Lappeenrannan teknillinen korkeakoulu
Tietotekniikan osasto / Tietoliikennetekniikka
1715 Tietotekniikan erikoistyöt

LOPPURAPORTTI

OpenTTCN protokollatesterin soveltaminen SNMP verkonhallintajärjestelmien testaamisessa internetverkoissa

Alkuraportti esitettiin 31.01.2000
Vastaanottaja: Professori Jouni Lampinen

Väliraportti esitettiin 12.06.2000
Vastaanottaja: Professori Jouni Lampinen

Lappeenrannassa 18.6.2000
Toni Suihko, 6242

TIIVISTELMÄ

Lappeenrannan teknillinen korkeakoulu
Tietotekniikan osasto / Tietoliikennetekniikka
Toni Suihko

OpenTTCN protokollatesterin soveltaminen SNMP verkonhallintajärjestelmien testaamisessa internetverkoissa

1715 Tietotekniikan erikoistyöt. Loppuraportti. 73 sivua, 36 kuvaa, 1 taulukko.

Paikka: 18.6.2000 Lappeenranta
Yritys: TKK/TML
Ohjaaja: Diplomi-insinööri Vesa-Matti Puro (OES)
Tarkastaja: Professori Jouni Lampinen
Hakusanat: ISO 9646(-3), X.292, SNMP, verkonhallinta, standardinmukaisuus-testaus, protokollatestaus, TTCN, ASN.1, Java, OpenTTCN

Erikoistyössä tutkitaan ISO 9646 metodologian soveltamista SNMP -pohjaisten verkonhallintajärjestelmien testaamisessa internetverkoissa. Testausalustana käytetään Open Environment Software Oy:n (OES) tuotetta (OpenTTCN), joka tulkitsee ISO 9646-3 (X.292) standardin mukaisia testiaineistoja. OES on luovuttanut tuotteen TML -laboratorion IPMAN -projektin käyttöön. Tutkimuksen tuloksena on yksinkertainen testausmalli, joka määrittelee testerin ja testattavan systeemin välisen arkkitehtuurin sekä testausstrategian ISO 9646-2 (X.291) metodologian mukaisesti. Lisäksi työssä toteutetaan sovitinohjelma (Gateway) testerin ja testattavan systeemin väliin sekä toteutetaan testerille muutamia ISO 9646-3 määrittelyn mukaisia testitapauksia, joilla pääasiassa todetaan testausmallin ja implementoidun Gateway:n toiminnan oikeellisuus. Gateway:n kehitysvaiheessa OpenTTCN protokollatesteriä käytetään myös testattavan systeemin mallintamiseen.

ABSTRACT

Lappeenranta University of Technology
Information Technology / Telecommunications
Toni Suihko

Applying of OpenTTCN protocoltester for testing of SNMP network management systems in internet networks.

1715 Information technology project. Final report. 73 pages, 36 figures, 1 table.

Place: 18.6.2000 Lappeenranta
Firm: HUT/TML
Instructor: Vesa-Matti Puro, M.S. (OES)
Inspector: Professor Jouni Lampinen
Keywords: ISO 9646(-3), X.292, SNMP, network management, conformance testing, protocol testing, TTCN, ASN.1, Java, OpenTTCN

The goal of this information technology project is to research how to apply ISO 9646 methodology for testing of SNMP based network management systems in internet networks. Testing platform that is used is product (OpenTTCN) of Open Environment Software Oy (OES). OpenTTCN interprets test suites that are following ISO 9646-3 (X.292) standard. OES has released OpenTTCN for free use by TML laboratory's IPMAN project. Result of the research is simple testing model that defines the architecture between tester and system under test and defines the testing strategy using ISO 9646-2 (X.291) methodology. In addition have to implement an adapter program (Gateway) between tester and system under test and few testcases that are following ISO 9646-3 definitions for tester. These testcases are mainly used to verify validy of testing model and functionality of Gateway. The system under test is also try on by OpenTTCN tester at the development phase of Gateway implementation in project.

SISÄLLYSLUETTELO

TIIVISTELMÄ ..... **I**

ABSTRACT ..... **II**

SISÄLLYSLUETTELO ..... **III**

LYHENNELUETTELO ..... **VI**

1 JOHDANTO **1**

2 ERIKOISTYÖN KUVAUS **2**

2.1 TAUSTAA 2

2.2 TAVOITTEET 3

3 ERIKOISTYÖSSÄ TARVITTAVAT TEKNOLOGIAT **4**

3.1 YLEISTÄ 4

3.1.1 Verkonhallintajärjestelmät *4*

3.1.2 Yleistä testauksesta *11*

3.1.3 Standardinmukaisuustestaus *13*

3.1.4 ISO 9646-3 (Tree and Tabular Combined Notation) *21*

3.1.5 Abstract Syntax Notation One *25*

3.1.6 Basic Encoding Rules *26*

3.1.7 Common Object Request Broker Architecture *32*

3.2 OPENTTCN PROTOKOLLATESTERI 32

3.2.1 Yleistä *32*

3.2.2 Toimintaidea *33*

3.2.3 Ominaisuudet / arkkitehtuuri *33*

3.2.4 Toiminta / käyttö *36*

4 ERIKOISTYÖSSÄ TÄLLÄ HETKELLÄ TOTEUTETUT ASIAT **37**

4.1 YLEINEN TESTAUSARKKITEHTUURI 37

4.1.1 Yleistä *37*

4.1.2 Gateway *38*

4.1.3 Testattavat elementit *46*

4.2 ISO 9646-2 STANDARDIN KÄSITTEET MÄÄRITELTÄESSÄ TESTAUSARKKITEHTUURIA 50
4.3 TESTITAPAUKSET 54
4.3.1 Yleistä 54
4.3.2 SNMP_GW_TEST/VALID_SNMP_GW_MESSAGES - ryhmän testitapaukset 55
4.3.3 SNMP_GW_TEST/VALID_SNMP_GW_MESSAGE_INTERACTION - ryhmän testitapaukset 58
4.3.4 SNMP_GW_TEST/VALID_SNMP_GW_MESSAGE_DATATYPES 61
4.3.5 SNMP_GW_TEST/INVALID_SNMP_GW_MESSAGE_DATATYPES 61
5 ISO 9646:N JA OPENTTCN TESTERIN SOVELTUMINEN VERKONHALLINTAJÄRJESTELMIEN TESTAAMISEEN 64
5.1 YLEISTÄ 64
5.2 ISO 9646 STANDARDI VERKONHALLINTAJÄRJESTELMIEN TESTAUKSESSA 64
5.3 OPENTTCN TESTERI VERKONHALLINTAJÄRJESTELMIEN TESTAUKSESSA 65
5.4 JOHTOPÄÄTÖKSET 66
5.4.1 Yleistä 66
5.4.2 Tutkimustulokset 67
5.4.3 Yhteenveto ja tulevaisuuden skenaariot 68
6 AIKATAULU 71
6.1 KÄYTETYT RESURSSIT 72
LÄHDEVIITTEET VIII
LIITE 1. GATEWAY OHJELMAN UML -KUVAUS; LUOKKADIAGRAMMI
LIITE 2. ESIMERKKI GATEWAY:N KONFIGUROINTITIEDOSTOSTA

KUVA 1: VERKONHALLINTAJÄRJESTELMÄN YLEINEN ARKKITEHTUURI /6/ 6
KUVA 2: VERKONHALLINTAJÄRJESTELMÄN SISÄLTÄMIEN LAITTEIDEN YLEINEN RAKENNE /5/....8
KUVA 3: SNMPV1 PDU -VIESTIEN RAKENNE /7/....10
KUVA 4: KÄSITTEELLINEN TESTAUSARKKITEHTUURI /13/ 17
KUVA 5: PAIKALLINEN TESTAUSMENETELMÄ /12, 13/ 19
KUVA 6: HAJAUTETTU TESTAUSMENETELMÄ /12, 13/....19
KUVA 7: KOORDINOITU TESTAUSMENETELMÄ /12, 13/....20
KUVA 8: ETÄTESTAUSMENETELMÄ /12, 13/....20
KUVA 9: TTCN:N KÄYTTÄMÄ YLEINEN RINNAKKAINEN TESTIARKKITEHTUURI /10/ 24
KUVA 10: ABSTRAKTIN SYNTAKSIN (ASN.1) JA SIIRTOSYNTAKSIN KÄYTTÖ /6/ 25
KUVA 11: BER -KOODAUSFORMAATTI: IDENTIFIER KENTÄN RAKENNE /7/....28
KUVA 12: BER -KOODAUSFORMAATTI: LENGTH KENTÄN RAKENNE /7/ 29
KUVA 13: OPENTTCN TESTAUSJÄRJESTELMÄ /17/....34
KUVA 14: YLEINEN TESTAUSARKKITEHTUURI VERKONHALLINTAJÄRJESTELMIEN TESTAAMISEKSI 38
KUVA 15: GATEWAY TOTEUTUKSIEN SIJOITTUMINEN KÄYTETTÄESSÄ RINNAKKAISTA TTCN:TA 42
KUVA 16: GATEWAY:N ARKKITEHTUURI....43
KUVA 17: VERKKOLAITTEEN YLEINEN TESTAUSARKKITEHTUURI 47
KUVA 18: VERKONHALLINTAPALVELIMEN YLEINEN TESTAUSARKKITEHTUURI 48
KUVA 19: GATEWAY:N YLEINEN TESTAUSARKKITEHTUURI 50
KUVA 20: VERKKOLAITTEEN ETÄTESTAUSMENETELMÄ 51
KUVA 21: VERKONHALLINTAPALVELIMEN HAJAUTETTU TESTAUSMENETELMÄ 53
KUVA 22: GATEWAY:N PAIKALLINEN TESTAUSMENETELMÄ....54
KUVA 23: TC_VSGM_001 TESTIN SEQUENCE -DIAGRAMMI....56
KUVA 24: TC_VSGM_002 TESTIN SEQUENCE –DIAGRAMMI 56
KUVA 25: TC_VSGM_003 TESTIN SEQUENCE -DIAGRAMMI....57
KUVA 26: TC_VSGM_004 TESTIN SEQUENCE -DIAGRAMMI....57
KUVA 27: TC_VSGM_005 TESTIN SEQUENCE -DIAGRAMMI 58
KUVA 28: TC_VSGMI_001 TESTIN SEQUENCE -DIAGRAMMI 59
KUVA 29: TC_VSGMI_002 TESTIN SEQUENCE -DIAGRAMMI....60
KUVA 30: TC_VSGMD_001 TESTIN SEQUENCE -DIAGRAMMI 61
KUVA 31: TC_ISGMD_001 TESTIN SEQUENCE -DIAGRAMMI....62
KUVA 32: TC_ISGMD_002 TESTIN SEQUENCE -DIAGRAMMI....62
KUVA 33: TC_ISGMD_003 TESTIN SEQUENCE -DIAGRAMMI....63
KUVA 34: TC_ISGMD_004 TESTIN SEQUENCE -DIAGRAMMI....63
KUVA 35: TC_ISGMD_005 TESTIN SEQUENCE -DIAGRAMMI....63
KUVA 36: ERIKOISTYÖN ALKURAPORTISSA ESITETTY AIKATAULU 72

TAULUKKO 1: SNMP:N KÄYTTÄMÄT TYYPIT JA NÄIDEN TUNNISTEET /5/ 31

LYHENNELUETTELO

ASN.1	Abstract Syntax Notation One
ASP	Abstract Service Primitive
ATM	Abstract Test Method
ATS	Abstract Test Suite
BER	Basic Encoding Rules
BIT	Basic Interconnection Tests
BNF	Backus Naur Form
CCITT	International Consultative Committee on Telephony and Telegraphy
CM	Coordination Message
CORBA	The Common Object Request Broker Architecture
CP	Coordination Point
CRT	Conformance Resolution Tests
CT	Conformance Tests
ETS	Executable Test Suite
ETSI	European Telecommunications Standards Institute
FRT	Functional Range Tests
HUT	Helsinki University Of Technology
ICS	Implementation Conformance Statement
IEC	International Electrotechnical Commission
IEEE	Institution of Electrical and Electronics Engineers
IPMAN	IP network Management
IP	Internet Protocol
ISO	International Organization for Standardization
ITU-T	International Telecommunications Union – Telecommunication Standardization Sector
IXIT	Implementation Extra Information for Testing
LAN	Local Area Network
LT	Lower Tester
LTCF	Lower Tester Control Function

MAC	Medium Access Control
MIB	Management Information Base
MOT	Means Of Testing
MPyT	Multi-Party Testing
MTC	Main Test Component
NMS	Network Management Station
OES	Open Environment Software Oy
OMG	Object Management Group
ORB	Object Request Broker
OSI	Open System Interconnection
PC	Personal Computer
PCO	Point of Control and Observation
PCTR	Protocol Conformance Test Report
PICS	Protocol Implementation Conformance Statement
PIXIT	Protocol IXIT
PTC	Parallel Test Component
SCS	System Conformance Statement
SCTR	System Conformance Test Report
SNMP	Simple Network Management Protocol
SPyT	Single-Party Testing
SS7	Signalling System no. 7
SUT	System Under Test
TCP	Transmission Control Protocol
TEKES	Teknologian kehittämiskeskus
TTCN	The Tree and Tabular Combined Notation
TML	Tietoliikenneohjelmistojen ja Multimedian Laboratorio
TOTI	Tietoliikenne- ja OhjelmistoTekniikan Instituutti
TVL	Type - Value - Length
UDP	User Datagram Protocol
UML	Unified Modelling Language
UT	Upper Tester

1 JOHDANTO

Tässä raportissa käydään läpi erikoistyön taustaa, asetettuja tavoitteita sekä esitellään työn lopullinen edistyminen suhteessa alkuraportissa esitettyihin arvioihin. Lisäksi raportissa esitellään tarkemmin työssä toteutuneet tulokset sekä arvio käytetyistä resursseista. Raportissa esitetään myös työhön liittyvät johtopäätökset ja mahdolliset tulevaisuuden skenaariot koskien erikoistyön aihetta.

Teknillisellä korkeakoululla (TKK) on alaisuudessaan useita tutkimusinstituutteja, jotka ovat TKK:n osastorajoja ylittäviä tutkimuksen yhteistyöyksiköitä. Yksi näistä yksiköistä on tietoliikenne- ja ohjelmistotekniikan instituutti (TOTI). TOTI on hanke, joka sitoo yhteen useita laboratorioita TKK:n sähkö- ja tietoliikennetekniikan sekä tietotekniikan osastoilta. Tietoliikenneohjelmistojen ja multimedian laboratorio (TML) on yksi tietotekniikan osaston ja tätä kautta TOTI:n alaisuudessa toimivista laboratorioista. /1, 2, 3/

Tämä erikoistyö tehdään IP network management (IPMAN) projektille, joka on yksi TML:n projekteista. Projektin rahoittajina toimivat TEKES, Nokia sekä Open Environment Software Oy (OES). TKK käynnisti IPMAN -projektin vuonna 1999. Tavoitteena projektissa on tutkia ja kehittää verkonhallintamallia suurille IP -verkoille. Mallia laatiessa on tarkoitus ottaa huomioon tällä hetkellä ja tulevaisuudessa tapahtuva räjähdysmäinen IP -verkkojen kasvu, joka osaltaan aiheutuu lisääntyneistä käyttäjämääristä, lisääntyneistä verkkolaitteista sekä uudentyyppisistä multimediasovellutuksista, jotka kuormittavat verkkoa suuresti, esimerkiksi IP -video ja ääni, IP -puhelut sekä videoneuvottelut jne. Tällä hetkellä projektissa keskitytään tutkimaan erityisesti erilaisten palveluiden sisällönhallintaa sekä käyttäjäkohtaista palveluprofilointia. /2, 4/

Erikoistyö ei liity suoraan IPMAN -projektin tämän hetkiseen tutkimusalueeseen (sisällönhallinta), vaan se on eriytetty omaksi kokonaisuudekseen, jonka laajuus on neljä henkilötyökuukautta. Työn tavoitteena on tutkia kuinka ISO 9646 standardin metodologia soveltuu Simple Network Management Protocol (SNMP) kehykseen

pohjautuvien verkonhallintajärjestelmien testaamiseen TCP/IP -verkoissa. Testausalustana käytetään OES:n projektille tarjoamaa OpenTTCN protokollatesteriä.

2 ERIKOISTYÖN KUVAUS

2.1 Taustaa

Verkkojen kokojen kasvaminen, erilaisten verkkotekniikoiden monimuotoisuuden lisääntyminen ja tästä johtuva verkkohierarkioiden muuttuminen monimutkaisemmiksi on asettanut suuren haasteen verkonhallintajärjestelmille. Lisäksi eri valmistajien toteuttamien erilaisten verkkolaitteiden markkinoille tulo on omalta osaltaan lisännyt verkonhallinnan problematiikkaa. Kun samalla tiedostetaan verkkojen korvaamaton merkitys yrityksille ja yhteisöille tänä päivänä, on tämän problematiikan selvittämiseen ja ratkaisemiseen kiinnitetty suurenevissa määrin huomiota. /4, 6/

Jo kauan sitten oli selvää ettei verkonhallintaa voida hoitaa pelkästään henkilöresurssein, vaan tarvitaan yksinkertaisia ja tehokkaita automatisoituja järjestelmiä, jotka ovat tuotavissa markkinoille mahdollisimman lyhyellä aikavälillä. Ongelmana verkonhallinnassa oli kuitenkin vallalla oleva kirjava käytäntö. Jokaisella valmistajalla oli omat järjestelmänsä, joilla pystyi hallinnoimaan vain kyseisen valmistajan omia tuotteita. Tämä johti ajatukseen yhdestä, kaikille laitteille yhteisestä, yksinkertaisesta ja tehokkaasta protokollasta, jota verkonhallintajärjestelmät voivat käyttää erilaisten verkkojen ja verkkolaitteiden kanssa neuvotellessaan mm. niiden tilasta ja konfigurointiasetuksista. /6/

SNMP eli yksinkertainen verkonhallintaprotokolla on standardisoinut yksinkertaisen tavan verkonhallintajärjestelmille neuvotella erilaisten verkkojen ja verkkolaitteiden kanssa sekä päinvastoin. Tämä on puolestaan vaikuttanut positiivisesti useiden kaupallisten verkonhallintajärjestelmien markkinoille tuloon, varsin nopeallakin tahdilla. SNMP -protokollakehyksen käyttäminen on antanut ohjelmistotaloille varsin suoraviivaisen tavan verkonhallintaohjelmistojen kehittämiseen standardeja noudattamalla. /5/

Tämä johtaa kysymykseen, kuinka nämä eri valmistajien tuottamat erilaiset järjestelmät/ohjelmistot olisivat helposti ja nopeasti testattavissa yhdenmukaisella tavalla niin loogisen toimivuuden kuin suorituskyvynkin kannalta.

2.2 Tavoitteet

Erikoistyössä pyritään löytämään vastaus kysymykseen kuinka eri valmistajien erilaiset verkonhallintajärjestelmät (ohjelmistot) olisivat helposti ja nopeasti testattavissa yhdenmukaisella tavalla loogisen toimivuuden kannalta. Yhdenmukainen tapa tarkoittaa mm. sitä, että testit täytyy pystyä kuvaamaan/suorittamaan erilaisille ympäristöille samanlaisella (abstraktilla) testausympäristöstä tai testin suoritusajankohdasta riippumattomalla tavalla. Lisäksi testien tulee olla toistettavia.

Erikoistyössä lähdetään etsimään vastausta esitettyyn kysymykseen tutkimalla ISO 9646 standardin soveltamista SNMP -pohjaisten verkonhallintajärjestelmien testaamisessa. Tämän tutkimuksen tuloksena saadaan yksinkertainen ISO 9646-2 metodologian mukainen testausmenetelmämalli ja testausstrategia.

Testausalustana käytetään OES:in IPMAN -projektille käyttöön tarjoamaa OpenTTCN protokollatesteriä, joka noudattaa ISO 9646 standardin määrittelyjä. Erikoistyössä toteutetaan protokollatesterin ja testattavan systeemin välille sovitinohjelma (Gateway), jonka tehtävänä on sovittaa testerin tarjoama Common Object Request Broker Architecture (CORBA) testausrajapinta ja verkonhallintajärjestelmien SNMP/UDP -protokollapino toimimaan keskenään. /9/

Testerille laaditaan pienilukuinen joukko esimerkkitestitapauksia, joilla voidaan testata ideoidun testausmallin ja implementoidun Gateway:n toimivuutta. Testitapauksien kuvaaminen tapahtuu käyttämällä TTCN -kuvauskieltä, joka on määritelty ISO 9646-3 standardissa sekä X.292 suosituksessa.

Tutkittaessa SNMP -verkonhallintajärjestelmien testaamista ISO 9646 standardin keinoin, kehitystyön aikana myös testattavaa systeemiä/ohjelmistoa mallinnetaan OpenTTCN testerillä. Mikäli työn suorituksen aikana löydetään esimerkkitapaukseksi

jokin sopiva verkonhallintaohjelmisto, voidaan saavutettuja tuloksia (Gateway implementaatio, testitapaukset) soveltaa tämän ohjelmiston testaamiseen. Jos aikataulu sallii, voidaan tätä ohjelmistoa silmälläpitäen laatia muutamia spesifistisempiä testitapauksia.

3 ERIKOISTYÖSSÄ TARVITTAVAT TEKNOLOGIAT

3.1 Yleistä

Ensimmäisessä vaiheessa työtä aloittaessa perehdyttiin työn suorittamisessa tarvittaviin yleisiin teknologioihin, standardeihin, suosituksiin, työkaluihin ja ohjelmiin kuten SNMP, ISO 9646, X.290-X.292, TTCN, ASN.1, BER, CORBA, Java, Javadoc, OpenTTCN jne.

Osin perustietämyksen hankkimisen aikana ja välittömästi tämän jälkeen aloitettiin suunnittelemaan yksinkertaista testausmallia, josta käy ilmi testattavat kohteet yleisellä tasolla, näiden testausstrategia jne. Samaan aikaan suunniteltiin testauskonfiguraatiota, joka määrittelee testerin, Gateway:den ja testattavan systeemin välisen arkkitehtuurin. Arkkitehtuurin suunnittelu tapahtui osin yhtäaikaa Gateway implementaation suunnittelun kanssa, koska näillä on keskinäisiä riippuvuuksia ja jonkin ominaisuuden sisällyttäminen arkkitehtuurimalliin vaikuttaa Gateway:n sisäiseen toteutukseen ja päinvastoin.

3.1.1 Verkonhallintajärjestelmät

Verkonhallintajärjestelmä kokonaisuutena on joukko työkaluja (ohjelmia, laitteita), joilla voidaan monitoroida ja hallita verkkoa sekä verkossa olevia laitteita. Verkonhallintajärjestelmän tulee pääpiirteissään noudattaa seuraavia periaatteita: /5, 6/

- Verkkoa täytyy pystyä hallitsemaan muutaman (mieluummin yhden) hallintarajapinnan kautta, käyttämällä suppeaa ja tehokasta käskyjoukkoa.
- Verkonhallintaa varten saa ottaa käyttöön vain minimaalisen määrän ylimääräistä laitteistoa. Suurin osa verkonhallinnan vaatimista ohjelmistoista tulee

voida sisällyttää jo olemassa oleviin verkkolaitteisiin. Jokainen laite, joka voi sisällyttää pienen määrän ohjelmistoa, voi osallistua verkonhallintajärjestelmän toimintaan.

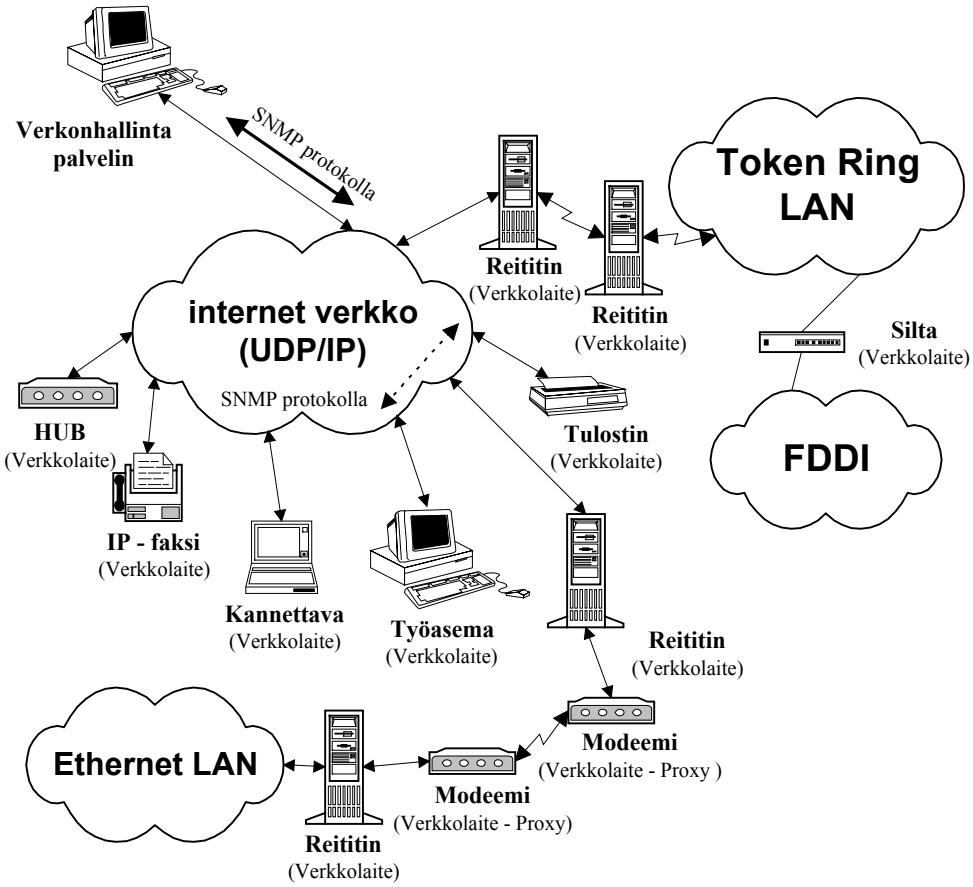
- Verkkolaitteet sisältävät informaatiota niistä itsestään (paitsi verkonhallinta-palvelin, joka sisältää myös muiden verkossa olevien verkkolaitteiden tietoja). Tällaisia tietoja voisivat esimerkiksi olla laitteen konfigurointitiedot mm. PC:n sisältämän verkkokortin MAC -osoite, verkkokortin nopeus jne.
- Verkonhallintajärjestelmä on suunniteltu näkemään koko verkko yhtenäisenä arkkitehtuurina siten, että jokaisella verkkosolmulla on oma verkko-osoite. Verkon aktiiviset elementit tarjoavat säännöllisesti verkonhallintajärjestelmälle tietoa omasta tilastaan.
- Verkonhallintapalvelin keskustelee kaikkien verkkolaitteiden kanssa samalla tavalla käyttäen Simple Network Management (SNMP) protokollaa.

IP -verkoissa on lukemattomia erilaisia laitteita, joita pitää pystyä valvomaan verkon-hallintajärjestelmillä. Laitteet voivat olla kytkettyjä suoraan organisaation/yhteisön internetverkkoon tai erilaisten lähiverkkoratkaisujen kautta. IP -verkko ja lähiverkot ovat kytkettyjä toisiinsa erilaisilla verkkolaitteilla kuten silloilla ja reitittimillä. Verkon ja laitteiden valvominen on mahdollista verkonhallintajärjestelmää käyttämällä. Tällöin järjestelmä pystyy keräämään tietoa verkon tilasta, verkossa olevien laitteiden tilasta sekä hallinnoimaan verkkoa ja siellä olevia laitteita.

Nämä toiminnot mahdollistavat sen, että verkonhallintajärjestelmän toiminta näkyy käyttäjälle lisääntyvänä verkon ja verkkolaitteiden toimintavarmuutena ja näin ollen mahdollistaa verkon ja laitteiden tehokkaamman käytön.

Verkonhallintajärjestelmään kuuluvat seuraavat pääkomponentit: /5, 6, 7/

- Verkonhallintapalvelin (NMS)
- Verkkolaite
- Hallintatietokanta (MIB)
- Yksinkertainen verkkohallintaprotokolla (SNMP)



Kuva 1: Verkonhallintajärjestelmän yleinen arkkitehtuuri /6/

Yleensä verkonhallintapalvelin on tehtävään suunniteltu stand-alone tyyppinen laite, joka toimii rajapintana ihmisen ja verkonhallintajärjestelmän välillä. Verkonhallintapalvelin pitää sisällään joukon erilaisia hallintaohjelmistoja mm. datan analysointia, virheistä toipumista yms. varten. Perinteisessä keskitetyssä verkonhallintamallissa yhdellä verkkosolmulla (esim. työasema) on verkonhallintapalvelimen rooli ja muutamat muut verkkosolmut toimivat varapalvelimina. Muut verkkosolmut ovat tavallisia verkkolaitteina (hub, reititin, työasema, PC, tulostin, modeemi jne.) /6/

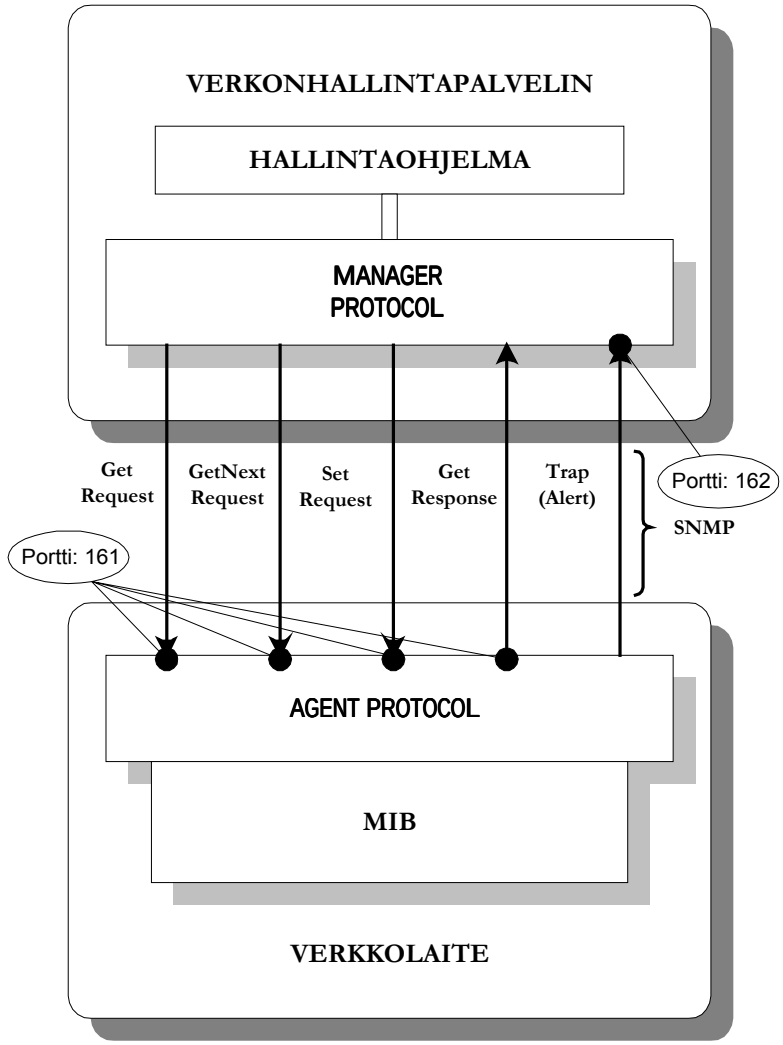
Kun verkkojen koot kasvavat, käyvät tällaiset keskitetyt ratkaisut tehottomiksi. Tällaisessa tilanteessa kohdistuu liian paljon kuormaa verkonhallintapalvelimelle sekä verkolle, koska jokaisen verkkolaitteen täytyy toimittaa tilastaan kertovat viestit keskitetylle verkonhallintapalvelimelle. Kyseessä olevassa tilanteessa hajautettu järjestelmä toimii parhaiten. Verkko on tällöin jaettu osiin ja kutakin osaa hallitsee oma verkonhal-

lintapalvelin (proxy) ja näitä proxy palvelimia edelleen ’ylemmän tason’ verkonhallintapalvelimet. /6/

Verkonhallintapalvelin sisältää hallintasovellusten lisäksi Manager -protokollan, jonka tehtävänä on toimia osana SNMP -protokollapinoa (*kts. Kuva 2*). Tämä tarkoittaa sitä, että kyseinen protokolla muodostaa SNMP -viestejä verkonhallintasovellusten ohjeiden mukaisesti ja lähettää nämä viestit edelleen verkossa olevalle/oleville verkkolaitteille. Manager -protokolla ottaa myös vastaan verkkolaitteen lähettämän vastauksen ja ohjaa sen hallintasovelluksen käyttöön.

Manager -protokolla voi muodostaa kolmenlaisia SNMP (versio 1) viestejä, SetRequest, GetRequest ja GetNextRequest sekä vastaanottaa kahdenlaisia viestejä, GetResponse ja Trap. Ottaessaan vastaan verkkolaitteiden lähettämiä Trap viestejä Manager -protokolla käyttää sidottua porttia, jonka numero on 162. Lähettäessään viestejä Manager voi käyttää mitä tahansa laitteen vapaata porttia. /5/

Toinen verkon avainelementti on tavalliset verkkolaitteet, jotka tukevat SNMP -protokollaa. Tällaiset laitteet kuten palvelimet, reitittimet ja hubit ovat varustettu Agent -protokollalla. Tämän protokollan tehtävänä on ottaa vastaan verkonhallintapalvelimen lähettämät SNMP -viestit ja näiden perusteella suorittaa hakuja/modifiointeja verkkolaitteen yhteydessä olevaan MIB -tietokantaan. Kun haut/modifioinnit on tehty, muodostaa Agent -protokolla vastauksen verkonhallintapalvelimen lähettämään pyyntöön ja lähettää sen takaisin palvelimelle. Ottaessaan vastaan palvelimen lähettämiä SNMP -viestejä, Agent -protokolla käyttää sidottua porttia 161. Lähettäessään vastausviestejä verkonhallintapalvelimelle Agent käyttää myöskin porttia 161. Lähettäessään Trap -viestejä verkonhallintapalvelimelle Agent voi käyttää mitä tahansa vapaata porttia. /5/



Kuva 2: Verkonhallintajärjestelmän sisältämien laitteiden yleinen rakenne /5/

On huomioitavaa, että vaikka SNMP -protokolla ja MIB -tietokannat ovat standardisoituja teknologioita niin verkonhallintasovellukset itsessään eivät sitä ole. Ne ovat valmistajakohtaisia ja ne käyttävät alla olevaa standardisoitua SNMP -sovelluskehystä toteuttaessa haluttuja verkonhallintapalveluita.

Management Information Base (MIB) on looginen tietokanta, jonka jokainen verkonhallintajärjestelmän laite sisältää. Kyseessä on looginen tietokanta siinä mielessä, että todellisuudessa tietokannan sisältö voi olla tallennettu esim. laitteen flash muistiin, kytkinasetuksiin, tiedostoihin, laitelaskureihin jne. /5/

MIB:in tulee sisältää oleellinen tieto laitteesta kuten mm. sen konfigurointiasetukset, tilatiedot jne. Verkonhallintapalvelin voi SNMP -protokollaa käyttämällä kysellä lait-

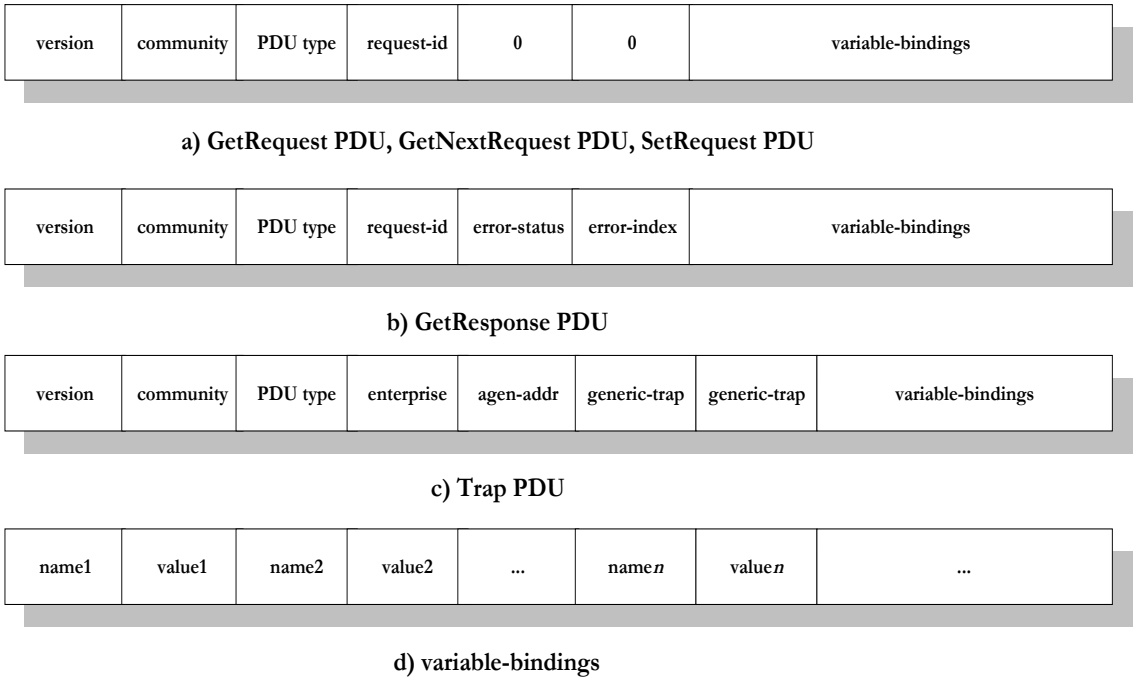
teelta sen MIB -tietokannassa olevia tietoja (objekteja) ja näin saada selville kyseisen laitteen tilan. Verkonhallintapalvelin voi myös muuttaa näitä kyseisiä asetuksia ja näin konfiguroida laitteen asetuksia uudestaan. /5/

Joillekin näistä MIB -tietokannassa olevista objekteista on voitu asettaa hallittavan verkkolaitteen toimesta erillinen 'liipaisin' jollekin tapahtumalle. Tämä voi tarkoittaa esimerkiksi sitä, että jos tarkasteltava laite on reititin ja verkonhallintapalvelin pyytää reitittimeltä tietoa tämän jostakin verkkorajapinnasta (verkkokortista) ja saa vastauksena, että verkkokortti on epästabiilissa tilassa. Tällöin verkonhallintapalvelin voi asettaa reitittimen MIB -tietokannassa olevan kyseisen verkkokortin enabled -objektin pois päältä SetRequest -viestillä, jolloin objektiin kytketty liipaisin joko initialisoi verkkokortin uudelleen tai asettaa sen pois käytöstä. /5, 7/

Erilaisten laiteryhmien sisältämät MIB -tietokannat on standardoitu. Tämä tarkoittaa sitä, että esimerkiksi eri reitittin valmistajien reitittimistä täytyy löytyä tuki samoille MIB -objekteille.

Verkonhallintapalvelin ja verkkolaitteet ovat linkitettyjä keskenään SNMP -protokollalla. SNMP on yksinkertainen verkonhallintaprotokolla, jonka toiminta perustuu muutamiin datagrammiviesteihin (<10). SNMPv1:ssa on mukana seuraavat aikaisemmin mainitut viestit: SetRequest, GetRequest, GetNextRequest, GetResponse ja Trap.

Viestien rakenne on seuraavanlainen:



Kuva 3: SNMPv1 PDU -viestien rakenne /7/

Näitä viestejä käyttämällä verkonhallintajärjestelmä voi kysellä verkossa olevilta laitteilta tietoja näiden toiminnasta ja sisäisistä tiloista tai vaihtoehtoisesti konfiguroida uudestaan laitteiden asetuksia. Myös verkkolaitteet voivat käyttää viestejä ilmoittaessaan toimintaolosuhteidensa muuttumisesta eli yleisemmin havaitusta virheestä, joka on tapahtunut laitteen toiminnassa.

SNMP -protokollaviestien sisältämien kenttien tyypit ovat osajoukko ASN.1 kuvauskielen muuttujien tyypeistä. SNMPv1:ssa on mukana seuraavat tyypit. /5/

Perustyypit:

- INTEGER
- OCTECT STRING
- OBJECT IDENTIFIER
- NULL

Perustyypeistä johdetut tyypit:

- Counter
- Gauge
- TimeTicks
- IpAddress
- NetworkAddress
- Opague

SNMP -protokolla toimii internetverkoissa UDP/IP -protokollapinojen päällä. SNMP on toteutettu myös muunlaisille verkkoympäristöille (FDDI, Token Ring, Frame Relay jne.), mutta tässä erikoistyössä keskitytään pelkästään internetverkkoihin.

SNMP on noussut yksinkertaisuutensa, suoraviivaisuutensa ja laajennettavuutensa takia suosituimmaksi verkonhallintaprotokollaksi. Siitä on tehty useita eri versioita (SNMPv1, SNMPv2 ja SNMPv3). Tässä erikoistyössä toteutetaan tuki pelkästään SNMPv1:lle.

3.1.2 Yleistä testauksesta

Testaus on yleisesti osa yrityksien laatujärjestelmää, jonka tavoitteena on parantaa tuotteiden laatua ja luotettavuutta, kehitysprosessin tuottavuutta ja pitkällä tähtäimellä vähentää automatisoiduilla järjestelmillä testaukseen kohdistuvaa työmäärää, jonka on ohjelmistoprosesseissa arvioitu olevan 30-50% kokonaistyömäärästä. Kehitettyjen ohjelmistojen toiminnan varmuutta voidaan oleellisesti lisätä testaamalla ohjelmisto yleisesti hyväksytyin ja standardoiduin menetelmin. /9/

Ensimmäisessä vaiheessa testausprosessia täytyy selvittää testattavan ohjelmiston määrittelyt (mahdollisesti standardit) ja ominaisuudet, joiden perusteella joko hankitaan valmiina olevat tai laaditaan itse sopivat testiaineistot.

Määrityksien mukaan tiedetään testattavan ohjelmiston käyttäytyminen kullakin testiaineistossa mukana olevalla syötteellä ja näin ollen tiedetään joko määritellyt tai ennustettavissa olevat vastearvot kullekin syötteelle. Mikäli testausprosessin myötä saatu arvo

poikkeaa odotetusta, voidaan testattavassa systeemissä todeta suurella todennäköisyydellä olevan virhe. Testatessa täytyy kuitenkin huomioida myös se seikka, että testausprosessissakin saattaa olla virheitä ja tämän takia virheen löytyessä täytyy verifioida sekä testausprosessi, että testattava systeemi. Virheen syy analysoidaan tarkemmin ja tehdyn analyysin tuloksien pohjalta virhe voidaan korjata. Tehdyt kokemukset voidaan kirjata testausraporttiin. /9, 10/

Testausmenetelmät voidaan jakaa kolmeen eri luokkaan: rakenteiseen testaukseen (White Box testing), lasilaatikkotestaukseen (Glass Box testing) sekä toiminnalliseen testaukseen (Black Box testing). Toiminnallinen testaus riippuu ohjelman ajonaikaisista ominaisuuksista, kun taas rakenteinen testaus keskittyy ohjelman staattisen rakenteen tutkimiseen. Toiminnallinen testaus varmistaa ohjelman ajonaikaisen oikeellisen toiminnan ja rakenteinen testaus varmistaa ohjelmistomoduulien logiikan yhteistoiminnan. /10/

Toiminnallisessa testauksessa testattava systeemi nähdään ’mustana laatikkona’, jonka tarkempaa sisäistä toimintaa ei tunneta. Testattavalle systeemille toimitetaan jokin määrittelyjen pohjalta tunnettu syöte, jonka määritelty vaste myöskin tunnetaan. Tällöin voidaan verrata tunnettua vastearvoa ’mustan laatikon’ palauttamaan arvoon ja näin verifioida testin oikeellisuus. Toiminnallinen testaus voidaan suorittaa alhaalta ylös tai vaihtoehtoisesti ylhäältä alas menetelmillä.

Rakenteinen testaus ei liity ohjelman ajonaikaisiin ominaisuuksiin, vaan se keskittyy ohjelman rakenteen tutkimiseen. Tätä menetelmää käyttäessä testaaja johtaa testiaineiston ohjelman logiikkaa tutkimalla. /10/

Lasilaatikkotestauksessa testaaja on tarkalleen tietoinen ohjelmiston rakenteesta ja sisäisestä toiminnasta. Tämän perusteella hän pystyy laatimaan spesifistisiä testiaineistoja, joilla voidaan testata halutessa ohjelmiston jokainen lause ja haara. Kun tietty testien kattavuusprosentti on saavutettu, testaus voidaan lopettaa. /10/

Testaus voidaan jakaa kolmeen eri pääalueeseen: standardinmukaisuustestaus, suorituskykytestaus ja toiminnallisuustestaus. Tässä työssä käytetään standardinmukaisuustestauksen menetelmiä.

3.1.3 Standardinmukaisuustestaus

Standardinmukaisuustestauksessa pyritään varmistamaan toteutetun protokollan/ohjelmiston ja standardin välinen yhdenmukaisuus niin, että implementaatio toteuttaa yksiselitteisesti standardin sille asettamat vaatimukset. Standardinmukaisuustestaus pitää sisällään staattisten ominaisuuksien sekä dynaamisen käyttäytymisen testauksen. Se ei ota kantaa suorituskyvyn tai luotettavuuden testaamiseen. /13/

Staattiset ominaisuudet ovat ominaisuuksia, jotka kaikkien standardin toteutuksien täytyy sisältää. Tällaisia ominaisuuksia voivat olla esimerkiksi protokollaviestien minimi/maksimi koko, viestien staattiset rakenteet, ajastimien kestoalueet, globaalien muuttujien arvoalueet jne. Dynaamiset ominaisuudet esimerkiksi SNMP -protokollaa ajatellen ovat, SNMP -viestien koodaaminen, vastaanotettuihin syötteisiin vastaaminen sopivilla vasteilla jne. /10/

Standardinmukaisuustestaus on määritelty International Organisation for Standardization (ISO) tuottamassa, viisi osaa sisältävässä, ISO 9646 standardijoukossa sekä vastaavasti International Telecommunication Union - Telecommunication Stardardization Sector:in (ITU - T) tuottamassa X.290 - X.296 suositusjoukossa. Nämä yhtenevät standardit/suositukset määrittelevät standardinmukaisuustestauksen testausmetodologian ja -kehyksen sekä menetelmät testiaineistojen laatimiseen avoimet rajapinnat toteuttavien järjestelmien (OSI järjestelmät) yhdenmukaisuustestauksessa. /9, 10, 13/

Standardinmukaisuustestaus noudattaa mustalaatikkotestauksen toimintaperiaatteita, joka tarkoittaa sitä, että testattavasta systeemistä tunnetaan vain sen ulkoinen rajapinta ja odotettu toiminta. Sisäistä toteutusta ei tunneta.

ISO 9646-1 määrittelee yleiset käsitteet OSI -järjestelmien standardinmukaisuustestausmenetelmille ja testauskehykselle. /13/

ISO 9646-2 standardi esittää viitekehyksen testiaineistojen kehittämisprosessille ja määrittelee kuinka abstraktit testiaineistot (Abstract Test Suite) ja testitulokset voidaan kuvata yleisellä, testinotaatiosta riippumattomalla tavalla. Standardoidut testiaineistot tulee toteuttaa jokaiselle OSI -protokollalle niin, että testiaineistoa voivat käyttää protokollan toteuttajat, käyttäjät, testauslaboratoriot, ylläpitäjät jne. Jotta testiaineisto voitaisiin standardoida, sen laadinnan pohjalla tulee olla kansainvälisesti hyväksytty metodologia ja määrittelyt. /10, 13/

ISO 9646-2 standardissa esiintyvät mm. seuraavat pääkäsitteet:

- Implementation Under Test (ITU)
- System Under Test (SUT)
- Point of Control and Observation (PCO)
- Abstract Test Suite (ATS)
- Means Of Testing (MOT)

ITU tarkoittaa testattavaa implementaatiota, joka voi olla yksittäinen protokolla tai protokollaperhe. Standardinmukaisuustestausta voi protokollatestauksen lisäksi soveltaa myös mm. hajautettujen CORBA -pohjaisten ohjelmistojen ja suurten järjestelmien testaamiseen. ITU on osa suurempaa kokonaisuutta SUT:ta, joka kattaa testattavan implementaation sekä tätä ympäröivän järjestelmän. Ympäröivä järjestelmä voi olla ohjelmisto-/laitekokonaisuus esimerkiksi sulautetun järjestelmän käyttöjärjestelmä, joka tarjoaa suoritusympäristön ja palveluita IUT:lle. /10/

PCO on havainnointipiste, jossa testerin ja IUT:n välisiä kommunikointiprimitiivejä (ASP)/protokollaviestejä (PDU) pystytään tarkkailemaan. Eli toisin sanoen PCO on toiminnallisuus, joka välittää viestejä testerille ja testeriltä niin, että se pystyy samalla tarkkailemaan kyseisiä viestejä.

ATS on abstrakti testiaineisto, joka on kuvattu ISO 9646-3 (X.292) standardin määrittelemällä Tree and Tabular Combined Notation (TTCN) kuvauskielellä. /10, 13/

MOT on ATS:a laajempi kokonaisuus, joka testitapahtumien kuvaamisen lisäksi ottaa huomioon semanttiset ja testausympäristöön liittyvät seikat. Tällaisia seikkoja ovat mm. testauksessa tarvittavien parametrien arvojoukot, työkalut suoritettavien testien valintaan, suoritettavan testiaineiston (ETS) johtaminen abstraktin testiaineiston pohjalta ja ETS:n suorittaminen sekä tulosten keruun testauslokeihin. MOT on todellinen testausjärjestelmä, joka yhdistää laitteiston ja testimateriaalin tavalla, jossa testausprosessi voidaan suorittaa testituloksen aikaansaamiseksi. ATS ei ole koskaan suoritettavissa sellaisenaan vaan tarvitaan tietoja käytettävien parametrien arvoista ja valituista testitapauksista. Testien suorittajan (testausohjelmiston) täytyy muuttaa ATS suorituskelpoiseksi prosessiksi. /10/

Termistön ja käsitteistön määrittämisen lisäksi ISO 9646-2 esittää viitekehyksen testiaineiston kehittämisprosessille sekä määrittelee kuinka testiaineistot ja tulokset kuvataan yleisellä testauskohteesta riippumattomalla tavalla.

ISO 9646-3 määrittelee Tree and Tabular Combined Notation (TTCN) kuvauskielen, jota käytetään kuvaamaan standardinmukaisuustestauksen abstrakteja testiaineistoja. /10/

Standardinmukaisuusvaatimukset voidaan luokitella yleisellä tasolla seuraaviin luokkiin: /13/

- Pakolliset vaatimukset. Testausprosessin tulee verifioida nämä vaatimukset aina.
- Ehdolliset vaatimukset. Testausprosessin tulee verifioida nämä vaatimukset mikäli standardissa esitetyt ehdot käyvät toteen.
- Vaihtoehtoiset vaatimukset. Testausprosessin tulee verifioida nämä vaatimukset vain silloin mikäli standardissa esitetyt vaihtoehtoiset ominaisuudet on otettu toteutukseen mukaan.

Jotta tietyn järjestelmän standardinmukaisuutta pystyttäisiin arvioimaan, on tarpeellista tietää mitkä ominaisuudet järjestelmään on toteutettu. Toteutetuista ominaisuuksista tulee standardinmukaisuustestausta ajatellen laatia Implementation Conformance Statement (ICS) lausunto. ICS:iin kuuluu kolme osa-aluetta: Protocol Implementation Conformance Statement (PICS), profile Implementation Conformance Statement (profile ICS) sekä System Conformance Statement (SCS). /13/

PICS on ICS, joka on tarkoitettu yhdelle protokollalle. PICS -lausunnossa tulee erotella pakolliset, vaihtoehtoiset sekä ehdolliset staattiset standardinmukaisuusvaatimukset itse protokollalle sekä erilliset lisävaatimukset protokollan riippuvuuksien testaamiseksi muista protokollista ja palveluiden tarjoajista. /13/

Profile ICS koostuu joukosta eri protokollien PICS -lausuntoja, joista jokainen toteuttaa jonkin vaatimuslistan (Requirement List) asettamat vaatimukset eli täyttää jonkin profiilin luokituksen. /13/

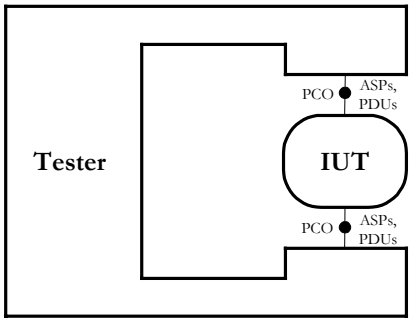
SCS on lyhennelmä joukosta systeemin ICS -lausuntoja, jota tarvitaan kuvaamaan kaikkia spesifikaatioita joita systeemi noudattaa. SCS:a käytetään kuvaamaan systeemin konfigurointimahdollisuuksia sekä kuvaamaan mitkä spesifikaatiot ovat standardinmukaisuustestauksen kohteita. /13/

ICS -lausuntojen lisäksi testauslaboratoriot tarvitsevat lisätietoa testattavasta protokollasta/järjestelmästä (IUT) sekä testausympäristöstä. Tämä tieto toimitetaan asiakkaan toimesta Implementation Extra Information for Testing (IXIT) lausunnon muodossa testauslaboratoriolle. IXIT -lausunto ei saa olla ristiriidassa vastaavien ICS -lausuntojen kanssa. /13/

IXIT -lausuntoa, joka keskittyy tiettyyn protokollaan kutsutaan Protocol IXIT (PIXIT) lausunnoksi. PIXIT voi sisältää joitakin seuraavista kohdista: tietoa testattavasta systeemistä (SUT), PICS -lausuntoa tarkentavaa tietoa sekä lisätietoa, joka määrittelee tarkemmin mitä ominaisuuksia testattava systeemi tukee yms. /13/

Standardinmukaisuustestauksen tulokset dokumentoidaan joukkoon standardinmukaisuus testiraportteja. Näitä raportteja on kahden tyyppisiä: System Conformance Test Report (SCTR) sekä Protocol Conformance Test Report (PCTR). Testauksen tulee tuottaa aina SCTR -dokumentti, joka on lyhennelmä systeemin standardinmukaisuustilasta. PCTR tuotetaan jokaista testiaineistoa kohden ja se sisältää suoritettujen testien tulokset.

Abstraktit testausmenetelmät (Abstract Test Method) kuvaavat yleisellä tasolla, kuinka IUT:ta testataan testausjärjestelmää käyttäen. Testausjärjestelmä näkee testaustilanteen mustalaatikkotestauksena, jolloin testausjärjestelmä lähettää IUT:lle abstrakteja palveluprimitiivejä (ASP) tai protokollaviestejä (PDU). IUT käsittelee nämä viestit ja primitiivit ja tilanteen mukaan lähettää testausjärjestelmälle vastauksena oman protokollaviestinsä/palveluprimitiivinsä. Vastauksen saatuaan testausjärjestelmä voi päätellä onko IUT toiminut tämän testin osalta halutulla tavalla standardia noudattaen. ATM:iin kuuluu myös PCO -pisteiden kuvaaminen suhteessa testausjärjestelmään. /10, 13/



Kuva 4: Käsitteellinen testausarkkitehtuuri /13/

Abstraktit testausmenetelmät voidaan jakaa koostuvaksi neljään abstraktiin testausfunktioon: /13/

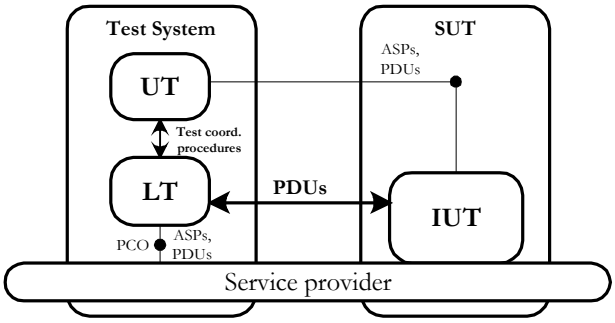
- Yläpuolinen testeri (Upper Tester)
- Alapuolinen testeri (Lower Tester)
- Alapuolisen testerin koordinointifunktiot (Lower Tester Control Function)
- Testien koordinointiproseduurit (Test Coordination Procedures)

Testausjärjestelmä on mahdollista jakaa kahteen eri yhteyteen (context), joissa testitapauksia voidaan spesifioida: Single-Party Testing (SPyT) sekä Multi-Party Testing (MPyT). SPyT -ratkaisua tarvitaan, kun testit vaativat IUT:ta kommunikoimaan vain yhden systeemin kanssa. MPyT -ratkaisua tarvitaan, kun testit vaativat, että IUT:n on kommunikoitava useiden erillisten systeemien kanssa. Tässä erikoistyössä keskitytään pelkästään SPyT -ratkaisuihin. /12, 13/

SPyT -skenaariossa käytettävät abstraktit testausfunktiot ja niiden sijoittuminen toisiinsa nähden on seuraava. Yläpuolinen testeri sijaitsee IUT:n yläpuolella ja hallitsee kyseistä rajapintaa. UT on IUT:n tarjoamien palveluiden käyttäjä. Alapuolinen testeri sijaitsee IUT:n alapuolella ja hallitsee puolestaan kyseistä rajapintaa. LT lähettää viestejä IUT:lle ja analysoi tämän lähettämät vasteet. Testien koordinointiproseduureja käytetään LT:n ja UT:n välillä (*kts. Kuva 6*). TCP:ien pääasiallinen tehtävä on mm. näiden kahden testauskomponentin välinen keskinäinen ajastus esimerkiksi kättelyrutiinit. /10, 13/

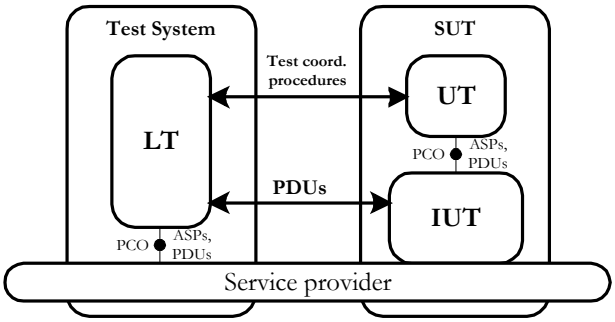
Käsitteellisestä testausmallista on johdettu neljä erilaista abstraktia testausmetodia (Abstract Test Method): paikallinen tai ulkoinen, hajautettu, koordinoitu ja etämenetelmä. Testausmetodin tehtävänä on kuvata abstrakti testausarkkitehtuuri käyttäen abstraktien testausfunktioiden, testattavan systeemin (SUT) ja testausjärjestelmän välisiä suhteita. Jokainen ATM määrittelee PCO:t ja testitapahtumat (ASP, PDU), joita käytetään kyseisen ATM:n abstrakteissa testitapauksissa. /10, 12/

Paikallisessa menetelmässä UT on sijoitettu LT:n kanssa samaan testausjärjestelmään, mikä tarkoittaa sitä, että testausjärjestelmä ohjaa suoraan ylempää ja alempaa testeriä. Tämä voi tarkoittaa esimerkiksi sitä, että LT:n ja UT:n välistä tiedonsiirron tahdistusta eivät määrää LT ja UT itse vaan näitä ympäröivä testausjärjestelmä, joka pitää huolen sekventiaalisen tiedonsiirron ajastuksista. Tällainen järjestelmä on harvinainen, koska tällöin testeri ja testattava systeemi ovat sidottuja samaan järjestelmään ja UT:n kytkeminen IUT:hen edellyttää IUT:lta standardinmukaista palvelurajapintaa. /10, 12/



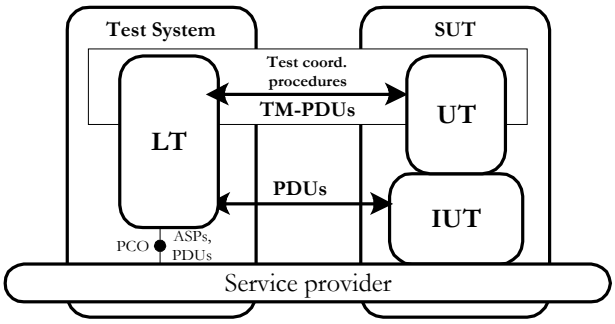
Kuva 5: Paikallinen testausmenetelmä /12, 13/

Hajautetussa testausmenetelmässä UT sijaitsee IUT:n kanssa samassa järjestelmässä. Eri järjestelmissä sijaitsevien LT:n ja UT:n välistä ohjausrajapintaa ei ole määritelty. IUT ylärajapinta voi olla joko käyttäjälle tarkoitettu käyttöliittymä tai standardisoitu ohjelmointikielinen rajapinta, jota voi käyttää erilaisiin testaustarkoituksiin. /10, 12/



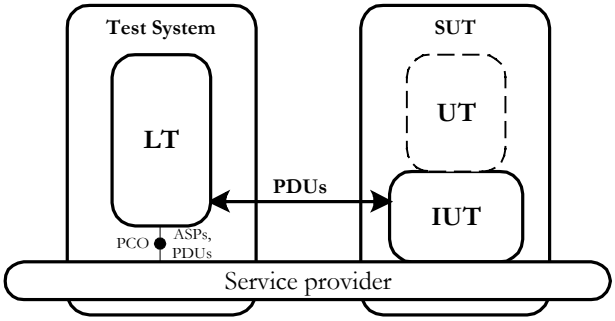
Kuva 6: Hajautettu testausmenetelmä /12, 13/

Koordinoidussa testausmenetelmässä on mahdollista vaihtaa ohjaussanomia UT:n ja LT:n välillä, jolloin LT ohjaa UT:n toimintaa. Ohjaussanomat koodataan IUT:n käyttämän siirtoprotokollan protokollaviestien dataosuuteen, josta UT saa ne itselleen käyttämällä IUT:n UT:lle tarjoamaa palvelurajapintaa.



Kuva 7: Koordinoitu testausmenetelmä /12, 13/

Etämenetelmässä testitapaukset on laadittu siten, ettei UT:n toteutuksella eikä ohjaussanomilla ole merkitystä. Etämenetelmässä LT huolehtii yksin testaustulosten analysoinnista IUT:n LT:lle lähettämien vaste ASP:den perusteella.



Kuva 8: Etätestausmenetelmä /12, 13/

Standardinmukaisuustestauksen tulisi antaa yksiselitteinen vastaus siitä, onko testattavajärjestelmä yhdenmukainen standardin kanssa. Tällainen kokonaisvaltainen testaus on monessa tapauksessa mahdoton toteuttaa sellaisenaan lähinnä testattavien asioiden kompleksisuuden ja määrän takia. Tämän takia standardinmukaisuustestaus jaetaan neljään osa-alueeseen testauksen kattavuutta ajatellen: /10, 13/

- Perustestit (Basic Interconnection Tests - BIT)
- Toimintotestit (Functional Range Tests - FRT)
- Standardinmukaisuustestit (Conformance Tests - CT)
- Päätöstestit (Conformance Resolution Tests - CRT)

Perustesteillä tutkitaan ovatko protokollan perustoiminnat määrittelyjen mukaisia. Toimintotestit määräävät toteutuksen tukemat ominaisuudet, esimerkiksi mitä protokolla viestejä/kenttiä/versioita testattavassa systeemissä tuetaan. Usein toimintotestien määräämät ominaisuudet ovat osin vaihtoehtoisia. Standardinmukaisuustestaus kertoo määritysten ja toteutuksen vastaavuuden. Päätöstestaus antaa lopullisen ja yksikäsitteisen vastauksen toteutuksen standardinmukaisuudesta. /9, 10/

3.1.4 ISO 9646-3 (Tree and Tabular Combined Notation)

ISO 9646-3 (X.292) määrittelee Tree and Tabular Combined Notation (TTCN) kuvauskielen, jota käytetään avoimien järjestelmien (OSI) välisten protokollien standardinmukaisuustestaukseen (conformance testing). TTCN soveltuu kaikkien OSI 1. kerroksen yläpuolella olevien protokollien testaukseen erityisesti mukaan lukien ASN.1:een pohjautuvat protokollat. TTCN -kieltä voi käyttää myös tuotekehitysaikana järjestelmätestauksessa tai CORBA -pohjaisten hajautettujen ohjelmistojen/-järjestelmien testaamiseen. TTCN -testaus on riippumaton testausmenetelmistä, kerroksista, verkkoprotokollista tai testilaitteista ja se noudattaa abstraktin testausmetodologian käsitteitä, jotka on määritelty ISO 9646-1 ja ISO 9646-2 standardeissa. Se on laajalti käytetty valmistajien, käyttäjien ja testilaboratorioiden piirissä ja se mahdollistaa standardoitujen testiaineistojen laadinnan. /9, 11/

TTCN -kuvauskielellä jokaiselle OSI -standardille määritellyn standardisoidun testiaineiston tulisi toteuttaa seuraavat kohdat:

- Saavuttaa yleisesti laaja hyväksyntä ja luottamus standardinmukaisuustestauksen tuloksiin.
- Tarjota luottamus eri laitteiden väliseen kykyyn toimia keskenään.

TTCN -kieli on tarkoitettu reaktiivisten järjestelmien testaamiseen. Eli toisin sanoen testaustilanne noudattaa mustalaatikkotestauksen käsitteitä, jolloin testattavan järjestelmän sisäistä toimintaa ei tunneta ja testattavan järjestelmän toimintaa tarkastellaan ulkopuolelta käsin. /9/

TTCN:n perusperiaate on testitapausten määritteleminen tavalla, joka ajokelpoisen testitapauksen luomisen standardissa kuvatun abstraktin protokollamäärittelyn pohjalta. TTCN on suunniteltu täyttämään seuraavat kohdat: /9, 10, 11/

- TTCN tarjoaa notaation, jolla abstraktit testitapahtumat voidaan ilmaista standardoidussa testiaineistossa järjestelmäriippumattomalla tavalla. TTCN:n käyttö tekee mahdolliseksi saman testiaineiston käytön eri testisysteemeissä. Eli toisin sanoen eri työkaluilla toteutetut testaussysteemit eivät vaadi testiaineiston uudelleen suunnittelua.
- TTCN tarjoaa notaation, joka on riippumaton testausmetodeista, kerroksista ja protokollista ja joka noudattaa ISO 9646 standardeissa määriteltyä abstraktia testausmetodologiaa.
- TTCN soveltuu testien laatimiseen kaikille protokollille.
- TTCN mahdollistaa testaustapahtumien automatisoinnin ja täten nopeuttaa tuotekehitysprosessia, tuottaa kustannussäästöjä, lisää testauksen systemaattisuutta ja tarkkuutta jne.
- TTCN mahdollistaa valmiiden testiaineistojen hyväksikäytön mm. ATM, DECT, GSM, ISDN, SS7, TETRA jne. protokollista on saatavilla valmiit standardoidut TTCN -testiaineistot.
- TTCN parantaa avointen järjestelmien (OSI) yhteistoimintaa standardoimalla testausta.
- TTCN on helposti opittava merkintätapa testien suunnittelijoille, jolloin parannetaan testistandardien yleistä ymmärrettävyyttä.
- TTCN tukee rinnakkaisten testiprosessien suorittamista.

TTCN:sta on olemassa kaksi muotoa: graafinen lomakemuodossa oleva TTCN.GR sekä ohjelmallinen TTCN.MP. Ensimmäinen on tarkoitettu pääasiassa standardien kuvaamiseen ja se soveltuu standardin soveltajille. Jälkimmäisessä muodossa testit kuvataan Backus Naur Form (BNF) säännöillä. Tämä muoto soveltuu pääasiassa tietokoneiden väliseen tiedonsiirtoon. Nämä kaksi muotoa ovat keskenään semanttisesti ekvivalentteja. /9, 10, 11/

TTCN on osa ISO/IEC 9646 testausmetodologiaa ja sitä käytetään määrittämään testitapauksista koostuvia testiaineistoja. Testitapaukset sisältävät testin varsinaisen toiminnallisuuden eli tapahtumat, jotka ovat tavallisesti systeemissä kulkevia TTCN:lla ja ASN.1:llä kuvattuja palveluprimitiivejä/protokollaviestejä. TTCN -testiaineisto sisältää hierarkkisesti kuvatun testirakenteen, joka jaottelee testiaineiston määrittelyihin (declarations), rajoittimiin (constraints), operaatioihin (operations), testiryhmiin (groups), testiaskeliin (steps), testitapauksiin (cases) ja testitapahtumiin (behaviours). Testien toiminnallisuus on määritelty puumaisina sanomasekvensseinä, joissa kuvataan lähetettävät ja vastaanotettavat viestit sekä ajastintapahtumat. Viestien lähettäminen/vastaanottaminen sekä ajastintapahtumat ovat perustestitapahtumia. /9, 10, 11/

ASN.1 on yleinen standardoitu merkintätapa tietotyyppinen ja arvojen määrittämiseksi, joilla voidaan tarjota riippumaton esitystapa siirtomuodolle. ASN.1:tä käytetään tietoliikenneprotokollien ja -sovellusten protokollaviestien ja tietoalkioiden esittämiseen standardeissa. Mahdollisuus käyttää ASN.1 -määrityksiä TTCN -kielessä varmistaa yhdenmukaisuuden järjestelmämääritysten ja testimääritysten välillä (*kts. 3.1.5 Abstract Syntax Notation One*). /10/

Testitapaus rakennetaan käyttämällä testikomponentteja ja määrittämällä niiden välille yhteyksiä. Testitapauksissa, joissa enemmän kuin yksi käyttäytymiskuvaus pyörii rinnakkaisesti, käytetään TTCN -rinnakkaisuus (concurrency) ominaisuuksia. ISO 9646-3 määrittelee rinnakkaiseen TTCN:ään liittyvät termit sekä käsitteet, kuinka rinnakkaisuutta voidaan käyttää TTCN:ssa abstraktien testitapauksien määrittelyssä. Alkuperäinen TTCN -standardi määritteli testitapauksen yhdeksi prosessiksi, joka kykeni kommunikoimaan ympäristönsä kanssa käyttämällä yhtä tai kahta PCO:ta, jotka mahdollisesti saattoivat olla fyysisesti eri järjestelemissä. Rinnakkainen TTCN tarjoaa rajattoman määrän PCO- ja CP -yhteyksiä testikomponenttien välillä. /10, 11/

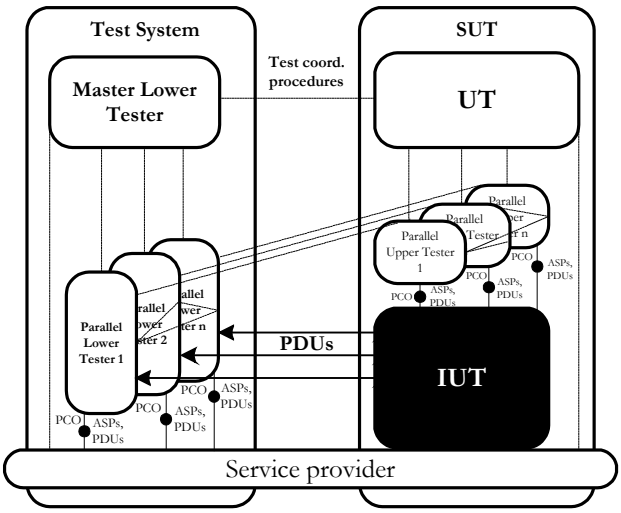
TTCN mahdollistaa rinnakkaisten testikomponenttien laatimisen niin, että ne voidaan suorittaa yhtä aikaa rinnakkaisesti. Testiaineiston ei ole pakko sisältää rinnakkaisuutta sisältäviä elementtejä vaan rinnakkaisuus TTCN:ssa on vaihtoehtoinen ominaisuus. Testeri koostuu päätestikomponentista (Main Test Component) sekä nollasta tai useammasta rinnakkaistestikomponentista (Parallel Test Component). Mikäli kyseessä

on ei rinnakkaisuutta käyttävä testiaineisto, käytössä on vain yksi testikomponentti, joka on automaattisesti tyyppiä MTC, ilman erillisiä määrittelyjä. /11/

MTC on vastuussa PTC:n luomisesta ja valvomisesta sekä testituloksen laskennasta ja määräämisestä. PTC on MTC:n valvonnassa oleva rinnakkaistestikomponentti, joka suorittaa oman testiosa-alueensa testit sekä määrää näille testituloksen, jota käyttämällä MTC voi määrittää lopullisen testituloksen. /11/

Testikomponentit voivat kommunikoida IUT:n kanssa yhden tai useamman PCO:n kautta. Komponentit voivat keskustella toistensa kanssa vaihtamalla koordinointiviestejä (Coordination Message) koordinaattipisteiden (Coordination Point) kautta. MTC pystyy tarkkailemaan PTC:a ilman koordinointiviestejä siinä mielessä, että se havaitsee mikäli PTC:n suoritus on päättynyt. TTCN:ssa pystytään määrittelemää erilaisia konfiguraatioita määritellyille komponenteille esimerkiksi jossakin konfiguraatiossa saattaa olla yksi MTC ja viisi PTC:a ja jossain toisessa konfiguraatiossa yksi MTC, yksi PTC ja yksi CP.

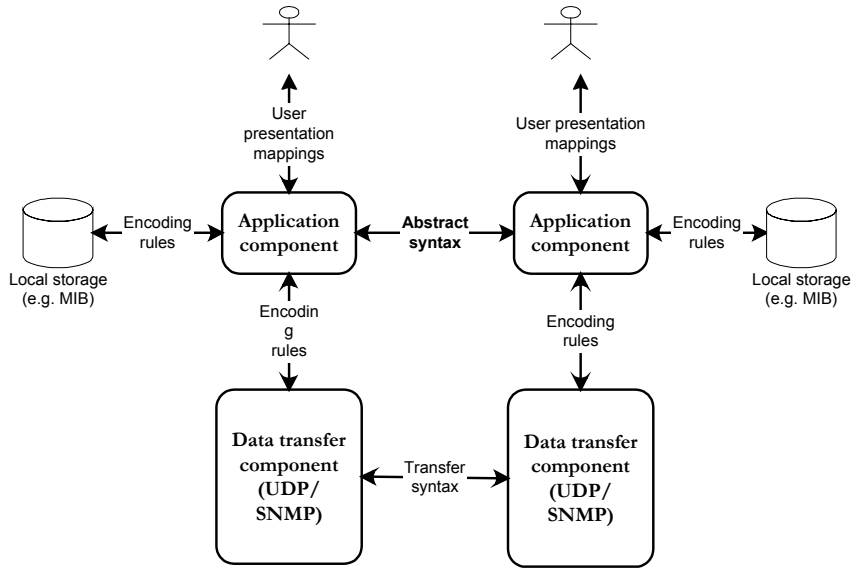
Koordinaatio viestit (CM) noudattavat samanlaisia määrittelyjä kuin abstraktit palveluprimitiivit (ASP) ja protokollaviestit (PDU). Ne ovat muutenkin käyttäytymisensä suhteen ASP:en ja PDU:en kaltaisia. Testikomponentit on mahdollista toteuttaa yhdellä koneella tai hajauttaa usealle eri koneelle. Seuraava kuva esittää rinnakkaista testausarkkitehtuuria, joka selventää määriteltyjen käsitteiden suhteita.



Kuva 9: TTCN:n käyttämä yleinen rinnakkainen testiarkkitehtuuri /10/

3.1.5 Abstract Syntax Notation One

Abstract Syntax Notation One (ASN.1) on abstrakti kuvauskieli rakenteelliselle informaatiolle, joka on riippumaton laitteistoista, verkoista, käyttöjärjestelmistä, ohjelmointiympäristöistä jne. Sitä käytetään yleisesti OSI- sekä TCP/IP -standardien kehittämisessä. ASN.1 on tarkoitettu mm. hajautettujen järjestelmien suunnittelijoille, joiden tehtävänä on kuvata protokollia, joilla järjestelmän eri komponentit keskustelevat keskenään. ASN.1 mahdollistaa protokollien kuvaamisen yleisellä tasolla niin ettei suunnittelijoiden tarvitse tuntea yksityiskohtaisesti varsinaista protokollaviestien siirtotekniikkaa (*kts. 3.1.6 Basic Encoding Rules*). /6, 14/



Kuva 10: Abstraktin syntaksin (ASN.1) ja siirtosyntaksin käyttö /6/

Verkkosolmun voidaan katsoa koostuvan kahdesta pääkomponentista: tiedonsiirtokomponentti (Data transfer component) sekä sovelluskomponentti (Application component). Kun tarkastellaan näiden kahden komponentin tapaa esitellä dataa, havaitaan huomattavia eroja; tiedonsiirtokomponentti käsittelee dataa bitti- ja tavujonoina, kun taas sovelluskomponentti käsittelee dataa abstraktilla tasolla, joka on riippumaton verkkosysteemistä. /6/

OSI -mallin käsitteiden mukaisesti kahden verkkosysteemin välinen datansiirto sovellustasolla voi tapahtua käyttämällä sovelluskerroksen PDU:ita, jotka on kuvattu abstraktilla tasolla eli käyttäen ASN.1 -notaatiota. Todellinen tiedonsiirto tapahtuu tiedonsiirtokomponenttien välillä binäärisessä muodossa käyttäen linkki-/verkkokerroksen PDU:ita (esimerkiksi SNMP käyttää UDP -protokollan datagrammeja, joiden dataosuus on BER -koodatussa muodossa). /6/

ASN.1 -syntaksi täytyy muuttaa käyttäjälle sopivaksi informaatioksi tilanteesta riippuvia kuvaussääntöjä käyttämällä. Tiedonsiirtokomponentille ASN.1 -syntaksi täytyy muuttaa sopivaksi käyttämällä erilaisia koodaussääntöjä, joista Basic Encoding Rules (BER) on yleisin. /6, 14/

3.1.6 Basic Encoding Rules

Basic Encoding Rules (BER) on Consultative Committee on International Telephony and Telegraphy:n (CCITT) kehittämä ja standardoima koodausmääritys, joka löytyy CCITT:n suosituksesta X.209 ja ISO:n standardista ISO 8825. Nämä standardit määrittelevät metodit, joilla jokainen ASN.1 -kielen tyyppi voidaan koodata sopivaan muotoon tiedonsiirtoa varten eri koneiden välillä verkossa.

Koodauksen tarkoitusta voidaan verrata tietokoneohjelmaan, joka on kirjoitettu korkeantason ohjelmointikielellä. Jotta tietokone pystyisi suorittamaan ohjelman, täytyy se ensin kääntää tietokoneen ymmärtämään konekieliseen muotoon. Vastaavalla tavalla ASN.1 -määrittelyt täytyy muuttaa verkossa siirrettävään muotoon. BER:in käyttö mahdollistaa ettei sovellussuunnittelijoiden eikä useampien ohjelmoijien, jotka voivat käyttää abstrakteja datamäärittelyitä (ASN.1), tarvitse tuntea varsinaista siirtosyntaksia yksityiskohtaisesti. /5, 7, 14/

Koska BER ja ASN.1 luotiin samaan aikaan samojen ihmisten toimesta, ovat ne erittäin läheisesti linkitettyjä toisiinsa. BER -säännöt määräävät yhden tai useamman tavan koodata mikä tahansa ASN.1 -tyyppi siirrettävään muotoon.

SNMP -protokollan tapauksessa ASN.1 -tyypit koodataan tavujonoiksi, koska tämä on luonnollinen siirtomuoto UDP -protokollan ollessa kyseessä. /5, 7/

Koodaussäännöt perustuvat yleensä Type-Length-Value (TLV) rakenteen käyttöön. Tämä tarkoittaa sitä, että mikä tahansa ASN.1 -tyypin arvo voidaan koodata kolmen joukkona seuraavilla komponenteilla: /7, 14/

- Type (tyyppi). Osoittaa koodattavan arvon ASN.1 -tyypin ja luokan, johon tyyppi kuuluu. Lisäksi se osoittaa koodausmuodon eli onko koodaus primitiivien vai rakenteellinen.
- Length (pituus). Osoittaa ASN.1 -tyypin koodatun arvon pituuden.
- Value (arvo). ASN.1 -tyypin koodattu arvo.

TLV -rakenne on rekursiivinen eli jokaisen TLV -rakenteen V -osa voi sisältää uuden TLV -rakenteen ja edelleen, jonka V -osuus voi sisältää uuden TLV -rakenteen jne. BER -koodauksen suosio pohjautuu sen joustavuuteen, joka johtuu koodattujen viestien ’itsemäärittävyydestä’. Tämä tarkoittaa sitä, että jokainen viestin kenttä itse tietää minkä tyyppinen se on, kuinka pitkä se on ja mikä sen arvo on. Näin ollen koko viesti on itsemääräytyvä. /5, 7/

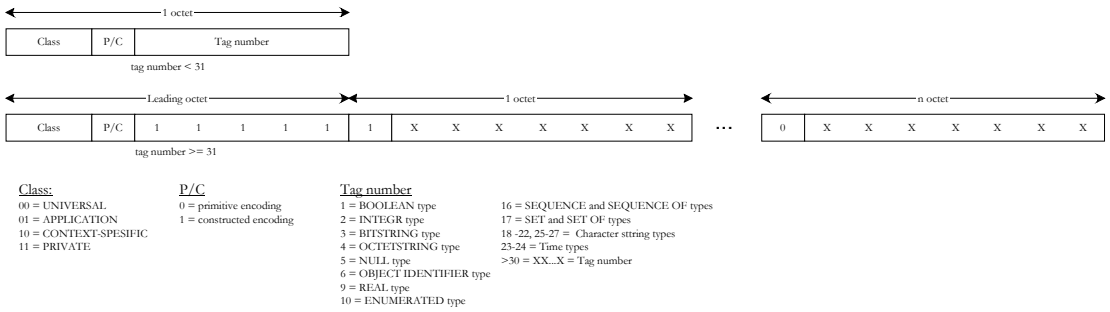
On olemassa kolme metodia, joilla ASN.1 arvo voidaan koodata: /7/

- Primitiivinen määrätty pituus (Definite-Length)
- Rakenteellinen määrätty pituus (Definite-Length)
- Rakenteellinen määräämätön pituus (Indefinite-Length)

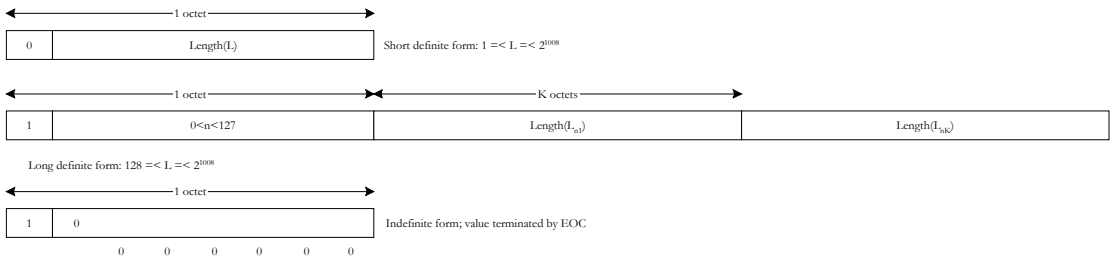
Metodin valinta riippuu koodattavan arvon ASN.1 -tyypistä sekä siitä, että tiedetäänkö koodattavan arvon pituus sen tyypin perusteella. Primitiivinen määrätty pituus metodia voidaan käyttää yksinkertaisten tyyppien tai implisiittisesti yksinkertaisista tyypeistä johdettujen tyyppien koodaamiseen. Rakenteellinen määrätty pituus metodia voidaan käyttää yksinkertaisten merkkijono tyyppien, rakenteellisten tyyppien sekä näistä implisiittisesti johdettujen tyyppien koodaamiseen. Rakenteellinen määräämätön pituus metodia voidaan käyttää samojen tyyppien koodaukseen kuin edellisessä metodissa mutta erona on se, että nyt koodatun arvon pituutta ei tarvitse tietää etukäteen. /7/

BER -koodauksessa TVL -formaatti koostuu kolmesta kentästä (*kts. kuva 10*): /5, 7, 14/

- Identifier (tunnistin). Tämä kenttä koodaa tyyppi tagin (luokka ja tag numero) koodattavalle arvolle. Ensimmäiset 2 bittiä osoittavat mihin luokkaan koodattavan arvon tyyppi kuuluu. Seuraava bitti osoittaa koodauksen muodon. Jos bitin arvo on 0, kyseessä on primitiivinen koodaus. Jos arvo on 1, kyseessä on rakenteellinen koodaus. Loput viisi bittiä ilmaisevat tagin numeron, jonka tarkoituksena on ilmaista koodattavan arvon tyyppi kyseisen luokan sisällä. Mikäli tagin numero on suurempi tai yhtä suurin kuin 31, asetetaan viimeisten viiden bitin arvoksi 11111. Tämä ilmaisee, että tagin numero ilmaistaan lisätavuissa (*kts. Kuva 11*).
- Length (pituus) Tämä kenttä määrittää Contents kentän pituuden tavuissa tai ilmaisee jollakin tapaa miten koodatun arvon loppu voidaan havaita (ilman, että tunnetaan varsinainen kentänpituus). Mikäli sisällön pituus on suurempi kuin 127, käytetään pituuden esittämiseen lisätavuja. Jos kyseessä on rakenteellinen määräämätön pituus metodi, pituus kenttään ei laiteta Contents kentän varsinaista pituutta vaan alku tag (80'h), joka ilmaisee Contents kentän alkua ja Contents kentän jälkeen sijoitetaan toinen tag (0000'h), joka ilmaisee kyseisen kentän loppua (*kts. Kuva 12*).
- Contents (sisältö). Tämä kenttä sisältää koodatun ASN.1 -arvon tavuina.



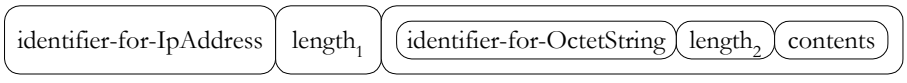
Kuva 11: BER -koodausformaatti: Identifier kentän rakenne /7/



Kuva 12: BER -koodausformaatti: Length kentän rakenne /7/

BER -koodausformaatin Identifier -numero ilmaisee koodattavan arvon tyypin. Koska ASN.1:tä käytetään useiden satojen erilaisten protokollien määrittämiseen, joista jokainen sisältää lukuisia erilaisia datatyyppejä, on erittäin suuri työ varmistaa, että jokaisella uudella tyypillä on oma uniikki Identifier -numero. Toinen hankaluus on varmistaa kaikkien olemassa olevien ASN.1 -määrittelyjen keskinäinen yhdenmukaisuus tilanteessa, jossa jonkin tyypin määrittelyä ja tätä kautta mahdollisesti tyyppinumeroa päätetään muuttaa. Lisäksi kymmeniä tai satoja BER Identifier -kenttiä sisältyy mahdollisesti yhteen viestiin, tämä antaa syyn pitää kenttien Identifier tunnistimet mahdollisimman pienenä, koska datakommunikointiin halutaan mahdollisimman vähän ’overheadia’. /5/

IMPLICIT määrittely ASN.1 -kielessä mahdollistaa ’overheadin’ pienenemisen BER -koodauksessa. Tilannetta voi mallintaa seuraavalla esimerkillä. Jos alkuperäinen rakenteellinen ASN.1 -määrittely (ilman IMPLICIT lausetta) on BER -koodattu seuraavanlaiseksi rakenteeksi: /5, 14/



Niin IMPLICIT määrittelyä käyttämällä koodaus saadaan muotoon:

Näin ollen sama informaatio saadaan pienempää tilaan jolloin protokollanviestin, jossa kyseistä tyyppiä käytetään, informaatio-osuuden ’overhead’ pienenee. Luonnollisestikaan IMPLICIT lauseen käyttö ei sovellu kaikkien ASN.1 -tyyppien määrittelyjen yhteydessä.

Yleisesti ongelma on ratkaistu BER -koodauksessa jakamalla datatyypit neljään luokkaan: /5/

- Universal (Universaali). Tämä pitää sisällään primitiiviset datatyypit, joita kaikki protokollat ym. määrittelyt voivat käyttää.
- Application (Sovellus). Tämän luokan datatyyppejä voivat käyttää tietyt sovellukset kuten esimerkiksi SNMP -verkonhallintasovellutukset.
- Context-spesific (Asiayhteys-spesifistinen). Tämän luokan datatyypit sisältyvät johonkin suurempaan datatyyppiin.
- Private (Yksityinen). Yksityisten organisaatioiden määrittelemät datatyypit.

SNMP:n käyttämät datatyypit (*kts. 3.1.1 Verkonhallintajärjestelmät*) jakaantuvat näihin luokkiin seuraavalla tavalla (samalla on esitetty jokaisen datatyypin Identifier tunnistimen arvo heksadesimaalilukuna):

Primitive ASN.1 Types	***Identifier in hex***
INTEGER	02
BIT STRING	03
OCTET STRING	04
NULL	05
OBJECT IDENTIFIER	06
Constructed ASN.1 type	
SEQUENCE	30
Primitive SNMP application types	
IpAddress	40
Counter	41
Gauge	42
TimeTicks	43
Opague	44
Context-spesific types within on SNMP messages	
GetRequest-PDU	A0
GetNextRequest-PDU	A1
GetResponse-PDU	A2
SetRequest-PDU	A3
Trap-PDU	A4

Taulukko 1: SNMP:n käyttämät tyypit ja näiden tunnisteet /5/

Seuraavana muutama esimerkki ASN.1 tyyppien BER -koodauksesta:

Counter -tyyppinen muuttuja, joka on implisiittisesti periytetty INTEGER -tyypistä sisältää arvon 4795. Kyseisen muuttujan BER -koodausformaatti on seuraava (huom. esityksessä on käytetty heksadesimaalilukuja):

4795_{10} → (BER) → 41_{16} 02_{16} 12_{16} 97_{16}

Object Identifier, jonka arvo on 1.3.6.1.2.2.250.143. Tällöin BER -koodausformaatti on seuraava. Huom. Object Identifier:in koodauksessa on omia erikoissääntöjä. (Lisätietoa BER -koodauksesta löytyy lähdeviitteistä 5, 7, ja 14):

1.3.6.1.2.2.250.143 → (BER) → 06_{16} 09_{16} 43_{16} 06_{16} 01_{16} 02_{16} 02_{16} 81_{16}
$7A_{16}$ 81_{16} $0E_{16}$

3.1.7 Common Object Request Broker Architecture

Common Object Request Broker Architecture (CORBA) on Object Management Group:in (OMG) kehittämä tekniikka, jolla mahdollistetaan hajautettujen ohjelmistojen eri komponenttien välinen toiminta huolimatta niiden varsinaisesta sijainnista verkossa. Nämä komponentit voivat sijaita erilaisissa verkkojärjestelmissä, ne voivat olla eri henkilöiden suunnitelmia ja toteuttamia, ne voivat toimia erilaisissa käyttöjärjestelmissä ja voivat olla toteutettuja eri tyyppisissä ohjelmointiympäristöissä. Toisin sanoen CORBA tekee näistä eri komponenteista läpinäkyviä toistensa suhteen ja näin ollen mahdollistaa niiden yhteistoiminnan.

CORBA on tarkoitettu mm. hajautettujen Asiakas - Palvelin ohjelmistojen toteutukseen ja se mahdollistaa, että asiakas voi esittää pyyntöjä palvelimelle ilman, että se tarkemmin tietää missä palvelinohjelmisto sijaitsee, minkälaisessa ympäristössä sitä ajetaan, millä ohjelmointikielellä se on toteutettu jne. Palvelimen etsiminen ja asiakkaan palvelupyyntöjen sekä palvelimen vastauksien sovittaminen asiakkaan ja palvelimen (sekä näiden järjestelmien) välillä on CORBA -tekniikan pääkomponentin, Object Request Broker:in (ORB), tehtävä. Se sovittaa eri järjestelmien lähettämät palveluprimitiivit ja näiden vastaukset toisiinsa sopiviksi ja näin mahdollistaa järjestelmäriippumattomuuden. /15/

Aiheesta löytyy enemmän tietoa mm. Robert Orfalin kirjasta: Client/Server programming with JAVA and CORBA (*kts. lähdeluettelo*) tai Object Management Group:in Internet sivuilta (http://www.omg.com/).

3.2 OpenTTCN protokollatesteri

3.2.1 Yleistä

Tässä kappaleessa käsitellään OpenTTCN protokollatesterin arkkitehtuuria ja rakennetta erikoistyön kannalta. Tässä ei esitellä esimerkiksi OpenTTCN:n eri versioita (OpenTTCN Workstation Tester, OpenTTCN Demo Tester ja OpenTTCN Tester ToolKit), vaan pitäydytään mahdollisimman yleisellä tasolla.

3.2.2 Toimintaidea

OpenTTCN protokollatesteriä käytetään pääasiassa automatisoimaan protokollien ja hajautettujen järjestelmien standardinmukaisuustestausta. Tämän lisäksi sitä voidaan käyttää ohjelmistojen tuotekehitysvaiheen testauksessa, toimivuustestauksessa sekä verkkojen integraatiotestauksessa (Network Integration Testing). /17/

Standardinmukaisuustestausta ajatellen testeriä voidaan käyttää perinteisten protokollien kuten ATM, DECT, ISDN, GSM jne. testaukseen, joista yleensä on toteutettu valmiit standardoidut testiaineistot eri standardointijärjestöjen (esimerkiksi ETSI) toimesta. Perinteisten televerkoissa käytössä olevien protokollien lisäksi testeriä voidaan käyttää internetprotokollien testaukseen mukaan lukien CORBA -palvelut, Mobile IP, Voice over IP ja SNMP -pohjaiset verkonhallintajärjestelmät jne. /9, 17/

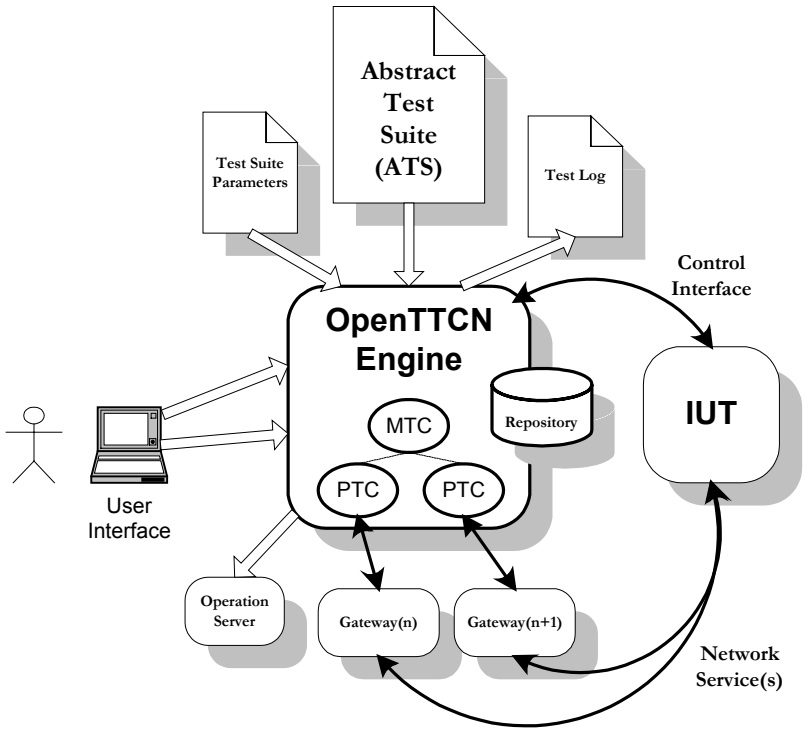
3.2.3 Ominaisuudet / arkkitehtuuri

OpenTTCN protokollatesteri on avoin ratkaisu, jonka toiminta pohjautuu ISO/IES 9646 standardien (ITU - T:n X.290 suosituksien) määrittelemään standardinmukaisuustestaus metodologiaan. OpenTTCN protokollatesteri suorittaa testit niin, että se tulkitsee abstraktit testitapaukset suoraan ajonaikana eikä operaatiossa vaadita erillisiä käännös- / linkitysvaiheita. /9, 17/

Abstraktit testitapaukset on kuvattu ISO 9646-3 standardissa määritellyllä TTCN -kielellä Machine Processable (TTCN.MP) muodossa. Abstrakti testiaineisto (ATS) kuvataan testitiedostoissa, joista testeri pystyy lataamaan testit tietokantaansa ja edelleen suorittamaan nämä.

Koska testien kuvaaminen tapahtuu abstraktilla tasolla (TTCN, ASN.1), testattavalla systeemillä ei ole vaikutusta testien kuvaamiseen. Testeri on testattavan systeemin suhteen geneerinen ratkaisu ja näin ollen sen yhteyteen tulee aina toteuttaa erillinen sovitinohjelma (Gateway), jonka tehtävänä on sovittaa testeri testattavaan implementaatioon (IUT). Toisin sanoen, testeri käyttäytyy aina samalla tavalla riippumatta testattavasta systeemistä ja vain Gateway toteutus vaihtelee testattavasta kohteesta riippuen. /8, 9/

OpenTTCN testerin arkkitehtuuri rakentuu seuraavista komponenteista ja niiden välisistä suhteista: User Interface, OpenTTCN Engine, Repository, Operation Server ja yksi tai useampi Gateway. User Interface, OpenTTCN Engine ja Repository ovat osa OpenTTCN ohjelmiston ydintä. Gateway:t ja vaihtoehtoinen Operation Server eivät kuulu testerin ydin toteutukseen vaan ovat osa kokonaistestausjärjestelmää. OpenTTCN testeri kokonaisuudessaan, Operation Server sekä Gateway:t muodostavat yhdessä hajautetun testausjärjestelmän.



Kuva 13: OpenTTCN testausjärjestelmä /17/

Testeri tarjoaa käyttäjälle komentorivipohjaisen käyttöliittymän (User Interface), jolla käyttäjä voi säädellä testerin toimintaa. Käyttöliittymä tarjoaa työkalut testiaineistojen lataamiseen tiedostoista testerin tietokantaan, testien valitsemiseen sekä testien ajamiseen. Lisäksi käyttöliittymä mahdollistaa standardinmukaisien testauslokien tuottamisen.

Repository ja TTCN Engine muodostavat systeemin, jota käytetään tallettamaan ja suorittamaan testiaineistoja. Abstrakti testiaineisto talletetaan Repository:n yhdessä testi-

aineistoon liittyvien parametrien kanssa. TTCN Engine:ä käytetään valitsemaan ja suorittamaan testitapauksia käyttöliittymä työkalujen ohjeiden mukaisesti.

OpenTTCN Engine on geneerinen TTCN -virtuaalikone, joka suorittaa abstrakteja testiaineistoja tulkitsemalla. Testiaineistojen tulee olla kirjoitettu standardinmukaisella TTCN -notaatiolla. Virtuaalikone muodostaa abstrakteista testiaineistoista suoritettavia testiaineistoja (Executable Test Suite). Tulkkaamisesta johtuen ei suoritettavan testiaineiston muodostamisessa tarvita välivaiheita kuten kääntämistä ja linkitystä.

Operation Server on palvelinkomponentti, joka toteuttaa testiaineiston määrittelemät operaatiokuvaukset. Mikäli operaatiot on kuvattu testiaineistossa proseduraalisesti, operaatiot suoritetaan suoraan OpenTTCN Engine virtuaalikoneen toimesta. Tällöin erillistä Operation Server ratkaisua ei tarvita.

Repository on tietokanta, jota käytetään 'säilömään' abstraktit testiaineistot ja niihin liittyvät parametrit. Säilöminen tapahtuu muodossa, jossa OpenTTCN voi tulkita säilöttyä testiaineistoa.

OpenTTCN testeri tarjoaa Gateway:lle avoimen oliopohjaisen OMG:n määrittelemän CORBA -ohjelmointirajapinnan, jonka avulla Gateway voidaan kytkeä testeriin ja jonka avulla voidaan siirtää dataa testeriltä Gateway:lle ja päinvastoin. Gateway toteuttaa varsinaisen PCO -määrittelyn, joka on esitetty testiaineistossa. Gateway:n tehtävänä on kytkeä OpenTTCN Engine verkko(siirto)palveluiden tarjoajaan ja tätä kautta edelleen testattavaan implementaatioon (IUT). /17/

Testerin ja SUT:in välillä oleva Gateway toteutus riippuu täysin testattavasta systeemistä. Jos testerillä testattaisiin esimerkiksi jotain älyverkko komponenttia, olisi Gateway:n ja SUT:n välinen protokollapino mahdollisesti Signaling System no. 7 (SS7). Tässä työssä Gateway:n ja testattavan systeemin välinen kommunikointiprotokolla on UDP, koska työn mielenkiinto suuntautuu SNMP -protokollaan, joka toimii IP -verkossa.

3.2.4 Toiminta / käyttö

Seuraavaksi esitellään pelkistetysti varsinaisen testausprosessin kuvaus OpenTTCN protokollatesteriä käyttäen:

- Käynnistetään OpenTTCN testeri
- Käynnistetään IUT ja tarvittaessa SUT.
- Käynnistetään tarvittavat Gateway implementaatiot. Näitä voi olla yksi tai useampi.
- Ladataan haluttu abstrakti testiaineisto (ATS) testerin tietokantaan käyttöliittymä komennoilla.
- Käyttöliittymää käyttämällä ladataan halutut parametrit, jotka liittyvät ladattuun testiaineistoon, testerin tietokantaan. Standardinmukaisuustestausta ajatellen nämä parametrit ilmoitetaan yleensä IXIT- ja PIXIT -lausunnoissa.
- Mikäli testiaineisto määrittelee erityisiä operaatioita, käynnistetään erillinen operaatiopalvelin (Operation Server), joka kytketään testeriin käyttämällä käyttöliittymäkomentoja.
- Kytketään Gateway implementaatiot testerin yhteyteen käyttämällä käyttöliittymää.
- Testaus aloitetaan käyttöliittymältä annettavalla komennolla. Tällöin on mahdollista valita jokin tietty tai mahdollisesti useampia suoritettavia testitapauksia.
- Lopuksi käyttäjä voi tutkia OpenTTCN:n tuottamia testauslokeja.

4 ERIKOISTYÖSSÄ TÄLLÄ HETKELLÄ TOTEUTETUT ASIAT

4.1 Yleinen testausarkkitehtuuri

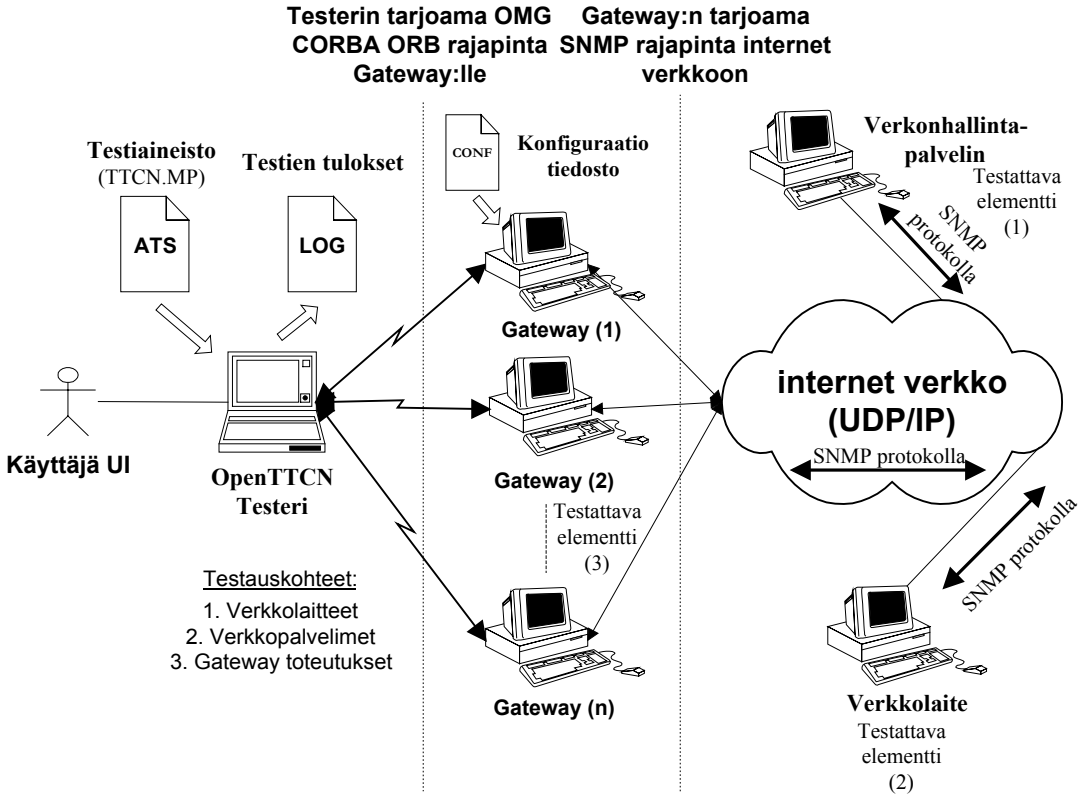
4.1.1 Yleistä

Erikoistyössä määritelty yleinen testausarkkitehtuuri verkonhallintajärjestelmien testaamiseksi rakentuu kolmesta pääkomponentista: OpenTTCN protokollatesteri, Gateway implementaatiot sekä verkonhallintajärjestelmä. Verkonhallintajärjestelmä koostuu kahdesta osajoukosta: verkonhallintapalvelimet ja tavalliset verkkolaitteet (*kts. Kuva 14*).

Protokollatesteri muodostaa suoritettavia testiaineistoja (ETS) abstrakteista testiaineistoista (ATS) tulkitsemalla näitä ajon aikana. Testit muodostuvat perustapahtumien suorittamisesta, joita ovat mm. protokollaviestien (PDU) lähetys ja vastaanotto sekä erilaiset ajastintoiminnot. Tarkemmin OpenTTCN testeristä on kerrottu kappaleessa *3.2 OpenTTCN protokollatesteri*.

Gateway:tä käytetään sovittamaan OpenTTCN testerin tarjoama CORBA -liityntärajapinta verkonhallintajärjestelmän SNMP -protokollapinoon. Toisin sanoen Gateway on sovitinohjelma, joka vastaanottaa testerin lähettämiä PDU:ita ja koodaa ne SNMP -viestien muotoon ja lähettää ne edelleen testattavalle järjestelmälle. Luonnollisesti Gateway vastaanottaa testattavan järjestelmän lähettämät vastaukset, dekoodaa ne PDU:ksi ja lähettää edelleen testerille. Näin mahdollistetaan testerin ja testattavan järjestelmän välinen kommunikointi. Gateway implementaatiosta on kerrottu tarkemmin kappaleessa *4.1.2 Gateway*. Verkonhallintajärjestelmistä on kerrottu tarkemmin kappaleessa *3.1.1 Verkonhallintajärjestelmät*.

Verkonhallintajärjestelmässä on kaksi pääelementtiä, joita voidaan testata protokollatesterillä: verkonhallintapalvelin ja verkkolaite. Tässä työssä otetaan mukaan vielä kolmas testattava elementti, Gateway implementaatio. Tarkemmin testauskohteista on kerrottu kappaleessa *4.1.3 Testattavat elementit.*



Kuva 14: Yleinen testausarkkitehtuuri verkonhallintajärjestelmien testaamiseksi

4.1.2 Gateway

Gateway:n perusidea on toimia sovitinohjelmana OpenTTCN testerin ja testattavan systeemin (verkonhallintajärjestelmä) välillä. Gateway vastaanottaa testeriltä protokollaviestejä (PDU:ita), jotka se koodaa SNMP -viesteiksi ja lähettää edelleen verkonhallintajärjestelmälle. Luonnollisesti Gateway vastaanottaa verkonhallintajärjestelmän lähettämiä SNMP -viestejä, jotka se dekoodaa PDU:ksi ja lähettää edelleen testerille. Näin Gateway mahdollistaa testerin ja verkonhallintajärjestelmän välisen kommunikoinnin.

OpenTTCN tarjoaa Gateway toteutuksille Object Request Broker (ORB) liityntärajapinnan, jota käytetään liittäessä abstrakteissa testiaineistossa (ATS) määritellyt PCO:t Gateway toteutuksiin, jotka varsinaisesti toteuttavat testiaineistoissa määriteltyjen PCO:den toiminnallisuudet. Liityntärajapinta tarjoaa Gateway:n käytettäväksi CORBA -metodeja, joilla voidaan siirtää dataa OpenTTCN testeriltä Gateway:lle ja päinvastoin. Gateway:n tulee kyetä keskustelemaan testerin lisäksi myös testattavan systeemin kanssa. Tähän käytetään SNMP -protokollaa, jota verkkolaitteet ja verkonhallintapalvelimet ymmärtävät.

Koska Gateway:n liityntä OpenTTCN testeriin ja tiedonsiirto testerin ja Gateway:n välillä tapahtuu CORBA -metodeja käyttämällä ja toisaalta Gateway:n tulee pystyä kommunikoimaan verkonhallintajärjestelmän kanssa käyttämällä SNMP -protokollaa, valittiin Gateway:n toteutuskieleksi Java. Java:n valintaa tukivat mm. seuraavat seikat:

- Java tukee OMG:n standardoimaa CORBA -teknologiaa.
- Java mahdollistaa UDP/IP -protokollatason ohjelmoinnin tarjoamalla yksinkertaisia ja kohtuullisen tehokkaita metodeja sokettien määrittelyyn ja UDP -tason tiedonsiirtoon.
- Java ohjelmointikieli on järjestelmäriippumaton eli samaa koodia voidaan käyttää tarvittaessa niin Linux kuin Windows alustalla ilman komplikaatioita.
- Koska SNMP -viestien välittäminen ei vaadi täydellistä reaaliaikaisuutta, Java:n suorituskyky on riittävä.
- Java ohjelmointikieli on helppo ja nopea omaksua, se mahdollistaa mm. säikeiden sisällyttämisen ohjelmaan helposti. Java koodin dokumentointiin on olemassa ilmaisia ja helppokäyttöisiä työkaluja.

Vaihtoehdot Java:lle olivat pääasiassa C- tai C++ ohjelmointikielet. Siihen miksi näitä kieliä ei valittu toteutuskieliksi vaikuttivat mm. seuraavat seikat: TML -laboratorion Lappeenrannan projekteissa käytetään yleisesti oliopohjaisia ratkaisuja ja C -kielen käyttäminen oliopohjaisen CORBA -liitynnän toteuttamiseen ei ole paras vaihtoehto, C- ja C++ kielet ovat monimutkaisempia verrattuna Java:an. Lisäksi Gateway:lla ei ole tarvetta olla täysin reaaliaikainen. Java:lla toteutettu Gateway pystyy välittämään muutamia SNMP -viestejä sekunnissa ja tämä on riittävä tehokkuus Gateway:lle.

Gateway:ta toteutettaessa ensimmäisenä on määriteltävä Gateway:n arkkitehtuuri ja eritoten liitynnät OpenTTCN testeriin ja verkonhallintajärjestelmään. Gateway:n arkkitehtuurin sekä liitäntöjen perusteella voidaan määritellä testausarkkitehtuuri, jossa käy ilmi testerin, Gateway:n ja testattavan systeemin välinen konfiguraatio (*kts. Kuva 14*). Testausarkkitehtuurin ja Gateway:n liitäntöjen määrittelyjen välillä on riippuvuuksia ja näin ollen Gateway:n arkkitehtuuri, liitäntöjen osalta, vaikuttaa siihen millaisia konfiguraatioita testerin, Gateway:n ja testattavan systeemin välillä on mahdollista toteuttaa.

Koska Gateway on sovitinohjelma kahden systeemin välillä, on arkkitehtuurin sisällettävä seuraavat määrittelyt: liitännät ulkoisiin systeemeihin, kooderi ja dekooderi elementit, elementit viestien vastaanottoon ja lähettämiseen.

Gateway:n liitynnät testeriin tapahtuvat PCO -määrittelyjen kautta. Gateway:n arkkitehtuurin kannalta oleellinen määrittely on montako PCO -liityntää yksi Gateway toteuttaa. Tässä työssä päädyttiin ratkaisuun, jossa Gateway toteuttaa yhden PCO -määrittelyn (*kts. Kuva 16*). Ratkaisuun vaikuttivat pääasiassa seuraavat seikat:

- Ratkaisu, jossa Gateway toteuttaa yhden PCO:n on tarpeeksi yleinen, jotta sitä voidaan soveltaa useimpiin testausarkkitehtuureihin.
- Tässä työssä toteutettavat testitapaukset ovat työn prototyyppi luonteesta johtuen niin yksinkertaisia, että testausarkkitehtuurissa määritellään vain konfiguraatioita, joissa jokaiselle PCO:lle on määritelty oma Gateway.

Vaihtoehto tälle ratkaisulle on tietenkin se, että yksi Gateway toteuttaa useampia PCO -määrittelyjä. Se kuinka monta PCO:ta yksi Gateway toteuttaa on oleellinen seikka määriteltäessä testausarkkitehtuuria. Esimerkiksi jos Gateway toteuttaa yhden PCO:n niin käytännössä tämä tarkoittaa sitä, että jokaista ATS:ssa ollutta PCO -määrittelyä kohden täytyy käynnistää oma Gateway ohjelma (*kts. Kuva 15*).

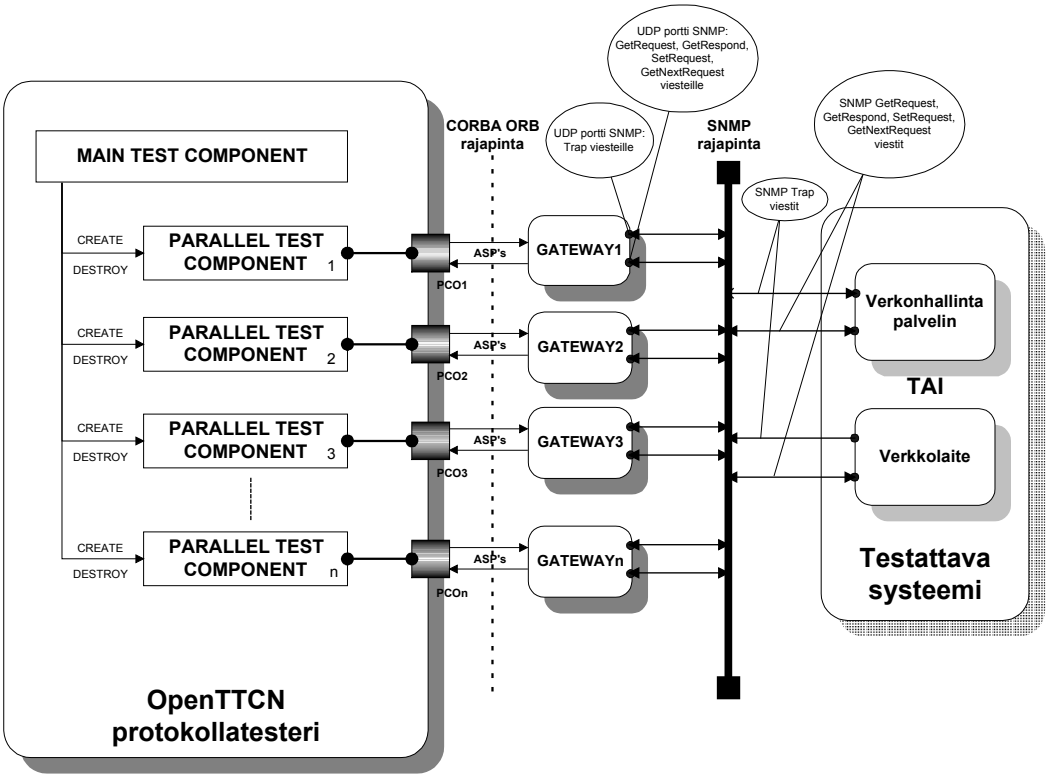
Jos Gateway toteuttaa useampia kuin yhden PCO -määrittelyn, esimerkiksi viisi, niin tällöin näitä viittä ATS:ssa ollutta PCO -määrittelyä kohden tarvitsee käynnistää vain yksi Gateway. Tällainen ratkaisu voisi soveltua tilanteeseen, jossa testerille syötetään

testiaineistoja, joissa määritellään suuri määrä PCO:ita. Tällöin Gateway toteutuksien käynnistäminen käsin jokaista PCO:ta kohden voi olla suuri operaatio ja parempi ratkaisu olisi käynnistää Gateway toteutukset automaattisesti yhden ohjelmarungon sisällä eri prosesseissa. Tämä mahdollistaisi lisäksi sen, että Gateway:lle voitaisiin antaa parametrina tieto siitä montako PCO:ta sen täytyy toteuttaa.

Gateway toteutettiin sisäisen arkkitehtuurin osalta niin, että sitä on helppo muuttaa. Tämän option takia Gateway on suhteellisen helppo määritellä toteuttamaan esimerkiksi kymmenen PCO:ta.

Gateway:n toteuttajan täytyy myös määritellä kuinka Gateway liitetään verkonhallintajärjestelmään. Työssä päädyttiin standardinmukaiseen ratkaisuun, jossa Gateway käyttää liitäntään kahta sokettia, joiden portti numerot voidaan määritellä konfigurointitiedostoissa (selitetään myöhemmin tässä kappaleessa). Toinen soketti käsittelee SNMP Trap -viestejä ja toinen SNMP GetRequest, GetNextRequest, SetRequest -viestejä (*kts. 3.1.1 Verkonhallintajärjestelmät*).

Mikäli abstraktissa testiaineistossa käytetään rinnakkaista TTCN:ää ei tällä ole vaikutusta Gateway:n arkkitehtuuriin. Tässäkin tapauksessa jokaista rinnakkaisissa testiprosesseissa olevaa PCO:ta varten tarvitaan oma Gateway toteutus, joka toteuttaa kyseisen PCO:n toiminnallisuuden (*kts. Kuva 15*). Edellä esitetyllä tavalla jokaisella Gateway:lla on oma sokettiparinsa, jolla se voi keskustella verkonhallintajärjestelmän kanssa.



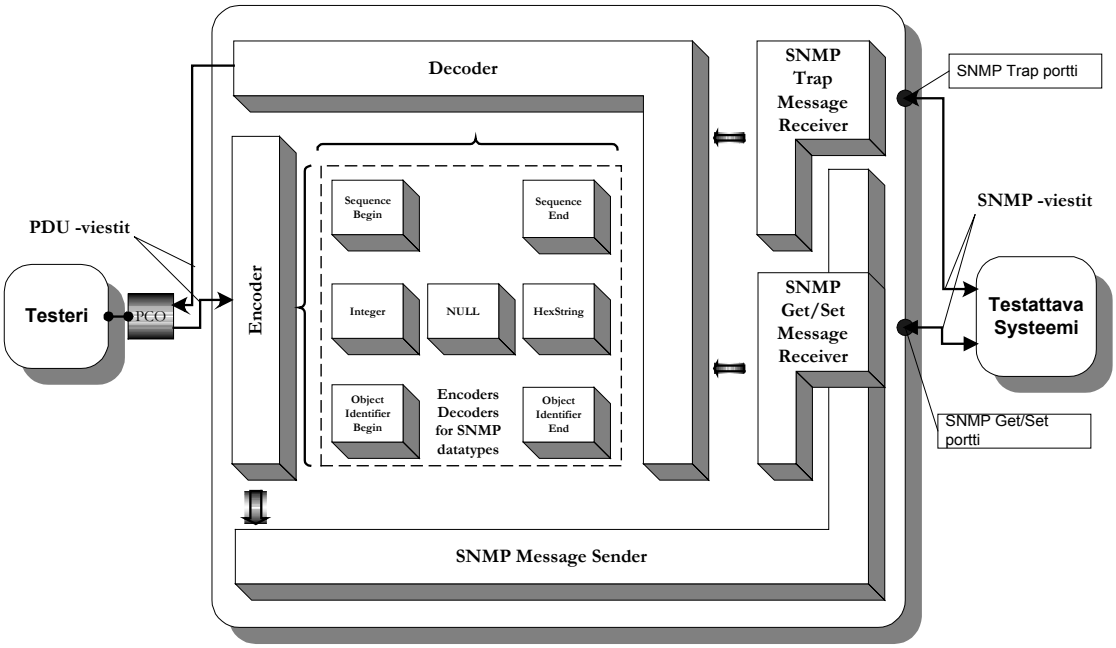
Kuva 15: Gateway toteutuksien sijoittuminen käytettäessä rinnakkaista TTCN:ta

Gateway:n arkkitehtuuri koostuu viidestä pääkomponentista: Decoder, Encoder, SNMP Message Sender, SNMP Trap Message Receiver, SNMP Get/Set Message Receiver. Näiden lisäksi arkkitehtuurissa on mukana seuraavat komponentit: Sequence Begin, Sequence End, Integer, NULL, HexString, Object Identifier Begin, Object Identifier End (*kts. Kuva 16*).

SNMP Trap Message Receiver komponentti vastaanottaa testattavalta systeemiltä (verkonhallintajärjestelmä) SNMP Trap -viestejä. Nämä viestit se välittää edelleen Decoder komponentille, joka dekoodaa viestit PDU:ksi ja välittää edelleen testerille. SNMP Get/Set Message Receiver komponentti toimii muuten vastaavalla tavalla kuin SNMP Trap Message Receiver komponentti mutta SNMP Trap -viestien sijaan se vastaanottaa kaikkia muita SNMP -viestejä.

Encoder komponentti vastaanottaa testeriltä abstrakteja PDU:ta, jotka se koodaa SNMP -viesteiksi ja välittää edelleen SNMP Message Sender komponentille, joka lähettää ne testattavalle systeemille (verkonhallintajärjestelmä).

Muut komponentit ovat koodaus/dekoodaus komponentteja, joita Encoder komponentti käyttää koodatessaan vastaanottamiaan PDU:ita SNMP -viesteiksi ja joita Decoder komponentti käyttää dekoodatessaan vastaanottamiaan SNMP -viestejä PDU:ksi. Kukin näistä komponenteista koodaa/dekoodaa tietyn tyyppisen kentän arvon. Esimerkiksi Integer komponentti koodaa PDU:ssa olevan Integer kentän arvon SNMP -viestiin BER -koodauksen mukaisesti. Vastaavasti toiseen suuntaan Integer komponentti dekoodaa (purkaa) BER -koodauksen ja sijoittaa Integer kentän arvon PDU:hun.



Kuva 16: Gateway:n arkkitehtuuri

Edellä Gateway:ta on esitelty yleisen arkkitehtuurin näkökulmasta. Tätä käytettiin pohjana Gateway ohjelman suunnittelussa ja toteutuksessa. Seuraavaksi Gateway:ta esitellään toteutusnäkökulmasta. Koska Gateway toteutettiin Java ohjelmointikielellä, on se täysin oliopohjainen. Tällöin ohjelman luokkarakenne voidaan esittää helposti käyttämällä Unified Modelling Language (UML) kuvauskieltä. UML -kieltä käytetään ohjelmistojen mallintamisessa kuvaamalla ja määrittelemällä ohjelmistokomponentteja (luo-

kat, luokkien ilmentymät jne.). Gateway:n luokkarakenteen UML -kuvaus on esitetty liitteessä 1.

Liitteessä 1 esitetyn Gateway ohjelman UML -kielisen luokkadiagrammin pohjalta voidaan esittää seuraavat toteutuksen pääkohdat (implementoitujen luokkien toiminta yleisellä tasolla):

Gateway ohjelman 'starting point' on ohjelman pääluokka, **SNMPGateway**. Se sisältää ohjelman käynnistämiseen tarvittavat metodit. Ohjelman käynnistämisen jälkeen eli sen jälkeen kun ohjelma on luonut ilmentymän (olion) SNMPGateway luokasta, tämä ilmentymä suorittaa toiminnallisuuksia, jotka parsivat ohjelmalle parametrina annetun konfigurointitiedoston sekä initialisoivat ohjelman näillä käyttäjän konfiguroimien tietojen perusteella.

SNMPGateway ja **SNMPThread** luokalla on UML -määrittelyjen mukaisesti (yhden suhde yhteen) koostumussuhde. Kun ohjelma luo ilmentymän SNMPGateway luokalle, samalla luodaan ilmentymä SNMPThread luokalle. SNMPThread ilmentymän tarkoituksena on luoda Gateway ohjelmaan säie, jonka sisällä ohjelman varsinainen toiminnallisuus (viestien lähettäminen, vastaanottaminen, koodaaminen, dekoodaaminen jne.) suoritetaan.

Aikaisemmin tässä kappaleessa mainittiin Gateway:n olevan toteutettu niin, että sen arkkitehtuuria on helppo muuttaa siten, että Gateway toteuttaisi useamman kuin yhden PCO -määrittelyn. Koska Gateway:n toiminnallisuus tapahtuu yhden säikeen sisällä, voidaan toiminnallisuuksien määrää (toteutettavien PCO -määrittelyjen määrää) lisätä määrittelemällä SNMPGateway ja SNMPThread luokkien välinen koostumussuhde niin, että SNMPGateway koostuu useammasta kuin yhdestä SNMPThread ilmentymästä (yhden suhde moneen yhteys). Eli toisinsanoin lisätään säikeiden määrää Gateway ohjelman sisällä.

SNMPThread luokka koostuu SNMPPco luokasta, joka määrittelee Gateway:n PCO -liitynnän OpenTTCN testeriin. Luotaessa ilmentymä SNMPThread luokasta, luodaan samalla ilmentymä SNMPPco luokasta.

SNMPPco luokalla on koostumussuhde seuraavien luokkien kanssa: SNMPGatewaySocketReader (yhden suhde kahteen), SNMPMsgSender (yhden suhde kahteen) ja EncoderEntity (yhden suhde yhteen). Eli näiden luokkien ilmentymät luodaan samaan aikaan kuin SNMPPCo luokan ilmentymä. SNMPPco luokan päätehtävä on toteuttaa testiaineistossa määritellyn PCO:n toiminnallisuus (viestien välittäminen testerin ja testattavan systeemin välillä). SNMPPco ilmentymä hoitaa kommunikoinnin testerin ja Gateway:n välillä eli se toteuttaa metodit, joilla voidaan vastaanottaa ja lähettää abstrakteja protokollaviestejä (PDU) testerin ja Gateway:n välillä.

SNMPPco käyttää SNMPGatewaySocketReader ja SNMPMsgSender luokkien metodeja keskustellessaan verkonhallintajärjestelmän kanssa. EncoderEntity luokkaa se käyttää testeriltä vastaanottamiensa PDU:en koodaukseen SNMP -viesteiksi.

SNMPGatewaySocketReader luokan ilmentymiä käytetään lukiessa Gateway:n soketteja (2 kpl). SNMPGatewaySocketReader luokalla on koostumissuhde (yhden suhde yhteen) DecoderEntity luokan kanssa, jota käytetään dekoodaamaan vastaanotetut SNMP -viestit PDU:ksi.

SNMPMsgSender luokan ilmentymät hoitavat testeriltä saapuneiden PDU:en, jotka on jo koodattu SNMP -viesteiksi, eteenpäin lähettämisen verkonhallintajärjestelmälle.

DecoderEntity luokan ilmentymä dekoodaa verkonhallintajärjestelmältä vastaanotetut SNMP -viestit PDU:ksi, jotka SNMPPco luokan ilmentymä välittää edelleen testerille.

EncoderEntity luokan olio koodaa testeriltä vastaanotetun PDU:n SNMP -viestiksi, joka se lähettää edelleen verkonhallintajärjestelmälle käyttäen SNMPMsgSender luokan ilmentymiä.

Sekä DecoderEntity ja EncoderEntity luokan oliot käyttävät seuraavien luokkien palveluita: **ASN1Integer**, **ASN1Null**, **HexString**, **ObjectIdentifierBegin**, **ObjectIdentifierEnd**, **SequenceBegin** ja **SequenceEnd**. Näiden luokkien ilmentymien tehtävänä on koodata Gateway:n testeriltä vastaanottamien PDU:den kenttien arvot

SNMP -viesteihin BER -koodauksella tai dekoodata Gateway:n verkonhallintajärjestelmältä vastaanottamien SNMP -viestien PDU:ksi, joka lähetetään testerille.

Gateway ohjelman muuttujien arvot on mahdollista alustaa käyttämällä konfigurointitiedostoa. Tällöin käyttäjä voi määritellä tiedostoon mm. Gateway:n sokettien porttien arvot, Gateway:n toteuttaman PCO:n nimen jne. Esimerkki konfigurointitiedoston rakenteesta on esitetty liitteessä 2.

4.1.3 Testattavat elementit

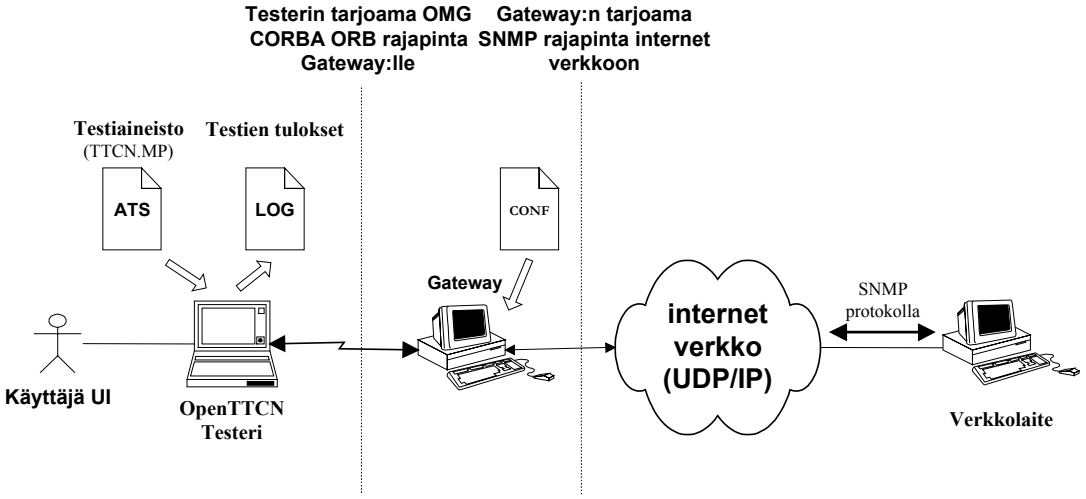
Verkonhallintajärjestelmässä on kaksi pääelementtiä, joita voidaan testata protokollatesterillä: verkonhallintapalvelin ja verkkolaite. Tässä työssä otetaan mukaan vielä kolmas testattava elementti, Gateway implementaatio (*kts. Kuva 14*). Verkonhallintapalvelimen ja verkkolaitteen testaaminen on käsitelty pääosin teoreettiselta kannalta ISO 9646-2 standardin käsitteitä soveltaen. Käytännön työssä keskityttiin Gateway implementaation testaamiseen.

Verkkolaite muodostaa testattavan systeemin (SUT) ja varsinainen testattava implementaatio (IUT) on verkkolaitteen 'sisällä' pyörivä Agent -protokolla (*kts. Kuva 2*). Agent -protokollan tehtävät on kuvattu tarkemmin kappaleessa *3.1.1 Verkonhallintajärjestelmät*.

Agent -protokollalla on kaksi rajapintaa. Ensimmäinen on SNMP -rajapinta, jolla Agent keskustelee verkonhallintapalvelimen ja testaustilanteessa Gateway:n välityksellä OpenTTCN protokollatesterin kanssa. Toinen rajapinta on Agent -protokollan ja MIB -tietokannan välinen, verkkolaitteen sisäinen rajapinta, jota Agent käyttää operoidessaan MIB -tietokantaa verkonhallintapalvelimelta tai testeriltä vastaanottamiensa viestien perusteella. Luonnollisesti tämän rajapinnan toteutus vaihtelee sen mukaan miten loog· nen MIB -tietokanta on toteutettu ohjelmisto-/rautatasolla.

Tässä työssä on toteutettu yksi testitapaus, jolla voidaan testata esimerkin omaisesti Linux käyttöjärjestelmään saatavia ilmaisohjelmia (*/usr/sbin/snmpd* ja */usr/bin/snmptrap*). Näistä ensimmäinen toteuttaa verkkolaitteiden yleisessä rakenne-

mallissa (*kts. Kuva 2*) määritellyn Agent -protokollan toiminnallisuuden. Jälkimmäistä käytetään generoimaan SNMP Trap -viestejä. Kyseisestä testitapauksesta ja mainituista ilmaisohjelmista kerrotaan tarkemmin kappaleessa *4.3 Testitapaukset*.



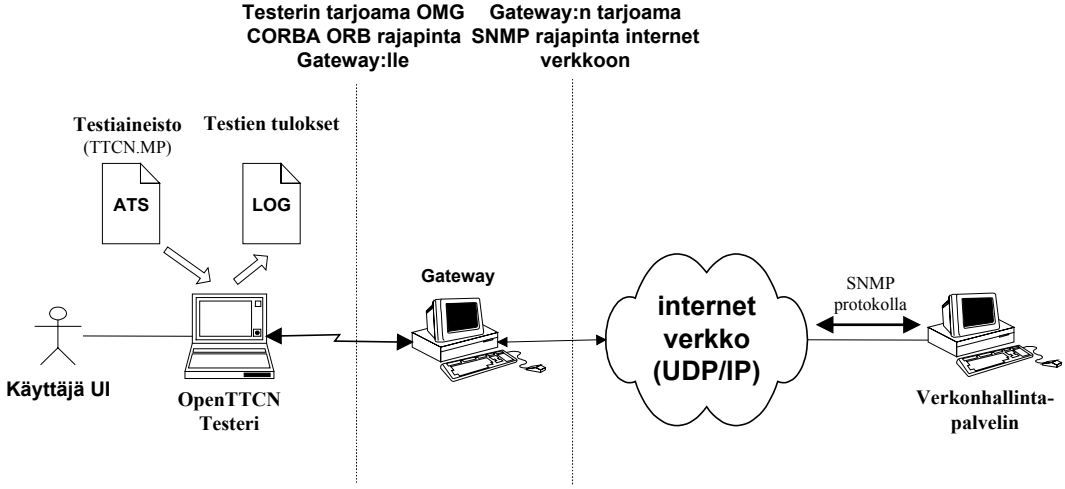
Kuva 17: Verkkolaitteen yleinen testausarkkitehtuuri

Verkkolaitteen testausarkkitehtuuri rakentuu niin, että OpenTTCN testeri kytketään ORB -liityntärajapintaa käyttäen Gateway:hin, joka hoitaa testerin lähettämien viestien välityksen internetverkon yli verkkolaitteelle ja verkkolaitteiden lähettämien viestien välityksen testerille. Agent -protokollan testaus on esitetty tarkemmin kappaleessa *4.2 ISO 9646-2 käsitteet määriteltäessä testausarkkitehtuuria*, jossa esitellään Agent -protokollan testaus ISO 9646-2 standardissa esitetyin käsittein.

Verkonhallintapalvelin muodostaa testattavan systeemin (SUT). Varsinainen testattava implementaatio muodostuu palvelimen ’sisällä’ pyörivistä Manager -protokollasta ja hallintaohjelmista (*kts. Kuva 2*). Manager -protokollan tehtävät on kuvattu tarkemmin kappaleessa *3.1.1 Verkonhallintajärjestelmät*.

Verkonhallintapalvelimia ajatellen on mahdollista muodostaa kaksi erillistä testausskenaariota: ensimmäisessä skenaariossa testataan yksistään Manager -protokollaa, toisessa testataan Manager -protokollan ja hallintaohjelmiston toimintaa yhdessä. Jos testataan hallintaohjelmiston toimintaa, tulee myös Manager -protokollan toiminta testata, koska se tarjoaa palveluitaan hallintaohjelmistolle, joiden toiminta on testattava ennen tai yhtäaikaa niiden käyttöön oton kanssa.

Manager -protokollalla on kaksi rajapintaa, joista toinen on Manager -protokollan ja verkon välinen SNMP -rajapinta, jota Manager käyttää keskustellessaan muiden verkossa olevien laitteiden kanssa. Hallintaohjelmistojen ja Manager:in välinen rajapinta on verkonhallintapalvelimen sisäinen rajapinta, koska verkonhallintaohjelmistot eivät ole standardoituja. Tällöin myöskään niiden rajapintoja ei ole sovittu ennalta vaan jokainen hallintaohjelmisto ja Manager -protokollan 'yläpuolinen' osuus omaa erilaisen rajapinnan riippuen hallintaohjelmistojen toteuttajasta. Tällaiset valmistajakohtaiset rajapinnat aiheuttavat sen, että toteutettaessa Gateway ratkaisuja testerin ja testattavan implementaation väliin, joudutaan jokaisen valmistajan rajapinnalle mahdollisesti toteuttamaan erilainen Gateway ratkaisu.



Kuva 18: Verkonhallintapalvelimen yleinen testausarkkitehtuuri

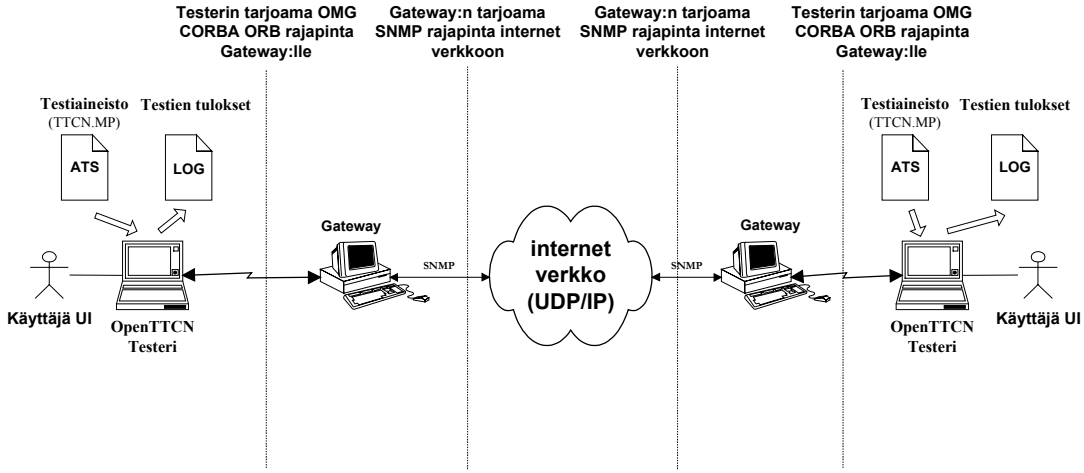
Verkonhallintapalvelimen testausarkkitehtuuri rakentuu vastaavalla tavalla kuin verkkolaitteen testauksessa eli OpenTTCN testeri kytketään Gateway:hin, joka hoitaa viestien välityksen testeriltä testattavalle systeemille ja päinvastoin. Tarkemmin Manager -protokollan testaus on esitetty kappaleessa *4.2 ISO 9646-2 käsitteet määriteltäessä tes-*

tausarkkitehtuuria, jossa esitellään Manager -protokollan ja hallintaohjelmistojen testaus ISO 9646-2 standardissa esitetyin käsittein.

Tässä työssä on toteutettu yksi testitapaus, jolla voidaan testata esimerkin omaisesti Linux:ssa pyörivä */usr/sbin/snmptrap* ilmaisohjelmaa. Tämä ohjelma toteuttaa osin verkonhallintapalvelimen Manager -protokollan toiminnallisuuden toimimalla demo ohjelmana, joka ottaa vastaan verkkolaitteiden lähettämiä SNMP Trap -viestejä. Kyseisestä testitapauksesta ja mainitusta ilmaisohjelmasta kerrotaan tarkemmin kappaleessa *3.3 Testitapaukset*.

Vaikka Gateway ei kuulu verkonhallintajärjestelmän komponentteihin, on sen toteutus ollut oleellinen osa tätä erikoistyötä. Työssä keskitytäänkin pääasiassa testaamaan Gateway toteutusta johtuen mm. seuraavista syistä:

- Jotta verkonhallintajärjestelmän komponentteja (verkkolaitteet ja verkonhallintapalvelimet) voitaisiin testata, täytyy ensin varmentaa Gateway:n toiminnallisuus.
- Gateway:n testaaminen antaa pitkälti käsityksen ISO 9646 metodologian ja OpenTTCN protokollatesterin soveltuvuudesta SNMP -verkonhallintajärjestelmien testaamiseen internetverkoissa.
- Erikoistyölle allokoitujen resurssien määrä on rajallinen ja työn tarkoitus on olla enemmän tutkimus kuin sovellusperäinen, jolloin tietyn verkonhallintasovelluksen tai verkkolaitteen Agent -protokollan testaaminen on turhan kapea-alainen projektin tulos.



Kuva 19: Gateway:n yleinen testausarkkitehtuuri

Gateway:ta testatessa muodostettiin testausarkkitehtuuri, jossa OpenTTCN testeri kytketään kahteen Gateway toteutukseen, jotka keskustelevat keskenään SNMP -protokollaviestejä käyttäen testerin ohjeiden mukaisesti. Määritelty testausarkkitehtuuri pohjautuu siihen seikkaan, että yksi Gateway toteuttaa yhden abstraktissa testiaineistossa (ATS) määritellyn PCO:n. Tällaisella testausarkkitehtuurilla päästään verifioimaan Gateway:n toiminnallisuuksia mm. koodaus/dekoodaus prosessien oikeellisuutta, viestien lähettämistä ja vastaanottoa, Gateway:n ja testerin välistä kommunikointi jne.

Tässä työssä on Gatewayn testaamiseksi toteutettu kymmenen testitapausta, jotka esitellään tarkemmin kappaleessa *4.3 Testitapaukset*. Tarkemmin Gateway:n testaus on esitetty kappaleessa *4.2 ISO 9646-2 käsitteet määriteltäessä testausarkkitehtuuria*, jossa esitellään Gateway:n testaus ISO 9646-2 standardissa esitetyin käsittein.

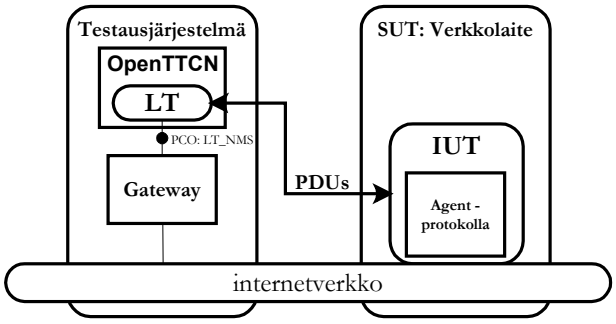
4.2 ISO 9646-2 standardin käsitteet määriteltäessä testausarkkitehtuuria

ISO 9646-2 standardi määrittelee käsitteen abstraktit testausmenetelmät (Abstract Test Methods). Tämän käsitteen tarkoituksena on kuvata yleisellä tasolla kuinka IUT:ta testataan testausjärjestelmää käyttäen (*kts. 3.1.3 Standardinmukaisuustestaus*).

Laadittaessa ISO 9646-2 käsitteiden mukaista mallia verkkolaitteen testauksesta, voidaan testausarkkitehtuuri esittää abstraktia etätestausmenetelmää käyttäen seuraavasti (*kts. Kuva 20*). Testausjärjestelmä koostuu OpenTTCN protokollatesteristä, joka muo-

dostaa alemman testerin (LT) toiminnallisuuden, sekä Gateway implementaatiosta. Gateway toteuttaa abstraktissa testiaineistossa (ATS), jota OpenTTCN tulkitsee, määritellyn PCO -toiminnallisuuden. Testattava systeemi (SUT) on verkkolaite, joka sisältää testattavan implementaation (IUT), joka on verkkolaitteen sisällä pyörivä SNMP Agent -protokolla. Viestien välittäminen testausjärjestelmän ja testattavan systeemin välillä tapahtuu internetverkon tarjoamia palveluita (UDP/IP) käyttäen.

Testaustapahtumat rakentuvat proseduurin ympärille, jossa LT lähettää PDU:ita IUT:lle ja vastaanottaa tämän lähettämiä PDU:ita. Lähettäessään PDU:ita IUT:lle PDU:n kulku tapahtuu todellisuudessa niin, että LT toimittaa PDU:n Gateway:lle, joka muokkaa LT:n lähettämästä PDU:sta SNMP -viestin, jonka se lähettää alla olevan internetverkon palveluita käyttäen IUT:lle. Tämän jälkeen IUT (Agent -protokolla) käsittelee vastaanottamansa viestin ja lähettää toimintansa mukaisen vastausviestin, jonka Gateway vastaanottaa. Gateway muuttaa vastaanottamansa viestin PDU -muotoon ja lähettää edelleen LT:lle. OpenTTCN protokollatesteri analysoi vastaanottamansa PDU:n ja vertaa sitä testiaineistossa määriteltyyn PDU:hun, jota odotetaan saapuvaksi. Mikäli vastaanotettu PDU on odotetun kaltainen, voidaan IUT:n sanoa olevan tämän testisekvenssin osalta toimiva. Jos PDU poikkeaa odotetusta, voidaan IUT -toiminnan todeta olevan testisekvenssin toiminnallisuuden osalta virheellinen.



Kuva 20: Verkkolaitteen etätestausmenetelmä

Verkonhallintapalvelimen ISO 9646-2 standardinmukainen testausmalli voidaan esittää abstraktia hajautettua testausmenetelmää käyttäen seuraavasti (*kts. Kuva 21*). Testausjärjestelmä rakentuu paljolti edeltä esitetyn verkkolaite testausmallin tapaan. Eli testattava systeemi koostuu OpenTTCN protokollatesteristä, joka muodostaa alemman testerin (LT) toiminnallisuuden ja Gateway:stä. Testattava systeemi on verkonhallintapalve-

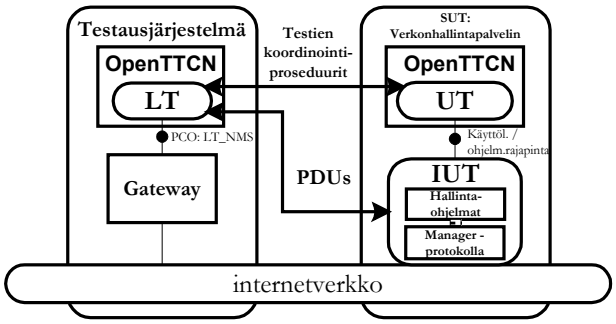
lin, joka sisältää testattavan implementaation (IUT). IUT koostuu Manager -protokollasta sekä hallintaohjelmista.

Lisäksi testattava systeemi sisältää ylemmän testerin (UT), joka on OpenTTCN muodostama testeri toiminnallisuus. UT:n ja IUT:n välillä oleva PCO voi olla esimerkiksi hallintaohjelmien käyttöliittymä tai verkonhallintaohjelmien standardoimaton ohjelmointirajapinta. Viestien välittäminen testausjärjestelmän ja testattavan systeemin välillä tapahtuu internetverkon tarjoamia palveluita (UDP/IP) käyttäen .

Testitapahtumat rakentuvat siten, että LT lähettää IUT:lle PDU:ita ja vastaanottaa tämän lähettämiä vaste PDU:ita. Gateway:n tehtävänä on tässäkin mallissa toimia koodaus-/dekoodausohjelmana PDU:iden ja SNMP -viestien välillä. Mikäli LT:n vastaanottama PDU vastaa testiaineistossa määriteltyä PDU:ta, jota odotetaan saapuvaksi, voidaan IUT:n todeta toimivan tämän testisekvenssin osalta.

Tilanne poikkeaa verkkolaitteen testausmenetelmä mallista siten, että nyt mukana on myös UT -komponentti, joka tarkkailee IUT:n yläpuolista rajapintaa ja samalla keskustelee LT:n kanssa mahdollistaen LT:n ja UT:n väliset koordinointiproseduurit mm. ajastus tarkoituksessa. Yleensä verkonhallintajärjestelmien testausta ajatellen, UT:n ja IUT:n välinen PCO on käyttöliittymärajapinta. Tällöin käyttäjän vastuulla on koordinointiproseduurien hoitaminen LT:n ja UT:n välillä (esimerkki testitapaus, TC_VSGMI_002, *kts. 4.3 Testitapaukset*).

Käyttöliittymärajapinta on yleinen varsinkin, jos testataan jo valmista verkonhallintaohjelmistoa. Jos standardinmukaisuustestausta sovelletaan tuotekehitysvaiheessa olevien verkonhallintaohjelmistojen testaukseen niin tällöin on mahdollista toteuttaa erillinen rajapintaliittymä IUT:n ja UT:n välille, jolla mahdollistetaan automatisoitujen koordinointiproseduurien käyttö. Erillisen rajapinnan toteuttaminen vaatii luonnollisestikin ohjelmointityötä IUT:n ylärajapinnan toteutuksessa eli toisin sanoen verkonhallintaohjelmistoon täytyy ohjelmoida uusi rajapinta testausta ajatellen.

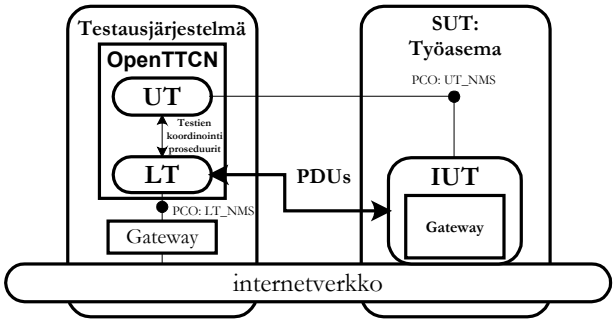


Kuva 21: Verkonhallintapalvelimen hajautettu testausmenetelmä

Gateway:n testaus voidaan esittää käyttäen abstraktia paikallista testausmenetelmää (*kts. Kuva 22*). Testausjärjestelmä rakentuu niin, että OpenTTCN testeri muodostaa toiminnallisuuden alemmalle (LT) ja ylemmälle testerille (UT). Lisäksi testausjärjestelmään kuuluu Gateway toteutus, joka varsinaisesti toteuttaa abstraktissa testiaineistossa (ATS) määrittelyn PCO:n. Nyt tulee huomata, että LT ja UT molemmat sijaitsevat testausjärjestelmän sisällä. Tällöin niiden väliset koordinointiproseduurit ovat suoraan testerin ohjaamia. Jotta tällainen testausmenetelmä olisi mahdollinen, on IUT:n ylärajapinnan oltava testerin tiedossa. Testattava systeemi (SUT) on Gateway implementaatio, jota ajetaan tavallisessa työasemaympäristössä. Gateway:ta testatessa UT:n ja IUT:n välinen PCO muodostuu OpenTTCN:n tarjoamasta ORB -liityntärajapinnasta. Viestien välittäminen testausjärjestelmän ja testattavan systeemin välillä tapahtuu internetverkon tarjoamia palveluita (UDP/IP) käyttämällä.

Testiproseduuri etenee seuraavasti. LT välittää PCO:n kautta testausjärjestelmässä olevalle Gateway:lle PDU:n, jonka tämä koodaa SNMP -viestiksi ja lähettää alla olevan internetverkon palveluita käyttäen testattavassa systeemissä sijaitsevalle Gateway:lle. Tämä vastaanottaa kyseisen viestin, dekoodaa sen ja välittää PCO:n kautta UT:lle. Koska UT tuntee IUT:n tarjoaman rajapinnan, voi se tätä rajapintaa käyttämällä lukea IUT:n testaussysteemiltä vastaanottamasta SNMP -viestistä dekoodaaman PDU:n.

Luettuaan PDU:n UT voi verrata sitä LT:n lähettämään PDU:hun ja näin verifioida Gateway:n dekoodaus/koodaus prosessien oikeellisuutta. Jos lähetetyt ja vastaanotetut PDU:ta täsmäävät, voidaan todeta Gateway koodausprosessien toimivan vastaavien viestien koodauksessa/dekoodauksessa. Samalla testataan Gateway:n kykyä vastaanottaa ja lähettää erilaisia SNMP -viestejä. Gateway:n toiminnan oikeellisuuden lisäksi tällä testausmallilla voidaan samalla testata OpenTTCN testerin tarjoaman ORB -liityntärajapinnan toiminnallisuutta ja soveltuvuutta SNMP -pohjaisten verkonhallintajärjestelmien testaukseen.



Kuva 22: Gateway:n paikallinen testausmenetelmä

4.3 Testitapaukset

4.3.1 Yleistä

Tässä erikoistyössä toteutetut testitapaukset (13 kappaletta) keskittyvät pääasiassa Gateway implementaation testaamiseen. Samaisen testauksen yhteydessä voidaan sivutuotteena havainnoida OpenTTCN protokollatesterin Gateway toteutukselle tarjoaman ORB -liityntärajapinnan soveltuvuus SNMP -pohjaisten verkonhallintajärjestelmien testaamiseen. Testauksen yhteydessä arvioidaan luonnollisesti ISO 9646 testausmetodologian ja tämän toteuttavan OpenTTCN ohjelmiston soveltamista verkonhallintajärjestelmien testaamiseen.

Implementoidut testit kuuluvat Basic Interconnection Tests (BIT) testiryhmään (*kts. 3.1.3 Standardinmukaisuustestaus*) ja niiden tarkoituksena on varmistaa, että toteutettu Gateway noudattaa toimintansa osalta SNMP -protokollan määrittelyitä. Tämä tarkoittaa mm. sitä, että testeillä tarkistetaan tukeeko Gateway implementaatio standardinmu-

kaisia SNMP -viestejä (*kts. 3.1.3 Verkonhallintajärjestelmät*) niin viestien rakenteen kuin koon osalta. Näillä testeillä varmistetaan myös Gateway:n ja OpenTTCN testerin välinen kytkentä eli toisinsanoin ORB -liitynnän toiminnallisuus.

ISO 9646-2 käsitteitä käyttäen voidaan testausarkkitehtuuri, joissa näitä testejä pääasiassa ajetaan, esittää kuvan 22 avulla. Eli testatessa Gateway:ta käytetään paikallista testausmenetelmää, jonka mukaisesti sekä LT ja UT sijaitsevat loogisesti samassa testausjärjestelmässä. Jos mietitään samaista testausarkkitehtuuria enemmän verkkotasolla, esittää kuva 19 mallin kyseisestä tilanteesta. Muutamia testitapauksia on toteutettu myös hajautettua testausmenetelmää (*kts. Kuva 21*) sekä etätestausmenetelmää (*kts. Kuva 20*) mallina käyttäen.

Työssä toteutettiin kolmetoista kappaletta erilaisia perustestitapauksia (BIT). Testitapaukset jakaantuvat neljään eri ryhmään:

- SNMP_GW_TEST/VALID_SNMP_GW_MESSAGES
- SNMP_GW_TEST/VALID_SNMP_GW_MESSAGE_INTERACTION
- SNMP_GW_TEST/VALID_SNMP_GW_MESSAGE_DATATYPES
- SNMP_GW_TEST/INVALID_SNMP_GW_MESSAGE_DATATYPES

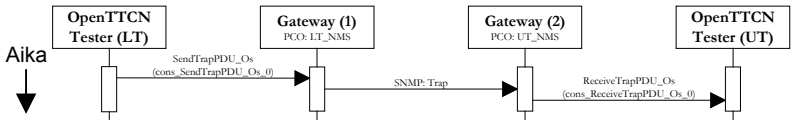
4.3.2 SNMP_GW_TEST/VALID_SNMP_GW_MESSAGES - ryhmän testitapaukset

Tähän testiryhmään kuuluu viisi testitapausta, joiden tarkoituksena on varmistaa, että Gateway tukee kaikkia standardinmukaisia SNMPv1 -viestejä.

TC_VSGM_001

LT lähettää Gateway(1) toteutukselle, joka sijaitsee testausjärjestelmässä (*kts. Kuva 22*), Trap PDU:n. Kyseinen PDU sisältää yhden Octet String tyyppisen kentän PDU -viestin variable-bindings osuudessa (*kts. kappale 3.1.1 Verkonhallintajärjestelmät*). Testerin lähettämän PDU:n tarkoituksena on mallintaa verkkolaitteen lähettämää SNMP Trap -viestiä. Gateway(1) muuttaa vastaanottamansa PDU:n SNMP -viestin muotoon ja lähettää Gateway(2) toteutukselle, joka on testattava implementaatio (*kts. Kuva 22*). Gateway(2) muuttaa vastaanottamansa SNMP -viestin takaisin PDU -muotoon ja lä-

hettää UT:lle. OpenTTCN vertaa UT:n vastaanottamaan PDU:ta LT:n lähettämään PDU:hun. Mikäli viestit täsmäävät, testi on mennyt läpi. Tämä tarkoittaa, että Gateway toteutuksien dekoodaus/koodaus prosessit ovat toimivia toistensa vastakohtia ja viestien

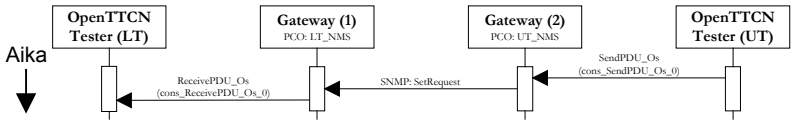


lähettäminen ja vastaanottaminen onnistuu, kun tarkastellaan SNMP Trap -viestien toiminnallisuutta.

Kuva 23: TC_VSGM_001 testin Sequence -diagrammi

TC_VSGM_002

UT lähettää Gateway(2) toteutukselle, joka on testattava implementaatio (*kts. Kuva 22*), SetRequest PDU:n. Gateway(2) välittää SNMP -viestiksi koodatun PDU:n Gateway(1) toteutuksella, joka dekoodaa saadun viestin ja välittää PDU:n edelleen LT:lle. Vastaavalla tavalla kuin TC_VSGM_001 testitapauksessa onnistuneen testin tulos on varmuus Gateway:n koodaus/dekoodaus prosessien toimivuudesta sekä viestien lähetyksestä ja vastaanotosta SNMP SetRequest -viesteille.



Kuva 24: TC_VSGM_002 testin Sequence –diagrammi

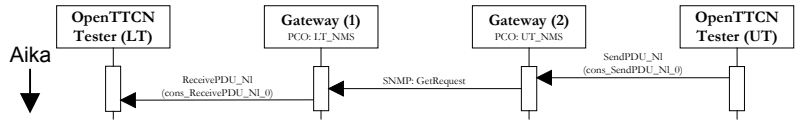
TC_VSGM_003

LT lähettää Gateway(1) toteutukselle, joka sijaitsee testausjärjestelmässä, GetResponse PDU:n. Gateway(1) välittää SNMP -viestiksi koodatun PDU:n Gateway(2) toteutuksella, joka dekoodaa saadun viestin ja välittää PDU:n edelleen UT:lle. Vastaavalla tavalla kuin TC_VSGM_002 testitapauksessa onnistuneen testin tulos on varmuus koodaus/dekoodaus prosessien toimivuudesta sekä viestien lähetyksestä ja vastaanotosta SNMP GetResponse -viesteille.

Kuva 25: TC_VSGM_003 testin Sequence -diagrammi

TC_VSGM_004

UT lähettää Gateway(2) toteutukselle, joka on testattava implementaatio, GetRequest PDU:n. Gateway(2) välittää SNMP -viestiksi koodatun PDU:n Gateway(1) toteutuksella, joka dekoodaa saadun viestin ja välittää PDU:n edelleen LT:lle. Vastaavalla tavalla kuin TC_VSGM_003 testitapauksessa onnistuneen testin tulos on varmuus koodaus/dekoodaus prosessien toimivuudesta sekä viestien lähetyksestä ja vastaanotosta SNMP GetRequest -viesteille.

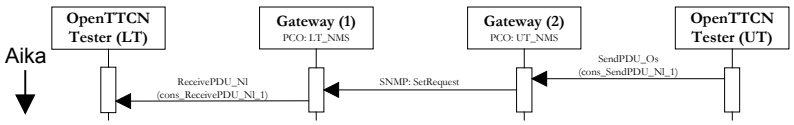


Kuva 26: TC_VSGM_004 testin Sequence -diagrammi

TC_VSGM_005

UT lähettää Gateway(2) toteutukselle, joka on testattava implementaatio, GetNextRequest PDU:n. Gateway(2) välittää SNMP -viestiksi koodatun PDU:n Gateway(1) toteutuksella, joka dekoodaa saadun viestin ja välittää PDU:n edelleen LT:lle. Vastaavalla tavalla kuin TC_VSGM_004 testitapauksessa onnistuneen testin

tulos on varmuus Gateway:n koodaus/dekoodaus prosessien toimivuudesta sekä viestien lähetyksestä ja vastaanotosta SNMP GetNextRequest -viesteille.



Kuva 27: TC_VSGM_005 testin Sequence -diagrammi

4.3.3 SNMP_GW_TEST/VALID_SNMP_GW_MESSAGE_INTERACTION - ryhmän testitapaukset

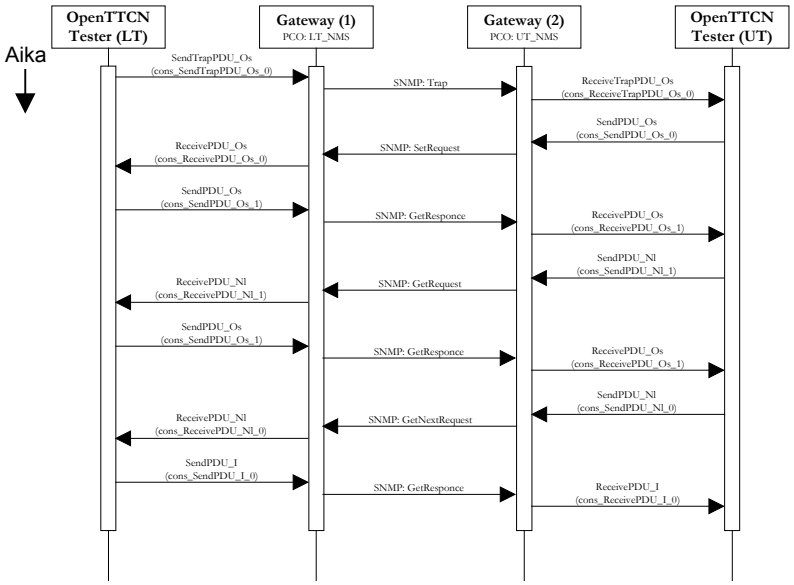
Tähän ryhmään kuuluu kaksi testitapausta, joista TC_VSGMI_001 noudattaa kuvan 22 mukaista paikallista testausmenetelmä mallia. Kyseisen testin tarkoituksena on testata Gateway toteutusta. TC_VSGM_002 poikkeaa muista tässä työssä toteutetuista testitapauksista, koska ne noudattaa kuvan 20 mukaista etätestausmenetelmä mallia, jonka mukaisesti testattavana kohteena ei ole Gateway toteutus vaan verkkolaitteen Agent -protokolla.

TC_VSGMI_001

Tämän testin ideana on testata Gateway:n toimintaa useamman viestisekvenssin ollessa kyseessä. Lisäksi testin ideana on lähettää kaikkia mahdollisia SNMPv1 -viestejä Gateway:n koodattavaksi/dekoodattavaksi noudattaen todellisia verkonhallintajärjestelmien viestisekvenssejä. Tämä tarkoittaa esimerkiksi sitä, että mikäli verkonhallintapalvelin lähettää verkkolaitteelle SetRequest viestin, odottaa se verkkolaitteen lähettävän vastaukseksi GetResponse viestin. Vaikka tässä testissä noudatetaan järjestelmällisyyttä viestien lähettämisessä ja vastaanottamisessa Gateway:n toiminnallisuutta ei ole mitenkään sidottu tiettyihin viestien lähetys-/vastaanottojärjestyksiin.

Viestien lähettäminen alkaa sillä, että LT lähettää Gateway(1) toteutukselle Trap PDU:n, jonka Gateway(1) koodaa ja välittää edelleen Gateway(2) toteutukselle. Tämä dekoodaa vastaanottamansa SNMP Trap -viestin ja lähettää PDU:n UT:lle. Mikäli UT odottaa kyseistä PDU:ta saapuvaksi, on testitapaus tämän yhden viestin lähettämisen ja

vastaanottamisen osalta mennyt läpi. Tämän jälkeen UT lähettää Gateway(2) toteutukselle SNMP SetRequest -viestin, jonka Gateway(2) koodaa ja välittää edelleen Gateway(1) toteutukselle jne. Viestien vaihtoa tapahtuu UT:n ja LT:n välillä kunnes kaikki viestit on saatu onnistuneesti lähetettyä ja vastaanotettua. Tällöin testi tulkitaan onnistuneeksi ja se saa ISO 9646 standardinmukaisen ’tuomion’ - PASS, joka ilmaisee testin onnistumisen. Mikäli jonkin viestin lähetys/vastaanotto tai koodaus/dekoodaus ei onnistu, saa testi ’tuomion’ - FAIL, joka ilmaisee testin epäonnistumisen.



Kuva 28: TC_VSGMI_001 testin Sequence -diagrammi

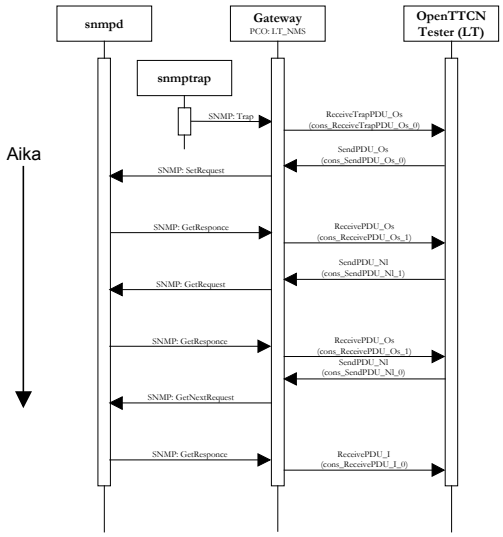
TC_VSGMI_002

Tämä testi eroaa muista toteutetuista testeistä, koska siinä testataan Gateway:n sijaan todellista verkkolaitteen Agent -protokolla toiminnallisuudet täyttävää sekä Trap viestien generointia varten tarkoitettua ohjelmaa (*kts. 4.1.3 Testattavat elementit*). Kyseessä ovat Linux käyttöjärjestelmään saatavat ilmaisohjelmat */usr/sbin/snmpd* ja */usr/bin/snmptrap*. Testi erottuu muista toteutuneista testeistä myös siinä, että sen mallintamiseen käytetään muista poiketen etätestausmenetelmää (*kts. Kuva 20*).

Testin ideana on testata *snmpd* ohjelman (IUT) toimintaa siten, että testausjärjestelmä lähettää IUT:lle kaikkia mahdollisia SNMPv1 -viestejä, joita verkonhallintapalvelin todellisuudessa voi lähettää verkkolaitteelle. Vastaavasti kuin TC_VSGMI_001 testita-

pauksessa, viestisekvenssit on muodostettu niin, että LT lähettää Gateway:lle SNMP -viestejä, joihin se odottaa vastauksia *snmpd* ohjelmalta.

Testi alkaa sillä, että käyttäjä käyttää hajautetusta testausmenetelmästä tuttua käyttöliittymä PCO -määrittelyä ja käyttää *snmptrap* työkalua generoimaan SNMP Trap -viestin, jonka käyttäjä lähettää Gateway toteutukselle. Gateway dekoodaa SNMP Trap -viestin Trap PDU:ksi ja lähettää eteenpäin LT:lle. Mikäli LT:n vastaanottama Trap -viesti on sellainen, jota se odottaa, on testi mennyt tämän vastaanotetun viestin osalta läpi. Tällöin testaus voi jatkua niin, että LT puolestaan lähettää PDU:ita Gateway:n välityksellä testattavalle implementaatiolle (*snmpd*). *snmpd* ohjelma käsittelee vastaanottamansa viestit ja lähettää Agent -protokollatoiminnallisuuden mukaiset vastaukset, jotka OpenTTCN testeri aikanaan verifioi ja näiden pohjalta muodostaa testaustuloksen.



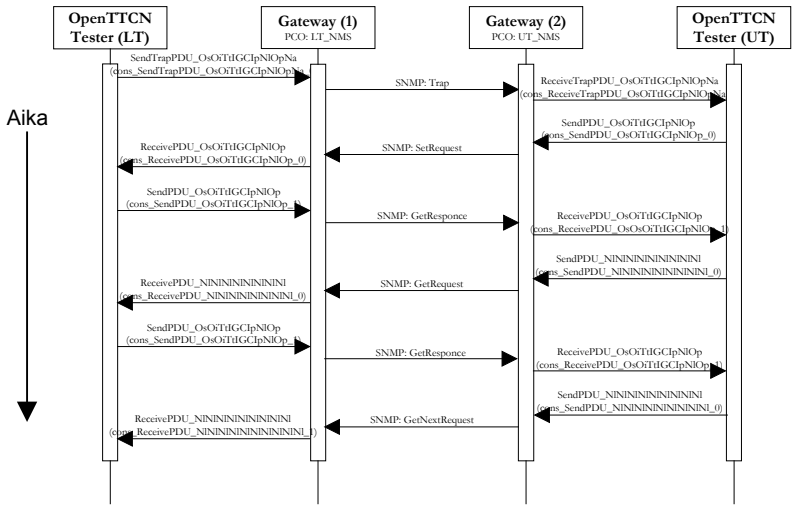
Kuva 29: TC_VSGMI_002 testin Sequence -diagrammi

4.3.4 SNMP_GW_TEST/VALID_SNMP_GW_MESSAGE_DATATYPES

Tähän ryhmään kuuluu yksi testitapaus, TC_VSGMD_001, joka noudattaa kuvan 22 mukaista paikallista testausmenetelmä mallia.

TC_VSGMD_001

TC_VSGMD_001 testin tarkoitus on testata Gateway:n kykyä käsitellä SNMP -protokollan tukemia datatyyppejä. Testi rakentuu vastaavalla tavalla kuin aikaisemmin esitetty TC_VSGMI_001 testi mutta sillä eroavaisuudella, että nyt jokaisessa SNMP -viestissä on mukana tämän viestin variable-bindings osuudessa kaikki mahdolliset SNMP -protokollan tukemat datatyypit (*kts. 3.1.1 Verkonhallintajärjestelmät*). Testi menee hyväksytysti läpi mikäli Gateway pystyy käsittelemään kaikkia vastaanotettuja ja lähetettyjä viestejä oikein.



Kuva 30: TC_VSGMD_001 testin Sequence -diagrammi

4.3.5 SNMP_GW_TEST/INVALID_SNMP_GW_MESSAGE_DATATYPES

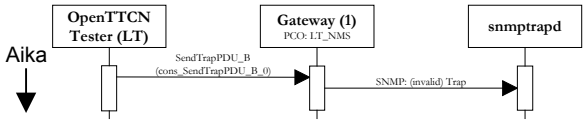
Tähän ryhmään kuuluu viisi testitapausta. Nämä testit poikkeavat muista toteutetuista testeistä, koska ne perustuvat kuvassa 21 esitettyyn abstraktiin hajautettu testausmenetelmä malliin. Testillä pyritään testaamaan esimerkinomaisesta Linux käyttöjärjestelmän mukana tulevaa ilmaisohjelmaa *snmptrapd*, jonka pääasiallinen tehtävä on ottaa vastaan

verkkolaitteiden lähettämiä SNMP Trap -viestejä. Eli toisin sanoen *snmptrapd* ohjelma toteuttaa SNMP Manager -protokollan toiminnallisuuksia.

Lisäksi testit poikkeavat muista toteutetuista testeissä myös siinä, että testerin lähettämät viestit ovat invalideja eli ne sisältävät datatyyppejä, joita SNMP -protokolla ei tue. *snmptrapd* ohjelman testauksen ohella saadaan kokemuksia Gateway:n käyttäytymisestä tilanteessa, jossa se joutuu lähettämään/vastaanottamaan ei standardinmukaisia SNMP -datatyyppejä

TC_ISGMD_001

Testeri lähettää Gateway(1) toteutusta käyttäen IUT:lle (*snmptrapd* ohjelma) invalidin Trap -viestin. Viestin sisältää Boolean kentän, joka ei kuulu SNMPv1 -määrittelyihin. Koska testi noudattaa kuvan 21 mukaista hajautettu testausmenetelmä mallia, voidaan *snmptrapd* ohjelmaa tarkkailla käyttäen IUT:n yläpuolelle sijaitsevaa PCO:ta, joka on *snmptrapd* ohjelman käyttöliittymä.

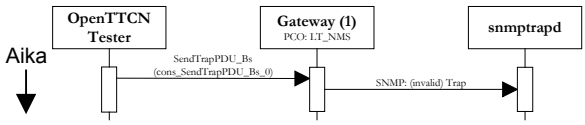


Kuva 31: TC_ISGMD_001 testin Sequence -diagrammi

Testi menee hyväksytysti läpi mikäli Gateway ja *snmptrapd* huomaavat virheellisen tyypin ja ilmoittavat tästä käyttäjälle asiaankuuluvalla ilmoituksella.

TC_ISGMD_002

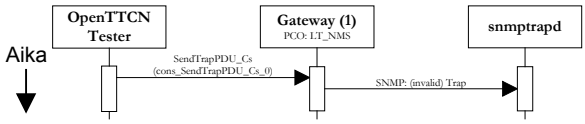
Vastaavanlainen testi kuin TC_ISGMD_001, paitsi että nyt viestin mukana lähetetään BitString kenttä, joka ei kuulu SNMPv1 -määrittelyihin.



Kuva 32: TC_ISGMD_002 testin Sequence -diagrammi

TC_ISGMD_003

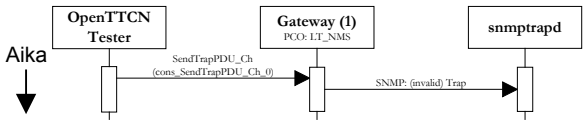
Vastaavanlainen testi kuin TC_ISGMD_001, poikkeuksena, että nyt viestissä lähetetään Character String kenttä, joka ei kuulu SNMPv1 -määrittelyihin.



Kuva 33: TC_ISGMD_003 testin Sequence -diagrammi

TC_ISGMD_004

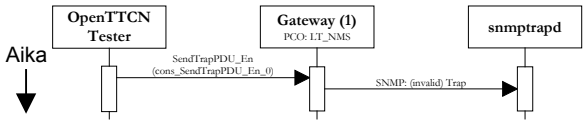
Vastaavanlainen testi kuin TC_ISGMD_001, paitsi että viestissä lähetetään Choice kenttä, joka ei kuulu SNMPv1 -määrittelyihin.



Kuva 34: TC_ISGMD_004 testin Sequence -diagrammi

TC_ISGMD_005

Vastaavanlainen testi kuin TC_ISGMD_001, paitsi että nyt viestissä lähetetään Enumerated kenttä, joka ei kuulu SNMPv1 -määrittelyihin.



Kuva 35: TC_ISGMD_005 testin Sequence -diagrammi

5 ISO 9646:N ja OPENTTCN TESTERIN SOVELTUMINEN VERKONHALLINTAJÄRJESTELMIEN TESTAAMISEEN

5.1 Yleistä

Työtä suorittaessa kiinnitettiin huomiota seuraavassa kappaleessa esitettyihin seikkoihin OpenTTCN protokollatesterin ja ISO 9646 standardin soveltuvuudesta SNMP -pohjaisten verkonhallintajärjestelmien testaamiseen.

5.2 ISO 9646 standardi verkonhallintajärjestelmien testauksessa

+ Verkonhallintajärjestelmien (verkkolaitteet, verkonhallintapalvelimet) testausarkkitehtuurimalli mukautuu luonnollisesti ISO 9646 standardijoukon määrittelemiin metodeihin ja käsitteisiin (*kts. 4.1.3 Testattavat elementit*).
+ ISO 9646 mahdollistaa standardoitujen MIB -objektiryhmien standardinmukaisuustestauksen.
+ ISO 9646 metodologia mahdollistaa automatisoidun testauksen.
+ TTCN -kielen ASN.1 osuus mahdollistaa SNMP -protokollamäärittelyn siirtämisen suoraan standardeista abstraktiin testiaineistoon (ATS).
+ ASN.1 tukee suoraan kaikkia standardeissa määritettyjä SNMP -tyyppejä, koska SNMP -tyypit ovat ASN.1:n tyyppien osajoukko (*kts. 3.1.1 Verkonhallintajärjestelmät*).
+ Kaikki SNMP -viestit pystytään kuvaamaan suoraviivaisesti TTCN -kielellä käyttäen ASN.1 notaatiota.

– SNMP -verkonhallintaohjelmistot eivät ole standardoituja ja näin ollen eri valmistajien ohjelmistot vaativat mahdollisesti erilaiset testiaineistot.

5.3 OpenTTCN testeri verkonhallintajärjestelmien testauksessa

+ OpenTTCN noudattaa täysin ISO 9646 standardia mikä mahdollistaa sen, että verkonhallintajärjestelmien testausstrategia voidaan suunnitella käyttäen ISO 9646 standardin määrittelemiä käsitteitä niin ettei testeri aiheuta rajoitteita suunnittelulle.
+ OpenTTCN mahdollistaa nopean testiaineiston käyttöönoton. Tämä johtuu siitä, että testeri ei käännä abstrakteja testiaineistoja (ATS) erikseen suoritettaviksi testiaineistoiksi (ETS) vaan tulkkaa ne suoraan ajonaikana. Tämä mahdollistaa sen, että erilliset käännös/linkitysvaiheet voidaan jättää testausprosesseista pois.
+ Selkeä käyttöliittymä.
+ Koska OpenTTCN on geneerinen protokollatesteri, sen yhteyteen tarvitsee aina toteuttaa erillinen Gateway elementti, joka liittää testerin testattavaan systeemiin (*kts. 4.2 OpenTTCN protokollatesteri*). OpenTTCN mahdollistaa CORBA -rajapinnan kautta joustavan ja suhteellisen yksinkertaisen tavan liittää Gateway testerin yhteyteen ja näin nopeuttaa huomattavasti erilaisten testausjärjestelmien laadintaprosessia.

− SNMP -verkonhallintaohjelmistot eivät ole standardoituja ja näin ollen eri valmistajien ohjelmistot vaativat mahdollisesti erilaiset Gateway toteutukset UT -rajapinnan osalta.
− OpenTTCN:n yhteydessä ei ole erillistä graafista editoria, jolla abstraktit testiaineistot voitaisiin laatia. Tällä hetkellä testit tulee kirjoittaa käsin TTCN.MP muodossa, mikä on aluksi aikaa vievää ellei testien laatija itse toteuta esimerkiksi script toteutuksia, jotka generoivat parametrien perusteella valmiiksi sopivia TTCN -kielen komponentteja.
− OpenTTCN oli erikoistyötä tehdessä vielä testaus ja kehitysvaiheessa ja näin ollen se ei ollut täysin stabiili kaikin osin. Tilanne kehittyi työn edetessä ja loppuvaiheessa työtä, testeri oli toteutetun Gateway:n ja testitapauksien osalta vakaa. Todennäköisesti testerissä on vieläkin pieniä puutteita, jotka selvinnevät myöhempien käyttösovelluksien yhteydessä.

5.4 Johtopäätökset

5.4.1 Yleistä

Standardinmukaisuustestaus on jo pitkän aikaa ollut käytössä protokollien testaamisessa perinteisten televerkkojen puolella. ISO 9646 metodologian käyttö on tarjonnut keinot mm. ATM, SS7, GSM, ISDN, DECT jne. protokollien automatisoituun standardinmukaisuustestaukseen. Se on saavuttanut suuren suosion formaalina protokollien testausmenetelmänä ja sen tehokkuuteen luotetaan yleisesti eri ohjelmistovalmistajien keskuudessa.

Kun tarkastellaan internetprotokolliin pohjautuvia verkkoja, on tilanne kokonaan toinen. Siinä missä standardinmukaisuustestaus on tarjonnut televerkoille yhdenmukaisen tavan testata implementoitujen protokollien standardinmukaisuutta, ei internet maailmassa ole ollut yhtenäistä käytäntöä, joka olisi mahdollistanut protokollien standardinmukaisuustestauksen vastaavalla, formaalilla, tavalla. Testausjärjestelmät ovat olleet enemmänkin kunkin instanssin itse kehittämiä ratkaisuja, jotka yleensä eivät ole sopineet muiden tahojen käytettäväksi. Pääosin yhtenäisten testausmenetelmien puute internet maailmassa johtuu internet teknologioiden uutuudesta ja jatkuvasta muutosvauhdista. Testausmenetelmät eivät ole ehtineet vakiintua alati muuttuvassa teknologiakentässä. Lisäksi perinteiset yhteyspohjaiset televerkot ja pakettipohjaiset internetverkot poikkeavat toisistaan paljon periaatteellisella tasolla.

Kaikesta huolimatta ISO 9646 standardi eli standardinmukaisuustestaus on nostamassa päätään myös internet maailmassa ja sen soveltamista erilaisten internetprotokollien ja palveluiden testaamiseen tutkitaan tällä hetkellä kasvavassa määrin ympäri maailmaa. TKK:n TML -laboratorio on projekteissaan tutkinut standardinmukaisuustestauksen soveltamista mm. mobiilien internetprotokollien (mobiili-IP), CORBA -pohjaisten palvelukomponenttien, reititysprotokollan (OSPF) ja nyt tässä erikoistyössä SNMP -pohjaisten verkonhallintajärjestelmien, jotka toimivat UDP/IP protokolla pinon päällä, testaamiseen.

5.4.2 Tutkimustulokset

Tässä työssä tutkittiin standardinmukaisuustestauksen soveltuvuutta verkonhallintajärjestelmien testaamiseen käyttäen OpenTTCN testeriä, joka toteuttaa ISO 9646 standardin määritykset. Työn tarkoituksena oli kartoittaa niin ISO 9646 metodologian kuin OpenTTCN:n soveltuvuus.

Työssä tarkasteltiin standardinmukaisuustestauksen soveltuvuutta teknologianäkökulmasta eli tutkittiin miten ISO 9646 metodologia soveltuu menetelmineen ja käsitteineen SNMP -verkonhallintajärjestelmien testaukseen ja miten OpenTTCN protokollatesteri mahdollistaa testitapauksien kuvaamisen ISO 9646-3 standardin määrittelemää TTCN -kuvauskieltä käyttäen ja miten testeri on mahdollista liittää testattavaan ympäristöön.

Tutkimuksen tuloksena selvisi, että ISO 9646 tarjoaa varsin tehokkaat keinot määritellä mm. testausarkkitehtuurimalleja, joita käytetään varsinaisten verkonhallintajärjestelmien testausarkkitehtuurien suunnitteluun ja tätä kautta pohjana koko testaukselle. Esimerkiksi kappaleessa *4.2 ISO 9646-2 standardin käsitteet* on käytetty ISO 9646 standardin kuvaamia abstrakteja testausmenetelmäkäsitteitä määriteltäessä testausarkkitehtuurimalleja verkonhallintajärjestelmien testaamiseksi. Mallit sisältävät mm. PCO -käsitteen, jota käyttämällä voidaan helposti suunnitella arkkitehtuuritasolla liittymät testausjärjestelmän ja testattavan systeemin välillä. Verkonhallintajärjestelmien verkkokomponentit pystytään sijoittamaan suoraan standardinmukaisuustestauksen testausmenetelmämalleihin (*kts. 4.2 ISO 9646-2 standardin käsitteet määriteltäessä testausarkkitehtuuria*).

Ensimmäinen oleellinen asia tässä työssä oli ISO 9646 metodologian soveltuvuuden tutkiminen verkonhallintajärjestelmien testaamisessa. Toinen asia oli tutkia miten OpenTTCN testerin liittäminen onnistuu testattavaan implementaatioon (IUT), joka yleensä on työasemassa, joka on kiinni internetverkossa, pyörivä ohjelmisto. Ohjelmisto käyttää kommunikointiin UDP/IP -protokollapinojen päällä pyörivää SNMP -protokollaa.

Liittyminen SUT:iin tapahtuu testerin tarjoaman CORBA -rajapinnan kautta. CORBA mahdollistaa testerin liittämisen Gateway:n kautta helposti internetverkkoihin. Koska CORBA on teknologia, joka mahdollistaa hajautettujen ohjelmistokomponenttien operoinnin verkkojen yli, pystyttiin Gateway:n ja testerin välinen testausarkkitehtuuri suunnittelemaan varsin joustavasti. Gateway:n toteuttaminen verkonhallintajärjestelmien testaamiseksi oli varsin suoraviivainen toimenpide ja testerin tarjoaman CORBA -rajapinnan sovittaminen verkonhallintajärjestelmien SNMP -protokollapinoon onnistui varsin helposti.

Kolmas asia jota työssä tutkittiin oli OpenTTCN testerin soveltuminen testien suorittamiseen ja TTCN:n soveltuminen testiaineistojen laadintaan.

Standardit kuvaavat verkonhallintajärjestelmien sisältämien komponenttien kommunikointiin käyttämän SNMP -protokollan ASN.1 notaatiolla. Koska TTCN:n tukee ASN.1 määrittelyjä, on abstraktien testiaineistojen laatiminen verkonhallintajärjestelmien testaamiseksi varsin suoraviivaista. Esimerkiksi SNMP -protokollaviestien määrittely on mahdollista siirtää sellaisenaan standardeista testiaineistoihin.

OpenTTCN:n toiminta perustuu abstraktien testiaineistojen tulkitsemiseen ajonaikana ja näin ollen se sopii erittäin hyvin tuotekehityksen alla olevien tuotteiden/ohjelmistojen testaamiseen, esimerkiksi tässä erikoistyössä testeriä käytettiin koko Gateway:n kehitysvaiheen ajan sen testaamiseen. Testerin tulkkipohjaisuus mahdollistaa sen, että testit voidaan ajaa heti haluttaessa ilman, että tarvitsee käydä läpi erillisiä käännös/linkitysvaiheita, joissa abstrakteista testiaineistoista luotaisiin erikseen suoritettavia testiaineistoja. Lisäksi jatkuvaa testausprosessia nopeutti OpenTTCN:n suhteellisen yksinkertainen ja selkeä käyttöliittymä.

5.4.3 Yhteenveto ja tulevaisuuden skenaariot

Edellisessä kappaleessa esitettyjen tuloksien pohjalta voidaan todeta, että ISO 9646 standardi on sovellettavissa myös verkonhallintajärjestelmien testaamiseen internetverkoissa. Myös OpenTTCN testeri soveltuu teknologiansa puolesta hyvin verkonhallintajärjestelmien testaamiseen. Geneerinen testeri ja erilliset Gateway implementaatio mah-

dollistavat erilaisten testausjärjestelmien kehittämisen erilaisille hallintaohjelmistoille varsin nopeallakin aikavälillä.

Myös testiaineiston laatiminen verkonhallintaohjelmistoille on suoraviivainen prosessi ja TTCN:n käyttäminen mahdollistaa SNMP -standardimäärittelyjen siirtämisen suoraan testiaineistoon, joka selkeyttää huomattavasti testiaineistojen ja Gateway implementaation toteutusta.

Vaikka työn tulos ISO 9646 standardi ja OpenTTCN:n soveltamisesta verkonhallintajärjestelmien testaamiseen on pääasiassa näille teknologioille myönteinen, havaittiin työssä myös muutamia negatiivisia aspekteja. Yksi lienee se, että varsinaiset verkonhallintaohjelmistot eivät ole standardoituja ja näin ollen ohjelmistovalmistajat voivat hieman vieroksua käsitettä standardinmukaisuustestaus verkonhallintaohjelmistojen yhteydessä.

Vaikka testien laatiminen periaatteellisella tasolla teknologia näkökulmasta oli helpohkoa, oli varsinaisten testien kuvaaminen TTCN.MP tasolla suhteellisen hidasta. Varsinkin, kun testiaineiston koko kasvoi useaan tuhanteen riviin. Asianmukaisen graafisen editorin käyttö olisi helpottanut tilannetta suuresti. Työssä testattiin Da Vinci TTCN -editoria mutta siitä luovuttiin varsin nopeasti johtuen editorin epävakaudesta ja hitaudesta. Koska toteutettujen testitapauksien määrä oli suhteellisen pieni, päätettiin kaikki testit toteuttaa suoraan TTCN.MP formaatissa.

Erikoistyö keskittyi ISO 9646 standardin soveltuvuuden tutkimiseen verkkojenhallintajärjestelmien testaamisessa ja se keskittyi teknologialähtöisiin seikkoihin kuten testausjärjestelmän liittämiseen SUT:iin, testausmallien laatimiseen ISO 9646-2 käsitteitä käyttäen jne. Jos tämän työn ajatellaan olevan isomman kokonaisuuden ensimmäinen vaihe niin seuraavassa vaiheessa, mikäli sellainen tulee, mahdollisia tehtäväkokonaisuuksia saattaisivat olla:

- Olemassa olevan Gateway implementaation parantaminen
- Jonkin case tyyppisen esimerkkitapauksen etsiminen, johon voidaan soveltaa tässä työssä esiteltyjä teknologioita. Esimerkiksi jokin verkonhallintaohjel-

misto, jolle laaditaan testimateriaalit, jotka kattavat hallintaohjelmin toiminnallisuuden. Tällöin voidaan lopullisesti todeta ISO 9646 ja OpenTTCN:n lopullinen soveltuvuus verkonhallintaohjelmistojen testaamiseen. Samalla esimerkkitapaus toimii ’business case’ mallina OpenTTCN testerin markkinointia ajatellen.

6 AIKATAULU

Erikoistyön aihealue jaettiin alkuraportissa (esitettiin 31.01.2000) seuraaviin tehtäväkokonaisuuksiin (*kts. Kuva 36*), joita erikoistyössä pyrittiin noudattamaan ja toteuttamaan:

- Perehtyminen työn suorittamisessa tarvittaviin teknologioihin/työkaluihin (ISO 9646, X.290, TTCN, ASN.1, BER, SNMP, CORBA, OpenTTCN, Java, Javadoc jne.).
- Yksinkertaisen testausmallin ideoiminen
- Gateway:n toteutus:
  - Gateway:n ja OpenTTCN:n kytkeminen toisiinsa CORBA -rajapinnalla
  - SNMPv1 -enkooderi /dekooderi
  - Yksinkertainen liikenne OpenTTCN <---> Gateway <---> Gateway <---> OpenTTCN.
  - Yksinkertaisien testitapauksien ideointi ja toteutus
  - Monimutkaisempien testitapauksien ideointi ja toteutus
- Dokumentointi/koodin viimeistely sekä testausmallin modifioiminen ja viimeistely Gateway:n ja testitapausten toteutuksen aikana opittujen asioiden pohjalta.
- Erikoistyön raportit (3 kpl)

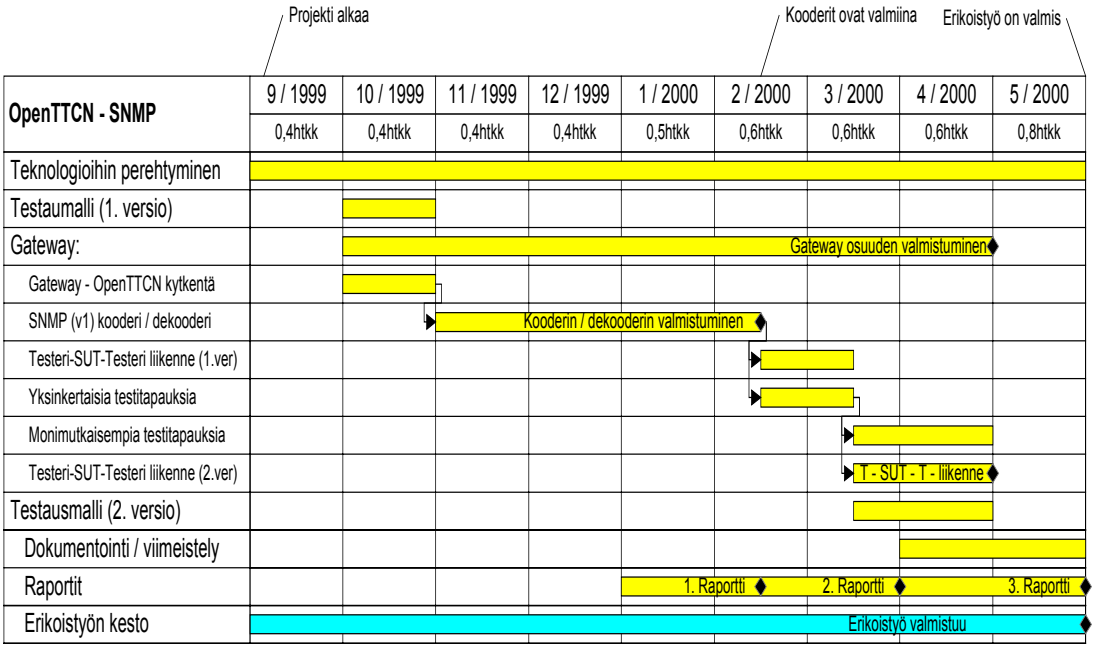
Alkuraportissa esitetyn alkuperäisen aikataulun mukaan erikoistyöprojektin tuli olla valmis 31.05.2000 mennessä. Pääpiirteissään projekti pysyi aikataulussa ja melkein kaikki yläpuolella esitetyt tehtäväkokonaisuudet saatiin vietyä loppuun suhteellisen pienellä aikatoleranssilla.

Suurin viivästyminen projektissa tapahtui erikoistyöraporttien laadinnassa. Alkuperäisen aikataulun mukaan viimeinen erikoistyöraportti, joka toimii samalla projektin loppuraporttina, tuli olla valmis 31.05.2000 mennessä, jonka jälkeen raportti toimitettaisiin IPMAN -projektin johtoryhmän evaluoitavaksi. Näin ollen kuvassa 36 esitetty raportit tehtäväkokonaisuus myöhästyi kolme viikkoa ja samalla viivästytti saman verran koko erikoistyöprojektia. Suhteellisesti tämä viivästyminen on minimaalista, koska koko

projekti ajallinen kesto oli noin yhdeksän kuukautta eli myöhästyminen on noin 8% projektin kokonaiskestosta.

Eri tehtäväkokonaisuuksien sisällä jouduttiin työn aikana suorittamaan pientä tinkimistä tavoitteista. Esimerkiksi testitapauksien laadinnassa oli alkuperäisen suunnitelman mukaan tarkoitus testata hieman enemmän realistisia verkonhallintaohjelmistoja verrattuna siihen mitä nykyisellään tehtiin mutta resurssien puutteen vuoksi ja erikoistyön tekijän siirtyessä muihin työtehtäviin, jäädytettiin projektin tulokset toukokuun lopulla saavutettuun tasoon.

Tarkasteltaessa projektia kokonaisuutena, voi toteuttajan näkökulmasta todeta projektin aikataulusuunnittelun onnistuneen kohtuullisen hyvin ja tätä kautta itse projektin toteutuksen sujuneen aikataulun kannalta ilman suurempia ongelmia.



Kuva 36: Erikoistyön alkuraportissa esitetty aikataulu

6.1 Käytetyt resurssit

Erikoistyölle allokoitiin IPMAN -projektin puitteissa resursseja neljä henkilötyökuukautta (4 htkk). Henkilötyötunneissa tämä on noin kuusisataa neljäkymmentä (640 htt). Erikoistyö projektissa käytettiin kokonaisuudessaan resursseja aikavälillä 01.09.1999 -

31.05.2000 mennessä 4,7 htkk. Tämä ei kuitenkaan ole todellinen arvo, koska työnsuorittajalla oli muitakin tehtäviä TML -laboratorion muissa projekteissa. Todellinen työmäärä, joka käytettiin erikoistyöprojektiin lienee noin 3,8 htkk. Eli resurssien arvioinnin/käytön suhteen projekti on sujunut hyvin.

LÄHDEVIITTEET

/1/ Teknillinen korkeakoulu [Internet sivut]. [viitattu 19.01.2000]. Saatavissa: http://www.hut.fi

/2/ Teknillinen korkeakoulu: TOTI - Instituutti, TSE - Institute (Telecommunications and Software Engineering) [Internet sivut]. 7.8.1997 [viitattu 19.01.2000]. Saatavissa: http://www.cs.hut.fi/TOTI/

/3/ Teknillinen korkeakoulu: Tietoliikenneohjelmistojen ja multimedian laboratorio [Internet sivut]. [viitattu 19.01.2000]. Saatavissa: http://www.tcm.hut.fi/

/4/ Uosukainen, Liisa & Lilja, Tuomas & Metso, Lasse & Ylänen, Stiina & Ihalainen, Seppo & Karvo, Jouni & Taipale, Ossi. IP Network Management - Interim report of the IPMAN project. TML-B2 Helsinki University of Technology - Laboratory of Telecommunications Software and Multimedia. Jun 1999.

/5/ Dr. Feit, Sidnie & Ranade, Jay. SNMP: A guide to Network Management. McGraw-Hill. 1995. 674 s. ISBN 0-07-020359-8.

/6/ Stallings, William. Data and Computer Communications. Fifth Edition. New Jersey: Prentice-Hall International, Inc. 1997. 798 s. ISBN 0-13-571274-2.

/7/ Stallings, William. SNMP, SNMPv2, and CMIP: The practical guide to network management. Fourth printing. Addison-Wesley. 1994. 625 s. ISBN 0-201-63331-0.

/8/ International Standard - ISO/IEC 9646-3. Information technology - Open Systems Interconnection - Conformance testing methodology and framework. Part 3: The Tree and Tabular Combined Notation. Second edition. 265 s.

/9/ Puro, Vesa - Matti. 1999. TTCN - testaus 03031999 /1.1/. Open Environment Software Oy

/10/ Toivanen, Ilkka & Vepsäläinen, Esa. TTCN, Tree and Tabular Combined Notation. Lappeenrannan teknillinen korkeakoulu - 1676 Avointen palveluverkkojen sovellukset: OSI jatkokurssi, seminaarityö. 1996. 44 s.

/11/ X.292. International Telecommunication Union. Series X: Data networks and open system communications - Open Systems Interconnection - Conformance testing: OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications - The Three and Tabular Combined Notation (TTCN). Revised 09/98. 1999.

/12/ X.291. International Telecommunication Union. Series X: Data networks and open system communications - Open Systems Interconnection - Conformance testing: OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications - Abstract Test Suite Specification. Revised 04/95. 1996.

/13/ X.290. International Telecommunication Union. Series X: Data networks and open system communications - Open Systems Interconnection - Conformance testing: OSI conformance testing methodology and framework for protocol Recommendations for ITU-T applications - General Concepts. Revised 04/95. 1996.

/14/ Steedman, Douglas. Abstract Syntax Notation One (ASN.1): The Tutorial & Reference. Twickenham: Technology Appraisals Ltd. 1993. 171 s. ISBN 1-871802-06-7.

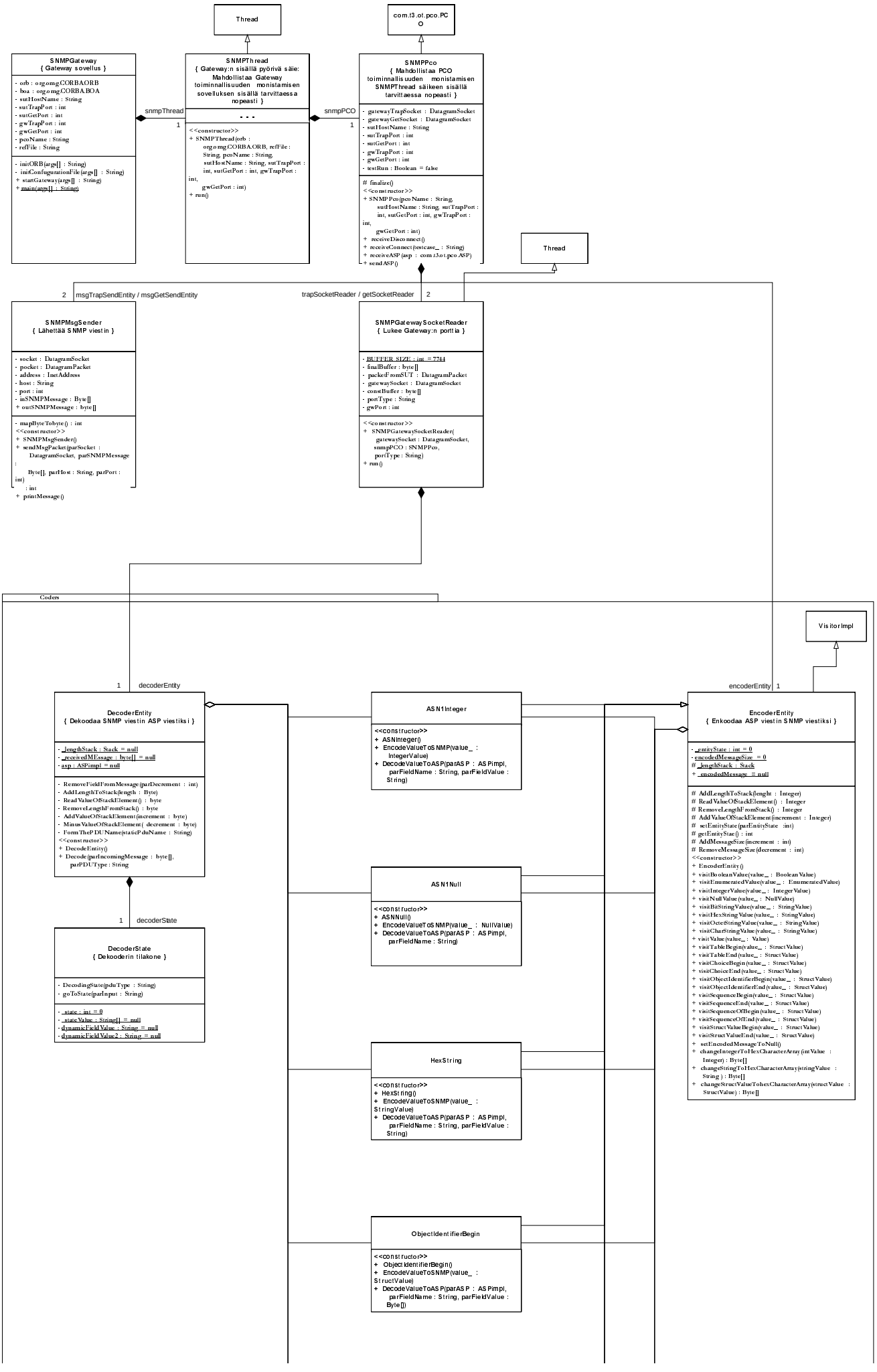
/15/ Object Management Group (OMG) [Internet sivut]. [viitattu 09.05.2000]. Saatavissa: http://www.omg.org/

/16/ Orfali, Robert & Harkey, Dan. Client/Server Programming with JAVA and CORBA. Second Edition. New York: Wiley Computer Publishing. 1998. 1022 s. ISBN 0-471-24578-X.

/17/ Open Environment Software Inc. OpenTTCN Products and Tutorial for Running OpenTTCN Tester: Automated Conformance Testing in Workstation Environment Using OpenTTCN. Version 2.1. 2000. 26 s.

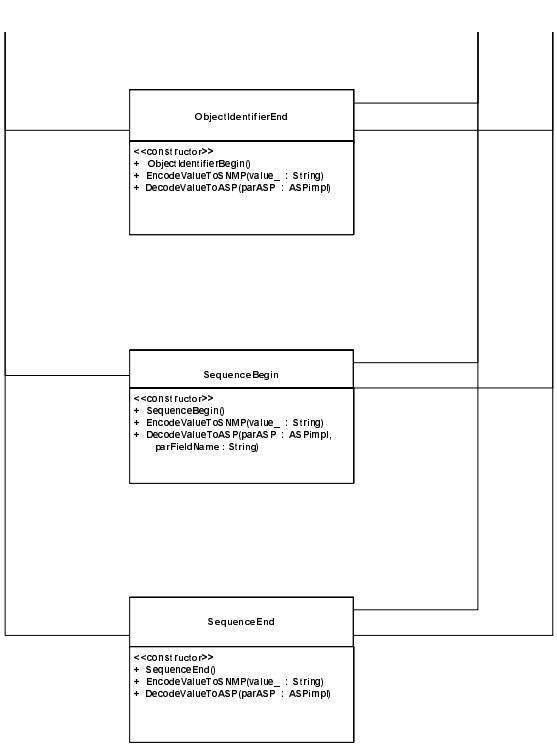
Tiedostonimi:	1715 Tietotekniikan erikoistyöt - Loppuraportti.doc
Hakemisto:	D:\Works\Koulu jutut\010715000 Tietotekniikan erikoistyöt\Loppuraportti
Malli:	E:\ohjelmatiedostot\Microsoft Office\Mallit\Normal.dot
Otsikko:	OpenTTCN protokollatesterin soveltaminen SNMP verkonhallintajärjestelmien testaamisessa internet verkoissa.
Aihe:	010715000 Tietotekniikan erikoistyöt - Loppuraportti
Tekijä:	Toni Suihko
Avainsanat:	ISO 9646, X.290, SNMP, OpenTTCN, standardinmukaisuustestaus, protokollatestaus
Kommentit:	Lopullinen versio!
Luontipäivä:	18.06.00 17:48
Version numero:	2
Viimeksi tallennettu:	18.06.00 17:48
Viimeksi tallentanut:	Toni Suihko
Kokonaismuokkausaika:	20 minuuttia
Viimeksi tulostettu:	18.06.00 18:14
Viimeisestä täydestä tulostuksesta	
Sivuja:	84
Sanoja:	13 823 (noin)
Merkkejä:	114 737 (noin)

Liite 1. Gateway ohjelman UML -kuvaus; luokkadiagrammi



(jatkuu)

(Liite 1 jatkoa)



Liite 2. Esimerkki Gateway:n konfigurointitiedostosta

```
BEGIN

# Defines name of Point of Control and Observation (PCO) between Tester --
Gateway
pco LT_NMS

# Defines name and location of object reference file for Gateway
ref ../ref/ipmangw.ref

# Defines host name of System Under Test (SUT)
sut minni.tcm

# Defines trap port number of System Under Test (SUT)
strap 8022

# Defines get / set port number of System Under Test (SUT)
sget 8021

# Defines trap port number of Gateway
gwtrap 8022

# Defines get / set port number of Gateway
gwget 8021

END
```