132	normal	4.65	0	0.149	0.764	11.784	192.75	3	0.274	0.288	12.585	24.26	0	0.189	0.715	11.830
25	Triangular Grid	120	315	normal	5.99	0	0.124	0.618	14.314	388.00	0	0.156	0.588	16.028	42.17	0	0.199	0.426	15.072
26	Path	128	127	sparse	9.54	0	0.053	0.704	30.189	432.73	21	0.212	0.034	20.030	44.14	5	0.265	0.148	26.017
27	Binary Tree	255	254	sparse	45.04	0	0.367	0.257	21.62	> 1000					186.21	37	0.635	0.000	20.751
28	Path	256	255	sparse	37.93	2	0.086	0.572	41.61	> 1000					181.65	32	0.519	0.000	27.785
29	Triangular Grid	210	570	normal	30.88	0	0.127	0.563	19.57	2110.06	435	0.266	0.045	19.162	131.63	4	0.219	0.196	20.465
30	Square Grid	256	480	normal	71.78	0	0.118	0.701	20.79	> 1000					197.41	89	0.250	0.000	18.421

LIITE III. CostSpring-, Gem-, ja FR-algoritmien suoritusaikojen vertailu**.**

Materiaali perustuu Behzadin [Beh99] tutkimukseen.

		CostSpring		GEM		FR	
#	*Name*	*iter.*	*time[s]*	*iter.*	*time[s]*	*iter.*	*time[s]*
1	Binary Tree 15	46.2	0.017	45	0.022	669.6	0.115
2	Path 16	53.6	0.018	48	0.022	1000	0.148
3	Cycle 16	45.6	0.015	48	0.022	561.8	0.083
4	Star 24	59.6	0.035	72	0.061	1000	0.271
5	Binary Tree 31	93.8	0.116	93	0.137	993	0.434
6	Path 48	123.2	0.239	144	0.354	1000	0.968
7	Cycle 48	120.8	0.255	144	0.375	1000	0.972
8	Binary Tree 63	189.2	0.644	189	0.734	1000	1.616
9	Binary Tree 127	392	4.517	381	5.307	1000	6.289
10	Path 128	224	2.723	384	5.548	1000	7.244
11	Binary Tree 180	552.6	12.71	540	14.29	1000	13.13
12	Cycle 220	345	12.312	660	26.51	1000	24.30
13	Path 256	441.6	21.88	768	43.27	1000	27.00
14	Binary Tree 255	779.2	42.09	765	44.41	1000	27.29
15	Wheel	41	0.015	39	0.019	239	0.034
16	4×4 Square Grid	33.2	0.013	48	0.024	320	0.052
17	Hypercube4D	36.8	0.015	48	0.025	212.6	0.036
18	Dodecahedron	32.2	0.017	60	0.04	349	0.074
19	Triangular Grid 28	42.6	0.042	84	0.095	497.2	0.2
20	Hypercube5D	58.2	0.075	96	0.137	308.4	0.155
21	7×7 Square Grid	64.6	0.162	147	0.395	714.2	0.726
22	Triangular Grid 55	75.8	0.213	165	0.542	1000	1.416
23	Hexagonal Grid 96	96.2	0.696	288	2.802	1000	3.767
24	5×20 Square Grid	500	3.854	300	2.743	1000	4.036
25	10×12 Square Grid	135	1.690	360	4.456	1000	6.305
26	Triangular Grid 120	116.6	1.528	360	4.47	1000	6.11
27	Triangular Grid 210	325	14.16	630	23.14	1000	18.69
28	Hexagonal Grid 216	347	14.919	648	24.70	1000	19.94
29	16×16 Square Grid	159.8	10.38	768	43.67	1000	28.12
30	K12	38.4	0.014	36	0.018	813.6	0.137
31	K24	43.6	0.039	72	0.106	1000	0.573
32	K50	54	0.191	150	0.754	1000	2.532
33	Hypercube6D	106.8	0.451	192	0.887	1000	1.766
34	Hypercube8D	150.8	18.01	768	42.11	1000	28.95

Table 5.3: Running times and iteration counts for CostSpring, GEM, and FR.