TKK/Teletekniikan laboratorio

S-38.128 Teletekniikan erikoistyö (4 ov)

IFMP- ja TCP-liikenteen muodostus AX/4000-mittalaitteella

Tekijä:	**Kai Solehmainen** 40414B	Kai.Solehmainen@iki.fi
Ohjaaja:	**Mika Ilvesmäki**	lynx@luuri.hut.fi

Tiivistelmä

Tässä dokumentissa kuvataan ohjelmisto, jolla voidaan tuottaa AAL 5 CPCS-PDU-, IP-, TCP-, IFMP-tietopaketteja sekä PRBS-signaalia. Ohjelman on tarkoitus toimia em. tietopakettien muokkaus- ja tallennusympäristönä, jonka tietopaketeista tuottamat tiedostot voidaan ladata Adtech AX/4000 -mittalaitetta ohjaavaan ohjelmistoon ja sitä kautta syöttää mittalaitteelle. Dokumentissa kuvataan yleisesti myös käytetyt protokollat ja tietopakettityypit pääpainon ollessa tietopakettien rakenteessa ja niiden toteuttamisessa.

SISÄLLYSLUETTELO

Kuvaluettelo ..... **III**
Taulukkoluettelo ..... **IV**
Lyhenne- ja käsiteluettelo ..... **V**
1. Johdanto **1**
2. Käytetyt protokollat ja tietopaketit **3**
2.1 IP-tietopakettien kapselointi AAL5-kehyksiin 3
2.2 IP-tietopaketti 5
2.2.1 Tarkistussumma 5
2.3 TCP-tietopaketti 6
2.4 IFMP 7
2.4.1 IP-vuotyypit 7
2.4.1.1 Tyyppi 0 8
2.4.1.2 Tyyppi 1 8
2.4.1.3 Tyyppi 2 9
2.4.2 IFMP Adjacency Protocol 10
2.4.3 IFMP Redirection Protocol 10
2.4.3.1 Redirect Message 11
2.4.3.2 Reclaim Message 11
2.4.3.3 Reclaim Ack Message 12
2.4.3.4 Label Range Message 12
2.4.3.5 Error Message 12
2.5 PRBS 13
3. AX/4000-mittalaite ja ohjausohjelmisto **15**
4. PaGen-ohjelma **16**
4.1 Ohjelman toiminta 16
4.1.1 Liityntä AX/4000-ohjausohjelmistoon 17
4.2 Käyttöohjeet 18
4.3 Arkkitehtuuri 18
4.3.1 Käyttöympäristö 18

4.3.2 Ohjelmointiympäristö 18
4.4 Toteutus 19
4.4.1 Oliomalli 19
4.4.2 Käyttöliittymä 20
4.4.3 Tietopakettien kuvauskieli 21
4.4.4 Uuden tietopaketin lisääminen ohjelmaan 22
4.4.4.1 Tietopakettiluokan määrittely 23
4.4.4.2 Tietopaketin sivun määrittely 23
4.4.4.3 Tietopaketin lisääminen tunnettuihin tietopaketteihin 24
4.4.5 Tietopaketin ja sekvenssin tallentaminen tiedostoon/leikepöydälle 25
4.4.6 Tietopaketin tallentaminen binaarimuodossa tiedostoon 25
4.5 Testaus 25
4.6 Jatkokehitys 26
5. Yhteenveto 27
Lähteet 28
Liitteet 29
Liite A: OMT+-notaatio 29
Liite B: IP-tietopaketin esittely 30
Liite C: IP-tietopaketin parametrimäärittely 31
Liite D: IP-tietopakettisivun WM_INITDIALOG-tapahtuma 33
Liite E: IP-tietopaketti tallennettuna tiedostoon 34

Kuvaluettelo

Kuva 1. AAL5 CPCS-PDU 3
Kuva 2. IP PDU kapseloitu AAL5 CPCS-PDU:n (ei-ISO kapselointi) 4
Kuva 3. IPv4-tietopaketti 5
Kuva 4. TCP-tietopaketti 6
Kuva 5. IFMP ATM-vuotunnus 7
Kuva 6. IFMP vuotyypin 0 IP-tietopaketin kapselointi ATM-verkossa 8
Kuva 7. IFMP vuotyypin 1 tunniste 8
Kuva 8. IFMP vuotyypin 1 mukainen IP-tietopaketin kapselointi ATM-verkossa 9
Kuva 9. IFMP vuotyypin 2 tunniste 9
Kuva 10. IFMP vuotyypin 2 mukainen IP-tietopaketin kapselointi ATM-verkossa 9
Kuva 11. IFMP Adjacency Protocol Message 10
Kuva 12. IFMP Redirection Protocol Message 11
Kuva 13. IFMP Redirection Protocol Redirect Message 11
Kuva 14. IFMP Redirection Protocol Reclaim Message 12
Kuva 15. IFMP Redirection Protocol Reclaim Ack Message 12
Kuva 16. IFMP Redirection Protocol Label Range Message 12
Kuva 17. IFMP Redirection Protocol Error Message 13
Kuva 18. PRBS-9 signaalin lineaarinen siirtorekisteri 13

Kuva 19. Tietopaketin välilehti 16
Kuva 20. Relaatiomalli 19
Kuva 21. Packet-luokan perintä 20
Kuva 22. CPacketPage-luokan perintä 20
Kuva 23. OMT+-notaatio 29

Taulukkoluettelo

Taulukko 1. CPacket-luokan uudelleen määriteltävät funktiot 23
Taulukko 2. Tuetut ohjelmointikomponentit 24

Lyhenne- ja käsiteluettelo

AAL 5	ATM Adaptation Layer type 5, ATM:n sovitustaso
ASCII	American Standard Code for Information Interchange, tiedonsiirrossa käytetty merkistö, josta on sekä 7- että 8-bittiset versiot
ATM	Asynchronous Transfer Mode, nopea tietopakettien välitystekniikka
C++	Oliopohjainen ohjelmointikieli
CPCS	Common Part Convergence Sublayer, ATM:n sovitustason alikerros
CPI	Common Part Indicator, AAL5-solun kenttä, joka määrittelee solun siirtämän protokollan
CRC	Cyclic Redundancy Code, tarkistuskoodi
IEEE	Institute of Electrical and Electronics Engineers, Yhdysvaltalainen järjestö, joka lähinnä toimii tietoliikenteen standardoinnin alueella
IETF	Internet Engineering Task Force, TCP/IP standardoitiin liittyvä elin, joka tutkii ja ratkaisee teknisiä kysymyksiä
IFMP	Ipsilon Flow Management Protocol, protokolla, jolla voidaan välittää vuonkäsittelytietoja eri verkkoelementtien välillä
IP	Internet Protocol, Internetin verkkoprotokolla
IPv4	Internet Protocol version 4, Internetin verkkoprotokolla versio 4
ISO	International Standards Organization, kansainvälinen standardointijärjestö
LLC	Logical Link Control, IEEE 802.2 määritelty protokolla, jolla siirretään dataa IEEE 802-sarjan lähiverkoissa
MDI	Multiple Document Interface, Windows-käyttöjärjestelmän mukainen monidokumenttikäyttöliittymä. Käyttäjä voi käsitellä useaa dokumenttia yhtäaikaa
MFC	Microsoft Foundation Class Library, Microsoftin C++ luokkakirjasto windows-ohjelmointikomponenttien käsittelyä varten
MS	Microsoft, maailmanlaajuinen ohjelmistoyritys.
MTU	Maximum Transmission Unit, IP-paketin maksimikoko, joka voidaan siirtää ko. verkossa
OMT+	Object Modeling Technique, Objektien mallinnus notaatio. Perustuu OMT-notaatioon
OUI	Organizationally Unique Identifier, SNAP:n kenttä, joka sisältää valmistajakoodin
PaGen	Packet Generator, ohjelma, jolla voidaan erilaisia tietopaketteja luoda ja muokata
PDU	Protocol Data Unit, jonkin protokolla määrittelyn mukainen tietopaketti
PID	Protocol Identifier, siirretyn protokollan tunniste SNAP:ssa
PRBS	Pseudo Random Bit(Binary) Sequence, bittijono, joka näyttää satunnaiselta, mutta todellisuudessa se on laskennallisesti generoitavissa
RFC	Request for Comments, Internet-yhteisön yhteyskäytäntöjä ja kokeiluja kuvaava dokumentti sarja
SNAP	SubNetwork Access Protocol, LLC:n sisältämä kehysrakenne, joka ilmoittaa kuljetetun protokollan

TCP	Transmission Control Protocol, Internetin luotettava tiedonsiirtoprotokolla
TCP/IP	Protokollaperhe, joka koostuu verkkoprotokollasta (IP) ja kahdesta siirtoprotokollasta (TCP ja UDP) ja joukosta sovellusprotokollia
UDP	User Datagram Protocol, Internetin epäluotettava tiedonsiirtoprotokolla
UU	User-to-User indicator, AAL5-solun kenttä, joka sisältää käyttäjän määrittelemää tietoa
VCI	Virtual Circuit Identifier, ATM-verkon virtuaalikanavan tunniste
VPI	Virtual Path Identifier, ATM-verkon virtuaalipolun tunniste
XOR	Exclusive-Or, poissulkeva-tai bittioperaattori

1. Johdanto

Digitaalisessa pakettimuotoisessa tiedonsiirrossa käytetään erittäin suurta määrää erilaisia pakettityyppejä, jotka riippuvat käytetystä verkkotyypistä ja protokollasta. Paketit muodostavat usein sisäkkäisen rakenteen, jossa paketti kapseloidaan toisen sisään. Varsinainen kuljetettava tieto sijoitetaan sisimpään pakettiin.

Pakettien rakenne vaihtelee eri pakettityyppien välillä, mutta normaalisti loppukäyttäjän ei tarvitse välittää kuinka paketti rakentuu vaan kapselointi on peitetty jonkin protokollapinon taakse. Protokollapino muodostuu usein monesta eri kerroksesta, joista ylimpänä on usein jonkinlainen ohjelmointirajapinta ja alimpana verkkotyypistä riippu verkkokerros. Kukin kerros kapseloi tarvittaessa ylemmän kerroksen välittämän tiedon kerroksen mukaiseen pakettiin ja välittää paketin alemmalle kerrokselle. Tämä mahdollistaa eri tiedon välittämisen eri tyyppisten verkkojen yli samalla tavoin eikä ylemmällä kerroksella tarvitse välittää kuinka tieto kapseloidaan.

Joissakin tilanteissa on hyödyllistä generoida suoraan tietyntyyppisiä paketteja, jotka voidaan lähettää suoraan tietoverkkoon. Tällöin voidaan vaikuttaa suoraan pakettityypin eri ominaisuuksiin, jotka vaikuttavat paketin käsittelyyn eri verkkoelementeissä. Tästä on hyötyä esimerkiksi verkon testaamisessa ja erilaisten erikoistilanteiden luomisessa.

Erilaisia paketteja generoivia ohjelmia on olemassa, mutta usein ne on kehitetty tiettyyn tarkoitukseen kuten IP-pakettien generointiin. Tämä ei riitä mikäli halutaan testata esim. ATM-verkkoa käyttäen AX/4000-mittalaitetta, jolla ATM-verkon liikennettä voidaan tarkkailla tai generoida. Tarvittaisiin ohjelma, joka pystyy generoimaan useita erilaisia pakettityyppejä ja kapseloimaan eri paketit toistensa sisälle.

PaGen-ohjelman lähtökohtana oli tehdä ohjelma, jolla voidaan TCP- ja IFMP-liikennettä tuottaa AX/4000-mittalaitteelle. Ohjelman alkuvaiheen suunnittelun aikana huomattiin, että samalla voidaan tuottaa yleinen pakettien muokkaamisohjelmisto. Niinpä ohjelmistosta kehittyi laajempi kokonaisuus, joka pystyy vastamaan edellä mainittuihin vaatimuksiin. Lisäksi ohjelmassa on otettu huomioon AX/4000-mittalaitteen erityispiirteet, joten ohjelmalla voidaan tuottaa AX/4000-mittalaitteen kanssa yhteensopivia tiedostoja.

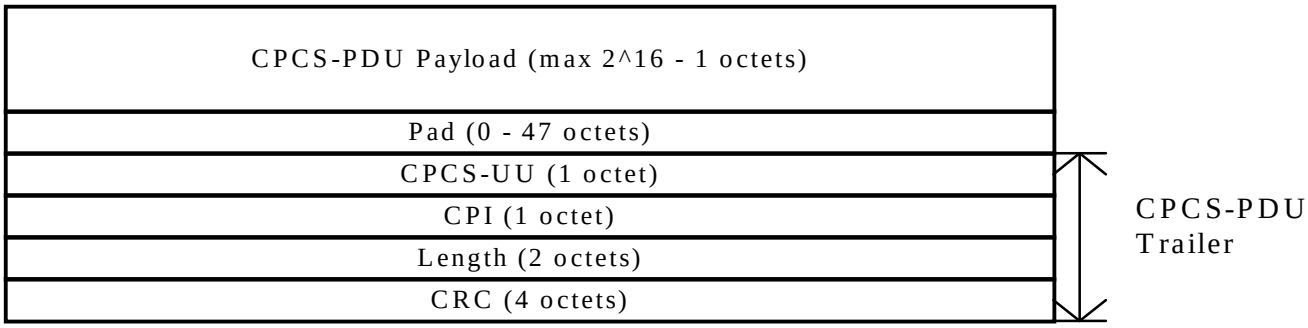
Tässä dokumentissa esitetään ensin katsaus *PaGen*-ohjelmassa käytettyihin paketti- ja protokollatyyppeihin, ja lopuksi käsitellään varsinaista ohjelman toteutusta.

2. Käytetyt protokollat ja tietopaketit

PaGen-ohjelma tuottaa useita eri tyyppisiä paketteja. Seuraavassa on esitetty yleiskatsaus näihin eri tyyppeihin ja niiden rakenteisiin.

2.1 IP-tietopakettien kapselointi AAL5-kehyksiin

ATM-verkossa voidaan siirtää eri tiedonsiirtoprotokollien mukaista liikennettä rinnakkain. Tämä on toteutettu käyttäen AAL5-kerroksen kehyksiä ja palveluja. Protokollien siirtämiseen on määritelty kaksi eri menetelmää. Ensimmäisessä menetelmässä eri protokollia siirretään yhdessä virtuaalikanavassa. Toinen menetelmä edellyttää, että eri protokollat siirretään kukin omassa virtuaalikanavassaan. Riippumatta käytetystä menetelmästä liikenne kapseloidaan AAL5 CPCS-PDU -kehyksen kuormaan [rfc1483]. Kuvassa 1 on esitetty AAL5 CPCS-PDU kaavakuvana.



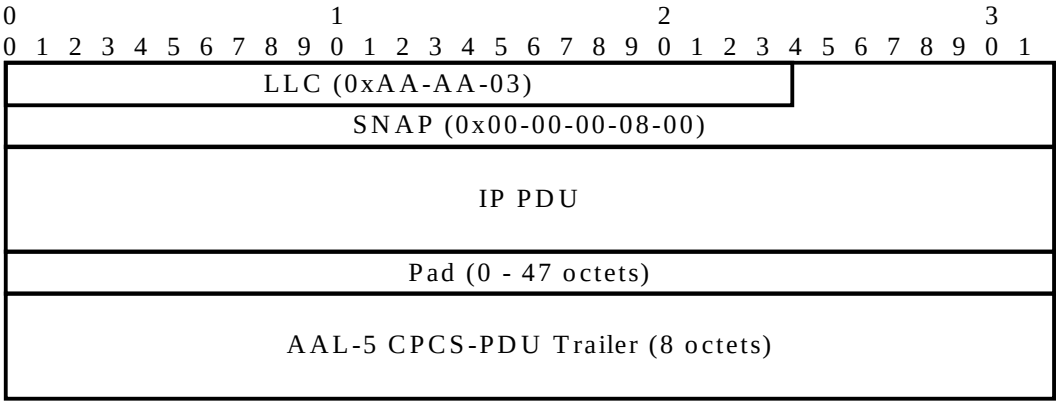
Kuva 1. AAL5 CPCS-PDU

Siirrettäessä useaa eri protokollaa samassa virtuaalikanavassa on käytettävä ns. LLC-kapselointia. Kehyksiin täytyy lisätä tunniste, jonka avulla vastaanottaja pystyy tunnistamaan kehyksen sisältämän protokollan ja käsittelemään kehyksen sisältöä oikein. LLC-kapseloinnissa kehykseen sijoitetaan IEEE 802.2 LLC-otsikkokenttä, jota mahdollisesti seuraa IEEE 802.1a SNAP-otsikkokenttä. Periaatteet ovat samanlaiset eri LLC-tyypeille, mutta otsikoiden muoto ja sisältö voivat olla erilaiset eri LLC-tyypeille. LLC-kenttä kertoo mikä on tietopaketissa käytetty protokolla sekä tiedon kehyksen tyypistä.

Mikäli kehyksessä siirrettävä tietopaketti on ISO-protokollan PDU ei SNAP-kenttää tarvita. Jos kyseessä ei ole ISO-prokolla PDU, on LLC-kentän perään lisättävä vielä SNAP-kenttä ja LLC-kentän

arvoksi on asettava 0xAA-AA-03. SNAP-kenttä muodostuu valmistajakoodista (OUI) ja protokollatunnisteesta (PID).

LLC- ja SNAP-kenttien jälkeen kehykseen sijoitetaan varsinainen siirrettävä protokollapaketti. Tietopaketin maksimi koko on 2^16-4 (65532) tavua, mikäli kyseessä on ISO-prokollan tietopaketti, ja 2^16-9 (65527) tavua mikäli kyseessä on ei-ISO-protokollapaketti. Tietopaketin jälkeen lisätään täytettä, jotta kokonaispituus saadaan jaolliseksi 48. Lopuksi lisätään vielä AAL-5 CPCS PDU -kehyksen ohjauskentät, jotka sisältävät mm. virheentarkistuksen ja kehyksen kokonaispituuden.

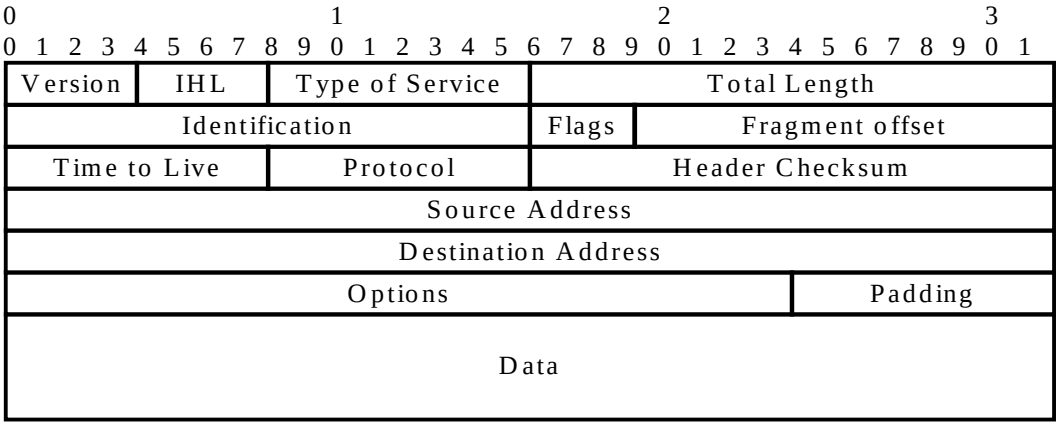


Kuva 2. IP PDU kapseloitu AAL5 CPCS-PDU:n (ei-ISO kapselointi)

IP-tietopaketti kapseloidaan kuten ei-ISO-protokollapaketti. Kuvassa 2 on esitetty IP-tietopaketin kapselointi. IP-tietopaketin MTU:n oletusarvoksi on määritelty 9180-tavua [rfc1626].

AX/4000-ohjainohjelmisto luo AAL5 CPCS-PDU:n täytteen (*engl. pad*) ja Trailer-osion automaattisesti. Tämän vuoksi *PaGen*-ohjelmassa on mahdollista tuottaa pelkkä AAL5 CPCS-PDU:n kuorma ilman em. osioita.

2.2 IP-tietopaketti



Kuva 3. IPv4-tietopaketti

IP(Internet Protocol)-tietopaketti on TCP/IP-protokollaperheen siirtokerroksen tiedonsiirtokehys, jolla IP siirtää datan paikasta toiseen. Tietopaketti muodostuu useasta eri kentästä, joiden perusteella tietopaketti siirretään oikeaan paikkaan oikealla tavalla. Nämä kentät muodostavat ns. tietopaketin otsikon. Otsikon pituus ei ole vakio, joten sen pituus ilmaistaan yhtenä otsikon arvona. Otsikon jälkeen seuraa varsinainen tietopaketin sisältämä data, jonka maksimikoko on 65536 tavua vähennettynä otsikon pituudella. Näin suuri tietopaketti on kuitenkin erittäin hankala käsitellä, joten suositellaan ettei suurempia paketteja kuin 576 tavua käytetä ellei tiedetä, että vastaanottaja pystyy vastaanottamaan suurempiakin paketteja. Kaikkien koneiden tulee pystyä vastaanottamaan ja välittämään 576 tavun kokoisia paketteja [rfc791]. Kuvassa 3 on esitetty version 4 IP-tietopaketti.

PaGen-ohjelmalla voidaan määritellä otsikon muut kuin optio- (*engl. Options*) ja täytekentät (*engl. Padding*). Ohjelma sijoittaa versionumeron ja laskee pituus- ja tarkistussummakentät automaattisesti.

2.2.1 Tarkistussumma

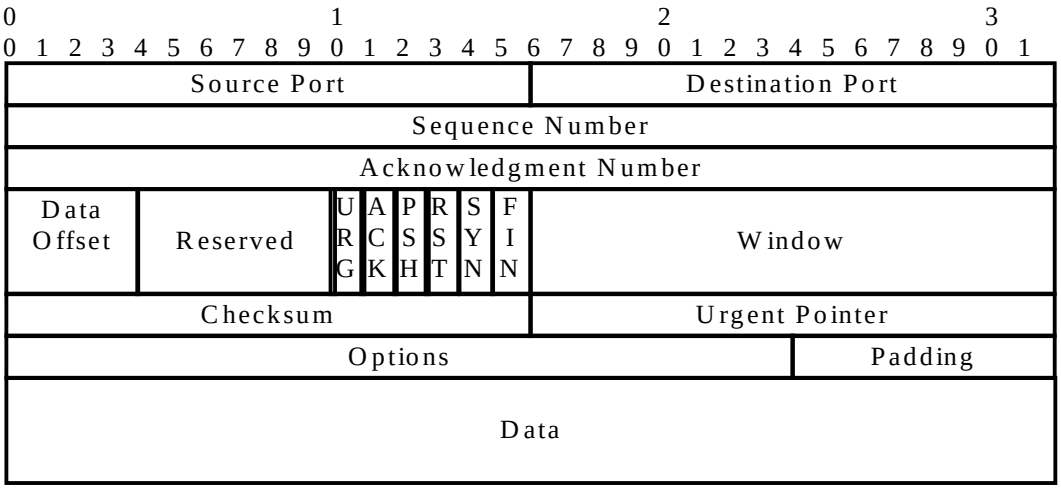
Tarkistussumma lasketaan pilkkomalla kaikki otsikko 16-bittiä pitkiksi sanoiksi ja summaamalla näin saadut sanat yhteen käyttäen yhden komplementtia. Tarkistussummakenttä asetetaan nollaksi laskemisen ajaksi. Saadusta summasta otetaan yhden komplementti, joka sijoitetaan tarkistussummaksi. Vastaanottaja laskee samanlaisen summan kaikista kentistä, mutta tällä kertaa käytetään tarkistussummakentässä todellista arvoa. Mikäli tarkistussumma on oikein, tulokseksi saadaan 0xffff, joka on 0 yhden komplementtina [rfc1071].

Yhden komplementti tarkoittaa lukua, joka saadaan muuttamalla lukua kuvaavan bittijonon 1:n sisältämät bitit 0:ksi ja 0:n sisältämät bitit 1:ksi (esim. 0100 -> 1011). Laskettaessa yhden komplementin summa kahdesta luvusta laskenta tehdään muuten normaalisti, mutta mahdollinen ylivuoto lisätään summaan.

Esim.	Normaali	Yhden komplementti	
	0xffde	0xffde	
	+0x00ff	+0x00ff	
	-------	-------	
	0x100dd	0x00dd	
		+0x0001	ylivuoto

		0x00de	

2.3 TCP-tietopaketti



Kuva 4. TCP-tietopaketti

TCP (Transmission Control Protocol)-tietopaketti on TCP/IP-protokollaperheen kuljetuskerroksen tiedonsiirtokehys, jolla TCP siirtää dataa . TCP on protokolla, joka tarjoaa luotettavan päästä-päähän yhteyden kahden eri tietokoneen välillä. Kuvassa 4 on esitetty TCP-tietopaketti. Tietopaketti muodostuu otsikosta ja siirrettävästä datasta. Otsikon pituus on vakio toisin kuin IP-tietopaketin otsikko.

Tietopaketin tarkistussumma lasketaan otsikosta ja kuljetettavasta datasta. Otsikon eteen lisätään nk. pseudo-otsikko, joka muodostetaan IP-tietopaketin otsikosta. Tämä pseudo-otsikko sisältää lähettäjän

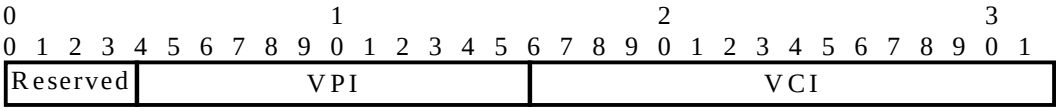
ja vastaaottajan IP-osoitteen sekä protokollan ja TCP-tietopaketin pituuden. Näin varmistetaan, ettei TCP-tietopakettia reititetä väärin. Pseudo-otsikon, otsikon ja datan sisältämät 16-bittiset sanat summataan yhteen yhden komplementin summana ja saadusta luvusta otetaan yhden komplementti, joka sijoitetaan tarkistussummaksi [rfc793].

PaGen-ohjelmalla voidaan määritellä otsikon muut kuin optio- (*engl. Options*) ja täytekentät (*engl. Padding*).

2.4 IFMP

IFMP (Ipsilon Flow Management Protocol) on protokolla, jolla voidaan määritellä IP-vuolle tunnus verkkoelementissä (reititin). Tunnus mahdollistaa vuon tehokkaamman reitityksen ja joissain tapauksissa vuota ei tarvitse reitittää vaan se voidaan suoraan välittää eteenpäin [rfc1953].

IP-vuo on sarja IP-paketteja, jotka ovat samaa protokollaa ja lähetystä eli lähettäjä ja vastaanottaja on sama kaikissa paketeissa. Tietopakettien siirtoa voidaan merkittävästi nopeuttaa, mikäli jokaista tietopakettia ei tarvitse erikseen tutkia ja reitittää. Tämä on mahdollista antamalla jokaiselle tietopakettivuolle oma tunniste. Tunniste on 32-bittinen luku. Kuvassa 5 on esitetty vuotunnus, jota käytetään ATM-verkossa. Tunniste muodostuu käytetyn virtuaalipolun ja -kanavan tunnisteesta.



Kuva 5. IFMP ATM-vuotunnus

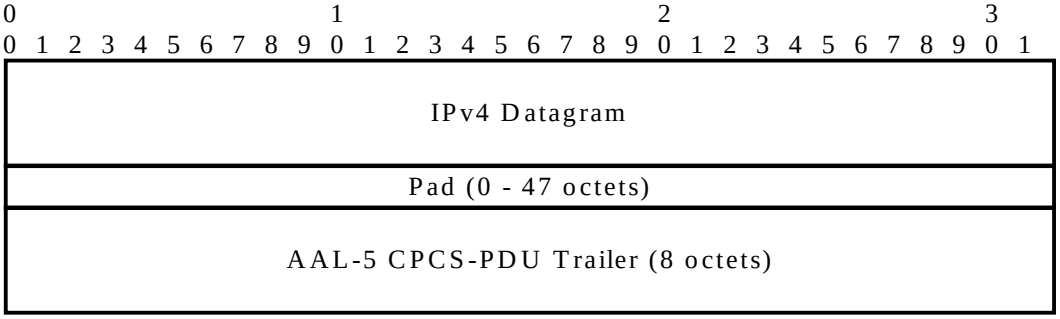
2.4.1 IP-vuotyypit

IP-vuotyyppejä on useita. Tällä hetkellä on määritelty tyypit 0, 1 ja 2. Jokaisella vuotyypillä on määritelty oma vuotunniste ja kapselointitapa. Tunniste muodostetaan IP-tietopaketin otsikon kenttien perusteella ja sillä tunnistetaan siirrettävä vuo. Kapseloinnilla tarkoitetaan tapaa, jolla IP-tietopaketti sijoitetaan verkkopakettiin. Tässä dokumentissa esitetään kapselointi, jota käytetään ATM-verkoissa. Kohdassa 2 on esitetty oletusarvoinen kapselointi.

PaGen-ohjelma tukee vuotyyppien 0,1,2 mukaisia IP-paketteja, jotka on kapseloitu AAL 5 CPCS-PDU:n kuormaksi. Täytettä eikä AAL 5 solun trailer-osiota ei ohjelma tuota.

2.4.1.1 Tyyppi 0

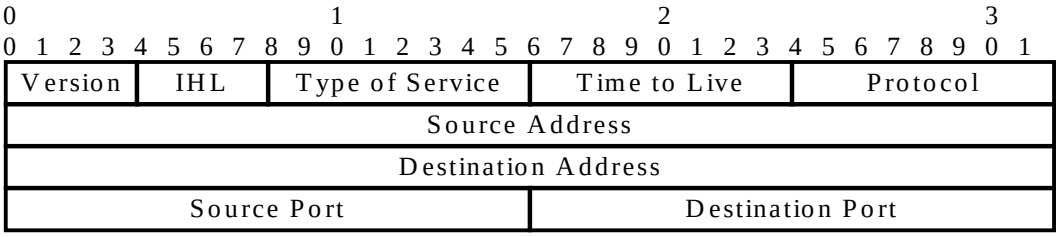
Tyypin 0 tunniste on nollan pituinen. Kaikki vuotyyppi 0:n IPv4 kehykset kapseloidaan AAL 5 CPCS-PDU-kehyksen kuormaksi ilman LLC ja SNAP-kenttiä. Menetelmä säästää 6 tavua. Kuvassa 6 on esitetty kapselointi.



Kuva 6. IFMP vuotyypin 0 IP-tietopaketin kapselointi ATM-verkossa

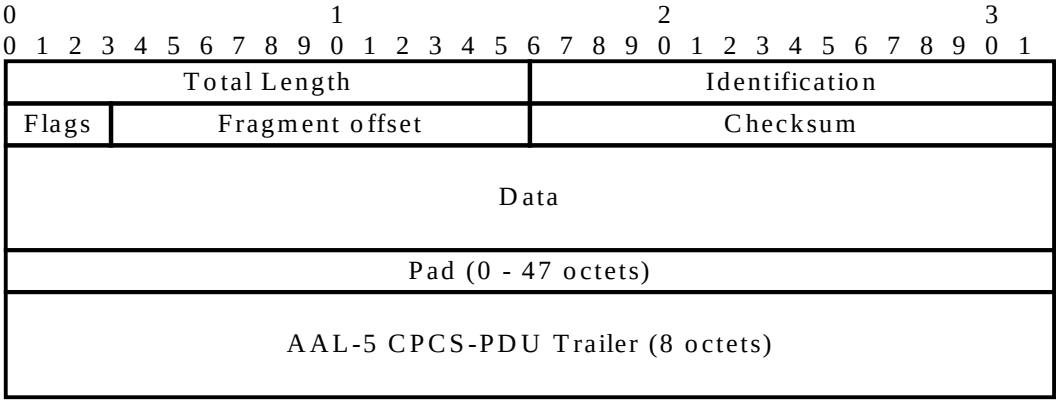
2.4.1.2 Tyyppi 1

Vuotyyppi 1 on suunniteltu TCP- ja UDP-tietopakettien siirtämistä varten. Tunniste muodostetaan IP-tietopaketin otsikon tietojen ja TCP- tai UDP-tietopaketin porttiosoitteiden avulla. Kuvassa 7 on esitetty vuotyypin 1 tunniste.



Kuva 7. IFMP vuotyypin 1 tunniste

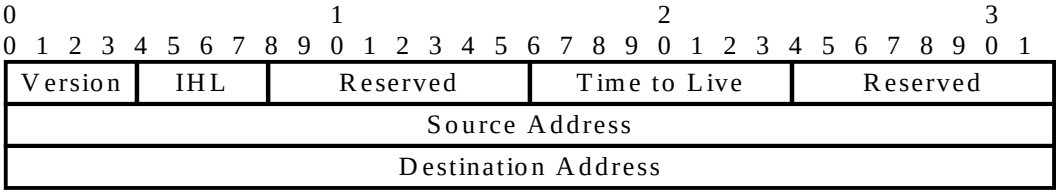
Vuotyypin 1 IP-tietopakettit kapseloidaan suoraan AAL 5 CPCS-PDU-kehyksen kuormaksi ilman LLC ja SNAP-kenttiä. Myöskään kaikkia IP-tietopaketin otsikon tietoja ei sijoiteta kehykseen sillä ne saadaan tarvittaessa tunnisteesta. Kuvassa 8 on esitetty AAL 5 CPCS-PDU, johon on kapseloitu IP-tietopaketti vuotyypin 1 mukaisesti.



Kuva 8. IFMP vuotyypin 1 mukainen IP-tietopaketin kapselointi ATM-verkossa

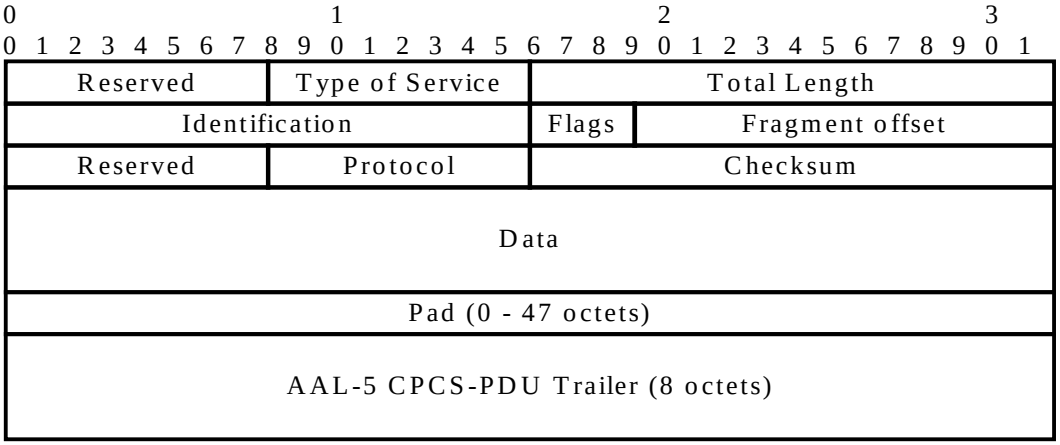
2.4.1.3 Tyyppi 2

Vuotyyppi 2 on suunniteltu kaikkia IP-paketteja varten. Tunniste muodostetaan IP-tietopaketin otsikon kenttien avulla. Kuvassa 9 on esitetty vuotyypin 2 tunniste.



Kuva 9. IFMP vuotyypin 2 tunniste

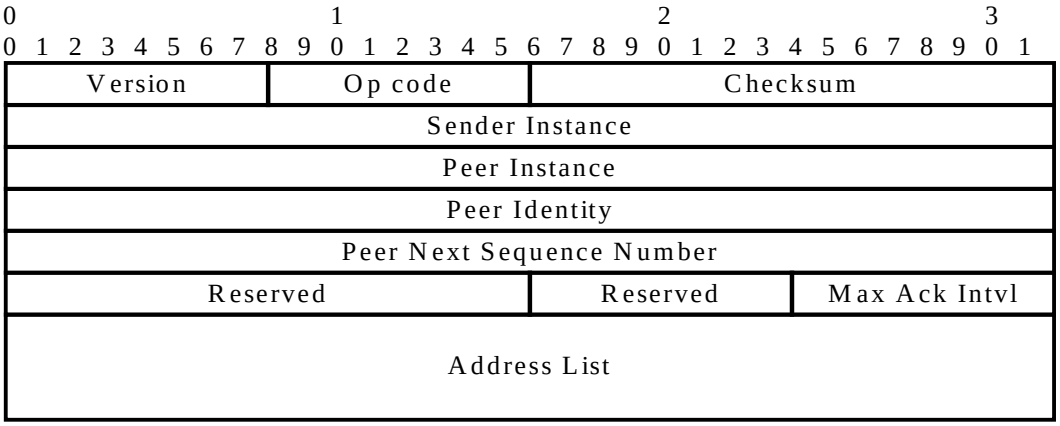
IP-tietopaketti kapseloidaan vuotyypin 1 lailla suoraan AAL 5 CPCS-PDU-kehyksen kuormaksi ilman LLC ja SNAP-kenttiä. IP-tietopaketin kentistä siirretään ne, joita ei tunnisteessa ole. Kuvassa 10 on esitetty AAL5 CPCS-PDU, johon on kapseloitu IP-tietopaketti vuotyypin 2 mukaisesti.



Kuva 10. IFMP vuotyypin 2 mukainen IP-tietopaketin kapselointi ATM-verkossa

2.4.2 IFMP Adjacency Protocol

Adjacency-protokollan avulla reititin tai isäntäkone voi löytää linkin toisessa päässä olevan vertaisolion (*engl. peer*) identiteetin. Protokollan avulla voidaan myös synkronisoida tila, jossa linkin eri päässä olevat oliot ovat, havaita milloin vertaisolio linkin toisessa päässä vaihtuu tai siirtää linkkiin sidotut IP-osoitteet. Kuvassa 11 on esitetty protokollan mukainen tietopaketti.



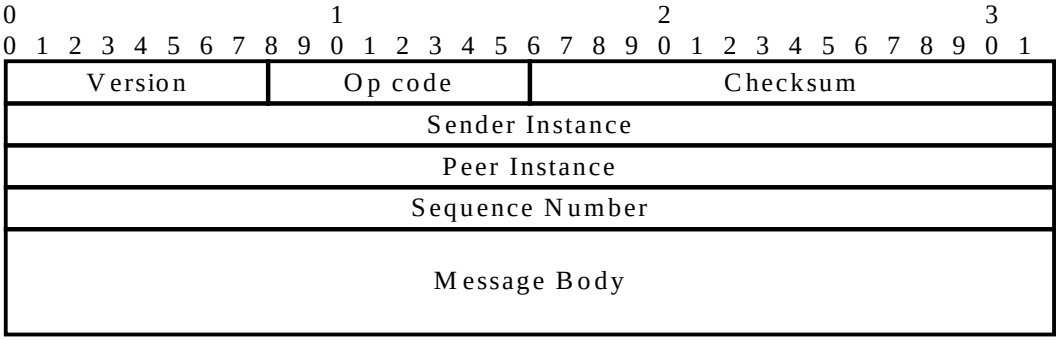
Kuva 11. IFMP Adjacency Protocol Message

Tietopaketti kapseloidaan version 4 IP-tietopaketiin. IP-tietopaketin vastaanottajan IP-osoitteeksi on laitettava 255.255.255.255 ja protokollaksi 101. Elinaika (*engl. Time to Live, TTL*) on asetettava 1:ksi.

Tarkistussumma lasketaan samoin kuin TCP-tietopaketille. Aluksi lasketaan yhden komplementin summa IP-tietopaketin otsikosta saadulle pseudo-otsikolle ja tietopaketin muille kentille. Lopuksi otetaan yhden komplementti edellä lasketusta summasta ja sijoitetaan tämä tarkistussummaksi.

2.4.3 IFMP Redirection Protocol

Redirection-protokolla muodostuu useasta eri viestistä, jotka kaikki kapseloidaan Redirection Protocol Message-tietopaketin Message Body -osioon. Tietopaketti puolestaan kapseloidaan IP-tietopakettin. Tietopaketin osoitteeksi laitetaan vertaisolion osoite, joka voidaan selvittää Adjcency-protokollan avulla. Tietopaketin elinajaksi laitetaan 1 ja protokollaksi 101. Tarkistussumma lasketaan kuten Adjacency-protokollan tietopaketin tarkistussumma. Kuvassa 12 on esitetty Redirection-protokollan tietopaketti.

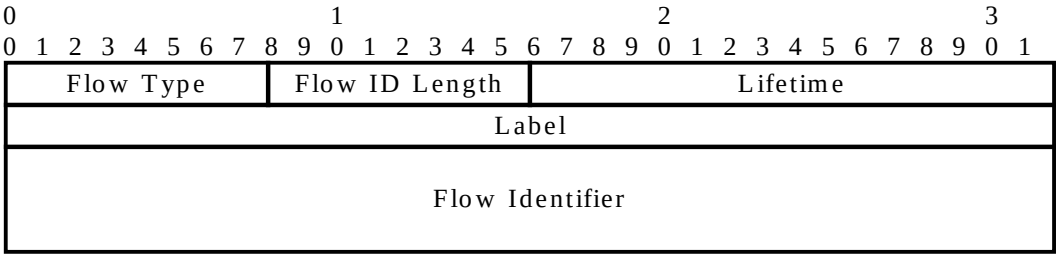


Kuva 12. IFMP Redirection Protocol Message

Tietopaketti voi sisältää yhden tai useamman samanlaisen viestin. Mahdollisia viestejä ovat Redirect Message, Reclaim Message, Reclaim Ack Message, Label Range Message ja Error Message.

2.4.3.1 Redirect Message

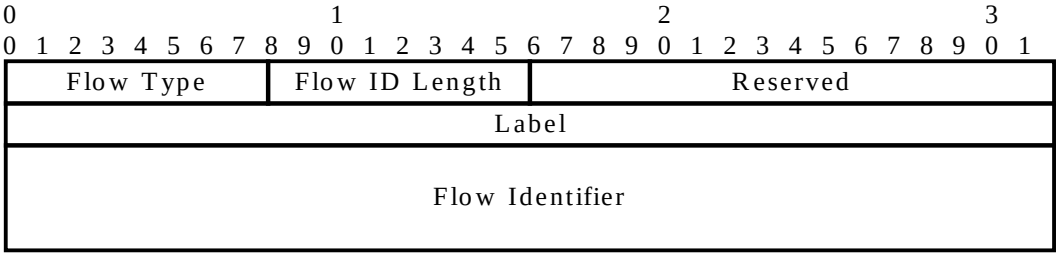
Redirect-viestillä voidaan välittää verkon viereiselle elementille käsky kytkeä annettu tunnus paketteihin, jotka kuuluvat määriteltyyn vuohon. Kytkentä on voimassa tietopaketissa annetun ajan (*engl. Lifetime*). Yhdellä viestillä voi muodostaa yhden kytkennän. Kuvassa 13 on esitetty kaaviokuva viestistä.



Kuva 13. IFMP Redirection Protocol Redirect Message

2.4.3.2 Reclaim Message

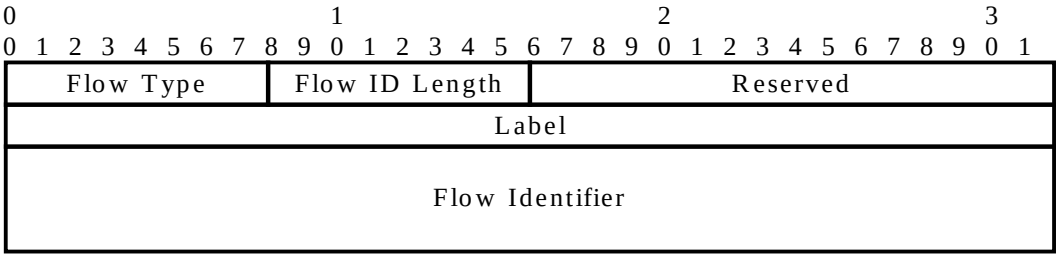
Reclaim-viestillä voidaan välittää verkon viereiselle elementille käsky purkaa kytkentä tunnuksen ja vuon välillä. Yhdellä viestillä voi purkaa yhden kytkennän. Kuvassa 14 on esitetty kaaviokuva viestistä.



Kuva 14. IFMP Redirection Protocol Reclaim Message

2.4.3.3 Reclaim Ack Message

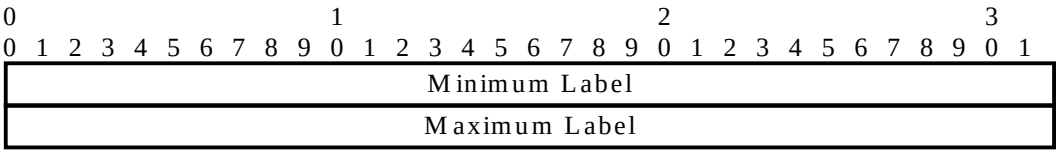
Reclaim Ack -viestillä vahvistetaan kytkennän purkaminen. Jokaiseen Reclaim-viestiin tulee vastata tällä viestillä. Kuvassa 15 on esitetty kaaviokuva viestistä.



Kuva 15. IFMP Redirection Protocol Reclaim Ack Message

2.4.3.4 Label Range Message

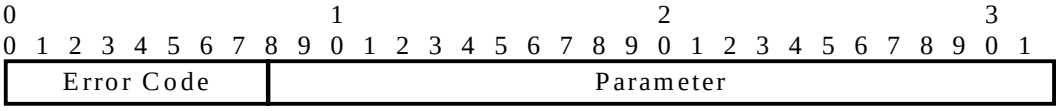
Label Range -viesti lähetetään vastauksena Redirect-viestille mikäli Redirect-viestin jonkin tunnus on suurempi tai pienempi kuin vastaanottaja voi käsitellä. Label Range -viestillä informoidaan lähettäjää rajoista, joiden sisällä tunnuksen tulee olla. Kuvassa 16 on esitetty kaaviokuva viestistä.



Kuva 16. IFMP Redirection Protocol Label Range Message

2.4.3.5 Error Message

Error-viesti voidaan lähettää vastauksena mille tahansa Redirection-protokollan viestille. Viestin sisältönä on virhekoodi ja mahdollinen parametri. Kuvassa 17 on esitetty kaaviokuva Error-viestistä.



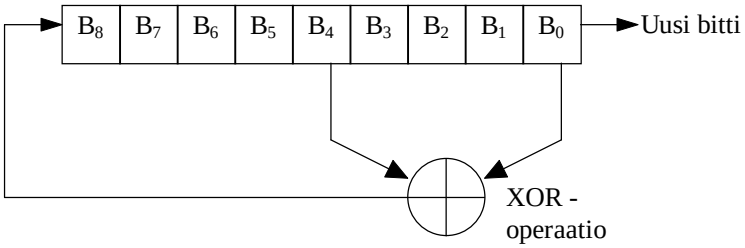
Kuva 17. IFMP Redirection Protocol Error Message

2.5 PRBS

PRBS (Pseudo Random Bit Sequence) on bittijono, joka muodostetaan jonkin määrätyn generoivan polynomin avulla. Bittijono simuloi satunnaista bittijonoa tilanteissa, joissa bittijonon tulee olla kuitenkin ennustettavissa. Tätä voidaan hyödyntää esimerkiksi testattaessa tiedonsiirtovirheitä. Toisesta päästä lähetetään PRBS-signaalia ja vastaanottaja päässä seurataan onko tiedonsiirrossa syntynyt virheitä.

PRBS-signaaleja on määritelty useita erilaisia kuten PRBS-9, PRBS-15 ja PRBS-23. Nimessä numero n tarkoittaa, että bittijono alkaa toistaa itseään 2^n-1 bitin jälkeen. PRBS-9 generoi siis bittijonon, joka alkaa alusta 511-bitin jälkeen. Jokaisella eri tyypillä on erilainen generoiva polynomia.

Tässä esitellään menetelmä, jolla PRBS-9 signaali tuotetaan. PRBS-9 signaali tuotetaan lineaarisella takaisinkytketyllä siirtorekisterillä, jonka generoiva polynomi on X^4+X^0. Kuvassa 18 on esitetty siirtorekisteri kaaviokuvana [axlib].



Kuva 18. PRBS-9 signaalin lineaarinen siirtorekisteri

Siirtorekisteri generoi uuden bitin seuraavasti. Ensin suoritetaan XOR-operaatio biteille numero 0 ja 4. Tulos laitetaan bitin numero 7 tilalle. Samalla siirretään kaikkien bittien arvoja yhden oikealle eli bitin 8 sisältö sijoitetaan bitin 7 sisällöksi jne. Bitin 0 sisältö siirretään uudeksi generoiduksi bitiksi.

PaGen-ohjelmalla tukee nykyisessä versiossaan vain PRBS-9-signaalia. Sekvenssi alkaa aina alusta riippumatta, missä kohtaa pakettia ollaan menossa. Tämä voi aiheuttaa ongelmia vastaanottopään synkronoinnin kanssa. Vastaanottaja ei pysty havaitsemaan kohtaa, jossa PRBS-signaali alkaa.

3. AX/4000-mittalaite ja ohjausohjelmisto

AX/4000-mittalaite on Adtech Inc. toteuttama laite- ja ohjelmistokokonaisuus, jolla voidaan analysoida ja generoida ATM-verkon liikennettä. Tämä mahdollistaa verkon ja sen eri toimilaitteiden, kuten reitittimien ja kytkimien, toiminnan tarkkailun ja ohjaamisen. Kokonaisuus muodostuu itse mittalaitteesta ja sen ohjausohjelmistosta. Ohjelmisto on muodostettu modulaarisesti eri komponenteista, joiden avulla käyttäjä voi toteuttaa erilaisia testejä. Lisäksi on saatavissa funktiokirjasto, jolla mittalaitetta voidaan ohjata suoraan ohjelmallisesti [axdoc].

Ohjausohjelmiston avulla voidaan muodostaa erilaisia liikennelähteitä, joiden avulla voidaan simuloida erilaisia liikennetilanteita. Ohjelmisto mahdollistaa myös ATM-solujen määrittelyn ja muokkaamisen. Solujen sisältö voidaan myös ladata ulkopuolisesta tiedostosta. *PaGen*-ohjelmalla voidaan tuottaa em. tiedostoja tallentamalla tietopaketti binaarimuodossa. Lisäksi AX/4000-ohjelmalla luotuja paketteja voidaan tallentaa levylle ja ladata uudelleen käsiteltäväksi.

AX/4000-ohjainohjelmisto pystyy myös käsittelemään sekvenssejä. Sekvenssi (*engl. sequence*) on sarja paketteja, jotka lähetetään mittalaitteen avulla verkkoon. Sekvenssiin voidaan määritellä hyppyjä ja toistoja. Sekvenssejä voidaan tallentaa levylle ja lukea uudelleen muistiin käsiteltäväksi. *PaGen*-ohjelma pystyy tuottamaan yksinkertaisen sekvenssitiedoston, joka ei sisällä hyppyjä tai muita erityisrakenteita. Mikäli näitä halutaan lisätä, on sekvenssiä käsiteltävä edelleen AX4000-ohjelman avulla.

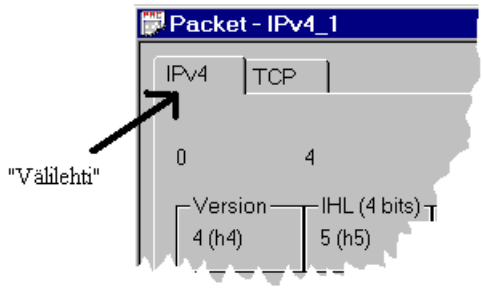
4. PaGen-ohjelma

Pagen on ohjelma, joka tarjoaa graafisen Windows 95 -yhteensopivan käyttöliittymän erilaisten verkkopakettien luomiseen, editoimiseen ja binaariseen muotoon kääntämiseen Windows 95 - tai Windows NT -ympäristössä. Tietopaketit esitetään havainnollisesti graafisessa muodossa, joka vastaa IETF:n RFC-dokumenteissa esitettyä notaatiota. Käyttäjä voi vapaasti muuttaa suurinta osaa tietopakettien parametreistä. Tietopakettien parametrit on alustettu oletusarvoilla, joten käyttäjän ei tarvitse määritellä kaikkia parametrejä itse.

4.1 Ohjelman toiminta

Avattaessa *PaGen*-ohjelma avautuu näkymä ohjelman työtilaan, joka on aluksi tyhjä. Käyttäjä voi joko luoda tai avata jo olemassa olevan tiedoston, jolloin työtilaan avautuu näkymä dokumenttiin, joka voi olla joko sekvenssi tai tietopaketti. Koska ohjelma on kehitetty monidokumenttijärjestelmän periaatteiden mukaisesti, voi käyttäjällä olla useita eri dokumentteja auki yhtäaikaa samassa ohjelmassa.

Mikäli käsitellään tietopakettia, näytöllä on näkyvissä lomake, joka sisältää erilaisia käyttäjän muuteltavissa olevia arvoja. Lisäksi lomakkeessa voi olla myös automaattisesti muuttuvia arvoja, joita käyttäjä ei voi muuttaa. Lomakkeen yläreunassa on ”välilehti”-kenttä (*engl. Tab*), joka ilmaisee tietopaketin tyypin. Jotkut tietopakettityypit (esim. IP-tietopaketti) voivat sisältää toisen tietopaketin. Tällöin lomakkeen yläreunassa on useita välilehtiä, joita klikkaamalla voidaan siirtyä tietopaketista toiseen. Vasemman puoleisin välilehti ilmaisee uloimman tietopaketin.



Kuva 19. Tietopaketin välilehti

Tietopaketti voidaan tallentaa ohjelman omassa formaatissa (ASCII) tiedostoon ja avata uudelleen editoitavaksi. Tietopaketti on myös mahdollista tallentaa binaarimuodossa. Binaarimuotoinen tallenne on verkon käyttämässä bittijärjestyksessä (*Big-Endian*, määräävin bitti ensimmäisenä [rfc791]), jolloin tallenteen voi lähettää sellaisenaan tai esim. *AX/4000*-mittalaitteen avulla verkkoon.

Sekvenssi *(engl. sequence*) on olio, joka voi sisältää yhden tai useamman tietopaketin. Tietopakettien muokkaaminen tapahtuu kaksoisklikkaamalla tietopaketin nimeä, jolloin näyttöön avautuu tietopaketin sivu, jonka arvoja käyttäjä voi muuttaa. Paketteja voidaan vapaasti lisätä ja poistaa. Niiden järjestystä voidaan myös muuttaa. Sekvenssin ensisijainen funktio on toimia eräänlaisena tietopakettikokoelmana. Tämä helpottaa esim. saman tietopaketin erilaisten versioiden säilyttämistä. Toinen funktio on tarjota mahdollisuus tuottaa *AX/4000*-ohjelmistolle valmis *sequence*-tiedosto, joka voidaan avata *AX/4000*-ohjelmistoon editoitavaksi.

PaGen voidaan assosioida käyttämiinsä tiedostopäätteisiin, jolloin klikattaessa tiedoston nimeä hakemistolistassa käynnistetään *PaGen* ja avataan tiedosto automaattisesti.

4.1.1 Liityntä AX/4000-ohjausohjelmistoon

PaGen-ohjelmalla luodut tietopaketit voidaan siirtää AX/4000-ohjelmistoon usealla eri menetelmällä. Kaikki menetelmät noudattavat seuraavaa kaavaa: paketti tai sekvenssi luodaan *PaGen*-ohjelmalla ja tallennetaan binaarimuodossa AX/4000-ohjelmaa varten.

Siirtomenetelmiä on kolme: 1) luodaan AAL5-CPCS-paketin kuorma, 2) luodaan AX4000 AAL5-PDU -paketti tai 3) luodaan sekvenssi AX/4000 AAL5-PDU -paketeista.

1) AAL5-CPCS-paketin kuorma (`AAL5_PAYLOAD_NON_ISO_PDU`) tallennetaan binaarimuodossa ja ladataan AX/4000-ohjelmassa AAL5-CPCS-PDU:n kuormaksi.

2) AX4000 AAL5-PDU -paketti tallennetaan myös binaarimuodossa, jolloin paketti tallennetaan AX/4000-ohjelman käyttämässä tiedostoformaatissa. Tällöin tallennettu tiedosto on käsiteltävissä kuten muutkin AX/4000-ohjelmiston kautta tallennetut AAL5 CPCS-PDU -paketit. Tiedosto voidaan ladata suoraan AAL5 CPCS-PDU -pakettina sekvenssin määrittelyn yhteydessä.

3) AX4000 AAL5-PDU -paketeista koostuva sekvenssi voidaan tallentaa AX/4000-ohjelman sekvenssiformaatissa. Tiedosto voidaan ladata AX/4000-ohjelman kautta normaalisti sekvenssinä ja sekvenssin editointia voidaan jatkaa.

AX/4000-ohjelman käyttämästä tiedostoformaatista ei ollut käytössä mitään spesifikaatiota, joten tiedostojen rakenne selvitettiin tutkimalla suoraan tiedostojen sisältöä. Kaikkia tiedoston kohtia ei pystytty tarkkaan selvittämään, joten täydellistä yhteensopivuutta ei voida taata.

4.2 Käyttöohjeet

Ohjelman käyttöohje on ohjelman yhteydessä konekielisessä muodossa. Ohje on Windows 95 standardikäyttöohjeen mukainen sisältäen linkityksen ohjeen osasta toiseen. Lisäksi ohjeesta voidaan etsiä tietoja käyttäjän määrittelemän hakusanan avulla. Käyttöohjetta voidaan lukea myös ilman, että *PaGen*-ohjelmaa käynnistetään. Ohje sisältää tekstin lisäksi myös asiaa havainnollistavia kuvia.

4.3 Arkkitehtuuri

4.3.1 Käyttöympäristö

Ohjelma vaatii joko Windows 95 (tai uudempi) tai Windows NT (versio 4.0 tai uudempi) -käyttöjärjestelmän. Ohjelma vie yhteensä n. 670 kB tilaa kovalevyllä ja ajettaessa n. 1MB riippuen kuinka monta eri sekvenssiä tai tietopakettia on käytössä. Käyttöjärjestelmässä ei tarvitse olla asennettuna mitään erikoiskirjastoja.

4.3.2 Ohjelmointiympäristö

Ohjelmointi on suoritettu käyttäen Microsoft Visual C++ 5.0 -ohjelmointiympäristöä. Toteutuksessa käytettiin hyväksi kehitysympäristön tarjoamia palveluja ja kirjastoja. Ohjelmaa ei tämän vuoksi voida kääntää muilla C++-kääntäjillä. Kehityskäyttöjärjestelmänä toimi Windows NT 4.0.

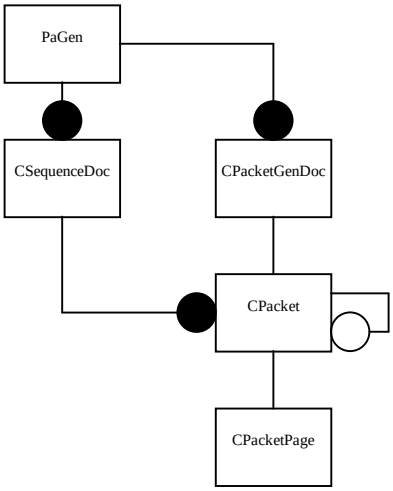
4.4 Toteutus

Pagen ohjelma on tehty olio-ohjelmoinnin periaatteiden mukaisesti. Toteutuksessa on käytetty perintää ja olioiden geneeristä käsittelyä sekä uudelleen määriteltäviä virtuaalimetodeja. Näin ohjelmasta on saatu joustava ja kohtuullisella työllä laajennettava.

4.4.1 Oliomalli

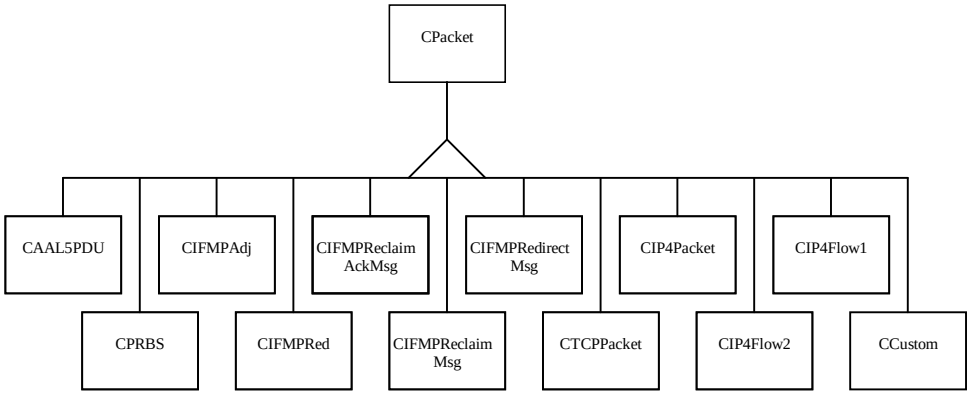
Jäsennettäessä ohjelman sisäistä rakennetta, on kuvaavinta käyttää oliomalleja. Malleihin ei ole sisällytetty kaikkia viittauksia MFC-kirjaston luokkiin, koska ne erittäin laajoja ja monimutkaistavat turhaa esitystä. Esitetyissä malleissa keskitymme olennaisimpiin kohtiin.

PaGen-ohjelman ydin muodostuu luokasta (`PaGen`), joka määrittelee sovelluksen ja huolehtii liitynnästä Windows-käyttöjärjestelmään. Luokalla voi olla nolla tai enemmän sekvenssi- (`CSequenceDoc`) ja tietopakettidokumenttia (`CSequenceDoc`). Tietopakettidokumentti sisältää yhden tietopaketin (`CPacket`), joka puolestaan sisältää yhden tietopakettisivun (`CPacketPage`). Lisäksi tietopaketilla voi rekursiivisesti olla alitietopaketti, jolla voi olla taas uusi alitietopaketti. Sekvenssi voi sisältää yhden tai monta tietopakettia, joiden ei tarvitse olla samanlaisia. Nämä tietopaketit voivat taas sisältää alipaketteja edellä kuvatusti. Edellä kuvattu relaatiomalli on esitetty kuvassa 20. Notaationa on käytetty OMT+-nootaatiota, joka on kuvattu liitteessä A [omt].

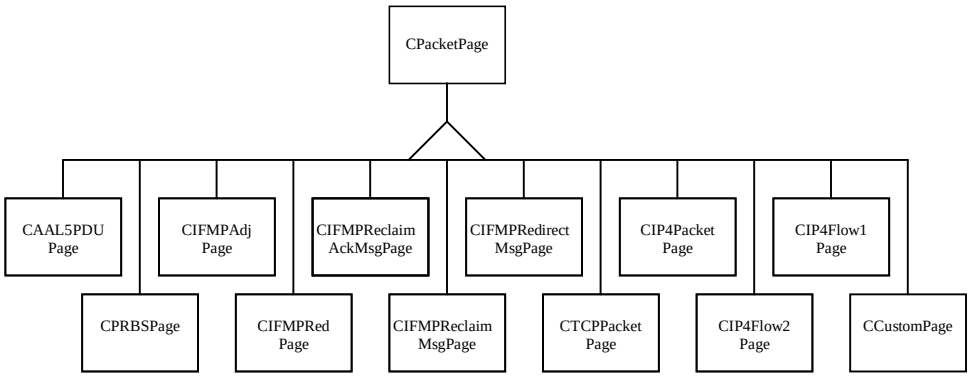


Kuva 20. Relaatiomalli

`CPacket`-luokka on abstraktio tietopaketista, joka kulkee tietoverkossa. Luokan avulla voidaan esittää erilaiset tietopaketit siten, että niitä pystytään geneerisesti käsittelemään. `CPacket`-luokkaan tai instanssiin viitataan yleisesti tässä dokumentissä käyttäen nimitystä *tietopaketti*. Tietopaketin ulkoasun abstraktio on puolestaan `CPacketPage`-luokka. Kuvassa 21 on esitetty `CPacket`-luokan perintäpuu ja kuvassa 22 `CPacketPage`-luokan perintäpuu.



Kuva 21. Packet-luokan perintä



Kuva 22. CPacketPage-luokan perintä

4.4.2 Käyttöliittymä

Ohjelman käyttöliittymä on rakennettu käyttäen Microsoft Visual C++ ja MFC-kirjastoa. Kirjasto tukee täysin Windows 95 -määrittelyä. Käyttöliittymän käyttämät ikkunointiympäristö komponentit ovat pitkälle standardeja, mutta joihinkin on jouduttu tekemään muutoksia. Tällaisia komponentteja ovat esimerkiksi puurakenne (*engl. TreeControl*) ja syöttöikkuna (*engl. ComboBox*).

Itse ohjelma on rakennettu käyttäen valmista monidokumenttisovelluksen (*engl. multiple document interface application, MDI*) runkoa. Tämä järjestelmä tarjoaa pitkälle valmiin ympäristön, jota voidaan

muokata halutuksi. Suurin muutos on tekstidokumenttipohjien muuttaminen *PaGen*-ohjelman käsittelemiksi tietopaketti- ja sekvenssipohjiksi.

Tietopakettien dokumenttipohja on tehty käyttäen ikkunointiympäristön komponenttia nimeltä `CPropertySheet`, jonka avulla voidaan siirtyminen tietopaketista toiseen hallita. Varsinainen tietopakettilomake on `CPropertyPage`-komponentti, jonka ulkoasu on määritelty Visual C++:n resurssieditorin avulla.

Jotta tietopakettisivujen päivittäminen olisi automaattista, on `CPacketPage`-luokkaan lisätty uusi viestienkäsittelijä, joka kutsuu tarvittavia päivitysrutiineja. Rutiinit puolestaan tunnistavat käsiteltävän komponentin ja suorittavat sen mukaiset päivitystoimenpiteet. Mikäli ohjelmaa halutaan laajentaa, siten että käytettävissä olisi laajempi valikoima ikkunointiympäristön ohjelmointikomponetteja, on em. päivitysrutiineja muutettava. Muualle ohjelmaan ei tarvitse tehdä muutoksia.

4.4.3 Tietopakettien kuvauskieli

Jokaisella tietopaketilla on joukko yksilöllisiä parametrejä kuten esimerkiksi pituus, versio, tunniste tai protokolla. Myös tietopaketin sisältämä data voidaan ajatella yhdeksi parametriksi tässä tapauksessa. Näiden parametrien merkitys vaihtelee riippuen tietopaketista ja tietopaketin muista parametreistä. Kuitenkin, riippumatta siitä minkä tietopaketin parametristä on kyse, yhteistä näille parametreille on, että niiden pituudet ja arvot voidaan määritellä. Tämä mahdollistaa parametrien käsittelyn tehokkaasti erilaisten algoritmien avulla. Algoritmit vaativat kuitenkin tarkan määritelmän parametristä, jota ne käsittelevät. Tähän on *PaGen*-ohjelmassa kehitetty erityinen kuvauskieli.

Kieli perustuu erilaisten makrojen ja tietorakenteiden käyttöön. Idea on seuraava. Kaikki parametrit sisältävät määrittelyn, joka kertoo miten tätä parametriä tulee käsitellä. Esimerkiksi tyyppi määrittelee kuinka parametri esitetään käyttöliittymässä ja kuinka sen arvon muutoksiin reagoidaan. Määrittely (`types.h::ValueDef`) sisältää seuraavat kohdat:

- tyyppi (`PACKET_VALUE`, `INT_VALUE`, `IP_VALUE`, `LIST_VALUE`, `TEXT_VALUE`)
- koko bitteinä
- tunnetut arvot ja niiden lukumäärä
- onko arvo yhtenäinen (= voiko parametri saada muita arvoja kuin tunnetut arvot?) ?

- voiko listan elementtien paikkoja muuttaa?
- tallennetaanko parametri tiedostoon?
- parametrin lyhyt ja pitkä nimi
- oletusarvo
- tiedostoon tulostettava avustusteksti
- voiko parametrin arvoa muuttaa?

Määrittely on rekursiivinen sillä tunnetulla arvolla voi olla uusi määrittely. Tämän avulla voidaan toteuttaa listatietorakenteita. Parametrit syötetään tietorakenteisiin erityisten makrojen avulla. Esimerkkinä on liitteessä C esitetty IP-tietopaketin parametrin määrittelyt.

Varsinainen parametrin tietorakenne (`types.h::Value`) sisältää viittauksen määrittelyyn ja eri tyyppisiä muuttujia: kokonaisluku, lista ja merkkijono. Parametrin arvo sijoitetaan johonkin näistä muuttujista riippuen parametrin tyypistä. Lisäksi tietorakenteessa on joukko apumuuttujia, jotka kertovat esimerkiksi onko arvon muuttunut.

4.4.4 Uuden tietopaketin lisääminen ohjelmaan

Suunnittelun alusta pitäen on pyritty huomioimaan uusien tietopakettien lisääminen. Samalla on pyritty yksinkertaistamaan ja automatisoimaan tietopaketin määrittely prosessia. Lopputulos ei kuitenkaan ajanpuutteen vuoksi ole aivan sitä mitä haluttiin vaan vielä joudutaan joitakin asioita tekemään manuaalisesti. Tarkoituksena oli, että tietopaketin koko visualisointi rakennettaisiin automaattisesti, mutta tällä hetkellä lomakkeen ulkoasu joudutaan määrittelemään käsin.

Uuden tietopaketin lisääminen koostuu seuraavista vaiheista (esitetty tiedostossa `packets/packets.h`):

1. Tietopakettiluokan määrittely
2. Tietopakettisivun määrittely
3. Tietopaketin lisääminen tunnettuihin paketteihin

4.4.4.1 Tietopakettiluokan määrittely

Uusi tietopakettiluokka peritään `CPacket`-luokasta käyttäen julkista perintää. Luokalle voidaan määrittää omia yksilöllisiä ominaisuuksia. Samoin luokan perittyjä metodeja voidaan uudelleen määritellä, jolloin tietopaketin toimintaa ja käsittelyä voidaan muuttaa. Taulukossa 1 on esitetty `CPacket`-luokan perityt metodit, jotka voidaan uudelleen määritellä.

Paluuarvo	Funktion nimi	Selitys
`unsigned int`	`GetChecksum()`	Laskee 1 komplementin summan kaikista parametreistä data pois luettuna.
`unsigned int`	`GetSize()`	Palauttaa tietopaketin kokonaiskoon.
`unsigned int`	`GetHeaderSize()`	Palauttaa otsikon koon.
`unsigned int`	`GetMaxDataSize()`	Palauttaa tietopaketin datan suurimman mahdollisen koon.
`Void`	`SetMaxSize(unsigned int bits)`	Asettaa tietopaketin suurimman mahdollisen koon.
`unsigned int`	`GetDataSize()`	Palauttaa datan koon.
`Void`	`SetDataSize(unsigned int size)`	Asettaa datan koon.
`Void`	`Update()`	Päivittää tietopaketin. Kutsutaan aina, kun tietopaketin arvoja on muutettu.
`Void`	`HeaderBinDump(CBitBuffer& buffer)`	Sijoittaa otsikon annettuun puskuriin binaarimuodossa.
`void`	`BinDump(CBitBuffer& buffer)`	Sijoittaa tietopaketin annettuun puskuriin binaarimuodossa.

Taulukko 1. CPacket-luokan uudelleen määriteltävät funktiot

Tietopaketille tulee määritellä parametrit ja niiden määritelmät. Lopuksi tietopaketti esitellään kaikille näkyväksi käyttäen `DECLARE_PACKET()`-makroa. Liitteessä B on IP-tietopaketin esittely kokonaisuudessaan sisältäen parametrimäärittelyt.

4.4.4.2 Tietopaketin sivun määrittely

Tietopaketille luodaan ulkoasu käyttäen MS Visual C++ ohjelmiston resurssieditoria. Editorilla luodaan dialogi (*engl. Dialog*), jonka tyypiksi valitaan `IDD_PROPPAGE_`. Osalle ohjelmointikomponenteista on määritelty valmiit käsittelyrutiinit, jolloin niitä käytettäessä ei tarvitse

huolehtia komponenttien päivittämisestä. Taulukossa 2 on esitetty nämä komponentit. Mikäli käytetään muita komponentteja, ohjelmaa muutettava siten, että ko. komponentteja osataan käsitellä.

Komponetti	**Kuvaus**	**Parametrin tyyppi, joka komponenttiin voidaan kytkeä**
`CStatic`	Staattinen teksti, jota käyttäjä ei voi muuttaa	`PACKET_VALUE`, `IP_VALUE`, `INT_VALUE`
`CComboBox`	Yleinen syöttöruutu, joka sisältää listan viimeksi syötetyistä arvoista sekä tunnetut arvot.	`PACKET_VALUE`, `IP_VALUE`, `INT_VALUE`
`CTreeControl`	Puurakenne, jota käytetään listojen ja vaihtelevan pituisten tietorakenteiden esittämiseen. Käyttäjä voi muuttaa tietoja, mikäli se on sallittu määrityksissä.	`LIST_VALUE`
`Cedit`	Vapaamuotoisen tekstin syöttöruutu.	`TEXT_VALUE`
`CListBox`	Tosi/epätosi -tyyppisten tietojen syöttöruutu.	`INT_VALUE`

Taulukko 2. Tuetut ohjelmointikomponentit.

Seuraavaksi luodaan sivulle luokka, joka aluksi tulee periä `CPropertyPage`-luokasta, mutta heti luomisen jälkeen perittävä luokka tulee manuaalisesti muuttaa `CPacketPage`-luokaksi. Tämä johtuu ohjelmointityökalun puutteellisesta ominaisuudesta käsitellä käyttäjän luomia luokkia. Käytännössä muuttaminen toteutetaan korvaamalla merkkijono `CPropertyPage` merkkijonolla `CPacketPage`.

Sivuluokalle tulee lisätä tapahtuma `WM_INITDIALOG` ja sijoittaa siihen koodi, joka kytkee sivulla käytetyt komponentit tietopaketin parametreihin. Lopuksi sivu esitellään kaikille näkyväksi käyttäen `DECLARE_PAGE()`-makroa. Liitteessä D on esitelty IP-tietopaketin `WM_INITDIALOG` -tapahtuma.

4.4.4.3 Tietopaketin lisääminen tunnettuihin tietopaketteihin

Jotta ohjelma tietää tietopaketin olemassaolosta, on tietopakettiluokka rekisteröitävä. Uuden tietopakettiluokan ja -sivun header-tiedostot tulee lisätä `#include`-direktiiveillä `packets/packets.h`-tiedostoon. Varsinainen rekisteröinti tehdään käyttäen `INSERT_PACKET()`-makroa. Rekisteröinnin jälkeen tietopaketti voidaan luoda käyttäen *PaGen*-ohjelman `New Packet` -komentoa.

4.4.5 Tietopaketin ja sekvenssin tallentaminen tiedostoon/leikepöydälle

Ohjelma tallentaa tietopaketin ja sekvenssin normaalisti ASCII-formaatissa. Tämä mahdollistaa tiedostojen käsittelyn myös muilla työkaluilla kuin *PaGen*-ohjelmalla, kuten käyttäen esim. tavallista tekstinkäsittelyohjelmaa. Jotta käyttäjä tietäisi miten tiedosto tulisi rakentaa, ohjelma tallentaa tiedostoon normaalin tiedon lisäksi myös avustetekstejä, joiden avulla voidaan tiedostoa käsitellä. Liitteessä E on esitetty tiedosto, johon on tallennettu IP-tietopaketti.

Mikäli käyttäjä kopioi tai leikkaa tietopaketin tai sekvenssin leikepöydälle, käytetään samaa formaattia kuin tallennettaessa tiedostoon.

4.4.6 Tietopaketin tallentaminen binaarimuodossa tiedostoon

Binaarimuotoinen tiedosto rakennetaan paketin kuvauksen mukaiseksi. Tämä tarkoittaa, että tiedosto sisältää paketin siinä muodossa kuin se on tietoverkossa. Ohjelmalla luodut paketti on siis mahdollista lähettää suoraan tietoverkkoon. Poikkeuksena tästä on AX/4000 AAL5-PDU -paketti, joka tallennetaan AX/4000-ohjelmiston käyttämässä formaatissa.

4.5 Testaus

Ohjelman toiminta on testattu sekä windows NT - että windows 95 -ympäristöissä. Testaus on sisältänyt käyttöliittymän, ohjeen ja tiedostojen käsittelyn sekä AX/4000-ohjelmistoliitynnän testauksen. Havaitut virheet korjattiin välittömästi.

Käyttöliittymä on testattu luomalla ja muokkaamalla mahdollisimman erilaisia paketteja sekä tallentamalla paketit tiedostoon ja lataamalla ne uudelleen työpöydälle. AX/4000-liitynnän toimivuus on testattu tallentamalla paketteja ja sekvenssejä binaarimuodossa tiedostoon ja lataamalla ne AX/4000-ohjelmistosta käsin.

Tietopakettien sisällön testausta on tehty tarkastelemalla binaarimuodossa tallennettuja tiedostoja. Tarkoitus oli testata ohjelmistoa todellisessa ympäristössä, mutta teletekniikan laboratorion laitteiston päivittämisen pitkittymisestä johtuen tätä ei päästy tekemään.

4.6 Jatkokehitys

PaGen on pyritty suunnittelemaan mahdollisimman yleiseksi erilaisten tietopakettien tuottamisympäristöksi. Tämä tarkoittaa, sitä, että uusien erilaisten tietopakettityyppien lisääminen ohjelmaan on kohtuullisen vaivatonta. Nykyisellään tietopakettityyppien määrä on vielä aika suppea, joten tämä vaatisi vielä lisäpanostusta.

Ohjelma pystyy tällä hetkellä tuottamaan AX/4000-ohjelman käyttämiä tiedostoformaatteja. Työn tekemisen aikana ei kuitenkaan ollut käytössä tarkkoja spesifikaatioita formaateista, joten formaattien rakenne täytyi tutkia empiirisesti tiedostojen perusteella. Tämän johdosta ei voida taata, että tiedostot olisivat täysin yhteensopivia joka tilanteessa. Mikäli spesifikaatiot saadaan käyttöön, olisi syytä tarkistaa ohjelman toiminta tältä osin.

`Custom`-tietopaketti lisättiin ohjelmaan aivan loppuvaiheessa ja niinpä siitä puuttuu vielä monia toivottavia ominaisuuksia, kuten kentän tyhjentäminen ja tiedon lataaminen tiedostosta sekä merkkien editoiminen muussa kuin ASCII-formaatissa. Jotta tietopaketin käyttöarvo olisi paras mahdollinen olisi em. toimenpiteet lisättävä.

IP- ja TCP-tietopaketin optioita ei tässä ohjelmaversiossa tueta. Ne pystytään toteuttamaan nykyisillä ohjelman rakenteilla, mutta joitain kohtia käyttöliittymässä tulisi hioa.

PRBS-signaali ei tällä hetkellä ota huomioon kohtaa, jossa se alkaa paketista. Tämä voi johtaa ongelmiin mikäli lähetyksen vastaanottaja yrittää synkronoitua ko. signaaliin. Tätä täytyisi tutkia lisää ja testata mittalaitteiden käyttäytyminen ko. tilanteessa.

5. Yhteenveto

PaGen-ohjelman lähtökohtana oli rakentaa ohjelma, jolla voidaan TCP- ja IFMP-liikennettä tuottaa AX/4000-mittalaitteelle. Suunnittelun aikana huomattiin, että pienellä lisätyöllä voidaan toteuttaa ohjelmisto, joka pystyy myös laajempaan pakettien muokkaamiseen. Lopputuloksena syntyi ohjelma, joka tukee useita eri paketti tyyppejä ja jolla voidaan tuottaa AX/4000-ohjelmiston käyttämiä tiedostoformaatteja.

Työn teon aikana ei kuitenkaan ollut käytettävissä tarkkoja spesifikaatioita em. tiedostojen rakenteesta, joten täyttä yhteensopivuutta joka tilanteessa ei voida taata. Testatuissa tilanteissa ei kuitenkaan ongelmia ilmaantunut.

Ohjelman on tällä hetkellä toimiva kokonaisuus, jonka käyttöarvoa laajemman pakettivalikoiman lisääminen vahvistaisi huomattavasti. Uuden pakettityypin lisääminen on otettu huomioon suunnittelussa, jolloin työmääräkään ei olisi suuri.

Lähteet

[axdoc] AX/4000 Series ATM Generator/Analyzer Operating Manual. Adtech Inc. 1994.

[axlib] AX/4000 C Function Library. Adtech Inc. 1997. CRC.c.

[omt] Kuusela & Aalto. OMT+ Guide. Opetusmoniste.Teknillinen korkeakoulu. 1994. 11s.

[rfc791] RFC 791. Internet Protocol DARPA Internet Program Protocol Specification. Information Sciences Institute. University of Southern California, 1981.

[rfc793] RFC 793. Transmission Control DARPA Internet Program Protocol Specification. Information Sciences Institute. University of Southern California, 1981.

[rfc1071] RFC 1071. Computing the Internet Checksum. BBN Laboratories, 1988.

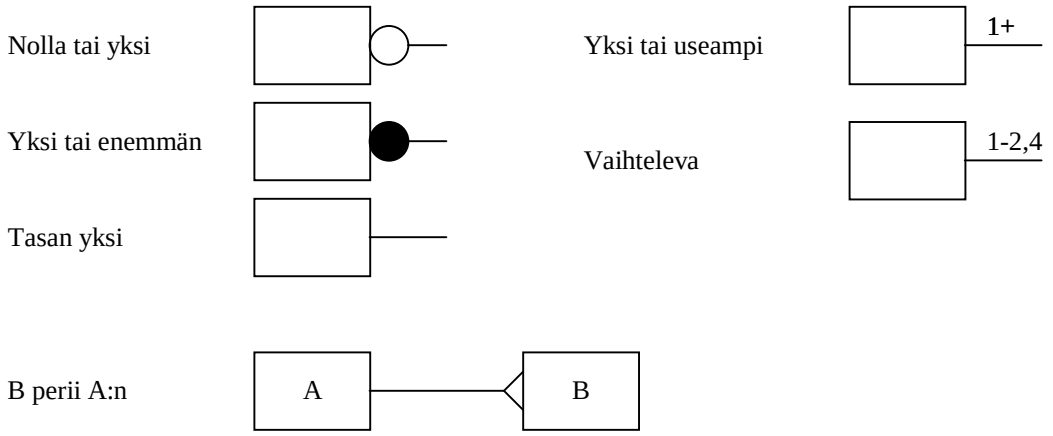
[rfc1483] RFC 1483. Multiprotocol Encapsulation over ATM Adaption Layer 5. Telecom Finland, 1993.

[rfc1626] RFC 1626. Default IP MTU for use over ATM AAL5. Naval Research Laboratory, 1994.

[rfc1953] RFC 1953. Ipsilon Flow Management Protocol Specification for IPv4. 1996.

Liitteet

Liite A: OMT+-notaatio



Kuva 23. OMT+-notaatio

Liite B: IP-tietopaketin esittely

```
#if !defined(AFX_IPPACKET_H)
#define AFX_IPPACKET_H

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// IP4Packet.h : header file
//

#include "macros.h"
#include "Packet.h"
#include "..\types.h"

/////////////////////////////////////////////////////////////////////////////
// CIP4Packet

#define IP4_NAME "IPv4"
#define IP4_DESC "IP version 4 -packet"
#define IP4_DEF_SIZE          576                                   // See RFC 791, page 13
#define TCP_PROTOCOL          6
#define IFMP_PROTOCOL         101

// Global defines
DECLARE_DEF(IP4VersionDef);
DECLARE_DEF(IP4IHLDef);
DECLARE_DEF(IP4TimeToLiveDef);
DECLARE_DEF(IP4ProtocolDef);
DECLARE_DEF(IP4SrcDef);
DECLARE_DEF(IP4DestDef);
DECLARE_DEF(IP4TimeToLiveDef);
DECLARE_DEF(IP4TypeOfServiceDef);
DECLARE_DEF(IP4PrecDef);
DECLARE_DEF(IP4DBitDef);
DECLARE_DEF(IP4TBitDef);
DECLARE_DEF(IP4RBitDef);
DECLARE_DEF(IP4Res1BitDef);
DECLARE_DEF(IP4Res2BitDef);
DECLARE_DEF(IP4LengthDef);
DECLARE_DEF(IP4IdDef);
DECLARE_DEF(IP4Res3BitDef);
DECLARE_DEF(IP4DFBitDef);
DECLARE_DEF(IP4MFBitDef);
DECLARE_DEF(IP4FragmentDef);
DECLARE_DEF(IP4ChecksumDef);
DECLARE_DEF(IP4DataDef);

// define values used in packet
BEGIN_DEFINE_VALUES(CIP4Packet)
          IP4VersionValue,
          IP4IHLValue,
          // Type of Service
                    IP4PrecValue,
                    IP4DBitValue,
                    IP4TBitValue,
                    IP4RBitValue,
                    IP4Reserved1BitValue,
                    IP4Reserved2BitValue,
          IP4LengthValue,
          IP4IdValue,
          //Flags
                    IP4Reserved3BitValue,
                    IP4DFBitValue,
                    IP4MFBitValue,
          IP4FragmentValue,
          IP4TimeToLiveValue,
          IP4ProtocolValue,
          IP4ChecksumValue,
          IP4SrcValue,
          IP4DestValue,
          IP4OptionsValue,
          IP4PaddingValue,
          IP4DataValue
END_DEFINE_VALUES(CIP4Packet)

class CIP4Packet : public CPacket
{
  public:
          CIP4Packet();
          ~CIP4Packet();

          // Overloaded members
          void SetDataSize(unsigned int size);
          void Update();                                                          // Values has been changed
};

DECLARE_PACKET(CIP4Packet);


#endif // !defined(AFX_IPPACKET_H)
```

Liite C: IP-tietopaketin parametrimäärittely

IPPacket.cc:

```
//////////////////////////////////////////////////
// Define types of values

// Version
BEGIN_VALUE_DEF(IP4VersionDef)
          SINGLE(4, DEFAULT_STRING),
END_VALUE_DEF(IP4VersionDef, INT_VALUE, 4, UNCONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                                "Version",        "version",          0,        "Version of internet protocol", TRUE);


// IHL
BEGIN_VALUE_DEF(IP4IHLDef)
          SINGLE(5, DEFAULT_STRING),
END_VALUE_DEF(IP4IHLDef, INT_VALUE, 4, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                                "IHL",            "Internet Header Length",     0,        "Internet Header Length", TRUE);

// Predence
BEGIN_VALUE_DEF(IP4PrecDef)
          SINGLE(0, "Routine"),
          SINGLE(1, "Priority"),
          SINGLE(2, "Immediate"),
          SINGLE(3, "Flash"),
          SINGLE(4, "Flash Override"),
          SINGLE(5, "CRITIC/ECP"),
          SINGLE(6, "Internetwork Control"),
          SINGLE(7, "Network Control"),
END_VALUE_DEF(IP4PrecDef, INT_VALUE, 3, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                                "Prec", "TOS: Precedence",  0,        "Type of Service: Precedence", TRUE);

// Total Length
BEGIN_VALUE_DEF(IP4LengthDef)
          SINGLE(576,          DEFAULT_STRING),
END_VALUE_DEF(IP4LengthDef, INT_VALUE, 16, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                                "Length",         "Length of DataGram",         0,        "Total length of datagram", TRUE);

// Identification
BEGIN_VALUE_DEF(IP4IdDef)
          SINGLE(0, DEFAULT_STRING),
END_VALUE_DEF(IP4IdDef, INT_VALUE, 16, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                                "Id",             "Identification",   0,        "Identification", TRUE);

// Fragment offset
BEGIN_VALUE_DEF(IP4FragmentDef)
          SINGLE(0, DEFAULT_STRING),
END_VALUE_DEF(IP4FragmentDef, INT_VALUE, 13, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                                "Frag", "Fragment Offset",  0,        "Fragment Offset", TRUE);

// Time to Live
BEGIN_VALUE_DEF(IP4TimeToLiveDef)
          SINGLE(32,           DEFAULT_STRING),
END_VALUE_DEF(IP4TimeToLiveDef, INT_VALUE, 8, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                                "TTL",  "Time to Live",     0,        "Time to Live", TRUE);

// Protocol
BEGIN_VALUE_DEF(IP4ProtocolDef)
          SINGLE(0, "IP"),
          SINGLE(1, "ICMP"),
          SINGLE(2, "IGMP"),
          SINGLE(3, "GGP"),
          SINGLE(TCP_PROTOCOL,          "TCP"),
          SINGLE(8, "EGP"),
          SINGLE(12,           "PUP"),
          SINGLE(17,           "UDP"),
          SINGLE(20,           "HMP"),
          SINGLE(22,           "XNS-IDP"),
          SINGLE(27,           "RDP"),
          SINGLE(66,           "RVD"),
          SINGLE(IFMP_PROTOCOL,         "IFMP"),
END_VALUE_DEF(IP4ProtocolDef, INT_VALUE, 8, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                                "Proto",          "Protocol",         0,        "Protocol", TRUE);

// Header Checksum
BEGIN_VALUE_DEF(IP4ChecksumDef)
          SINGLE(0, ""),
END_VALUE_DEF(IP4ChecksumDef, INT_VALUE, 16, CONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                                "Checksum", "Checksum",     0,        "Header Checksum", TRUE);

// Data
BEGIN_VALUE_DEF(IP4DataDef)
          SINGLE(NO_PACKET,   EMPTY_NAME),
          SINGLE(TCP_ID,                TCP_NAME),
          SINGLE(IFMPAdj_ID,  IFMPAdj_NAME),
          SINGLE(IFMPRed_ID,  IFMPRed_NAME),
          SINGLE(PRBS_ID,               PRBS_NAME),
          SINGLE(Custom_ID,   CUSTOM_NAME),
END_VALUE_DEF(IP4DataDef, PACKET_VALUE, 0, UNCONNECTED_REGION,
                                CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                                "Data", "Data",   0,        "Payload packet", TRUE);
```

```
// bits
BEGIN_VALUE_DEF(IP4DBitDef)
          SINGLE(FALSE_ID,    ""),
          SINGLE(TRUE_ID,               ""),
END_VALUE_DEF(IP4DBitDef, INT_VALUE, 1, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "D",    "TOS: Delay",       0,        "Type of Service: Delay", TRUE);

BEGIN_VALUE_DEF(IP4TBitDef)
          SINGLE(FALSE_ID,    ""),
          SINGLE(TRUE_ID,               ""),
END_VALUE_DEF(IP4TBitDef, INT_VALUE, 1, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "T",    "TOS: Throughput",            0,        "Type of Service: Throughput", TRUE);

BEGIN_VALUE_DEF(IP4RBitDef)
          SINGLE(FALSE_ID,    ""),
          SINGLE(TRUE_ID,               ""),
END_VALUE_DEF(IP4RBitDef, INT_VALUE, 1, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "R",    "TOS: Reliability",           0,        "Type of Service: Reliability", TRUE);

BEGIN_VALUE_DEF(IP4DFBitDef)
          SINGLE(FALSE_ID,    ""),
          SINGLE(TRUE_ID,               ""),
END_VALUE_DEF(IP4DFBitDef, INT_VALUE, 1, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "DF",   "Flags: Don't Fragment",      0,        "Flags: Don't Fragment", TRUE);

BEGIN_VALUE_DEF(IP4MFBitDef)
          SINGLE(FALSE_ID,    ""),
          SINGLE(TRUE_ID,               ""),
END_VALUE_DEF(IP4MFBitDef, INT_VALUE, 1, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "MF",   "Flags: More Fragments",      0,        "Flags: More Fragments", TRUE);


BEGIN_VALUE_DEF(IP4Res1BitDef)
          SINGLE(FALSE_ID,    ""),
END_VALUE_DEF(IP4Res1BitDef, INT_VALUE, 1, UNCONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                              "Res1", "TOS: Reserved1",   0,        "Type of Service: Reserved bit 1", TRUE);

BEGIN_VALUE_DEF(IP4Res2BitDef)
          SINGLE(FALSE_ID,    ""),
END_VALUE_DEF(IP4Res2BitDef, INT_VALUE, 1, UNCONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                              "Res2", "TOS: Reserved2",   0,        "Type of Service: Reserved bit 2", TRUE);

BEGIN_VALUE_DEF(IP4Res3BitDef)
          SINGLE(FALSE_ID,    ""),
END_VALUE_DEF(IP4Res3BitDef, INT_VALUE, 1, UNCONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                              "Res3", "Flags: Reserved",  0,        "Flags: Reserved bit", TRUE);


// IP Source Address
BEGIN_VALUE_DEF(IP4SrcDef)
          SINGLE(0, "0.0.0.0"),
END_VALUE_DEF(IP4SrcDef, IP_VALUE, 32, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "SrcAddr",        "Source Address",   0,        "IP Address of the source ", TRUE);

// IP Destination Address
BEGIN_VALUE_DEF(IP4DestDef)
          SINGLE(0, "0.0.0.0"),
END_VALUE_DEF(IP4DestDef, IP_VALUE, 32, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "DestAddr",       "Destination Address",        0,        "IP Address of the destination", TRUE);

// Options
BEGIN_VALUE_DEF(IP4OptionsDef)
          SINGLE(0, DEFAULT_STRING),
END_VALUE_DEF(IP4OptionsDef, INT_VALUE, 0, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                              "Options",        "Options",          0,        "Options of the packet", TRUE);

// Padding
BEGIN_VALUE_DEF(IP4PaddingDef)
          SINGLE(0, DEFAULT_STRING),
END_VALUE_DEF(IP4PaddingDef, INT_VALUE, 0, CONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, DO_NOT_SAVE_TO_FILE,
                              "Padding",        "Padding",          0,        "Padding", TRUE);

// Type Of service
BEGIN_VALUE_DEF(IP4TypeOfServiceDef)
          LIST(0, IP4PrecDef),
          LIST(1, IP4DBitDef),
          LIST(2, IP4TBitDef),
          LIST(3, IP4RBitDef),
          LIST(4, IP4Res1BitDef),
          LIST(5, IP4Res2BitDef),
END_VALUE_DEF(IP4TypeOfServiceDef, LIST_VALUE, 32, UNCONNECTED_REGION,
                              CAN_NOT_CHANGE_ORDER, SAVE_TO_FILE,
                              "TOS",  "Type of Service",  0,        "Type of Service", TRUE);
```

Liite D: IP-tietopakettisivun `WM_INITDIALOG`-tapahtuma

IP4Page.cpp:

```
BOOL CIP4Page::OnInitDialog()
{

        CPacketPage::OnInitDialog();
        ClearMap(ControlToValue);

        MAP_CONTROL_TO_VALUE(m_VersionStatic, IP4VersionValue);
        MAP_CONTROL_TO_VALUE(m_IHLStatic,     IP4IHLValue,   );
        MAP_CONTROL_TO_VALUE(m_DataCombo,     IP4DataValue);
        MAP_CONTROL_TO_VALUE(m_SourceCombo,   IP4SrcValue);
        MAP_CONTROL_TO_VALUE(m_DestCombo,     IP4DestValue);
        MAP_CONTROL_TO_VALUE(m_ProtocolCombo, IP4ProtocolValue);
        MAP_CONTROL_TO_VALUE(m_PredenceCombo, IP4PrecValue);
        MAP_CONTROL_TO_VALUE(m_TOSList,              IP4DBitValue);
        MAP_CONTROL_TO_VALUE(m_TOSList,              IP4TBitValue);
        MAP_CONTROL_TO_VALUE(m_TOSList,              IP4RBitValue);
        MAP_CONTROL_TO_VALUE(m_LengthStatic,  IP4LengthValue);
        MAP_CONTROL_TO_VALUE(m_IdCombo,              IP4IdValue);
        MAP_CONTROL_TO_VALUE(m_FlagsList,     IP4DFBitValue);
        MAP_CONTROL_TO_VALUE(m_FlagsList,     IP4MFBitValue);
        MAP_CONTROL_TO_VALUE(m_FragCombo,     IP4FragmentValue);
        MAP_CONTROL_TO_VALUE(m_LiveCombo,     IP4TimeToLiveValue);
        MAP_CONTROL_TO_VALUE(m_ProtocolCombo, IP4ProtocolValue);
        MAP_CONTROL_TO_VALUE(m_ChecksumStatic,IP4ChecksumValue);
        MAP_CONTROL_TO_VALUE(m_OptionsStatic,IP4OptionsValue);
        MAP_CONTROL_TO_VALUE(m_PaddingStatic,IP4PaddingValue);

        return TRUE;  // return TRUE unless you set the focus to a control
                      // EXCEPTION: OCX Property Pages should return FALSE
}
```

Liite E: IP-tietopaketti tallennettuna tiedostoon

```
# PaGen version 1.0
# Root of the file (Choices: Sequence, AAL5_PDU, IPv4, IPv4_IFMP_FlowType_1_ATM, IPv4_IFMP_FlowType_2_ATM, TCP,
IFMP_AdjMsg, IFMP_RedMsg, IFMP_RedirectMsg, IFMP_ReclaimMsg, IFMP_ReclaimAckMsg, PRBS, Custom)
IPv4 {
        # IP version 4 -packet
        # (Values: Prec, D, T, R, Id, DF, MF, Frag, TTL, Proto, SrcAddr, DestAddr, Data)
        Prec=0           # Type of Service: Precedence (Choices: 0 - 7, DEFAULT)
        D=0              # Type of Service: Delay (Choices: 0 - 1, DEFAULT)
        T=0              # Type of Service: Throughput (Choices: 0 - 1, DEFAULT)
        R=0              # Type of Service: Reliability (Choices: 0 - 1, DEFAULT)
        Id=0             # Identification (Choices: 0 - 65535, DEFAULT)
        DF=0             # Flags: Don't Fragment (Choices: 0 - 1, DEFAULT)
        MF=0             # Flags: More Fragments (Choices: 0 - 1, DEFAULT)
        Frag=0           # Fragment Offset (Choices: 0 - 8191, DEFAULT)
        TTL=32           # Time to Live (Choices: 0 - 255, DEFAULT)
        Proto=0          # Protocol (Choices: 0 - 255, DEFAULT)
        SrcAddr=0.0.0.0  # IP Address of the source  (Choices: 0.0.0.0 - 255.255.255.255, DEFAULT)
        DestAddr=0.0.0.0 # IP Address of the destination (Choices: 0.0.0.0 - 255.255.255.255, DEFAULT)
        Data=EMPTY       # Payload packet (Choices: EMPTY, TCP, IFMP_AdjMsg, IFMP_RedMsg, PRBS, Custom, DEFAULT)
}
```