SZAKDOLGOZAT

Lefter János

Debrecen
2008

Debreceni Egyetem
Informatika Kar

Térképező modul fejlesztése hálózati menedzsment rendszerhez.

Témavezető:
Gál Zoltán
DE TEK ITK Igazgató

Készítette:
Lefter János
Porgramozó Matematikus

Debrecen
2008

Köszönetnyilvánítás

Ez úton szeretnék köszönetet mondani Gál Zoltánnak, a Debreceni Egyetem Információtechnológiai Központ igazgatójának az egész munka során nyújtott segítségéért, hasznos ötleteiért és útmutatásáért.

Köszönöm Dr. Fazekas Attilának, a Debreceni Egyetem Informatikai Kar Komputergrafikai és Képfeldolgozás Tanszék docensének, hogy izgalmas, és érdekfeszítő előadásaival új utakat nyitott előttem.

Szeretném megköszönni Dr. Juhász Istvánnak, az Információ Technológia Tanszék egyetemi adjunktusának, hogy programozási alapvetéseivel útmutatást adott kódolási nehézségeim megoldására.

Továbbá köszönöm Kovács Sándor tanársegédnek, hogy gyakorlatain elsajátíthattam az objektum orientált programozás fortélyait, és szemléletét.

Köszönöm még Prepuk János hálózatüzemeltető kollégámnak a szakmai, gyakorlati munkában nyújtott segítségét.

Végül szeretném külön megköszönni családomnak, hogy támogatásukkal és türelmükkel hozzásegítettek dolgozatom elkészítéséhez.

Tartalomjegyzék

1. fejezet

Bevezetés

Mint minden más szolgáltatási ágazatban, így az internet szolgáltató cégeknél is az egyik legfőbb szempont az ügyfelek igényeinek megfelelő, jó minőségű és zavar mentes szolgáltatás nyújtása. Ennek alapvető technikai követelménye a jól felépített és átlátható hálózat, amelyben az esetlegesen felmerülő zavarok, hibák gyorsan felismerhetők és elháríthatók.
Az internetes hálózatok rohamos növekedésének következtében még nagyobb hansúlyt kell fektetni a azok monitorozására, menedzselésére, karbantartására. Amíg néhány éve jól képzett szakemberekből álló kis létszámú csapat szinte minden feladatot el tudott látni viszonylag nagy kiterjedésű hálózatokon, addig napjainkban a passzív, illetve aktív elemek számának jelentős növekedése, és a szolgáltatások kiszélesedésének következtében, a speciális feladatokra különböző csoportokat kell létrehozni. Az eltérő képességű és képzettségű, más-más szakterületen dolgozó emberek jól összehangolt munkájával lehet csak jó minőségű, üzembiztos szolgáltatást nyújtani. A szolgáltatás műszaki hátterét biztosító csoportok munkájának megkönnytéséhez, és összehangolásához elengedhetetlen feltétel egy közös műszaki felület megteremtése.

1.1. A témaválasztás indoklása

A szakdolgozatom témájául egy hozzám közel álló, lehetőség szerint hétköznapi valós probléma megoldását választottam. Erre lehetőségem nyílt, mivel hálózat üzemeltetőként dolgozom a Digi Zrt. internet szolgáltató vállalatnál, és a feladatkörömbe tartozik a hálózatmenedzsmentet segítő eszközök fejlesztése.

A már nálunk használt, és egyik kollégám által fejlesztett adatbázis alapú, webes felületű több funkciós menedzsment szoftverhez kellett egy bővítményt készítenem. Így a már meglévő rendszerhez igazodva a következő környezetben kellett dolgoznom: Linux operációs rendsze-

ren futó Apache webszerver, MySql adatbázis kezelő rendszer, valamint PHP5 programnyelvi környezet. Mivel multinacionális vállalatnál dolgozom, amely Magyarország területén több nagyvárosban is jelen van internetszolgáltatóként, nagyon fontos szempont volt, hogy bárhol használni lehessen az általam elkészített terméket. A másik szempont a továbbfejleszthetőség, és rugalmas használat, ezért igyekeztem a PHP5 nyújtotta objektum orientált lehetőségeket kihasználva, könnyen paraméterezhető és beilleszthető, jól dokumentált szoftvermodult írni. E bővítmény feladata, a városi szolgáltatási területen lévő aktív eszközök kapcsolódási hierarchiájának feltérképezése.

Eredetileg a technikusok feladata volt azt dokumentálni, hogy egy újonnan beüzemelt switch mely másik kapcsolóhoz csatlakozik. Ezt viszonylag pontosan végezték az arra kijelölt szakemberek, viszont az üzemeltetés folyamán bekövetkezett változások dokumentálása már gyakran elmaradt. Így sosem alakulhatott ki pontos kép, illetve olyan precíz nyílvántartás amely a valós kapcsolódási hierarchiát reprezentálná.

A debreceni rendszerben üzemelő aktíveszközök ezres nagyságrendű száma megköveteli a hálózat hierarchiájának automatikus nyílvántartását. Ez elengedhetetlen a hálózattechnikus csoport műszaki támogatásához, és a rendszerben felmerülő hibák, és egyéb információk keresésének meggyorsításához. Így merült fel az igény annak a programmodulnak a kifejlesztésére amelyet a szakdolgozatomban bemutatok.

A webes alkalmazások fejlesztése terén már szereztem némi tapasztalatot, viszont a PHP 5 adta új lehetőségeket ez eddig nem volt alkalmam kipróbálni. Tulajdonképpen az általam írt modulban nem a webes környezetben való alkalmazáson van a hangsúly azon túl, hogy HTML formátumú kimenetet is képes generálni. A hálózati programozás is egy járatlan terület volt számomra és izgalmas, kihívásokkal teli feladatnak ígérkezett. A fejlesztés során igen sok új ismeretet szereztem. A tervezésen és programozáson kívül jelentős időt arra kellett fordítanom, hogy a nálunk alkalmazott hálózati eszközök speciális tulajdonságait, és lehetőségeit megismerjem. Nagyrészt csak idegen nyelvű dokumentációkra támaszkodhattam, mind nyomtatott mind elektronikus formában. Sajnos nem minden esetben sikerült „szép” megoldást találnom egy-egy problémára, de ezekre majd külön kitérek a dolgozatom során.

Mivel a hálózat üzemeltetési feladatok ellátásához, a biztonsági előírások, és adatvédelmi intézkedések betartása érdekében titoktartási kötelezettséget vállaltam, ezért a dolgozatom néhány pontján csak általános információkat közölhetek.

A célomat sikerült megvalósítanom: egy jól működő, gyakorlatban használt programot írni, amellyel jónéhány ember - többek között a magam - munkáját sekítek megkönnyíteni. Ezért tartom érdemesnek ezt a témát szakdolgozat formájában ismertetni, valamint a fejlesztési munka

során megszerzett tapasztalataimat, és ismereteimet kívánom megosztani.

1.2. Néhány szó a fejlesztői környezetről

Fontosnak tartom, hogy néhány szóban kitérjek a fejlesztői környezetre. Mint már azt írtam Linux operációs rendszeren és rendszerre fejlesztettem. Ez azért fontos mert alkalmaztam néhány olyan eszközt amelyet más ismertebb operációs rendszerek nem nyújtanak. A monitorozó rendszer webes felületét Apache 2 webszerver szolgálja ki. A hálózati nyílvántartás adatbázisa a Mysql 5-ös verziójú adatbázis szerverén működik. Az adatbázist nem én terveztem, nekem már a kész szerkezethez kellett alkalmazkodnom, de egy-két módosítást javasoltam, és igyekeztem nem eltérni a standard SQL szintaktikától. A programozási nyelvi környezet, mint azt már említettem PHP5 szerver oldali szkript nyelv. Az 5-ös verzió által nyújtott, OO paradigma adta lehetőségeket kihasználva alkítottam ki a program szerkezetét.

Az Eclipse IDE grafikus fejlesztői környezetet használtam PDT, és WTP pluginnal kiegészítve. A program dokumentációját a PHP Documentor eszközrendszerrel készítettem el.

Munkakörömből kifolyóan abban a szerencsés helyzetben vagyok, hogy az általam írt program modult valós környezetben tudtam tesztelni, így a lehetséges problémák hamar felszínre kerültek a fejlesztés során.

2. fejezet

Az internet hozzáférést biztosító nagyvárosi hálózat.

A Digi Zrt. debreceni területén internet szolgáltatás tulajdonképpen egy nagy kiterjedésű ethernet hálózaton történik PPPoE protokol használatával. A szolgáltatási területet mérete alapján MAN (Metropolitan Area Network) hálózatnak tekinthetjük. Struktúrája hasonló egy nagyvállalat, illetve campus ethernet hálózatához csak a jelentős távolságok miatt a fizikai rétegben nagyságrendekkel több optikai szál, és adatátviteli eszköz van jelen. Az ügyfelek nagy számából kifolyóan (több ezer hoszt) pedig jóval több aktív kapcsoló eszközre, úgynevezett switchre van szükség.

Természetesen olyan autentikációt, és accoutingot ellátó berendezéseket is tartalmaz a rendszer, amelyek nem feltétlenül részei egy vállalati környezetnek.

A szolgáltatási terület hierarhikusan több alrendszerre van bontva. A felosztást egyrészt meghatározzák a használt eszközök és technológiák, valamint az internet szolgáltatási terület fizikai jellemzői.

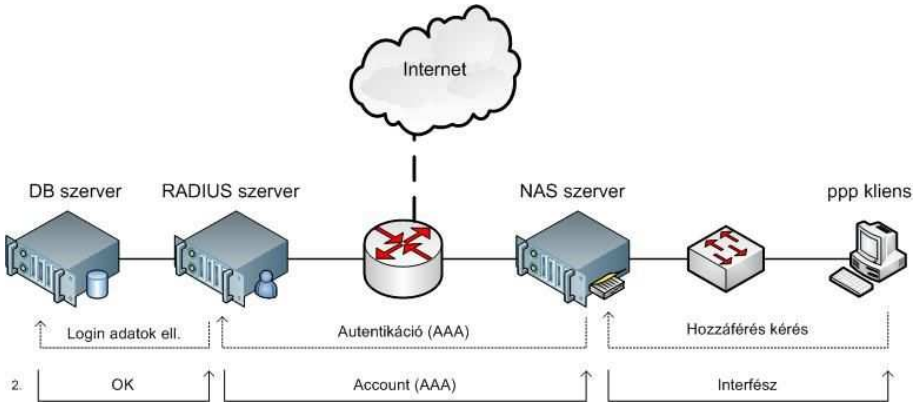
Az előzőek szerint a városi hálózatot logikai szinten is érdemes, felosztani több különálló rétegre, illetve alhálózatra. Ezt a rétegződést természetesen a területi adottságokon kívül, befolyásolja az előfizetők eloszlása a településen belül. Az adott szegmenst működtető aktíveszközök számítási kapacítása korlátozhatja a kiszolgálható hosztok számát. Itt arra gondolok, hogy milyen sávszélességű gerinc kapcsolatot tudunk egy-egy területhez rendelni, a hálózati forgalmat biztosító Layer 2-es eszközök mennyi fizikai címet, a Layer 3-as eszközök pedig mennyi ip címet, vagy tartományt tudnak biztonságosan kezelni, valamint az autentikációs rendszert hogyan tudjuk szervezni.

Ahogyan már a fentiekben azt említettem, a PPPoE rendszerben az előfizetők internet hoz-

záférésének szabályozásához speciális eszközrendszerre van szükség. Ezen összetevőkről, illetve berendezésekről csak egy vázlatos leírást adok, mivel nem állnak szoros összefüggésben az általam kiemelt témával.

2.1. Az előfizetői autentikáció és accounting megvalósítása

A PPPoE (Point-to-Point Protocil Over Ethernet) protokoll pont-pont kapcsolatot létesít ethernet hálózat fölött, a ppp keretek ethernet keretekbe való csomagolásával. Az ethernet hálózatra kapcsolódó hoszt a PPPoE protokoll használatával egy virtuális „betárcsázási" folyamaton keresztül létesít pont-pont kapcsolatot a kiszolgálóval, amely internet elérést biztosít számára. Felvázolok egy lehetséges több rétegű kliens-szerver architektúrát amelyet a 2.1 ábra szemléltet.



2.1. ábra. Hoszt internet hozzáférése PPPoE hálózaton

Első lépésben az előfizetői hálózati szegmensből a hoszt kliensként kapcsolatfelvételt kezdeményez a NAS[1] szerverrel. A NAS szerver feladata az IP cím kiosztása, valamint az internet felé történő hozzáférés biztosítása, és a forgalom sávszélesség szabályozása a kliens számára. A hoszt kapcsolódási kérelmének elfogadásához a NAS szintén kliensként kapcsolódik egy RADIUS[2] szerverhez amely a felhasználók azonosítását, engedélyezését vagy tiltását, és szabályozását látja el. A RADIUS internet szolgáltatók, és egyéb hálózatot üzemeltető vállalatok által széles körben használt hozzáférés menedzsment megoldás.

A RADIUS szintén kliensként kapcsolódik egy adatbázishoz, amely az előfizetők hozzáférésének megállapításához szükséges adatokat tárolja (felhasználói név, jelszó, internet cso-

[1]Network Access Server - Hálózati hozzáférést biztosító szerver
[2]Remote Authentication Dial In User Service

mag, dinamikus vagy fix IP cím, időkorlát, stb.). Ha az adatok megfelelőek, akkor a RADIUS visszaigazolja a NAS számára a kapcsolat engedélyezését, a NAS pedig felépít egy ppp interfészt a klienssel amit ősszekapcsol egy saját, az internet irányába forgalmazó interfészével. A RADIUS-tól kapott információk alapján szabályozza az adott klienshez tartozó ppp interfész forgalmát. A NAS és RADIUS között a kommunikáció egy speciális protokoll, az AAA[3] szerint történik.

Az eszközök megválasztása az internet szolgáltató költség-hatékonyság kalkulációja szerint történik. Léteznek célhardverek, amelyek a fent felvázolt architektúra több rétegét is magukban foglalják, illetve használható több eszköz a rétegek szétbontására aszerint, hogy a hálózati szegmensek hogyan oszlanak el. A gyártók, és szakmai fórumok által közreadott dokumentációkban, illetve tutoriálokban megtalálhatóak a különböző eszközök (szerverek, routerek, szoftverek, célhardverek) teljesítmény leírásai, amely alapján méretezhető az ügyfélszámmal szembeállított erőforrás igény.

2.2. A nagyvárosi Ethernet hálózat szerkezete.

Nagyvárosi környezetben a szolgáltatási terület nagy kiterjedése miatt elkerülhetetlen az Ethernet hálózat szegmentálása. Két irányú felosztásra kell gondolni:

- Horizontális felosztás - A hálózat logikai rétegződése.
- Vertikális szegmentálás - A hálózat fizikailag elkülönülő területekre bontása.

A cisco céghez fűződő három rétegű hierarchikus modell jól illeszkedik a szolgáltatási terület fenti felosztásához.

1. Core vagy Backbone Layer
2. Distribution Layer
3. Access Layer

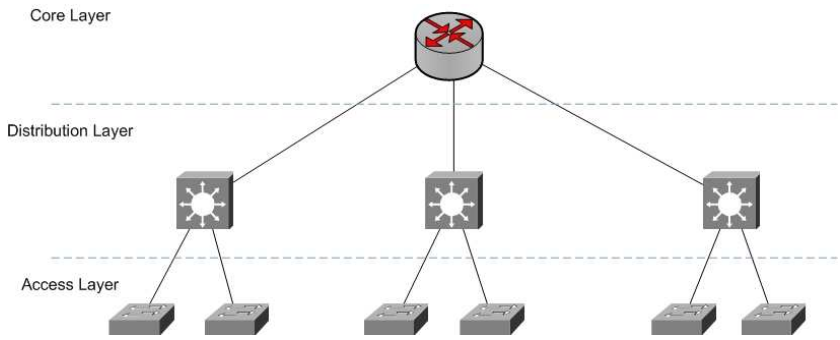
Az első és egyben legfelső *Core* réteg a teljes nagyvárosi hlózatot fogja össze, és kapcsolatot biztosít a szolgáltató internetes gerincéhez. Ebben a rétegben rendszerint nagyteljesítményű routerek vagy Multilayer switchek működnek.

[3] Authentication Authorization and Accounting

A második, azaz *Distribution* réteg a város területén egymástól távol levő, nagyobb alhálózatok központi csomópontjában működő eszközöket tartalmazza. Ezek rendszerint nagyteljesítményű Layer 3-as switchek, amelyek a VLAN-ok közti kapcsolatot biztosítják, tűzfal és hálózati szabály előírási funkciókat látnak el, valamint prioritás alapú forgalom irányítási szolgáltatást nyújtanak.

A harmadik és egyben legalsó *Access* szint a Layer 2-es kommunikációt biztosítja. Ebben a rétegben switchek működnek amelyek a felhasználók rendszerhez kapcsolódását biztosítják, valamint a hálózathoz csatlakozó hosztok VLAN-okkal történő csoportosítását végzik el.

A háromrétegű hierarchikus modellt szemlélteti a 2.2. ábra.



2.2. ábra. A három rétegű hierarchikus modell.

Az általam készített program modul a hálózat legalsó szintjén, az *Access* rétegben működő aktíveszközök kapcsolódási hierarchiájának feltérképezését, és annak adminisztrálását, valamint megjelenítését hivatott megoldani.

A hálózat vertikális szegmentálása a szolgáltatási terület fizikai adottságai szerint történik. A szolgáltató internetes gerinckapcsolati pontjától, a *Core* rétegbeli eszköztől távol, több fejállomást úgynevezett *HUB*-ot érdemes kialakítani, az egyetemi illetve vállalati hálózatok létesítmények szerinti tagolódásához hasonlóan. Ezek az állomások fogják össze egy-egy városrész alhálózatát, és nagy sávszélességű gerinckapcsolaton keresztül csatlakoznak a központi csomóponthoz. A hierarchikus modell *Distribution* eszközei állnak az egyes területeket működtető részhálózatok tetején, ellátva az előzőekben felvázolt feladatokat, összefogva az adott szegmens *Access* rétegének Layer 2-es switcheit.

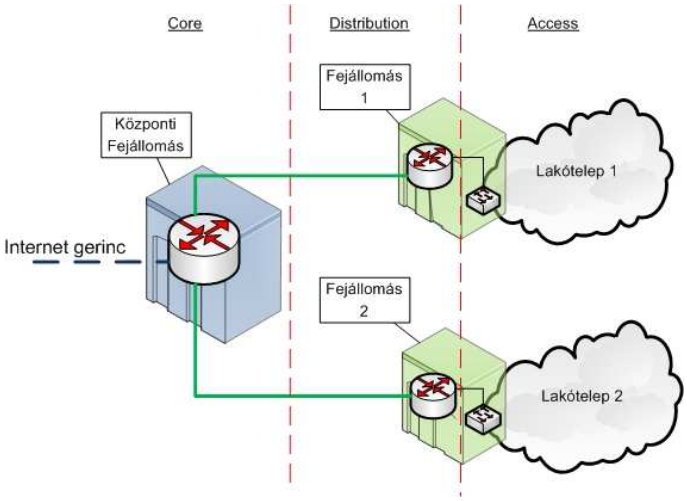
A debreceni internet szolgáltatók előfizetői területei tulnyomó részt a „nagy” lakótelepeket fedik le. Ezért a hálózat felépítését ezeknek a lakótelepeknek az elhelyezkedése, és utcaszerkezete határozza meg. A szolgáltatási végpontok nagyjából lépcsőházakra, illetve háztömbökre

korlátozódnak. Mivel az ethernet hálózatban a csavart érpáras kábelezés vonali hossza kb. 100 méterre korlátozódik, ezért ez a módszer nem alkalmas a végpontok gerincvonalainak kialakításához. Így a fizikai rétegben optikai szálakkal kell áthidalni a nagyobb távolságokat. Nagysávszélességű gerinckapcsolatok létesítéséhez pedig szintén elengedhetetlen az optikai adatátviteli eszközök alkalmazása. A lakóépületeken belüli kábelezés az előfizetők ellátására már csavartérpáras (UTP) kábellel történik. Így a szolgáltatási pontokon ethernet kapcsolat áll rendelkezésre.

2.2.1. A szolgáltatási terület központi csomópontjai

A legnagyobb telefonszolgáltató kihelyezett fokozataihoz hasonlóan a Digi Zrt. is több *HUB*-ot (2.3 ábra) működtet amelyek a központi informatkai, illetve kábel televízós fejállomáshoz kapcsolódnak. Az előző szakaszra visszautalva, ezen csomópontok adnak helyet a hierarchikus modell *Distribution* rétegének, illetve az *Access* szint alhálózatait összefogó nagyteljesítményű switcheknek. Ezen Layer 2-es eszközök képezik a térképező program által feldolgozott hálózati részfák rendezésének kiinduló pontját.

A fejállomások és a nagyvárosi hálózat központi csomópontja közti összeköttetést nagy sávszélességű (1 ill. több Gigabit/sec) optikai gerincvonalak biztosítják. A nagy távolság és amagas adatátviteli sebesség miatt mono (single) módosú kábelekre, és optikai interfészekre van szükség.



2.3. ábra. Fejállomások

Amennyiben lehetőség nyílik rá, a *HUB*-ok közötti redundáns gerinckapcsolatok létesítésével

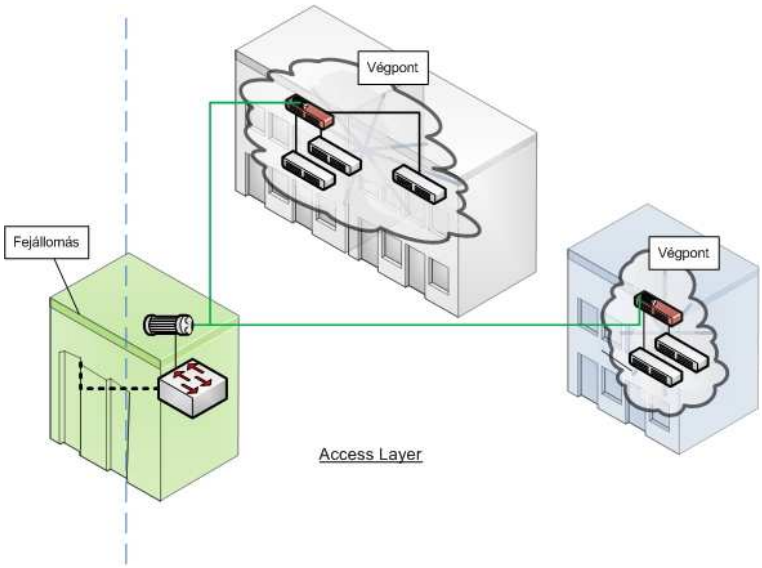
biztosítható az egyes hálózati szakaszok kiesése miatt (pl. kábel vágás) fellépő szolgáltatási szünet elkerülése.

Az előfizetői szolgáltatási csomópontok szintén optikai kábellel csatlakoznak az őket kiszolgáló fejállomásokhoz. Ezek az optikai szálak, a fénynyalábokat elektromos jellé átalakító média konverterek segítségével csatlakoznak az *Access* réteg tetején elhejezkedő switchekhez.

A fenti leírásból jól látszik, hogy egy-egy előfizető internetes forgalma, hányszor konvertálódik egyik közegből egy másikba a fizikai rétegben, már a városi hálózaton belül. Érzékelhető hogy mennyi eszközre és milyen komoly erőforrásokra van szükség egy ekkora hálózat biztonságos üzemeltetéséhez.

2.2.2. Az előfizetői szolgáltatási végpontok

A hálózatban egy végpontnak tekintjük az egy optikai gerinckapcsolaton keresztül megtáplált, egy szűk területen lévő előfizetőket kiszolgáló switchek összességét. E szerint egy lépcsőházban lehet több végpont is, ha az ott lévő előfizetőszám ezt indokolja; illetve egy végpont kiterjedhet több lépcsőházra vagy akár épülettömbre is, amennyiben a sávszélesség elegendő az ott található felhasználók számára, és a távolságok megengedik az aktív eszközök „rezes” összekötését. A 2.4 ábra a végpontok kialakítását szemlélteti.



2.4. ábra. A végpontok kialakítása

A végpontokon médiakonvertert alkalmazunk az optikai jel elektromos jellé alakítására. Ma már jó néhány típusú switch rendelkezik médiakonverter egységgel, így érdemes ilyen berendezéseket használni. Az előfizetői szolgáltatási csomóponton lévő eszközök hierarchiájának

tetején tehát egy optikai interfésszel ellátott switch áll, és ehhez kapcsolódik a többi berendezés.

Az előfizetők UTP kábellel csatlakoznak a végpontokra kihelyezett switchekhez. Mivel a felhasználói létszám eloszlása nem egyenletes, ezért nem érdemes nagy portszámú eszközökből megépíteni a végpontokat. Ahhoz, hogy ne legyenek kihasználatlan erőforrások több, kevesebb fizikai interfésszel rendelkező kapcsoló eszköz hierarchikus elrendezésével rugalmas csomópontokat alakíthatunk ki. Ezzel a módszerrel viszont jelentősen megnő a hálózatot működtető aktíveszközök száma (akár több ezer is lehet).

3. fejezet

A hálózat menedzsmentben alkalmazott technológiák, és módszerek

A hálózat teljes felügyeletéhez távolról elérhető, menedzselhető aktíveszközökre van szükség. Biztonsági szempontból viszont ezeket a berendezéseket az előfizetők számára „láthatalanná" kell tenni. Ennek egyik módja, hogy a menedzselhető eszközök IP cím tartományát csak a hálózat bizonyos szegmenséből, cím tartományából lehessen elérni, valamint szabályokat írjunk elő a berendezések kommunikációs csatornáinak használatára. Ezen túl az egyes felhasználói csoportokat különböző virtuális alhálózatokba, úgynevezett VLAN-okba kell rendezni, hogy már az adatkapcsolati rétegben el tudjuk szigetelni őket egymástól.

A felügyeleti rendszer és az adatforgalmat biztosító berendezések közti kapcsolattartást, távoli elérést megvalósító protokollok használatával valósítjuk meg. A menedzsment rendszer számára szükséges információk megszerzését, és az egyes célhardverek konfigurációs módosításait ezen kommunikációs felületek alkalmazásával érjük el.

A menedzsment szoftver és az általam készített programmodul a fent felvázolt, és a következőkben ismertetett módszereket és technológiákat hsználja fel működéséhez, illetve azok alkalmazásával teremt kapcsolatot a hálózat hardver eszközeivel.

3.1. A menedzselt switchek címkiosztása

A szolgáltatási területen működő aktíveszközöket egy, vagy több olyan IP tartományba kell helyezni amit csak a felügyeletet biztosító eszközök és felhasználók érhetnek el. A városi hálózat szegmentálása, és az aktíveszközök rendkívül nagy száma miatt több cím tartományra is szükség lehet.

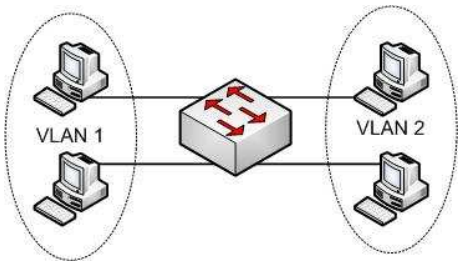
Az egyik fő kérdés a cím kiosztás módja. A cím kiosztás lehet statikus, például az adott eszköz üzembehelyezésekor előre beállított. Amennyiben olyan eszközökkel rendelkezünk amelyek DHCP kliensként képesek működni, akkor automatikus cím hozzárendelést is használhatunk.

Mérlegelni kell, hogy melyik megoldás felel meg legjobban céljainknak. Ha automatikus cím kiosztást alkalmazunk, talán kényelmesebb az installálás, viszont a folyamatos felügyelet érdekében olyan adatbázis struktúrát kell kialakítanunk, amelyet a DHCP szerver és a monitorozó rendszer közösen használ. Ha statikusan osztjuk ki a címeket akkor viszont az IP-ék és eszközök összerendelését nagyon pontosan kell nyílvántartanunk, különben ütközések léphetnek fel a hálózatban. Mindkét esetben fontos, hogy bármikor meg tudjuk határozni azt, hogy egy adott eszköz hol, és a működésének milyen fázisában van.

3.2. Virtuális lanhálózatok

Mint azt a fejezet bevezetésében felvázoltam, Layer 2-es szinten is van lehetőség a felhasználók csoportjainak szétválasztására. Ezt a technikát érdemes alkalmazni arra, hogy a felügyelthez tartozó felhasználókat (embereket, és gépeket) izoláljuk az előfizetőktől. Ehhez nyújt megoldást az úgynevezett virtuális lanhálózat, vagy röviden VLAN.

A VLAN egy szórási tartomány amelyet egy vagy több switchen hozunk létre. Ez által a kapcsoló eszköz nem továbbít adatkapcsolati szórási kereteket a különböző VLAN-ok között. Egy switch tartalmazhat több VLAN-t is amelyeket különböző portjaihoz rendel hozzá, ahogyan azt a 3.1 ábrán láthatjuk. Megdhatjuk, hogy a kapcsoló mely interfésze melyik VLAN-hoz tartozzon. Ezzel a technikával az aktív eszköz izolálja a különböző virtuális lanhálózathoz tartozó portokon jelentkező hosztokat.



3.1. ábra. Több VLAN egy switchen

A fizikai interfészekhez statikusan rendelt VLAN beállítási tecnikát *Port-based* VLAN-nak

nevezzük. Létezik más megoldás is, amely szerint nem azt mondjuk meg, hogy melyik interfészt melyik VLAN-nal rendeljük össze, hanem azt határozzuk meg, hogy egy adott MAC cím melyik VLAN-hoz tartozik. Az utóbbi technika alkalmazása nem igazán megfelelő internetes környezetben, mivel nem ismerjük elöre az előfizetők használt hálózati illesztők hardver címeit. Külön nehézséget okoz az, hogy a hordozható számítógépek és az üzletekben vásárolható hálózati eszközök árának rohamos csökkenése miatt egy internet szolgáltatási ponton más és más hálózati interfészek kapcsolódhatnak különböző időpontokban.

Az egyes aktíveszközök hálózati hierarchiában történő elhelyezkedésének megállapítása bizonyos szempontból szoros összefüggésben áll a VLAN technológiával. A hálózatot működtető kapcsolók egy adott menedzsment VLAN-ba vannak sorolva. Ezért a térképezés során ezen virtuális lanhálózat tagjainak adatait kell legyűjteni a felügyeleti rendszer adatbázisából, illetve magukból a hardverekből.

3.2.1. A TRUNK kapcsolatok

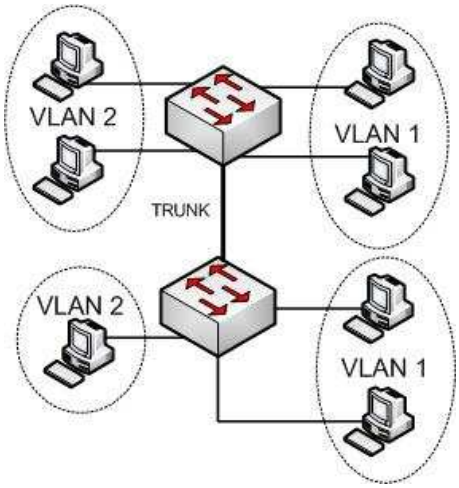
Eddig csak arra találtunk megoldást, hogy egy adott aktív eszközön szétválasszunk bizonyos felhasználói csoportokat. Mivel az ethernet hálózatban több aktíveszköz is jelen lehet, arra a kérdésre is választ kell adnunk, hogy ugyanaz a VLAN hogy jelenik meg több switchen, és az azonos virtuális lanhálózatba tartozó hosztok, hogyan kommunikálnak egymással.

Az alapvető módszer az lenne, hogy ahány VLAN-t használunk az eszközeinken annyi, az egyes virtuális lanhálózatoknak megfelelő fizikai interfésszel csatlakoztatnánk azokat egymáshoz. Ez a lehetőség nem megfelelő, és gyakran kivitelezhetetlen, mivel egyrészt a fizikai portok pazarlásához vezet, másrészt rendkívül rugalmatlan.

A „trunking” technika nyújt megoldást a fenti problémára. A hálózatban lévő switcheket úgynevezett trunk portokon keresztül kapcsoljuk össze, ezt szemlélteti a 3.2 ábra.

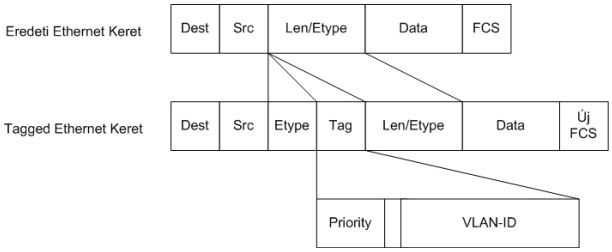
A *VLAN trunking* lényege, hogy egy fizikai interfészre vonatkozóan megadhatjuk, hogy milyen virtuális lanhálózatokat akarunk kiterjeszteni azon keresztül az összekapcsolt switchek között. A kapcsoló eszközök megjelölnek minden csomagot amelyet a trunk porton át továbbküldenek. Így a csomagot fogadó switch tudja, hogy az adott keretet küldő állomás mely VLAN-hoz tartozik, és csak azokra a rákapcsolt állomásokra küldheti tovább amelyek szintén ugyanabban a szórási tartományban vannak. Ezzel a módszerrel több VLAN-t valósíthatunk meg amelyeket több aktív eszközre terjeszthetünk ki.

Az IEEE kidolgozott egy szabványt a *trunking* technika megvalósítására, melyet a 802.1Q elnevezéssel látott el. Eszerint az eredeti Ethernet keret fejlécét kibővíti egy 4 bájtos extra bejegyzéssel. Ez a 4 bájtos fejléc tartalmazza a VLAN azonosítóját, és egy prioritás jelzőt. A



3.2. ábra. Több VLAN kiterjesztése több switchre TRUNK portok segítségével

következő ábra szemlélteti a 802.1Q fejléc és Ethernet fejléc szerkezetét.



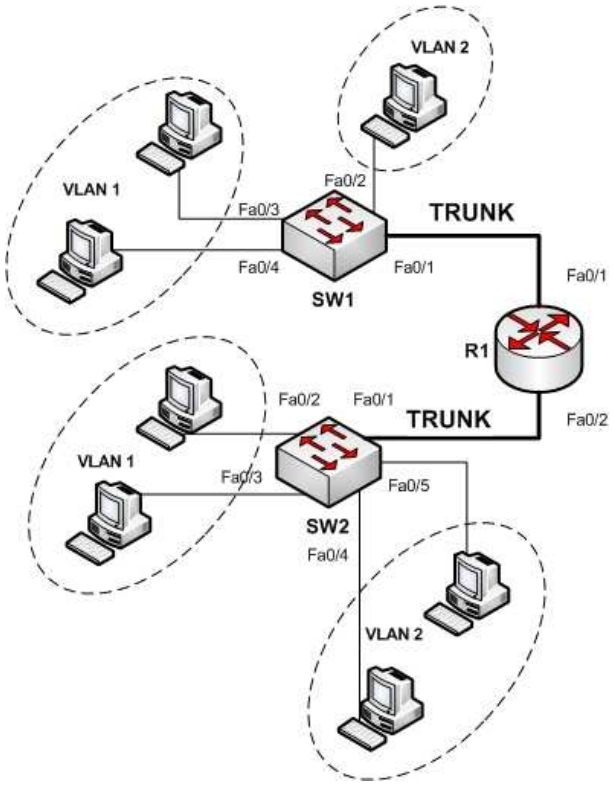
3.3. ábra. Több VLAN kiterjesztése több switchre TRUNK portok segítségével

Ennek a módszernek az a hátránya, hogy az Ethernet keret tartalmának megváltoztatása miatt újra kell számolni az ellenőrző összeget. A mai nagy számítási sebességű kapcsolóeszközöknek viszont nem csökkent a működési hatékonyságán ez a plusz tevékenység.

A *trunking* technika szabványosítása előtt már léteztek a 802.1Q-tól eltérő megoldások. Ilyen például a Cisco által kifejlesztett ISL protokoll. Itt az Ethernet fejléc változatlan marad, viszont minden egyes keret becsomagolódik egy ISL keretbe, amelynek a fej része tartalmaz a VLAN-ra vonatkozó adatokat, amelyeket az ISL-t értelmező switchek felhasználnak. E technika alkalmazásához, a hálózat összes kapcsolójának ismernie kell az ISL protokollt, így leszűkül a használható eszköztípusok köre. Az újabb típusú cisco eszközök már nem is használják ezt a technológiát.

3.2.2. A forgalom biztosítása különböző VLAN-ok között

A felhasználói szintek, illetve típusok VLAN-okkal való elkülönítésével a hálózatban csak az adatkapcsolati rétegben, a switchek szintjén teremtettük meg a menedzsment illetve az egyéb hosztok szétválasztását. Ahhoz, hogy az előfizetők és egyéb felhasználók számára „láthatatlanná" tegyük az üzemeltetést, a VLAN-okat egy az egyben külön csoportoknak kell megfeleltetnünk az IP címtartományok szintjén. Tehát az ügyfél VLAN-okhoz ügyfél IP cím tartományokat, a felügyeleti VLAN-(ok)hoz pedig menedzsment IP cím tartományokat kell rendelnünk.A különböző VLAN-okban lévő hosztok közti forgalom biztosítását szemlélteti a 3.4 ábra.



3.4. ábra. Több VLAN kiterjesztése több switchre TRUNK portok segítségével

Az egy felhasználói csoprtba, de más-más VLAN-ba, és egyúttal más címtartományba sorolt hosztok kommunikációját szintén csak routeren, illetve layer 3-as switchen kereszül lehet megoldani. Ez igen fontos, mivel a nagyszámú aktíveszközpark nem feltétlenül egy IP címtartományba kerül, noha ugyanabba a VLAN-ba tartozik. Persze választhatunk „elég nagy" címtartományt a menedzselhető eszközök számára, de a hálózat áttekinthetősége, és a különböző eszközcsoportok logikai szinten történő megkülönböztetése érdekében érdemes több, funkcio-

nalitás vagy területi elhelyezkedés szerinti kisebb cím osztályt választani.

A különböző területekhez tartozó útválasztókon speciális ACL technikák alkalmazásával szigorú, és biztonságos szabályrendszer alakítható ki az üzemeltető, az előfizető és egyéb felhasználói csoportok hozzáférési szabályozására.

A fenti feladatok elvégzését a hierarchikus hálózat *Distribution* rétegének eszközei látják el, melyekről a 2.2 szakaszban szóltam. A programban megtalálható egy konkrét osztály implementációja, amely a hálózatban működő Layer 3-as switcheket reprezentálja.

3.3. Az SNMP keretrendszer

Az SNMP[1] azaz egyszerű hálózat kezelő protokoll, a hálózatra kapcsolt eszközök vezérlését, illetve adataik, állapotuk lekérdezését hivatott szolgálni. Az SNMP protokollt az IETF[2] definiálta. Az SNMP alkalmazás szintű protokoll amely az IP fölött működő, az UDP 161, illetve 162-es portján történő speciális kiliens-szerver architektúrájú, kommunikációs rendszert ír le. Az SNMP szabvány[3] magában foglalja a hálózati menedzsmenthez szükséges alkalmazás szintű kommunikációs rendszert, egy speciális adatbázis sémát, és az adat objektumok halmazát.

Az SNMP keretrendszer a működés szempontjából alapjában véve három absztrakciós szintet határoz meg:

- Maga az SNMP protokol adatokat változókat állít be, visz át amelyek egy adott eszköz állapotát meghatározzák.
- A MIB-ek definiálják, hogy az átvitt változók mit jelentenek.
- Az SMI mondja meg azt, hogy ezek a változók, hogyan definiálják magukat a MIB-ben.

E három absztrakciós szintet implementálva valósíthatjuk meg a hálózati eszközök kezelését, távoli elérését, és állapotváltozásaik nyomonkövetését.

Az SNMP tipikus felhasználása, mikor számos eszközzel rendelkezünk amelyeket távolról szeretnénk menedzselni egy vagy több felügyeleti rendszer segítségével. Minden felügyelni kívánt eszközön fut egy szoftver komponens amelyet *ágensnek*, vagy *daemonnak* nevezünk. Ez a szoftver szolgáltat információkat az adott rendszerről, illetve kezelő felületet nyújt az eszköz paraméterezéséhez SNMP segítségével.

[1] Simple Network Management Protocol
[2] Internet Engineering Task Force
[3] RFC 3411

Az SNMP-vel menedzselt hálózatnak három alapvető komponense van:

- Menedzselhető eszközök.
- Ágensek
- Hálózat kezelő rendszerek (NMS[4])

Az első csoport magukat a hardvereket, vagy esetleg szoftvereket foglalja magába, amelyeket manipulálni akarunk, vagy adatokat akarunk kinyerni belőlük.

A második komponens a menedzselhető eszközön futó szoftver, amely kiszolgálja az SNMP kéréseket, interfészt biztosít az eszköz és a kezelő rendszer között.

A harmadik komponens pedig maga a hálózat kezelő rendszer, amely a felhasználóknak nyújt kényelmes felületet, illetve automatikus funkciókat lát el, amelyek a hálózat működését befolyásolják, felügyelik.

Az menedzselhető eszköz állapotáról információkat az SNMP `GET`, `GETNEXT`, és `GETBULK` protokoll műveletein keresztül lehet lekérdezni. A menedzselhető eszközön működő démon pedig a `TRAP`, illetve `INFORM` protokoll műveletekkel küld információkat az adott berendezés állapotáról. A felügyeleti rendszernek lehetősége van az eszközök távoli konfigurálására is, amelyet a protokoll `SET` műveletének segítségével tud megvalósítani.

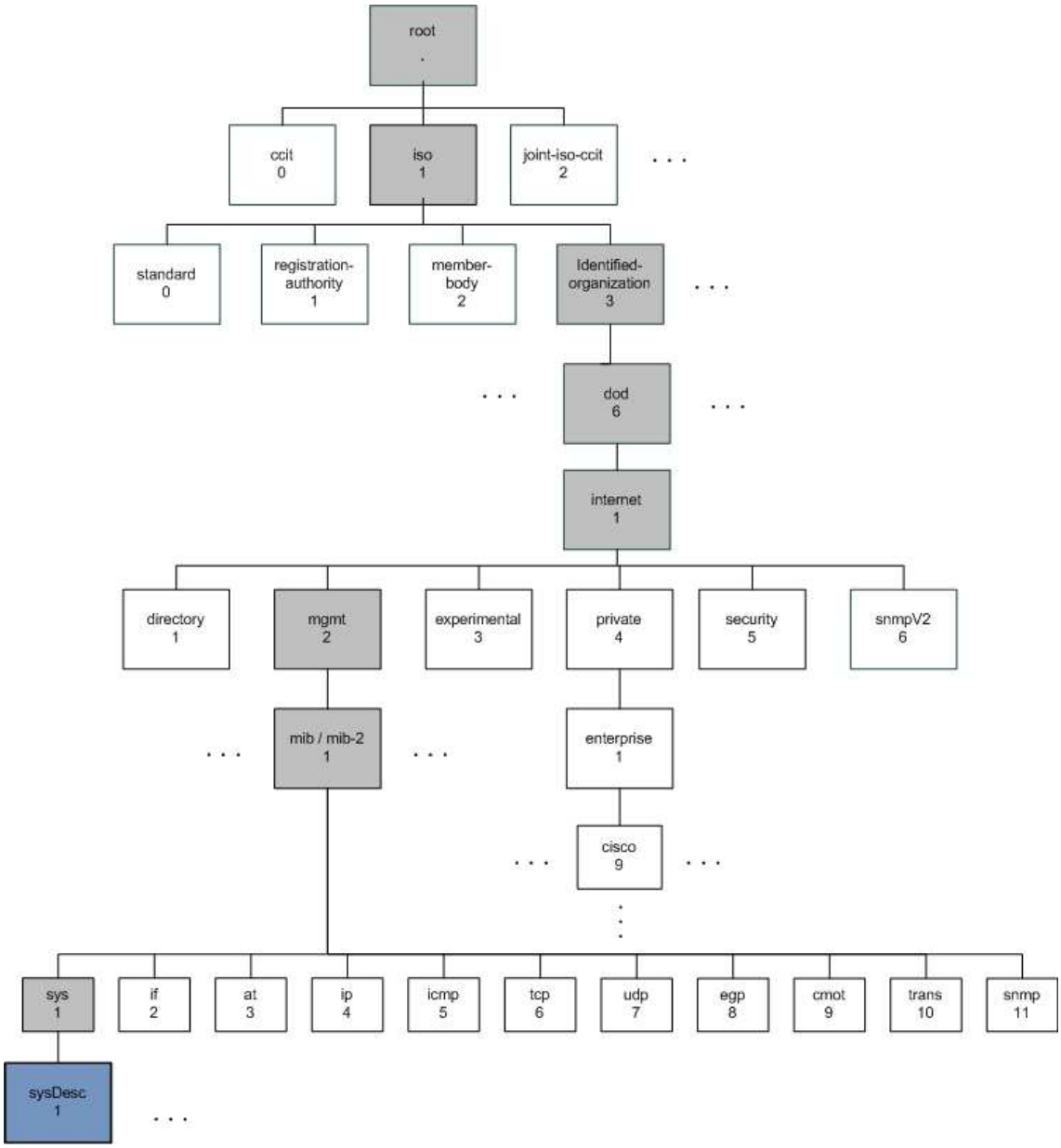
Az adott hardverek állapotát az ágens változókon keresztül teszi elérhetővé a felügyeleti rendszer részére (pl. rendszer név,memória foglaltság,IP cím,MAC cím, stb.). Az SNMP segítségével elérhető változók egy hierarhikus adatszerkezetben tárolódnak. A változókat, és metaadatokat tartalmazó hierarhiát IB[5]-nek nevezzük, amely egy virtuális adatbázis.

3.3.1. A MIB-ek

Maga Az SNMP nem definiálja, hogy egy konkrét menedzselhető rendszer milyen adatokat szolgáltasson, illetve milyen állapotát lehessen változtatni. Az SNMP-t úgy tervezték, hogy bármilyen eszközhöz általános hozzáférést biztosítson, és maga az adott eszköz egy bővíthető adatbázisban, a MIB-ben tárolja az őt leíró, és paraméterező attribútumokat. Mivel a MIB egy fa, így a benne található változókat egy-egy csomópont reprezentálja amelyeket az objektum azonosító OID[6] jelöl a hierarhikus névtérben. Egy csomópontot a hozzávezető

[4]Network-manegment System
[5]Management Information Base
[6]Object Identifier

ágakon keresztül lehet „megfogni” az OID-ekkel történő azonosítással. A keresett objektumhoz vezető élet úgy határozzuk meg, hogy az egy ponttal jelölt, névtelen gyökértől indulva az elágazási pontokat reprezentáló OID-eket felsoroljuk ponttal elválasztva. Például a következő MIB egy eszköz rendszerleírását tartalmazza szöveges formátumban: .1.3.6.1.2.1.1.1 vagy .iso.org.dod.internet.mgmt.mib-2.system.sysDescr Az OID által jelölt csomópont elhelyezkedését a MIB fában a 3.5 ábra reprezentálja.



3.5. ábra. Az .iso.org.dod.internet.mgmt.mib-2.system.sysDescr objektum helye a MIB-ben.

A MIB fában a különböző ágak gyökér csomópontjai nevesítve vannak. Ezekkel a nevekkel is hivatkozhatunk a kijelölt útvonalra illetve csomópontra. A MIB fa „névtelelen” gyökerének, amelyet egy ponttal jelölünk, három gyermek objektuma van:

- *ccit (0)* - Az ITU[7] szabványok ága.
- *iso (1)* - Az ISO szabvány ága.
- *joint-iso-ccitt (2)* - Az ISO és CCIT szabványok egyesített ága.

Mivel az SNMP segítségével végzett hálózati manipulácó lényeges számunkra, ezért a MIB fában a következő útvonalon kell haladnunk:

- Az *iso (1)* gyermeke az *org (3)* ág.
- Az *org (3)* ágon belül a *dod (6)* ág következik, amely a United States Department of Defense ről kapta a nevét.
- A *dod (6)* ágon belül található az *internet (1)* részfa.

Az SNMP szempontjából minden számunra lényeges információ a fent leírt útvonallal kijelölt részfában érhető el: .1.3.6.1, vagy *.iso.org.dod.internet*.

Ez a részfa hat további ágra bomlik amelyek a következők:

- *directory (1)* - Későbbi felhasználásra lefoglalt ág az ISO számára.
- *mgmt (2)* - Az elsődleges részfa, ahol a MIB objektumok elhelyezkednek. Ide tartozik egy részfa amely a *mib (1)* elnevezést kapta, és amelyet ma már *mib-2 (1)* néven használunk, amióta a MIB-II megszületett.
- *experimental (3)* A kidolgozás alatt álló szabványok számára tartalmaz objektumokat.
- *private (4)* A magán vállalatok, gyártók számára fenntartott ág. Ez az ág tartalmazza az *enterprise (1)* ágat amelyben a különböző gyártók saját névtérrel rendelkeznek, és az általuk készített eszközökre vonatkozó speciális objektumokat ezekben a névterekben tárolják.
- *security (5)* Biztonsági felhasználásra fenntartott ág.

[7]International Telecommunication Unit

- *snmpV2 (6)* Kimondottan az SNMP 2-es verziójához tartozó objektumok.

Az *enterprises (.1.3.6.1.4.1)* ág alatt minden egyes gyártót egy egyedi azonosító, egy egész szám jelöl. Az alábbi listában a példa kedvéért felsorolok néhány nevesebb gyártóhoz tartozó, *enterprises* ágat jelölő MIB-et:

- .1.3.6.1.4.1.2 - IBM
- .1.3.6.1.4.1.4 - Unix
- .1.3.6.1.4.1.9 - Cisco
- .1.3.6.1.4.1.11 - HP
- .1.3.6.1.4.1.36 - DEC
- .1.3.6.1.4.1.757 - Oracle

Egy konkrét MIB objektum, vagy részfa leírását egy szöveges állomány tartalmazza. Ebben a szöveges állományban találjuk az objektum elemeinek deklarációját, a típus leírását, és az objektum magyarázatát. A hálózatokban lévő eszközök sokszínűsége miatt, amely funkciójukban, megvalósításukban, paraméterezhetőségükben egyaránt jelentkezik, jelentős probléma ezen eszközök egységes kezelhetőségének kialakítása.

Az SNMP keretrendszer egy külön része, az SMI[8] biztosítja a MIB objektumok univerzalitását Az SMI az SNMP-ben olyan specifikáció amely a hálózati eszközök általános tulajdonságait reprezentáló, összetarozó MIB objektumok halmazát hivatott definiálni. Ezeket a halmazokat MIB moduloknak nevezzük. Az SMI szabályai meghatározzák, hogy az egyes MIB objektumokat, illetve MIB modulokat hogyan kell deklarálni. Az SMI-nek két fő szabványa van. Az eredeti *SMIv1*, amely az első SNMP keretrendszer, az *SNMPv1* része volt, és ar *RFC 1155*-ben rögzítették. Ez a verzió alapvető szabályokat írt le a MIB felépítésére és a változók leírására. A második az *SMIv2* amely az *SNMPv2p* részeként jött létre a *RFC 1442* szabványban, valamint az *SNMPv3*-ban továbbfejlesztették, amelyet az *RFC 2578*-ban rögzítettek. A legújabb verzió megegyezik az előzővel, csak újabb adat objektumok definícióival bővült.

Az SMI-ben MIB objektumok leírása egy adat definíciós nyelv, az ASN.1[9] jelölésrendszerének segítségével történik.

[8] Structure of Management Information

[9] Abstract Syntax Notation One

A telekomunikációs és informatikai hálózatokban az ASN.1 egy szabványos és rugalmas jelölés rendszert biztosít a különböző adatstruktúrák ábrázolására, átvitelére, kódolására és dekódolására. Az ASN.1 az egymástól független rendszerek közt átvitt adatok ábrázolásának egyeztetéséhez ad egy köztes felületet. A különböző adattípusok formális leírásához alkalmazott szabályok halmazát nyújtja, így az ábrázolt adatok mindig egyértelműek maradnak. Az SNMP az ASN.1 egy részhalmazát használja az SMI meghatározására.

3.3.2. A térképrző program által használt fonotsabb MIB objektumok.

Ahogyan az előzőekben írtam a hálózati eszközök általános információit az *SNMPv2-MIB*, illetve az egyes típusokra vonatkozó speciális tulajdonságokat az *SNMPv2-SMI::enterprises* csomópontok alatt találjuk. A program működéséhez szükségünk van az eszközök típusára vonatkozó információkra, a fizikai interfészeik listájára, valamint az egyes portokhoz bejegyzett menedzsment VLAN-ban jelentkező MAC címekre.

Az eszközökre vonatkozó általános típus leírást a *.iso.org.dod.internet.mgmt.mib-2.system.sysDescr* MIB bejegyzés tartalmazza. Ez a bejegyzés minden eszköznél megtalálható, természetesen az adott típusra vonatkozó formátumban. Ezért minden egyes aktíveszköz típusnál más-más módon kell feldolgozni a kapott információt, különböző rész karakterláncok kinyerésével.

A cisco eszközök esetén nem a fenti MIB-et használom. Sajnos nem lehet egységes módszert alkalmazni minden, ettől a gyártótól származó switch típusra. Viszont az *SNMPv2-SMI::mib-2.47.1.1.1.1.13 (entPhysicalModelName)* OID alatt található lista első eleme mindig az adott eszköz típusát adja meg.

Az interfészek listája szintén az *SNMPv2-SMI::mib-2* standard ágon szerepel, így minden egyes gyártmány esetén a *SNMPv2-SMI::mib-2.interfaces.ifTable* csomóponton férünk hozzá a szükséges bejegyzésekhez.

Pusztán az a kérdés, hogy a gyártók milyen interfész típusokat sorolnak a fenti bejegyzések közé. A cisco kapcsolók az *ifTable* listába nem csak a fizikai portokat jegyzik be, hanem a VLAN és egyéb interfészeket is. Ezért a `CSwitchCisco` osztály implementációjában a *.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr* listát olvasom ki, és csak a fizikai portoknak megfelelő bejegyzéseket gyűjtöm le.

A MAC címek listájának legyűjtése már sokkal bonyolultabb feladat, és sajnos egyes típusoknál nem is találtam megoldást a bejegyzések SNMP-én keresztül történő kiolvasására.

Meglepő módon pont a cisco eszközök is ez utóbbi csoporthoz tartoznak. Ezen kapcsolók esetén maradt a Telnet protokoll alkalmazása. Nem szép, de alkalmas megoldás a fenti problémára, amelynek megvalósítására a későbbiekben külön kitérek.

Mivel a térképezés során adott hálózati részfában ismert MAC címeket kell megkeresnem az ott működő eszközökön, számomra a hardver címhez tartozó port azonosítójának kinyerése a cél.

A Huawei eszközöknél a legegyszerűbb a megoldás, mivel az ARP tábla port bejegyzéseinek OID-jeit maguk a VLAN azonosítók és MAC címek jelölik a következő formában: *.iso.org.dod.internet.private.enterprises.2011.2.23.1.3.2.1.2.VlanID.byte.byte.byte.byte.byte.byte* . Ahol a *VlanID* a VLAN-t azonosító egész szám a *byte* sorozat pedig a MAC cím decimális egészek formájában. A bejegyzés értéke annak az interfésznek az azonosítója, ahol az adott MAC cím jelentkezik.

Eltérő tárolási módot alkalmaznak a ZTE, illetve SWH gyártmányú eszközök. Több azonos bejegyzésszámú listát vezetnek, és a listaindexek egyesztetésével nyerhető ki a keresett információ. Az *SNMPv2-SMI::enterprises.9304.100.2109.5.7.3.1.2* lista tartalmazza az ARP táblában található MAC címeket. Az *SNMPv2-SMI::enterprises.9304.100.2109.5.7.3.1.3* csomóponton pedig a portok azonosítói tárolódnak amelyek indexei megegyeznek a megfelelő fizikai cím bejegyzések indexével. A két lista elemei páronként adják a szükséges információt.

3.4. A menedzselhető eszközök elérése Telnet protokoll segítségével.

A legtöbb hálózati eszköz biztosít Telnet felületet a *command line interface* távoli eléréséhez. Ez az 1969 óta létező „ősi” technológia ma is népszerű és alkalmas eszközrendszert nyújt ilyen feladatok elvégzésére.

Az előző szakaszban utaltam rá, hogy a program működéséhez szükséges bizonyos infromációkat nem sikerült egyes hardver típusokból SNMP segítségével kinyerni. Ezért más módszert kellett alkalmaznom, nevezetesen a Telnet protokollt.

Minden általunk használt kapcsoló rendelkezik virtuális terminál felülettel, amelyre távolról bejelentkezve, a UNIX operációs rendszerhez hasonlóan egy parancs értelmezőt, esetleg menüvezérelt felületet biztosít. A szükséges beállítások, illetve információk kilistázása elvégezhető ezen módszer alkalmazásával.

A programban implementáltam egy egyszerű, minimális Telnet interfészt, amely képes tá-

voli bejelentkezésre, és adatok küldésére, illetve fogadására. Használatával interaktív kommunikációt valósíthatunk meg a hálózatban működő eszközökkel.

Eredetileg a Linux operációs rendszer által nyújtotta Telnet, illetve Expect programokat használtam, melyeket a PHP külső programok futtatására képes, `exec` függvényével hívtam meg. Azonban azt akartam elérni, hogy a hordozhatóság, és egyszerű telepítés érdekében, amennyire lehet lecsökkentsem az ilyen jellegű megoldásokat.

4. fejezet

Üzemeltetéssel kapcsolatos problémák, és nehézségek

A nagyvárosi ethernet hálózat sajátos architektúrájából kifolyóan számos probléma, illetve nehézség merülhet föl az üzemeltetés folyamán. A hálózat különböző rétegei, és szegmensei szerint más-más anomáliákat, zavarokat kell felismerni, elhárítani, illetve megelőzni a felügyelettel foglalkozó szakembereknek. A menedzsment rendszer egyik fő feladata ezen problémák felismerésének, illetve megoldásának támogatása. Az aktíveszközök hierarchiájának feltérképezése, a kapcsolódási pontok valóságnak megfelelő szemléletes ábrázolása, nagyban elősegítheti egyes problémák feltárását, illetve a hibák lokalizálását.

Az alábbiakban tömören összefoglalom a szolgáltatás folyamán leggyakrabban felmerülő nehézségeket:

4.1. A hálózat fizikai jellemzőiből adódó nehézségek.

- A végpontokon elhelyezett eszközök áramingadozásra érzékenyek.
- A hálózatban lévő aktíveszközök működését nagyban befolyásolhatja a jelentős hőmérséklet ingadozás, különösen a nyári időszakban történő erős felmelegedés.
- Az előfizetői oldalon történő UTP, illetve optikai kábelezést ritkán lehet ideális körülmények között végezni. Nincs lehetőség teljesen egységes kábelezési rendszer kialakítására. Ebből adódik, hogy a passzív hálózat hibafeltárása nem egyszerű feladat.
- Speciális hibák megoldásához több szakmai csoport közös munkájára van szükség: optikai hálózattechnikusok, hálózatszerelők, hálózatüzemeltetők.

- Bizonyos speciális konfigurációk beállításához egy időben több helyen kell módosításokat végezni a halózatban működő eszközökön.
- A hálózatban nem feltétlenül azonos gyártótól származó eszközök működnek. Ezért a szakembereknek többféle eszköztípusról kell folyamatosan ismereteket szerezniük. Valamint a hálózat menedzsment rendszert is fel kell készíteni erre a problémára.

4.2. A hálózat dinamikus növekedéséből adódó problémák, és feladatok.

- A növekvő előfizetői létszám miatt szükséges az IP tartományok folyamatos bővítése.
- A különböző hálózati szegmensek gerinckapcsolatainak sávszélességét folyamatosan növelni kell.
- Az új szolgáltatási területek beindításához, néha a hálózat struktúrájában is kell változásokat eszközölni.
- Egy-egy elkerülhetetlen szolgáltatási szünet, vagy karbantartás esetén, egyre nagyobb ügyfélcsoportot kell kiértesíteni.
- A hálózati erőforrások terhelésének egyensúlyban tartása nehéz feladat.
- Az eszközök monitorozása egyre komolyabb erőforrásokat igényel.

4.3. Előfizetők eszközeinek kiszámíthatatlan működéséből keletkező problémák.

- Az internet felhasználó hibás hálózati interfésze miatt zavarok keletkezhetnek azon kapcsoló működésében amelyhez az adott előfizető csatlakozik.
- Az internet felhasználó számítógépén lévő rosszindulatú alkalmazás futása nyomán történő SPAM szórás, MAC flood vagy egyéb zavart keltő folyamat keletkezik.
- Az internet előfizető számítógépe, vagy soho-routere részéről történő DHCP cím osztás. Ebben az esetben az adott végponthoz kapcsolt számítógépek automatikusan IP címet

kapnak. Amennyiben nem a PPPoE kapcsolat az alapértelmezett egy ilyen számítógépen, a felhasználó nem a saját internet hozzáférésén keresztül próbál kommunikálni.

4.4. Az emberi tényezőkből adódó nehézségek.

- Felületes informatikai ismereteik miatt az internet felhasználók gyakran nem elégséges, vagy valótlan információkat közölnek hibabejelentéskor.
- A hálózatban dolgozó szakemberek gyakran elmulasztják a műszaki dokumentációk elkészítését, pótlását, kiegészítését, ezért amennyire lehet automatizálni kell a hálózatban történt változások nyomonkövetését.
- Előfordulhat, hogy egy előre felkonfigurált aktíveszközt nem a megfelelő végponton helyez üzembe a feladattal megbízott technikus.
- Túl sok eszközt fűz fel egymásra a switcheket beüzemelő technikus. Ebben az esetben szűkül az adatátviteli keresztmetszet a hálózati hierarchiában lefelé haladva.
- Rosszakaratú „támadások” a hálózatban, a hálózatot működtető eszközök ellen.
- Ritkán előfordul a hálózati eszközök rongálása.

5. fejezet

Az üzemeltetés hatékonyságát növelő személyes javaslatok

A felügyeleti rendszert olyan képességekkel kell felruházni, amelyek valós időben, és szemléletes módon tájékoztatják az üzemeltetést végző csoportot a hálózatban történő változásokról. Természetesen a szakemberek leghatékonyabb „fegyvere" a célhardverek távoli elérésén keresztül hozzáférhető *command line interface*. Azonban sok esetben jelentősen megkönnyíthető az összetett feladatok elvégzése az informatív támogató rendszer használatával.

5.1. A nehezen behatárolható hibák megkeresése, előrejelzése

Nagy kiterjedésű hálózatokban, igen magas számú aktíveszköz állomány mellett, mint azt a 4.1 szakaszban is jeleztem bizonyos hibák felderítése igen fáradtságos, és sok időt felemésztő feladat. A hálózat méretének növekedésével még összetettebb problémák adódhatnak, és a megoldásuk is nagyobb felkészültséget igényel. Mivel egy-egy probléma megoldására fordított idő is kritikus tényező, az üzemeltetőknek érdemes nagy hangsúlyt fordítaniuk azok megelőzésére, illetve a hálózatban hibát gerjesztő folyamatok felismerésére, és előrejelzésére.

5.1.1. A gyakran, vagy kiszámíthatatlan időközönként újrainduló eszközök kiszűrése

Ahogyan azt a 4.1. szakaszban említettem a hálózat fizikai jellemzői miatt előfordulnak nem várt események amelyek az aktíveszközök működését befolyásolják. Egy-egy végponton

lévő switchek gyakori újraindulását okozhatják áramellátási problémák, tápegység meghibásodások, vagy valamilyen más zavar által okozott események. Az ilyen jellegű hibák felismerése, megtalálása nehéz feladat, a hibajelenség esetlegessége miatt.

Egy eszköz folyamatos pingelése igen erőforrás igényes módszer, viszont ha túl nagy időintervallumok telnek el a switch „életjeleit" ellenőrző *ICMP* üzenetek között könnyen átsiklunk egy újraindulási hiba fölött. Szebb megoldás, ha maga az aktív eszköz értesíti a felügyeleti rendszert arról, ha ilyen esemény történt. Ehhez ad alkalmas felületet az SNMP protokol TRAP mechanizmusa.

A legtöbb menedzselhető aktív eszköz képes indulást (COLDSTART) és újraindulást (WARMSTART) jelző SNMP TRAP riasztást küldeni. A fentiek szerint az aktíveszközökben be kell állítani, hogy hová küldjenek nem várt eseményekről figyelmeztető *TRAP* üzeneteket, és a felügyeleti rendszerben futtatni kell egy SNMP TRAP démont ami összegyűjti a hibajelzéseket. A hibajelzések alapján már könnyen vezethetünk valamilyen statisztikát amely alapján kiszűrhetjük a gyakran, vagy kiszámíthatalanul újrainduló eszközeinket.

5.1.2. A hálózatban lévő szűk keresztmetszetek lokalizálása

Az előfizetők részéről történő sebességcsökkenések okának kiderítése összetett feladat. Mivel a nagyvárosi Ethernet hálózatban az adatcsomagok több féle média típuson, és viszonylag sok aktíveszközön keresztül közlekednek, az ilyen jellegű hibák behatárolása idő, és erőforrás igényes. Érdemes a monitorozó rendszert úgy kialakítani, hogy előre meghatározott vagy esetlegesen kijelölt pontokon végzett mérések alapján bizonyos adatforgalmi jellemzők (pl. ki- és bemenő oldali adatforgalom, broadcast csomag szám, unicast csomag szám, eldobott csomag szám, stb.) határérték túllépése, vagy adott irányú túl nagy eltérése esetén jelezést küldjön az üzemeltetők felé.

5.2. A hálózat dinamikus változásainak előre becslése

Az üzemeltetés során igen fontos kérdés, hogy a meglévő erőforrásokat hogyan osszuk el optimálisan, valamint a költségeket lehetőség szerint minimalizálva végezzünk fejlesztéseket. Ennek előfeltétele, hogy amennyire lehet előre megbecsülhessük a szolgáltatási rendszer erőforrás bővítésének irányait és mértékét.

A fenti cél eléréséhez ki kell dolgozni valamilyen stratégiát arra, hogy az üzleti megfontolások alapján kitűzött, fejleszteni kívánt szolgáltatási területek szerint a várható ügyfélszám

növekedését jól lehessen becsülni. A becslések alapján pedig a műszaki oldal fejlesztési irányait optimálisan lehessen meghatározni, valamint a hálózati struktúrát alakítani.

Egy egyszerű példával élve, ha az üzletkötői oldalról, illetve ügyfélszolgálatokról beérkezett adatok alapján, előre lehet becsülni a szolgáltatási terület szegmensenkénti internet bekapcsolási igényeit, akkor ezen adatok ismeretében előre lehet tervezni, illetve szervezni az emberi és műszaki erőforrások elosztását.

6. fejezet

A hálózati menedzsment szoftver ismertetése

Ahogyan azt a bevezetőben vázoltam, az általam elkészített program modul a menedzsment rendszerbe beillesztett, de önállóan is működni képes program, illetve osztály gyűjtemény. A szoftver fő feladata a hálózatban működő switchek, aktíveszközök kapcsolódási hierarchiájának feltérképezése.

A programmodul különböző osztályokat tartalmazó csomag, illetve csomagok könyvtárainak összessége, amely rugalmas felületet nyújt a fenti funkció integrálásához a hálózatfelügyeleti szoftverbe. A programmodulban lévő osztályokat úgy terveztem és implementáltam, hogy néhány példányosítással, és metódushívással megvalósítható legyen a kívánt feladat elvégzése.

A tervezés során a következő fő szempontokat, és követelményeket fogalmaztam meg:

- A szoftver modul használható legyen a program szerkezet átfogó ismerete nélkül, pusztán a megfelelő interfészek dokumentációjának felhasználásával.
- A térképezési funkció adatbázis független legyen. Lényegtelen, hogy a kiinduló adatokat honnan kapja meg a program, csak megfelelően legyen paraméterezve.
- A szoftver funkcióji akkor is helyesen működjenek, ha a hálózatban struktúrális változások történnek.
- Amennyiben új típusú eszköz kerül a hálózatba ne legyen szükség a program módosítására, csupán a hardvernek megfelelő osztályt kelljen implementálni. Azt is úgy, hogy csak az eszközre vonatkozó speciális funkciókat legyen szükséges felüldefiniálni, illetve kibővíteni.

- A programmodul logikai rétegződése tegye lehetővé a könnyű továbbfejleszthetőséget. Ha új kiegészítéseket akarunk eszközölni, vagy egy-egy régi funkciót továbbfejleszteni, akkor ne kelljen változtatni a szoftver szerkezetén; csak az adott helyen, osztályban, metódusban lokális módosításokra legyen szükség.
- A szoftvermodul könnyen implementálható legyen másik programozási nyelven, illetve operációs rendszer környezetben.

6.1. A program funkciói.

A program működés szempontjából három fő feladatot lát el:

1. Az adott hálózati szegmensben működő aktíveszközök kapcsolódási hierarchiájának felépítése egy adatszerkezeti struktúrába, az eszközökből kiolvasott adatok alapján.
2. A felépített adatszerkezet adatbázisban történő tárolása, és frissítése.
3. A kívánt hálózati szegmensben működő aktív eszközök összekapcsolódási fájának megjelenítése grafikus felhasználói felületen.

A fenti funkciókat a következőkben ismertetem:

6.1.1. A térképező funkció.

A hálózat térképezés egy olyan hierarchikus adatszerkezet felépítése amely az eszközök kapcsolati struktúráját reprezentálja. Ez az adatszerkezet alapvetően egy fa struktúra a hálózat jellegéből adódóan.

A program teljes hálózat részfáit állítja elő az adott csomópontokra vonatkozóan. A részfák felépítéséhez a switchek ARP vagy Bridge tábláit olvassa ki SNMP illetve Telnet protokoll segítségével. Az ARP táblában megtalálható, hogy egy kapcsolót azonosító MAC cím a switch melyik fizikai portján jelent meg. Ezért ha ismerjük az aktíveszközök fizikai címeit be tudjuk azonosítani, hogy egy adott switch mely portján jelentkezik egy másik kapcsoló.

6.1.2. Az adatok tárolása, frissítése adatbázisban.

A térképezés elvégzéséhez szükséges adatokat a program egy relációs adatbázisból nyeri, valamint a rendezés elvégzése után a megfelelő adatokat frissíti az adatbázisban az aktuális állapotnak megfelelően.

A switchek úgy kerülnek ki a végpontokra, hogy a menedzsment szoftver egy konfigurációs modulja segítségével alapbeállításokat kapnak (IP cím, VLAN beállítások, SNMP paraméterek stb.), és az eszközre vonatkozó adatok a fizikai címmel egyetemben az adatbázisban eltárolódnak. Ezek az adatok adják a térképezés kiinduló paramétereit.

Az adott hálózati szegmensre vonatkozó térképezés során kerülnek be az adatbázisba az ott működő eszközök kapcsolódását reprezentáló információk:

- Az adott eszköz melyik eszközöz kapcsolódik.
- Az adott eszköz melyik portján keresztül kapcsolódik a szülő eszközhöz.
- A szülő eszköz melyik portjához kapcsolódik az adott eszköz.

Amennyiben olyan változás történik a hálózat valamely végpontján amely adminisztratív tevékenységgel jár (új eszköz kerül beüzemelésre, switch csere történik, leszerelnek egy kapcsolót) a menedzsment rendszer „kikényszeríti" az adott szegmens újratérképezését, és az adatok frissítését az adatbázisban.

Ahogyan a fejezet elején a tervezési szempontokban felvázoltam, a program szerkezetét úgy alakítottam ki, hogy a térképezési funkció az adatbázistól független legyen. Így a kapcsolódási felület megváltoztatásával lecserélhető a program „mögött" elhelyezkedő adatbázis, vagy adatbázis rendszer.

6.1.3. Az eszközök fastruktúrájának megjelenítése.

Ez a funkció valósítja meg az adatbázisban eltárolt hierarchikus elrendezés grafikus felületen történő megjelenítését.

Mivel az aktíveszközökből történő közvetlen adatlegyűjtés, és fa struktúra építés viszonylag hosszú időt vesz igénybe, ami nem kényelmes a felhasználóknak ezért a megjelenítéshez szükséges adatokat a program az adatbázisból olvassa ki. Amennyiben a felhasználó úgy találja, hogy a megjelenített fa struktúra nem megfelelő, például vannak olyan eszközök amelyek nem kapcsolódnak egymáshoz, akkor a felületen elhejezett *újra térképezés* feliratú gomb segítségével kikényszeríthető az adott végpontra vonatkozó információk adatbázis frissítése.

Mivel a felügyeleti szoftver web felületen működik. Ezért az általam készített bővítmény a megjelenítéséhez HTML formátumú kimenetet generál.

6.2. Hatékonyság növelő megoldások.

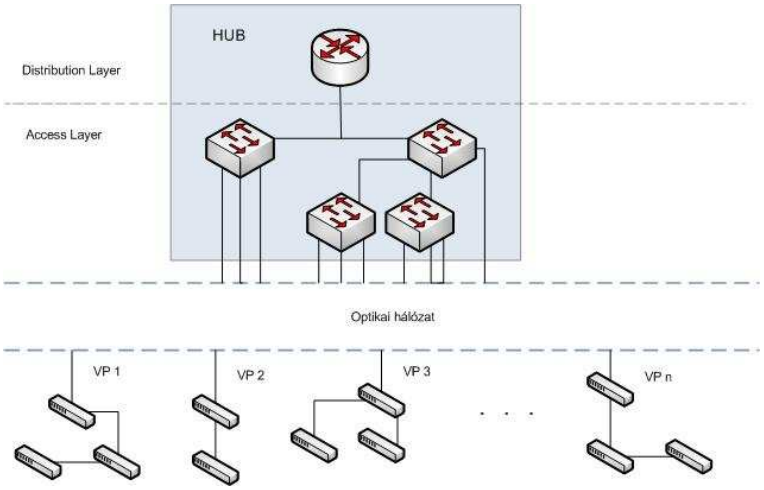
A térképezési folyamat a hálózatban működő kapcsolóeszközök jelentős darabszáma miatt igen sok időt vesz igénybe. Ezért a folyamat meggyorsítása érdekében érdemes a keresést a meglévő információk felhasználásával befolyásolni.

6.2.1. Az IP-ARP bejegyzések adatbázisban történő tárolása.

A futási idő szempontjából az egyik kritikus folyamat a layer 3-as eszközök IP-ARP táblájának lekérdezése, és feldolgozása. Mivel a menedzsment rendszerben előfordulhat, hogy egy időben több keresési, és térképezési folyamat is elindul, ezért érdemes az IP-ARP tábla tartalmát egy önállóan futó, időzített programrésszel olvastatni, és a bejegyzéseket adatbázisban tárolni. Így gyorsítható a keresési folyamatok részére a bejegyzések elérése, illetve leredukálható a Layer 3-as switchek SNMP kérésekkel való „bombázása”.

6.2.2. A hálózat több lépcsőben történő térképezése.

A hálózat felépítése miatt az egyes területek fáját horizontálisan el lehet vágni (6.1. ábra). Ahogyan a 2.2.2 alfejezetben írtam az egyes végpontok optikai szálon kapcsolódnak a fejállomásokhoz. Így az *Access* réteg végpontjainak alhálózati fáit külön-külön lehet térképezni, és a kész fákat összefűzni a fejállomásokon működő aktív eszközök fizikai interfészei szerint. Az



6.1. ábra. Egy adott hálózati szegmensben működő eszközök fájának horizontális rétegződése.

optikai alhálózatokat ősszefogó fejállomásokon működő switchek IP címeinek ismeretében a program képes ezen eszközök kapcsolódási fáját első körben külön térképezi, és feléépíteni. Az

elkészült fát bejárva, az aktuális eszköz ARP bejegyzései alapján, az eszköz interfészein jelentkező MAC címekhez tartozó IP-ARP bejegyzések, vagy az adatbázisban tárolt switch adatok felhasználásával a program egyesével felépíti a végpontokhoz tartozó részfákat. Amint elkészült egy részfa a szoftver frissíti az ott jelen lévő eszközök megfelelő adatait az adatbázisban. Természetesen, ha valamilyen ok miatt nem sikerül felépíteni egy csomóponthoz tartozó hierarchiát, akkor az adatbázis frissítés elmarad, de a hibáról log bejegyzés készül. Így a nem várt esemény visszakereshető a hibák feltárása végett.

6.3. A szoftverrendszer részletes architektúrája.

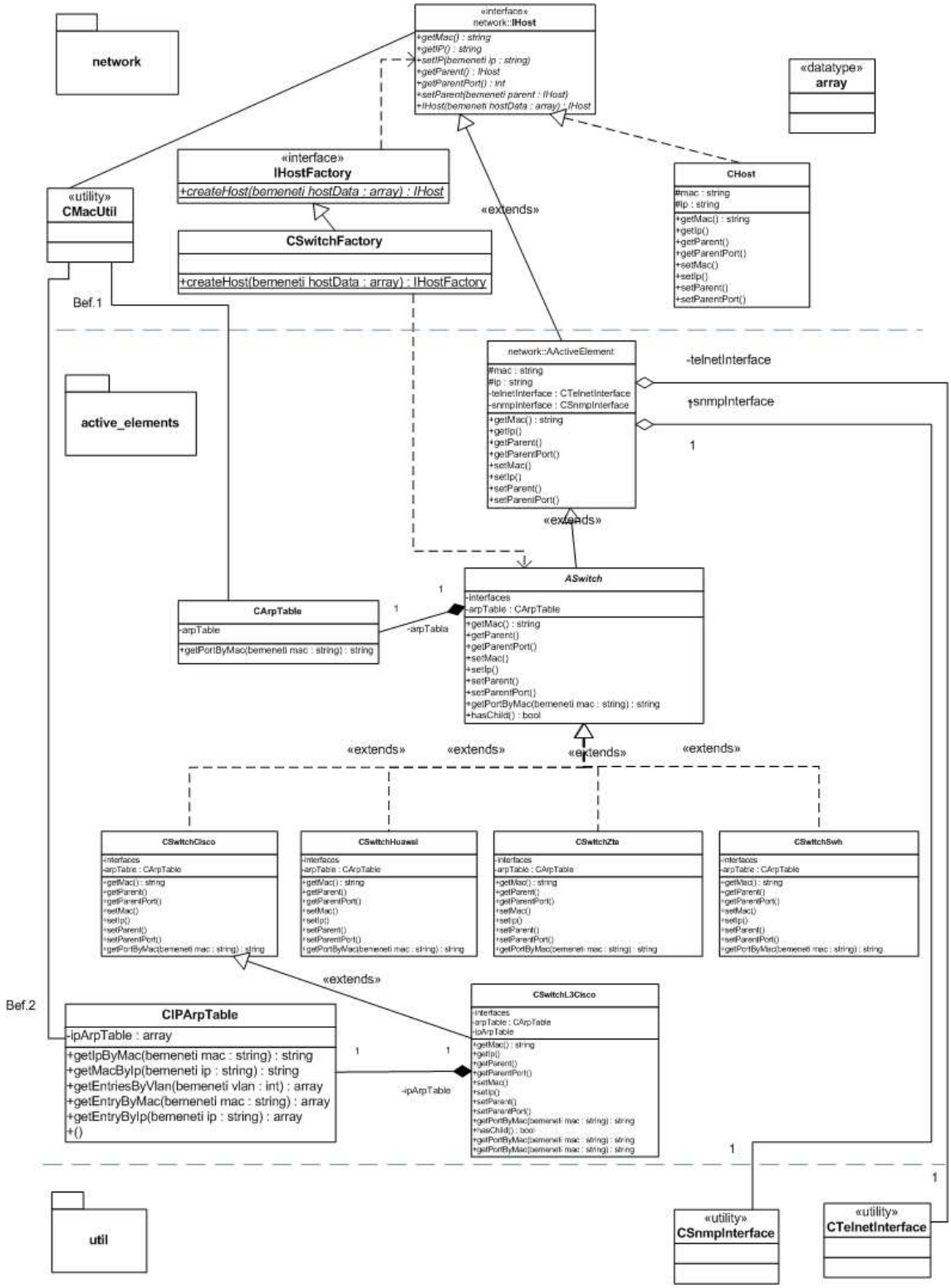
A programmodul számos interfészből, és az interfészeket implementáló osztályokból áll. Az osztályok a betöltött funkciójuk szerint csomagokba rendeződnek. Ezen alkotóelemek statikus kapcsolati szerkezetét ábrázolja a 6.2 ábrán látható UML diagram. A program csomagjai az alkotóelemek logikai rétegződését szemléltetik. Így külön választottam a hálózat topológiáját, az eszközöket, az adatbázis kapcsolati, és műveleti interfészét, valamint az egyéb alkotóelemeket megvalósító osztályokat.

A következőkben ismertetem az egyes csomagokban található elemek szerepét, illetve a 6.4 szakaszban részletesen kitérek azokra az osztályokra amelyek a program legfontosabb komponenseit alkotják.

6.3.1. A programmodult alkotó csomagok

A programmodul osztályait a következő csomagokba rendeztem:

- *default* - A program alapértelmezett csomagja. Ez a csomag tartalmazza a program minden osztálykönyvtárát.
- *network* - A hálózat osztályai.
- *active_elements* - A hálózatban lévő aktív eszközöket reprezentáló osztályok vannak ebben a csomagban.
- *exception* - A kivétel osztályokat összegyűjtő könyvtár.
- *util* - Segédosztályokat tartalmazó csomag.
- *db* - Az adatbázis kapcsolatot, és a megfelelő olvasási, tárolási, frissítési műveleteket biztosító osztályok.



6.2. ábra. A hálózat építőelemeit tartalmazó csomagok UML diagramja.

6.3.2. A default csomag tartalma

A default könyvtárban található az összes osztálykönyvtár valamint a hálózat térképezését elvégző `CNetworkMapper` osztály.

A `CNetworkMapper` osztály nyújt megfelelő metódusokat a térképezési folyamat elvégzéséhez. Az osztály implementációja nem egy általános megoldást kínál, hanem a 2.2 fejezetben felvázolt hálózati struktúra, Access rétegében található csomópontok fa struktúrájának felépítéséhez biztosít szolgáltatásokat.

A rendezést elvégző metódusok elsősorban a menedzsment rendszer adatbázisára támaszkodnak. Onnan nyerik a térképező algoritmus kiindulási paramétereit, és a sikeres futás eredményét is ott tárolják. Mivel a hálózat *Access* rétegének tetején álló cisco eszközök helye a hierarchiában nem változik, ezért a térképezési folyamatot csak a hozzájuk kapcsolt csomópontokon kell elvégezni. Így ezen eszközök adatait nem szükséges frissíteni, pusztán a rendezési algoritmus paramétereinek meghatározásához szükséges azokat felhasználni.

6.3.3. A *network* csomag osztályai

A network csomag tartalmazza azokat az interfészeket illetve osztályokat amelyek a hálózat struktúráját ábrázolják, illetve a struktúra felépítését segítik elő.

Az `IHost` interfész a hálózatban lévő összes eszköz közös felületét adja. Bármilyen a hálózatban működő eszközt megvalósító osztályt úgy kell megalkotni, hogy az `IHost` interfészt implementálja. Ezen interfész megfelelő metódusaival nyújtunk hozzáférést egy eszköz MAC, illetve IP címéhez, valamint a hálózati „szereplő” szülő eszközét jelölő adatokhoz.

Az `IHostFactory` interfészt megvalósító `CSwitchFactory` osztály a hálózatban lévő eszközök példányosítását végzi. Mivel igen nagy darabszámú, és különböző típusú eszköznek megfelelő objektumot kell előállítani a program futása során, így az OO Factory mintát követve választottam ezt a megoldást. Ezen interfész és osztály szerepét részletesen ismertetem a 6.4.4. részben.

Az `INetworkTree` interfész nyújt felületet a hálózat fájának, illetve részfáinak adatszerkezeti megvalósításához. A `CSwitchTree` osztály implementálja ezt az interfészt, és valósítja meg a hálózatban működő switchek fájának felépítését. A `CSwitchTree buildTree()` metódus implementációja maga a hálózati fa építési algoritmusa. A fát felépítő algoritmust

a 6.4.2. alfejezetben ismertetem.

A `CSwitchDbTree` osztály szintén a switchek fáját építi fel, viszont már az adatbázisban eltárolt adatok alapján. Ezen osztály a megjelenítést szolgálja. A web alapú, grafikus felhasználói felületbe ágyazva, az adatbázisban tárolt információk alapján elkészíti egy adott hálózati csomópont switcheinek fáját. A fa HTML formátumú reprezentációját szolgáltatja kimenetként, amelyet a megfelelő web oldalba beágyazhatunk.

A `CSwitchDbTree` osztály szerepét és működését részletesebben ismertetem a 6.4.5 részben.

A `CArpTable`, és `CIPArpTable` osztályok a switchek ARP, illetev IP-ARP táblájának megvalósítását szolgálják. Mindkét osztály egy-egy adatszerkezet, amely a megfelelő bejegyzések tárolását, keresését, és az ehhez kapcsolódó műveleteket nyújtja. Minden switchet megvalósító osztály példány rendelkezik egy `CArpTable` példánnyal, illetve a Layer 3-as eszközök `CIPArpTable` példánnyal is. A fa építését végző algoritmus ezen táblázatokban található adatok alapján rendezi hierarchiába az eszközöket megvalósító objektumokat.

6.3.4. Az aktíveszközök implementációit tartalmazó *active_elements* könvtár osztályai

Ez a csomag jelenleg két absztrakt osztályt tartalmaz: `AActiveElement, ASwitch`. Az `AActiveElement` absztrakt osztály a hálózatban lévő összes aktíveszköz közös őse. Amennyiben új eszközt akarunk implementálni akkor ebből az osztályból kell származtatni a konkrét osztályt, illetve amennyiben kapcsolóról van szó akkor az `ASwitch` osztályból, mivel az utóbbi már a switchekre vonatkozó speciális attributumokkal, illetve metódusokkal is rendelkezik. Ahhoz, hogy a térképezés megfelelően működjön, és általánosan használható legyen a fenti megkötéseket be kell tartanunk a fejlesztés során. Viszont így a program lényegi részén nem kell változtatnunk, csak a megfelelő `Factory` osztályt kell helyesen felparaméterezni az új eszközre vonatkozó adatokkal.

Az `ASwitch` absztakt osztályt kiterjesztő `CSwitchCisco`, `CSwitchHuawei`, `CSwitchZte` valamint `CSwitchSwh` a hálózatban üzemelő ezközöknek megfelelő konkrét osztályok. Ezen osztályok implementációi az adott típusok speciális tulajdonságaihoz igazodnak. Mivel minden gyártó bizonyos funkciókat eltérően valósít meg (pl. ARP tábla tárolása, VLAN beállítások,

SNMP MIB implementációk, Telnet tulajdonságok, stb.), ezért az osztályok bizonyos metódusait különféle módon kell implementálni, illetve új funkciókat kell bevezetni.

A `CSwitchCisco` osztályból származtatott `CSwitchL3Cisco` osztály az egyes hálózati szegmenseket összefogó, a distribution szinthez tartozó Layer 3-as switcheket valósítja meg. Az adott hálózati szegmens teljes fájának gyökér eleme egy `CSwitchL3Cisco` példány, amely rendelkezik egy `CIPArpTable` típusú attribútummal. Amennyiben a fa építés során nem kívánjuk felhasználni az ott működő switchekről az adatbázisban eltárolt adatokat (vagy a szükséges adatok nem állnak rendelkezésre), úgy a megfelelő `CSwitchL3Cisco` példány felhasználásával legyűjthetjük az IP-ARP bejegyzéseket.

6.3.5. A *util* csomag osztályai

Ebben a csomagban olyan osztályok implementációi szerepelnek, amelyek valamilyen szolgáltatást nyújtanak más osztályoknak, objektumoknak. Vannak amelyek adatelemként példányosítva használhatók, illetve van olyan is amely az „egyke" modellt megvalósítva csak statikus, osztályszintű metódusokat tartalmaz mivel a szolgáltatásai nem példányhoz kötöttek.

A `CMacUtil` osztály az utóbbi szemléletet tükrözi. Feladata a MAC címek formai tulajdonságához kötődő szolgáltatások nyújtása: MAC cím helyesség ellenőrzés, két MAC cím ősszehasonlítása, a MAC cím gyártók általi részének kinyerése. A program igen sok részén szükség van ezekre a műveletekre, de ezen feladatok elvégzését többször implementálni igen nagy pazarlás lenne. Ezért hoztam létre ezt az osztályt.

A `CSNMPInterface` osztály az SNMP kommunikációt valósítja meg. `CSNMPInterface` osztály példánya biztosítja az SNMP `get`, `walk`, illetve `set` protokol funkciókat egy-egy ezköz adatainak lekéréséhez, vagy távoli manipulációjához. Minden aktív eszköz objektum rendelkezik egy ilyen attributummal, amit példányosításkor hoz létre ha megkapja a megfelelő paramétereket. Nekem nem kellett protokol szinten megírnom a megfelelő metódusokat, mivel a PHP rendelkezik SNMP csomaggal, így csak a nyelvi környezet által nyújtott függvényeket kellett felhasználnom.

A `CTelnetInterface` osztály szintén hálózati kommunikációs felületet nyújt, de TELNET protokol alkalmazásával. Ezen osztály megírása során viszont már nekem kellett megvalósítanom a protokol szintű kommunikációt. Az általam írt osztály a Telnet protokol egy mini-

mális megvalósítása. Csak a számomra szükséges funkciókat implementáltam, mivel a hálózati eszközök adatainak lekérdezéséhez igen csekély funkciót kellett alkalmaznom. Megteremti a kapcsolatfelvételt a távoli hardverrel, segítségével a virtuális terminál ablakméretének meghatározására kérhetünk visszaigazolást, adatokat tudunk küldeni, illetve szöveget olvashatunk be meghatározott promptig.

6.3.6. A *db csomagban elhelyezett osztályok és szerepük.*

Jelenleg a *db* könyvtár kettő darab osztályt tartalmaz, amelyek a térképező program és az adatbázis közötti kapcsolatot hivatottak biztosítani: `CDBInterface`, `CNetworkDbTools`.

A fenti osztályok két réteget képeznek az adatbázis és a program között:

1. Adatbázis szerver típustól függő felület (`CDBInterface`).

2. Az adatbázis szerkezettől függő, a program számára inputot szolgáltató, és az output tárolását biztosító felület. (`CNetworkDbTools`).

Az első szint hivatott az adatbázis szerverhez illeszkedő, és az ehhez kapcsolódó műveleteket magábanfoglaló réteget képezni a program számára. A feladat az adatbázis kapcsolat felépítése, lebontása és az SQL kérések végrehajtása, valamint azok eredményeinek biztosítása.

A második réteg végzi a szükséges adatok olvasását az adatbázisból, és az adott funkciók számára megfelelő formátumra konvertálását. Természetesen a térképezési, rendezési algoritmus eredményét is ezen felületnek kell az adatbázis szerkezetéhez igazítania, és a frissítéshez szükséges SQL utasításokat elkészítenie.

Ezen osztályok használatával értem el, hogy amennyiben meg kell változtatni a program mögött álló adatbázis szervert, vagy az adatbázis szerkezetét azok a módosítások ne legyenek befolyással a szoftver többi elemére.

6.4. A fontosabb szoftver modulok.

A következőkben kiemelem azokat a programrészeket, osztályokat amelyek a szoftver gerincét képezik, és a hálózat menedzsmentben alkalmazott technológiák használatára épülnek. Ebben a fejezetben felvázolom ezen építőelemek működését, használatát, valamint kitérek arra, hogy a bemutatott programrészek hogyan támaszkodnak egymás szolgáltatásaira.

A szoftver legtöbb alkotóeleme általánosan felhsználható eszközt valósít meg, de egyes részek a gyakorlatban használt menedzsment rendszerhez, és konkrét hálózatmodellhez kötődnek.

6.4.1. Az egyes hálózati szegmensek eszközeinek rendezését, és az eredmény rögzítését végző CNetworkMapper osztály ismertetése.

A programmodul legfelső eleme a `CNetworkMapper` osztály. Ahogyan az előzőekben írtam ezen osztály nyújt metódusokat a hálózat térképezés elvégzéséhez. A hálózati hierarchia különböző szintjeihez egy-egy metódust kínál amellyel elvégezhető a kívánt feladat:

- Az `updateOnePort` az *Access* réteg tetején található cisco eszköz adott portjához tartozó végpontján üzemelő switchek fáját építi fel.
- Az `updateOnCiscoByID` metódus segítségével egy cisco eszköz összes portját bejárva, az alatta lévő csomópontok fáit rendezhetjük hierarchiába.
- Amennyiben a hálózat egy *Distribution* eszköze által összefogott szegmens minden részfáját szeretnénk felépíteni, az `updateOneDistributionElement` módszert kell használnunk.
- Ha egy városrészt ellátó *HUB*-hoz tartozó hálózati szegmens *Access* rétegének kapcsoló eszközeit szeretnénk feltérképezni akkor az `updateOneHub` metódus nyújt ehhez lehetőséget. *A*z adott település teljes hálózatában lévő minden terület Layer 2-es eszközeinek kapcsolódási részfáit az `updateAllHubs` metódus segítségével építhetjük fel.

A hálózat bármelyik rétegéhez kapcsolódó térképezési folyamat az első metódusra, a legalsó szinthez kapcsolódó `updateOnePort`-ra épül. A szinteken fölfelé haladva az adott metódus, mindig az alatta lévő réteget rendező metódust használja föl. Ez a megoldás kézenfekvő a hierarchikus felépítés miatt.

A `CNetworkMapper` osztály implementációja az általunk üzemeltetett hálózati struktúrához igadozik, és a nálunk működő menedzsment rendszer adatbázisára épül. Így az ezen verzióban megvalósított részfa rendezés az *Access* szint felső részében elhejezkedő cisco eszközökre nem terjed ki, csak a hozzájuk kapcsolódó, végpontokon üzemelő hálózati részfákra.

A teljes hálózat *Access* rétegéhez tartozó eszközök rendezését a következő programkód szemlélteti:

```
<?php

$CLASS_DIR="/usr/share/netmap/default/";
$configFile="/etc/netmap/netmap.conf";

include_once($CLASS_DIR."CNetworkMapper.php");

try {
```

```
  $mapper=new CNetworkMapper($configFile);
    $mapper->updateAllHubs();
      unset($mapper);
}
catch (Exception $e){
  print $e->getMessage();
  unset($e);
}

?>
```

A $CLASS_DIR változó tartalmazza az apértelmezett csomag elérési útját. Ezt a változó nevet globálisan használom a szoftver minden fájljában a szükséges állományok beágyazásához, ezért a főprogramban kötelező ezzel a névvel hivatkozni, és az értékét helyesen beállítani.

A térképező osztály egy konfigurációs állomány elérési útját várja paraméterül. Ebben az egyszerű szöveges fájlban kell megadni a futáshoz szükséges paramétereket, soronként <változó> <érték> párok formájában. A konfigurációs állomány tartalma a következő:

```
# A logoláshoz szükséges paraméterek
[log]
logFile /var/log/workspace/netmap.log

# Az adatbázis kapcsolathoz szükséges paraméterek
[db data]
hostname <adatbázis_szerver_hosztneve>
dbName <adatbázis_név>
user <db_felhsználó>
password <db_jelszó>

# Az Accsess réteg tetején lévő eszközök hozzáférési paraméterei
[cisco data]
cMgmtVlan <menedzsment_VLAN_azonosító>
cTelnetLogin <telnet_user>
cTelnetPassword <telnet_password>
cSnmpROCommunity <SNMP_RO_community>
cSnmpRWCommunity <SNMP_RW_community>
cSnmpVersion <SNMP_verzió>

# A végpontokon üzemelő eszközök hozzáférési paraméterei
[switch data]
mgmtVlan <menedzsment_VLAN_azonosító>
telnetLogin <telnet_user>
telnetPassword <telnet_passásword>
snmpROCommunity <SNMP_RO_community>
snmpRWCommunity <SNMP_RW_community>
snmpVersion <SNMP_verzió>
```

A fenti módszer használatával a prorammodul paraméterezése nem forráskód szinten történik, így akár háttérben való futtatáskor, akár web felületbe ágyazás esetén rugalmasan végezhető, és a felhasználási környezethez alakítható.

A program futása során a megadott *log* állományba bejegyzéseket rögzít, amelyek segítségével tájékozódhatunk a folyamat sikerességéről, illetve a keletkezett hibákról. A bejegyzéseket időpont megjelöléssel rögzíti, így a szöveges fájlban adott időszakokra is könnyen kereshetünk mintaillesztés alkalmazásával.

6.4.2. A hálózati hierarchiát reprezentáló CSwitchTree osztály

Az `INetworkTree`-t implementáló `CSwitchTree` osztály a hálózat egy adott végpontján működő, vagy akár egy *distribution* eszköz mögött lévő összes switch által alkotott fa reprezentációját valósítja meg. Az osztályból származtatott példány egy adatszerkezetet és az azt felépítő metódusokat bocsájtja rendelkezésre.

A 6.4.1 fejezetben ismertetett osztály `updateOnePort` metódusa egy ilyen objektumot állít elő, és használ az adott csomópont rendezésére, és adatainak frissítésére. Az elkészített fa iterátorának segítségével a rendezés utáni állapotnak megfelelő adatokat használja fel az adatbázis frissítéséhez.

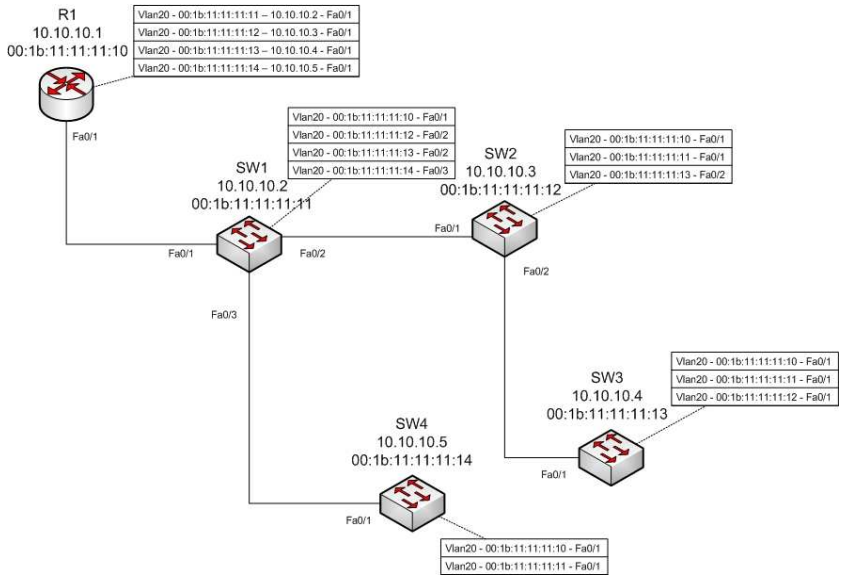
A fa felépítéséhez szükséges paramétereket a `CSwitchTree` osztály példányosításakor a konstruktorral kell megadnunk:

- `array $hostDataList`: A fába rendezendő eszközök adatainak, vagy maguknak az `ASwitch` osztályt konkrétan megvalósító osztályok példányainak a listája.
- `CSwitchFactory $hostFactory`: A switcheket gyártó osztály egy példánya.
- `string $routerMac`: Az adott hálózati szegmenset összefogó Layer 3-as distribution kapcsoló eszköz (mint a terület switcheinek alapértelmezett átjárója) menedzsment VLAN-hoz tartozó MAC címe.

A példányosítás folyamán meghívódik a `createHosts` metódus, amely a `CswitchFactory` osztály segítségével a switchek adatait tartalmazó listán végighaladva előállítja az eszközöknek megfelelő példányokat. Ekkor történik az egyes kapcsolók MAC address táblájának beolvasása, amely alapján a rendezés megtörténik. A gyártó osztály működését a 6.4.4 fejezetben részletesen ismertetem.

A switchek kapcsolódási fájának felépítését a `buildTree` metódus meghívásával végezhetjük el. A fa felépítése egy igen egyszerű algoritmus, amely a switchek bridge táblájában lévő, a menedzsment VLAN-hoz tartozó MAC cím bejegyzések alapján megkeresi minden egyes kapcsoló szülőeszközét. A 6.3 ábra szemléltet egy hálózati részfát amelyben az egyes switchek MAC address táblájának kivonata látszik. Az ábra szerint a menedzselhető hálózati

eszközöket a VLAN 20-ban felvett IP címeiken keresztül érjük el. Így az ARP táblájukban a VLAN 20-hoz tartozó bejegyzésekre van szükségünk.



6.3. ábra. Egy hálózati részfában lévő switchek és a hozzájuk tartozó ARP bejegyzések.

A bejegyzések alapján be tudjuk azonosítani, hogy egy adott kapcsoló eszköz melyik fizikai portján melyik MAC cím jelentkezik. Ha ismerjük az vizsgált hálózati szegmensben lévő aktíveszközök fizikai címeit meghatározhatjuk, hogy egy adott switch mely portjához tartozó alhálózatban találjuk, illetve nem találjuk meg a keresett eszközöket. Azt, hogy egy switch mely portja az úgynevezett *uplink* port, azaz mely interfésze csatlakozik a szülő eszközhöz abból tudjuk megállapítani, hogy mely portján jelentkezik az adott hálózati szegmens distribution eszközének menedzsment VLAN-hoz tartozó MAC címe. A fát felépítő algoritmus a fenti információkat felhasználva működik.

A fát felépítő algoritmus.

Az algoritmus indulásakor a switcheket reprezentáló példányokat a `CSwitchTree` objektum `$tree` listája tartalmazza. Ez a tömb képezi az algoritmus bemenetét, amelyet bejárva az eljárás minden elemhez megkeresi a közvetlen szülő eszközt, és amennyiben megtalálja azt, hozzákapcsolja az aktuális objektumot. Ha az algoritmus hiba nélkül befejezi működését, a `$tree` tömbben egy objektum szerepel, a hálózati fa gyökér eleme, és többi switch példány levél, vagy csomóponti elemként ehhez az objektumhoz kapcsolódik. Ellenben ha valamilyen

okból nem tud teljes rendezettséget előállítani a metódus, úgy a `$tree` lista több objektumot tartalmaz, amelyek vagy egyedül álló eszközök, vagy switchek részfái.

A fa felépítése során az eljárás a gyermek eszközt reprezentáló objektumot a szülő switch `$interfaces` tömbjének megfelelő indexéhez kapcsolja értékként, az ős `addHostToPort` nyílvános metódusa segítségével. Így alakul ki az adott csomópontot reprezentáló többágú fa.

A rendezést végző algoritmus forráskódja a következő:

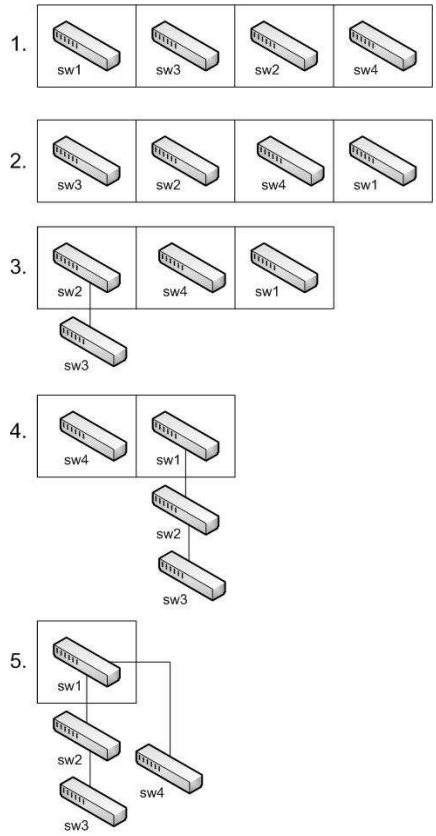
```
  /**
   * Az eszközök fáját felépítő metódus.
   *
   */
  public function buildTree(){
    if(!$this->tree){
      return;
    }
    $nemVegtelen=count($this->tree);

    while(count($this->tree)>1 && $nemVegtelen){
      ASwitch & $host=array_shift($this->tree);
        $this->minMac=MIN_MAC; // Elég nagy szám
        $this->parentPort=0;
          IHost & $tempParent=NULL;
      foreach($this->tree as $k=>$e){
        if(($r=$this->searchParent($host->getMac(),$e))){
          $tempParent=$r;
        }
      }
      if($tempParent){
        try {
          if($this->parentPort!=$tempParent->getUplinkPort()){
            $tempParent->addHostToPort($host,$this->parentPort);
              $host->setParent($tempParent);
                $host->setParentPort($this->parentPort);
          }
          else {
            array_push($this->tree,$host);
          }
        }
        catch(EDoubleMacException $e){
          unset($e);
          $tempHost=$tempParent->getHostByPort($this->parentPort);
            if(($p=$tempHost->getPortByMac($host->getMac()) &&
 $p!=$tempHost->getUplinkPort())){
            $tempParent->getArpTable()->addMacToPort(
sprintf("0000.0000.%04X",$nemVegtelen),$this->parentPort);
              $nemVegtelen++;
                array_unshift($host,$this->tree);
            }
            else {
              array_push($this->tree,$host);
```

```
            }
          }
        }
        else {
          array_push($this->tree,$host);
        }
        --$nemVegtelen;
      }
    }

    /**
     * Megkeresi az adott MAC címmel rendelkező hoszt szülő eszközét.
     * @param string $mac Az eszköz MAC címe aminek a szülő eszközét keressük
     * @param IHost $root A részfa gyökere, ahonnan a rekurzív keresés indul.
     * @return ASwitch A megtalált szülő eszköz vagy NULL.
     */
    public function searchParent ($mac,IHost & $root){
      $result=NULL;
      if($root instanceof ASwitch ){
        if(($p=$root->getPortByMac($mac))!=NULL && ($mm =
$root->getMacCountOnPort($p)) < $this->minMac){
          $this->minMac=$mm;
            $this->parentPort=$p;
              $result=$root;
        }
        if($root->hasChild()){
          foreach($root->getInterfaces() as $port=>$host){
            if($host && $r=$this->searchParent($mac,$host)){
              $result=$r;
            }
          }
        }
      }
      return  $result;
    }
```

Az iteráció folyamán az eszközök listájának aktuális első elemét kiemeljük. A tömb többi elemére nézve megvizsgáljuk, hogy a kiemelt switch MAC címe szerepel-e valamelyik objektum ARP táblájában. Ezt a keresést a `$tree` lista minden tagjára rekurzívan kell elvégezni, mivel a rendezés során részfák alakulhatnak ki alattuk. Ha az aktuális switch hardvercíme több objektumban is megtalálható, akkor úgy döntjük el, hogy melyik a közvetlen szülő, hogy a szóba jöhető jelöltek megfelelő interfészén lévő, a menedzsment VLAN-hoz tartozó MAC bejegyzések minimumát választjuk ki. Amennyiben nem sikerül megtalálni a listából kivett eszköz szülőjét, akkor az objektumot beillesztjük a `$tree` sor végére. Az iterációt addig folytatjuk, amig csak egy elem marad a listában, vagy megvizsgáltuk az összes switch példányt.

A `CSwitchTree` objektum `$tree` listája tartalmának, a 6.3. ábrán látható hálózati szerkezet felépítése folyamán történő változását a 6.4. ábra szemlélteti.

A MAC address táblák folyamatos frissülése miatt előfordulhat redundancia, mi szerint



6.4. ábra. Az eszközök listájának tartalma az algoritmus futása során.

olyan porthoz akarjuk kötni a vizsgált gyermek eszközt, amelyhez már kapcsolódik egy másik switch. Ebben az esetben meg kell vizsgálnunk, hogy az adott portra kapcsolódó eszköznek gyermeke-e az aktuális elem. Ha igen, akkor szülőeszköz adott interfészén lévő MAC bejegyzésekhez hozzáadunk egy hamis fizikai címet, hogy kiküszöböljük a bejegyzések darabszámának egyezését. Visszahelyezzük a `$tree` lista elejére az aktuális elemet, így adunk egy újabb esélyt a megfelelő közvetlen ős megkeresésére.

6.4.3. A fa elemeit képező, kapcsoló eszközöket megvalósító osztályok

A 6.3.4 szakaszban röviden bemutattam a hálózat *Access* rétegében működő Layer 2-es eszközöket reprezentáló osztályokat. Jelen részben részletesen kitérek szerepükre, és működésükre a programon belül.

Az `ASwitch` absztrakt osztályból származtatott `CSwitch` osztályok a hálózatban működő különböző kapcsolók típusaihoz igazodó implementációk. Feladatuk a program és a konkrét hardver eszközök közti kapcsolat megteremtése. Az `CSwitchTree` objektumok ezen osztá-

lyok példányit tartalmazzák adattagokként a `$tree` listában. A konkrét eszközökből történő adatolvasási feladatokat is ezek az objektumok végzik el.

Minden egyes `CSwitch` példány rendelkezik saját Telnet illetve SNMP interfésszel amelyeket a `CTelnetInterface`, és `CSNMPInterface` osztályok nyújtanak, valamint a rendezés elvégzéséhez szükséges MAC cím bejegyzések tárolását megvalósító collection osztály, a `CArpTable` egy egyedével. A hálózati kommunikációt végző felületek biztosítanak hozzáférést a hardverek adatainak legyűjtéséhez. Az ARP táblát reprezentáló adattag pedig a rendezési algoritmus számára tárolja a szükséges információkat, magukat a hardver cím bejegyzéseket, illetve azok kezeléséhez kínál alkalmas metódusokat.

Példányosításkor minden objektum lekérdezi az általa képviselt eszköz interfész listáját, és típusát SNMP kérések segítségével. A fa felépítése során az eredményül kapott portokhoz rendelődnek hozzá az adott switchez kapcsolódó gyermek eszközök, emelyek meghatározását a kiolvasott MAC address tábla alapján végzi el a program. A hardver cím bejegyzéseket is lekérdezhetjük konstruktor időben, vagy később ha azt célszerűbbnek találjuk. Ennek akkor van jelentősége, ha egy adott csomóponton található switchek száma olyan nagy, hogy a lineáris végrehajtás miatt menet közben kiürül, vagy befrissül valamely hardver MAC address táblája. Sajnos ilyenkor előfordulhat, hogy eredménytelenül fut le a térképezés az adott csomóponton.

Visszautalva a 3.3.2 szakaszban bemutatott, a program által használt MIB objektumokra beillesztek két forráskód részletet, amelyek az adott VLAN-hoz tartozó ARP bejegyzések legyűjtését végzik különböző típusú switcheken.

A CSwitchSwh osztály megfelelő metódusa:

```
/**
 * Az adott VLAN-ból jövő MAC címeket olvassa ki az eszközből.
 * @param int $vlan A VLAN azonosítója.
 * @return array A MAC címek listáját tartalmazó tömb. A tömb indexei a MAC
 * címek,az értékek, pedig az interfészek, amelyen a MAC cím jelentkezik.
 * @throws ENoSuchVlanException Ha nincs az eszközön az adott VLAN.
 */
protected function readMacListByVlan ($vlan){
  if(!($pl=$this->readPortsByVlan($vlan))){
    return NULL;
  }
  // MAC címek
  $mib1="SNMPv2-SMI::enterprises.9304.100.2109.5.7.3.1.2";
  // portok
  $mib2="SNMPv2-SMI::enterprises.9304.100.2109.5.7.3.1.3";
    $si=$this->getSnmpInterface();
    try {
      $ml=$si->walk($this->getIP(),$mib1);
      $il=$si->walk($this->getIP(),$mib2);
    }
    catch (ESNMPException $e){
```

```
      unset($e);
      return NULL;
    }
      if(!is_array($ml) || !is_array($il)){
        return NULL;
      }
        $result=NULL;
        foreach($il as $k=>$e){
          if(in_array(($port=substr($e,9)+1),$pl)){
              $mac=trim(substr(($ml[$k]),12));
              $result[CMacUtil::validMac($mac)]=$port;
            }
        }
  return $result;
}
```

A CSwitchCisco osztály megfelelő metódusa:

```
/**
 * Az adott VLAN-ból jövő MAC címeket olvassa ki az eszközből.
 * @param int $vlan A VLAN azonosítója.
 * @return array A MAC címek listáját tartalmazó tömb. A tömb indexei a MAC
 * címek,az értékek, pedig az interfészek, amelyen a MAC cím jelentkezik.
 * @throws ENoSuchVlanException Ha nincs az eszközön az adott VLAN.
 */
protected function readMacListByVlan ($vlan){
  $result=NULL;
  if(!($ti=$this->getTelnetInterface())){
    return NULL;
  }
  try{
    $ti->connect();
      $ti->setBufferSize(96000);
        $ti->setWindow(80,30000);
           $ti->readToPrompt("Username: ");
             $ti->send($this->getTelnetLogin()."\n");
               $ti->readToPrompt("Password: ");
                 $ti->send($this->getTelnetPassword()."\n");
                   $ti->readToPrompt("#");
                     $ti->send("show mac-address-table vlan $vlan\n");
                       $enters=4;
                         $list="";
                           $minta="/More-- $/";
                      while($enters){
                        $ti->readToPrompt("#",$minta);
                          if(preg_match("/(#)$/",($list.=$ti->toString()))){
                            break;
                          }
                          $ti->send(" ");
                       }
                       $ti->send("exit\n");
  }
  catch (ETelnetConnectException $e){
    $ti->disconnect();
```

```
        unset($e);
      return NULL;
    }
    catch (EBufferOverflowException $e){
      $ti->disconnect();
        unset($e);
      return NULL;
    }
    $ti->disconnect();
      $ti->clearBuffer();
    return  $this->textToMacListByVlan($list,$vlan);
  }
  /**
   * A telneten keresztül kapott szöveges fájlt, ami tartalmazza a MAC listát,
   * tömbbé alakítja.
   * @param string $str A MAC listát tartalmazó string.
   * @param int $vlan A vlan azonosító.
   * @return array A MAC címeket és hozzájuk tartozó portokat tartalmazó tömb.
   */
  protected function textToMacListByVlan(& $str,$vlan){
    if(!$str || !$vlan){
      return NULL;
    }
    $result=array();
      $rows=split("\n|\r\n",$str);
        foreach($rows as $k=>$e){
          $row=trim($e);
            $minta="/$vlan(\t| )+([0-9a-f]{4}.[0-9a-f]{4}.[0-9a-f]{4}).+
(Fa0\/[0-9]+|Gi0\/[0-9]+)/";
            if(preg_match($minta,$row,$tokens)){
              if($mac=CMacUtil::validMac($tokens[2])){
                $result[$mac]=$tokens[3];
              }
            }
        }
    return $result;
  }
```

6.4.4. A switcheket képviselő objektumok példányosítását végző *CSwitch-Factory* osztály.

A hálózatban működő eszközöket reprezentáló osztályok példányosítását a program folyamán úgynevezett *Factory* osztályok segítségével végzem. Ennek a megoldásnak fő okai a következők:

- Az objektumok nagy száma.
- A megfelelő osztályok példányosításához szükséges paraméterek száma.
- Az eszközök típusainak megfelelő osztályok automatikus kiválasztása.

- A térképezési algoritmus eszközfüggetlenségének biztosítása.
- A program új eszközökkel való bővítésének rugalmas megoldása.

Minden hálózatieszközt gyártó osztálynak az `IHostFactory` interfészt kell implementálnia. Ez a megkötés biztosítja, hogy egy `INetworkTree` interfészt implementáló osztály példánya használni tudja a gyártó osztály szolgáltatásait. A *factory* osztályt használat előtt fel kell paraméterezni egy tömbbel, amely a megfelelő eszközöket megvalósító osztályok azonosítási adatait tartalmazza. Ezen lista elemei asszociatív tömbök amelyek az egyes switch típusokhoz tartozó osztályok neveit, és az azokat azonosító gyártóspecifikus MAC cím részeket tartalmazzák:

array (**array** ("mac" => "00:0f:e2", "className" => "CSwitchHuawei"), **array** ("mac" => "00:d0:d0", "className" => "CSwitchZte"));

A `createHost` módszer a paraméterként kapott, az adott eszközre vonatkozó adatok közül a MAC cím alapján eldönti, hogy mely típusnak megfelelő osztályt kell példányosítania. A mennyiben nem talál a hardvercímhez illeszkedő bejegyzést, úgy egy általános `CHost` objektumot állít elő, és ad vissza.

Így a gyártó osztálynak a MAC címen kívül nem kell tudnia semmit a példányosítandó típusról, pusztán „mechanikus” munkát végez, és általános felületet nyújt a program egyéb részeinek számára.

6.4.5. A felépített részfákat megjelenítő CSwitchDbTree osztály használata.

Az elkészült részfák megjelenítéséhez szükséges HTML kimenetet a `CSwitchDbTree` osztály példánya szolgáltatja.

A használt adatbázis eléréséhez szükséges információkkal kell felparaméterezni a konstruktorban példányosításkor:

```
/**
 * Konstruktor
 *
 * @param string $hostname Az adatbázis szerver hosztneve vagy IP címe.
 * @param string $dbname Az adatbázis neve.
 * @param string $user Az adatbázis felhasználói név.
 * @param string $passwd Az adatbázis jelszó.
 * @param int $mgmtVlan A menedzsment VLAN azonosítója.
 */
 public function __construct($hostname,$dbname,$user,$passwd,$mgmtVlan)
```

Magát a fa felépítést az osztály `buildTree()` metódusának meghívásával végezhetjük el. Két paraméterként meg kell adnunk egy, a hálózat Access rétegének tetején álló Cisco switch adatbázis azonosítóját, valamint annak a portnak a nevét, vagy számát amelyhez kapcsolódó részfát akarjuk megjeleníteni. A metódus lefutása után az objektum `toString` metódusával tudjuk előállítani a fa szerkezetnek megfelelő HTML kimenetet.

Maga a HTML szöveg nem tartalmaz a formázásra vonatkozó információkat, pusztán a hierarchikus modellnek megfelelő `<div>`, `<ul>`, `<li>` elemekből és adatokból épül fel. Viszont minden HTML TAG-hez tartozik egy `id` azonosító, amelyre hivatkozva egy CSS stíluslap segítségével a számunkra megfelelő, szemléletes és esztétikus formára alakítható a kapott eredmény.

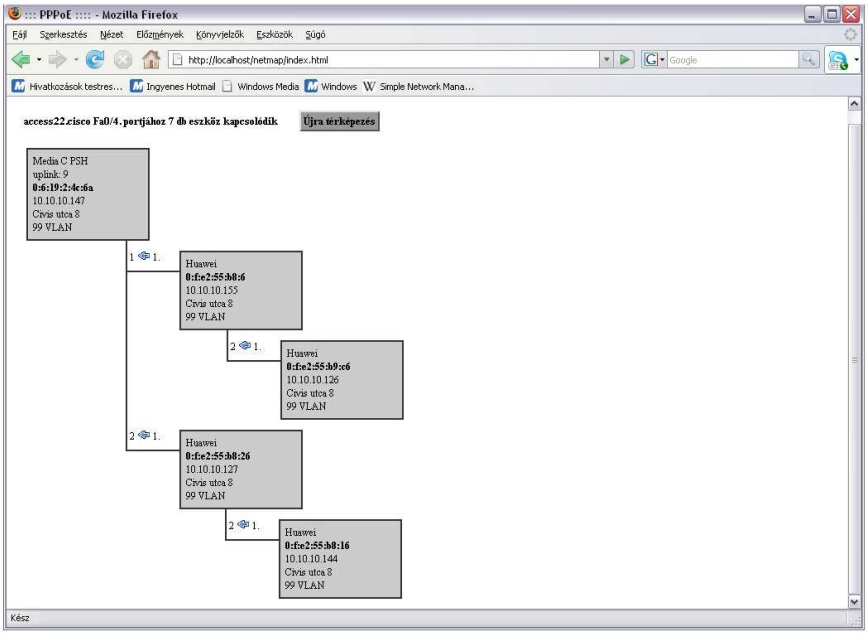
Egy feltérképezett hálózati csomópont aktíveszközeinek hierarchikus struktúráját a a 6.5 ábra szemlélteti, amelyen a böngésző ablakban megjelenített fa szerkezet látható. A dobozokat összekötő vonalakon van feltüntetve, hogy a szülő eszköz mely portjára, és melyik portján keresztül csatlakozik a gyermek switch.

A következő CSS forráskód szemléltet egy lehetséges megoldást, amely használatával a switchek adatait tartalmazó szövegdobozokat egymáshoz kapcsolódó fa struktúrában jeleníti meg. A fent említett ábrán látható HTML oldal formázását ezen stíluslap biztosítja.

```
#hostnev{
    font-size: 13px;
    font-weight: bold;
    color:black;
    text-align: left;
    margin-left:1em;
    margin-top:3em;
}
#hostnev a{
    margin-left:2em;
    border:outset 2px #999;
    padding:3px;
    background-color: #999;
    color:black;
    text-decoration: none;
}
#hostnev a:hover{
    color:blue;
}
#kontener{
    text-align: left;
    display:block;
    margin-left:1em;
}
#fa {
    font-size:12px;
    margin-top:1em;
    margin-left:0;
```

```
    padding-left:0;
    text-align:left;
}
#fa ul{
    margin-left:0;
    padding-left:10em;
}
#fa li {
    list-style-image: none;
    list-style: none;
    margin-left:0;
    padding-left:0;
}
#fa .uccso {

}
#fa .uccso #portnum {
    border-left:2px solid #333;
}
#fa .gyermek {
    border-left:2px solid #333;
}

#fa #sp{
    padding-top:1em;
}

#fa #esor #eszkadatok {
    display:table-cell;
    width:100pt;
    border:2px double #333;
    padding:1ex;
    margin-top:1ex;
    background-color: #ccc;
}
#fa #eszkadatok #mac {
    font-weight: bold;
}
#fa #portnum {
    padding: 1em 0  0 3px;
    border-bottom:2px solid #333;
    vertical-align:bottom;
    width:5em;
    height:2em;
    float:left;

}
#fa #portnum img {
    height:12px;
    width:16px;
}
#fa #gyermekek {
    display:block;
}
#fa .nemok {
```

```
    color:red;
    border-color:red;
}
```



6.5. ábra. Egy csomópont eszközeinek fája a web böngészőben.

7. fejezet

A hálózati menedzsment szoftver használatával kapcsolatos tapasztalatok

Ahogy a bevezetőben már említettem, abban a szerencsés helyzetben vagyok, hogy az általam fejlesztett szoftver modul nem csupán teszt környezetben, hanem „éles" körülmények között működik. Ebből kifolyóan számos tapasztalatot sikerült gyűjtenem, melyek újabb kérdéseket, megoldásra váró problémákat, illetve fejlesztési irányokat vetnek fel.

Az általam fejlesztett program modul két különböző környezetben fut: a menedzsment szoftver webes, grafikus felületébe integrálva, és a felügyeleti rendszertől külön egy kis teljesítményű „stand alone" pc-én.

A grafikus környezetbe illesztett modul, a térképezés nyomán feltöltött adatbázisra támaszkodva jeleníti meg a kiválasztott végpontokon működő eszközök hierarchiáját. Illetve a felhasználói felületen egy „gomb nyomással" elvégezhető a kívánt szegmens újratérképezése, és ezáltal adatbázis frissítése. Ezt felületet használják a hálózatüzemeltetők illetve technikusok az aktív eszközök helyének, valamint elrendezésének felderítésére. Az a tapasztalat, hogy a mindennapos hibaelhárítás során jelentős segítséget nyújt a technikusoknak, szerelőknek ez a funkció, különösen akkor ha olyan területen kell dolgozniuk amelyen még esetleg nem jártak, illetve amelyről nem rendelkeznek kellő helyismerettel.

A megjelenített fa struktúra, mutatja az eszközök állapotát, pontosabban eltérő színnel jelzi a hibás eszközöket, így rögtön látható, hogy mely területek esnek ki a szolgáltatásból. Ez a funkció többször segítséget nyújtott egy-egy hiba helyének meghatározásában.

Sajnos a térképezés egy adott végponton történő lefutásának idejét nagyban befolyásolja az ott működő eszközök száma illetve típusa. Mivel a különböző típusú switchek más-más hozzáférést nyújtanak az ARP táblájuk lekérdezéséhez, így nem lehet előre megbecsülni a fo-

lyamat hosszát. Minél több berendezés van egy adott végponton, annál nagyobb számú MAC cím bejegyzéssel, ARP tábla mérettel kell számolnunk aktíveszközönként. Az algoritmus, jelen pillanatban szekvenciálisan halad végig az egy végponton működő hardvereken, így ha túl nagy az eszközök száma előfordulhat, hogy a sorban „később" következők ARP táblája elévül. Ilyen esetben előfordulhat, hogy nem sikerül a hierarchia felépítése, ezért ki kell kényszeríteni a frissítést, és újra kell indítani a folyamatot. Sajnos ebben az esetben a felhasználó akár több perces várakozás után nem a teljes fát kapja, hanem maradnak olyan eszközök amelyek „árván" szerepelnek.

Az a program amely nem a menedszment szoftvert működtető gépen fut, végzi a hálózat folyamatos térképezését. Ennek két fő oka van: ne terhelje a felügyeleti rendszer erőforrásait ez a funkció, mivel folyamatosan működik. Valamint amíg jelentősebb fejlesztési, módosítási, tesztelési fázisok lehetségesek, ezek ne befolyásolják a menedzsment rendszer működését.

Ha a rendszer „tökéletesen" működik az utóbbi programnak tulajdonképpen csak egyszer kell lefutnia, hogy felderítse a hálózatot és ezután, amennyiben felügyeleti rendszer változást „érzékel" a hálózatban a grafikus környezetbe integrált modul automatikusan frissíti az adott szegmens hierarchiáját reprezentáló adatbázis bejegyzéseket. Viszont az előre nem várt hibák, és esetleges futási szünetek miatt biztosabb megoldás egy háttérben futó folyamat, amely rendszeresen térképezi a hálózatot.

A hálózat dinamikus növekedése miatt a program terhelése, futási ideje is nő. A teljes hálózat feltérképezését a szoftver jelen verziója több mint négy óra alatt végzi el. Természetesen ezt az időt lehetne rövidíteni a logolási funkció kikapcsolásával, így viszont nem nyernénk adatokat az estlegesen felmerülő hibákról, vagy a sikertelen térképezési szakaszokról.

8. fejezet

A hálózati menedzsment szoftver továbbfejlesztésének lehetőségei

Az informatikai hálózatok üzemeltetése folyamán az adott rendszer speciális tulajdonságai miatt, és szolgáltatásokat igénybe vevő csoportok részéről rendszerint újabb és újabb igények merülnek föl. A következőkben felvázolok néhány fejlesztési irányt, lehetőséget amelyek a mindennapos használat folyamán felmerültek, illetve amelyeket én szeretnék megvalósítani a saját munkám megkönnyítése, vagy pusztán szakmai érdeklődés végett.

8.1. A rendszer felkészítése más típusú aktíveszközök hálózatba kerülésére

Amennyiben az eddig használt eszközöktől különböző típusú switch kerül a hálózatba, nincs más tehendő, mint az eszköznek megfelelő osztály implementációjának elkészítése az `ASwitch` absztrakt osztály leszármazottjaként, és az új osztály importálása az `IHostFactory` interfészt implementáló osztályba.

Amennyiben nem switch, hanem más funkciójú aktíveszköz kerül a rendszerbe, akkor pedig az `AActiveElement` absztrakt osztályból kell származtatni az eszközt megvalósító ostályt, és felruházni a rá vonatkozó speciális lehetőségekkel.

Ha az új eszközök implementációjakor megtartjuk a származtatási hierarchiát, és helyesen implementáljuk a megfelelő absztrakt metódusokat, akkor a fa építő algoritmus helyesen működik tovább.

8.2. A térképező eljárás párhuzamos megoldása

Ahogy azt már a 7. fejezetben említettem a hálózat dinamikus növekedésével, nagyságrendekkel megnőhet a hálózat teljes feltérképezésének ideje. Valamint az olyan végpontokon, amelyeken túl nagy eszköz szám van, vagy a kihelyezett berendezések típus összetétele miatt meghiusúlhat, vagy csak többszöri próbálkozás után mehet végbe az aktív eszközök fájának felépítése. Erre a problémára adhat egyfajta megoldást, ha a térképező eljárás bizonyos tevékenységeit párhuzamosítani tudjuk.

Szekvenciális végrehajtás során az egy csomóponton lévő eszközök ARP táblájának elévülése megakadályozhatja szükséges adatok kinyerését, ezért célszerű lenne párhuzamosan lekérni a megfelelő információkat ezekből az eszközökből.

A futási idő csökkentését pedig azzal lehetne elősegíteni, ha bizonyos hálózati szegmenseket párhuzamosan térképeznénk föl. Ehhez azt kell figyelembe venni, hogy a hálózat struktúrájában, hol vannak azok a pontok, ahol már olyan információkkal rendelkezünk, hogy ezt meg tudjuk valósítani.

Mindkét gyorsítási eljárásnál erősen meg kell fontolni az erőforrás igényt, valamint azt, hogy az adatgyűjtési folyamatok mennyire terhelik meg a hálózat adatátviteli keresztmetszetét.

8.3. Hálózati fa mélység vizsgálata

Könnyen implementálható olyan funkció amely felismeri, hogy egy-egy végponton üzemelő eszközök kapcsolódási hierachiája nem optimális. Azaz túl sok eszköz van egymásra felfűzve, így túl mély az adott hálózati részfa, ami szűk adatátviteli keresztmetszetet okozhat. Ezt a funkciót tovább gondolva kifejleszthető olyan program modul is amely a felismerésen túl javaslatot is fogalmazhat meg az optimális hálózati struktúra kialakítására, az aktív eszközök terheltségének vizsgálata alapján.

8.4. Hosztok automatikus keresése a hálózatban.

Magas ügyfélszám esetén viszonylag gyakran előfordul, hogy bizonyos hosztokat be kell azonosítani és meg kell találni a hálózatban. Meg kell találni, hogy melyik eszköz melyik portján forgalmaz, illetve fizikailag hol van a keresett ügyfél illetve eszköz. Tipikusan akkor fordul elő ilyen eset, ha egy ügyfél soho-routere rosszul van rákapcsolva a hálózatra: nem a *wan* portján keresztül kapcsolódik, hanem valapelyik *switch* portján keresztül. Ilyen esetben

DHCP szerverként működik egy adott végponton. Illetve előfordulhat, hogy valamilyen rosszakaratú, vagy illegális tevékenység miatt kell megtalálni egy adott eszközt. Aszerint , hogy milyen információkkal rendelkezünk, a teljes hálózaton keresztül a legfelső szinttől elindulva az aktíveszközök ARP, illteve IP-ARP tábláiból kikeresve a megfelelő bejegyzéseket be tudjuk azonosítani, hogy mely switch mely fizikai interfészéhez kapcsolódik a keresett hoszt.

8.5. Automatikus ügyfél átcsoportosítás

Ahogyan azt a 3.2 fejezetben írtam a hálózat felhasználói csoportjait layer 2-es szinten, VLAN-okba rendezve szeparáljuk el egymástól. A magas ügyfélszám miatt, magukat az előfizetői csoportokat is érdemes több VLAN-ba sorolni, az adott hálózati szegmensben. Így az egyes ügyfélcsoportokat kiszolgáló NAS szerverek terheltségét rugalmasan lehet szabályozni. Ehhez viszont az adott ügyfélcsoportot összefogó switcheken is néha módosítani kell a VLAN beállításokat. Ahhoz hogy ezt a feladatot megoldjuk pontos nyílvántartás kell az adott területeken lévő ügyfélszámról, és aktív eszközökről.

A konfigurációs módosítások automatizálását két részre kell bontani:

1. Az aktív eszközök kézi konfigurációjának kiküszöbölése.

2. Az erőforrás elosztás optimális megoldásának keresése.

Az első logikai réteg jelen pillanatban is része a nálunk működő menedzsment rendzernek. Ezt a feladatot egy kényelmes web felületen el tudjuk végezni, így egy időben több berendezés beállításának automatikusan konfigurálható. Viszont az egyes területeken lévő ügyfélszám és fizikai erőforrások alapján történő becsléseket nekünk, üzemeltetőknek kell elvégezni. Valamint előre fel kell ismerni a dinamikusan változó ügyfélszám következtében kialakuló szűk keresztmetszetet.

A második logikai rész megvalósításához, pontos és naprakész adatok szükségesek. Mivel pontos nyílvántartással rendelkezünk az ügyfelekről, a hálózatról illetve erőforrásokról, valamint a menedzsment rendszer és az általam írt szoftver komponens segítségével folyamatosan nyomonkövethetők a hálózatban történt fizikai változások így jelen pillanatban is rendelkezésünkre áll a megoldáshoz szükséges adatbázis háttér.

A megfelelő adatforrások ismeretében lehetőség van egy optimális megoldást kereső eljárás implementálására, amely ajánlást tesz a területek átcsoprtosítására, vagy az első logikai réteggel együtműködve el is végzi a megfelelő konfigurációs beállításokat.

8.6. A térképező komponens más környezetben történő implementálása

Ahogyan azt a bevezetőben vázoltam a program modult vagy csomagot úgy terveztem, és a fejlesztés során olyan módszereket alkalmaztam, hogy be lehessen illeszteni más felügyeleti rendszerbe, illetve könnyen implementálható legyen eltérő programozási környezetben.

A szoftvert alkotó osztályok gond nélkül implementálhatók más programozási nyelveken, amelyek támogatják az objektumorientált programozási módszereket. Így a szintaktikai átíráson kívül, kevés változtatással a program túlnyomó része könnyen átírható például Java, C++, C# nyelvekre, illetve hozzáigazítható más operációs rendszerekhez.

9. fejezet

Összefoglalás

Szakdolgozatomban egy már működő hálózati menedzsment rendszerbe illeszkedő, általam fejlesztett programmodult mutattam be. Ezen szoftver komponens feladata a nagyvárosi ethernet hálózatban működő kapcsolók csatlakozási hierarchiájának feltérképezése és adminisztrálása.

Bemutattam, hogy egy igen egyszerű algoritmus segítségével könnyen elvégezhető a rendezés, és a valóságot reprezentáló fa struktúra felépítése amennyiben helyes, és kielégítő információkkal rendelkezünk a szóbanforgó eszközökről. Viszont az implementáció elkészítéséhez több témakörben kellett sokrétű ismeretekre szert tennem: hálózat menedzsment, speciális kommunikációs protokollok, gyártóspecifikus megoldások, objektum orientált tervezési minták alkalmazása, a PHP nyelv újabb lehetőségeinek megismerése, és kihasználása. Ezen apró építőelemek összegyúrásával jutottam el a feladat megoldásának jelenlegi fázisába.

A szoftver implementálása folyamán az egyes részfeladatokat megoldva, lépésről lépésre alakult ki a jelen verzió. A program alkotóelemeit funkciójuk, illetve helyetesíthetőségük, és újra felhasználhatóságuk szerint soroltam különböző rétegekbe. Ezen programrészek az egymásnak nyújtotta felületeken keresztül kapcsolódnak egymáshoz és ajánlanak, illetve használnak fel megfelelő szolgáltatásokat.

A szoftver elemeit úgy alakítottam ki, hogy minél általánosabb megoldást nyújtsanak az adott funkció ellátására. A moduláris szerkezet miatt könnyen kicserélhetők, továbbfejleszthetők az egyes komponensek. Ehhez nyújtott kényelmes megoldást a PHP5 nyelv által biztosított objektumorientált környezet.

A fejlesztés során az egyik legfáradságosabb feladat az egyes switch típusok MAC address tábláját megvalósító MIB modulok feltérképezése és értelmezése volt. Ugyanis ezeket az információkat kevés gyártó teszi közzé, és néha egyes dokumentációk megszerzése is reménytelen.

Volt olyan típus, amelynél ezt nem is sikerült megvalósítani, így nem feltétlenül „szép”, de célravezető megoldással kellett áthidalnom a problémát.

A szoftvermodul gyakorlati alkalmazása során szerzett tapasztalatok ismeretében, és a felmerült felhasználói igényeknek köszönhetően szükségessé vált a program továbbfejlesztése. Jelenleg csak a debreceni szolgáltatási területen működő menedzsment rendszer részét képezi, de a közeljövőben más településeken is bevezetésre kerül. Néhány újabb funkcióval kibővítve a jelenlegi felügyeleti szoftver több feladatát is ellátja majd. Az internet szolgáltatás támogatásán túl, a hálózat VoIP eszközeinek nyilvántartásában is szerepet kap az általam fejlesztett programmodul.

A. Függelék

1. számú melléklet

A.1. default package

A.1.1. CNetworkMapper

```
 include_once($CLASS_DIR."db/CDBInterface.php");
 include_once($CLASS_DIR."db/CNetworkDataTools.php");
 include_once($CLASS_DIR."network/CSwitchFactory.php");
 include_once($CLASS_DIR."network/CSwitchTree.php");
 /**
  * class CNetworkMapper
  * Egy konkrét hálózattérképező osztály amely az Access rétegben lévő switcheket
  * rendezi fa struktúrába.
  * @package default
  * @file CNetworkMapper.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.21.
  */
 class CNetworkMapper {
   /**
    * A konfig fájlból betöltött változók listája.
    * @var array
    */
   private $vars=array();
   /**
    * Adatbázis interfész.
    * @var CDBInterface.
    */
   private $dbInterface;
   /**
    * Az térképezéshez szükséges adatokat szolgáltató, illetve frissítő felület az
    * adatbázis és az adatszerkezet között.
    * @see CNetworkDataTools
    * @var CNetworkDataTools
    */
   private $dataTools;
   private $logFile;
   /**
    * Switcheket előállító osztály egy példánya.
    * @var CSwitchFactory
    */
   private $factory;
   /**
    * Az ismert switchtípusoknak megfelelő osztályokat tartalmazó lista.
    * A tömb indexei azok a MAC cím részletek, amellyel azonosítható az adott
    * switch típus.
    * @var array
    */
   private $switchTypes=array(array("mac"=>"00:19:c6","className"=>"CSwitchZte"),
 array("mac"=>"00:1e","className"=>"CSwitchCisco"),
array("mac"=>"00:0e:8","className"=>"CSwitchCisco"),
array("mac"=>"00:1c","className"=>"CSwitchCisco"),
array("mac"=>"00:1f","className"=>"CSwitchCisco"),
array("mac"=>"00:1b","className"=>"CSwitchCisco"),
array("mac"=>"00:19","className"=>"CSwitchCisco"),
array("mac"=>"00:06:19","className"=>"CSwitchSwh"),
array("mac"=>"00:d0:d0","className"=>"CSwitchZte"),
```

```
array("mac"=>"00:0f:e2","className"=>"CSwitchHuawei"));
  /**
   * SNMP interfész amely segítségével lekérdezhetjük a menedzsment VLAN-hoz
   * tartozó eszközöket.
   * @var CSNMPInterface
   */
  private $snmpInterface=NULL;
  /**
   * Hibaüzenet.
   * @var string
   */
  private $errorMsg=NULL;
  /**
   * Konstruktor
   * @param string $configFilePath A konfigurációs állomány elérési útja.
   * @throws EInvalidParameterException H a hibás, vagy hiányos valamelyik
   * paraméter.
   */
  public function __construct($configFilePath){
    if(!$this->loadConfig($configFilePath)){
        $this->errorMsg="Can't create configuration ";
          throw new EInvalidParameterException ($this->errorMsg,1);
      }
      else {
        if(!empty($this->vars["logFile"])){
          $this->logFile=$this->vars["logFile"];
          if(!is_file($this->vars["logFile"])){
            if(!touch($this->logFile)){
              $this->errorMsg="Can't create log file: ".$this->logFile;
                throw new EInvalidParameterException($this->errorMsg,1);
            }
          }
        }
        else {
          $this->errorMsg="There is not Log file path parameter!";
            throw new EInvalidParameterException ($this->errorMsg,1);
        }
        if(!empty($this->vars["hostname"]) && !empty($this->vars["user"])){
          $this->dataTools=new CNetworkDataTools(
new CDBInterface($this->vars["hostname"],
$this->vars["dbName"],$this->vars["user"],$this->vars["password"]));
        }
        else {
          $this->errorMsg="Can't create DB interface!";
            $this->log($this->errorMsg);
              throw new EInvalidParameterException ($this->errorMsg,1);
        }
        if(!empty($this->vars["snmpROCommunity"]) &&
!empty($this->vars["snmpVersion"])){
          $this->snmpInterface=new CSNMPInterface( $this->vars["snmpROCommunity"],
$this->vars["snmpRWCommunity"],$this->vars["snmpVersion"]);
        }
        else {
          $this->errorMsg="Can't create SNMP interface !";
            $this->log($this->errorMsg);
              throw new EInvalidParameterException ($this->errorMsg,1);
        }
      }
      $this->factory= new CSwitchFactory($this->swichTypes);
        $this->factory->setHostTypes($this->switchTypes);
  }
  /**
   * A hibaüzenethez ad hozzáférést.
   * @return string
   */
  public function getErrorMsg (){
    return $this->errorMsg;
  }
  /**
   * Beállítja az SNMP interfészt.
   * @param CSNMPInterface $snmpInterface Az SNMP interfész példány.
   */
  public function setSnmpInterface($snmpInterface){
    if($snmpInterface instanceof CSNMPInterface ){
      $this->snmpInterface=$snmpInterface;
    }
  }
  /**
   * Egy cisco porthoz tartozó eszközöket rendezi fába, és frissíti adatbázisban.
   * @param int $id A cisco eszköz adatbázis azonosítója.
   * @param string $port A cisco eszköz fizikai interfésze, amin a rendezés
   * történik.
   */
  public function updateOnePort($cid,$cport) {
    if(!$cid || !$cport) {
      return false;
    }
    $cdata=$this->dataTools->getCiscoDataById($cid);
      if(!is_array($cdata)){
```

```
        $this->errorMsg.="There is not data result in DB about [id=$cid] cisco";
          $this->log($this->errorMsg);
            $this->log("Updating wasn't succesfull on [id=$cid] cisco");
        return false;
      }
      else {
$cdata["snmpParams"]=array("roCommunity"=>$this->vars["cSnmpROCommunity"]
, $rwCommunity"=>$this->vars["cSnmpRWCommunity"],
"version"=>$this->vars["cSnmpVersion"]);
        $cdata["telnetLogin"]=$this->vars["cTelnetLogin"];
        $cdata["telnetPassword"]=$this->vars["cTelnetPassword"];
      }
      $this->log("Udating elements' tree on ".$cdata["hostname"].
" [id=$cid] cisco's $cport port");
      if(!$switchData=$this->createSwitchList
($this->dataTools->getSwitchDataListByCisco($cid,$cport))){
        $this->errorMsg.= "Empty switch data list n ".$cdata["hostname"].
" [id=$cid] cisco's $cport port.";
          $this->log($this->errorMsg);
            $this->log("Updating wasn't succesfull n ".$cdata["hostname"].
" [id=$cid] cisco's $cport port");
        return false;
      }
      if(!$this->isReachable($switchData,$this->snmpInterface)){
        $this->errorMsg="There is an unreachable element!";
          $this->log($this->errorMsg);
            $this->log("Updating wasn't succesfull n ".$cdata["hostname"].
" [id=$cid] cisco's $cport port");
        return false;
      }
      $this->ping(&$switchData);
        print_r($switchData);flush();
          $tree=new SwitchTree($switchData,
$this->factory,$this->dataTools->getRouterMacByCisco($cid));
            $tree->buildTree();
            if(!$tree->isValidTree()){
              $log="Not valid tree on n ".$cdata["hostname"].
" [id=$cid] cisco's $cport port. Invalid elements:";
              $it=$tree->iterator();
                while($it->hasNext()){
                  $sw=$it->next();
                  if(!$sw->getParent()){
                  }
                  if(!$sw->getUplinkPort() || !$sw->getParent()){
                    $log.="\n\tType:".$sw->getType()." MAC:".$sw->getMac().
" IP:".$sw->getIP();
                  }
                }
                $this->log($log);
                   $this->log("Updating wasn't succesfull n ".$cdata["hostname"].
" [id=$cid] cisco's $cport port");
                    unset ($tree);
                return false;
            }
            else { // adatbázis frissítés
              $this->dataTools->updateSwitchListInDb(&$switchData,$tree);
            }
              $this->log("Updating was succesfull on ".$cdata["hostname"].
" [id=$cid] cisco's  $cport port");
      unset($tree);
    return true;
  }
  /**
   * A switchel adatainak listáját állítja elő.
   * Kibővíti a paraméterül kapott rekordokat a térképezéshez szükséges adatokkal.
   * @param array $ipMacList Az IP és MAC címeket tartalmazó ista.
   * @return array Az elkészített adat lista.
   */
  public function createSwitchList ($ipMacList){
    if(!is_array($ipMacList)){
      return NULL;
    }
    $snmpParams= array("roCommunity"=>$this->vars["snmpROCommunity"],
"rwCommunity"=>$this->vars["snmpRWCommunity"],
"version"=>$this->vars["snmpVersion"]);
    foreach($ipMacList as $k=>$e){
      $e["vlans"]=array($this->vars["mgmtVlan"]);
      $e["snmpParams"]=$snmpParams;
      $e["telnetLogin"]=$this->vars["telnetLogin"];
      $e["telnetPassword"]=$this->vars["telnetPassword"];
      $result[]=$e;
    }
    return $result;
  }
  /**
   * Log bejegyzést tesz
   * @param string $msg A bejegyzés szövege
   * @return boolean A sikeresség
   */
```

```
  private function log($msg){
    if(!is_file($this->logFile) || !($fp=fopen($this->logFile,"a+"))){
      return false;
    }
    $msg=date("[Y-m-d H:i:s]")." $msg\n";
      fwrite($fp,$msg);
        fclose($fp);
    return true;
  }
  /**
   * Megállapítja, hogy az adott eszközök elérhetők-e.
   * @param array $entries A switchek listája.
   * @param CSNMPInterface $snmpInterface Megfelelő snmp interfész az elérhetőség
   * vizsgálatához.
   * @return boolean A sikeresség.
   */
  private function isReachable($entries,$snmpInterface) {
    if(!is_array($entries) || !($snmpInterface instanceof CSNMPInterface)){
      return false;
    }
    foreach($entries as $k=>$e){
      try{
        $result=$snmpInterface->get($e["ip"],"system.sysDescr.0");
      }
      catch(ESNMPException $e){
        $this->log("Element is unreachable! ip:".$e["ip"]." mac:".$e["mac"]."");
          return false;
      }
    }
    return true;
  }
  /**
   * Elkészít egy részfát az adatokat tartalmazó lista alapján.
   * @param array $switchDataList A switxchek adatait tartalmazó tömb.
   * @param string $routerMac Az eszközök átjárójának menedzsment VLAN-ban felvett
   * MAC címe.
   * @return CSwitchtree A felépített fa.
   */
  private function createSwitchTree ($switchDataList,$routerMac){
    if(!is_array($switchDataList)){
      return NULL;
    }
    for($i=0;$i<count($switchDataList);$i++){
      $switchDataList[$i]["vlans"]=array($this->mgmtVlan);
      $switchDataList[$i]["snmpParams"]=$this->snmpInterface;
      $switchDataList[$i]["telnetLogin"]=$this->telnetLogin;
      $switchDataList[$i]["telnetPassword"]=$this->telnetPasswd;
    }
    $tree=new CSwitchTree($switchDataList,$this->factory,$routerMac);
    return $tree;
  }
  /**
   * Az Access réteg tetején lévő cisco switch minden portjához kapcsolt switch fa
   * feltérképezése.
   * @param int $cid A cisco eszköz adatbázis ID-ja.
   * @return boolean A sikeresség.
   */
   public function updateOnCiscoByID ($cid){
    $cdata=$this->dataTools->getCiscoDataById($cid);
      if(!is_array($cdata)){
        $this->errorMsg.="There is not data result in DB about [$cid] cisco";
        $this->log($this->errorMsg);
          return NULL;
      }
      else {
    $cdata["snmpParams"] = array("roCommunity"=>$this->vars["cSnmpROCommunity"],
"rwCommunity"=>$this->vars["cSnmpRWCommunity"],
"version"=>$this->vars["cSnmpVersion"]);
        $cdata["telnetLogin"]=$this->vars["cTelnetLogin"];
        $cdata["telnetPassword"]=$this->vars["cTelnetPassword"];
        $cdata["vlans"]=array(130);
      }
        $cisco=new CSwitchCisco($cdata);
        if(is_array($ifList=$cisco->getInterfaces())){
          $result=true;
          $this->log("Update all ports of ".$cdata["hostname"].
" [id=$cid] cisco switch");
          foreach($ifList as $k=>$e){
            if($k[0]=="F" && $cisco->getMacCountOnPort($k)){
              $cport=substr($k,4);
                if(!$this->updateOnePort($cid,$cport)){
                  $result=false;
                }
            }
          }
        }
        else {
          $this->errorMsg.="There is not interface list on ".$cdata["hostname"].
" [id=$cid] cisco switch";
```

```
            $this->log($this->errorMsg);
              $result = false;
        }
        if($result) {
          $this->log("Updating was succesfull on all ports of "
.$cdata["hostname"]." [id=$cid] cisco switch");
        }
        unset($cisco);
    return $result;
  }
  /**
   * Az egy distribution L3 switchhez tartozó hálózati fa frissítése.
   * @param int $rid Az L3-as eszköz adatbázis ID-ja.
   */
  public function updateOneDistributionElement($rid){
    if(!($r=$this->dataTools->getCiscoListByDrId($rid))){
      $this->errorMsg.="There is not data result in DB about [$rid] L3 ".
"distribution switch";
        $this->log($this->errorMsg);
          return NULL;
    }
    $this->log("(START) Update all cisco switches on $rid L3 distributon switch");
    $result=true;
      foreach($r as $k=>$e){
        if(!$this->updateOnCiscoByID($e["id"])){
          $result=false;
        }
      }
    if($result){
      $this->log("(STOP) Updating was succesfull on $rid L3 distribution switch");
    }
    else {
      $this->errorMsg="Updating wasn't succesfull on $rid L3 distribution switch";
        $this->log("(STOP) ".$this->getErrorMsg());
    }
    return $result;
  }
  /**
   * Az egy adott HUB minden distribution eszköze alatt lévő hálózatzot frissíti
   * le.
   * @param int $hid A HUB adatbázis kulcsa.
   * @return boolean A sikeresség.
   */
  public function updateOneHub($hid){
    if(!($r=$this->dataTools->getDrListByHubId($hid))){
      $this->errorMsg.="There is not data result in DB about [id=$hid] HUB";
        $this->log($this->errorMsg);
          return NULL;
    }
    $this->log("(START) Update all distribution L3 switch at ".$r[0]["hub"].
" [id=$hid] HUB.");
      $result=true;
        foreach($r as $k=>$e){
          if(!$this->updateOneDistributionElement($e["id"])){
            $result=false;
          }
        }
    if($result){
      $this->log("(STOP) Udate was succesfull at  ".$r[0]["hub"].
" [id=$hid] HUB.");
    }
    else {
      $this->errorMsg="Udate wasn't succesfull at  ".$r[0]["hub"].
" [id=$hid] HUB.";
        $this->log("(STOP) Udate wasn't succesfull at  ".$r[0]["hub"].
" [id=$hid] HUB.");
    }
    return $result;
  }
  /**
   * A teljes hálózat frissítése.
   * @return boolean A sikeresség.
   */
  public function updateAllHubs(){
    if(!$this->dataTools->getHubIdList()){
      $this->errorMsg.="There is not data result in DB about network HUBs";
        $this->log($this->errorMsg);
          return NULL;
    }
    $this->log("(START) Update all HUBs in network");
      $result=true;
        foreach($r as $k=>$e){
          if(!$this->updateOneHub($e["id"])){
            $result=false;
          }
        }
    if($result){
      $this->log("(STOP) Udating of all HUBs was succesfull.");
    }
```

```
    else {
      $this->errorMsg="Udating of all HUBs wasn't succesfull.";
        $this->log("(STOP) ".$this->errorMsg);
    }
    return $result;
  }
  /**
   * Az adott eszközök pingelését végzi el.
   * @param array $lista Az eszközök IP címeit tartalmazó tömb.
   * @return boolean A sikeresség.
   */
  public function ping (&$lista){
    if(!is_array($lista)){
      return false;
    }
    $result=true;
    foreach($lista as $k=>$e){
      $ex="ping -c 4 ".$e["ip"];
          exec($ex,$o,$r);
            if($r!=0){
              $result=false;
            }
    }
    return $result;
  }
  /**
   * Beolvassa a konfigurációs állományt, és a tartalma alapján inicializálja a
   * megfelelő változókat.
   * @param string $filePath A konfigurációs állomány elérési útja.
   * @return boolean A sikeresség.
   */
  private function loadConfig($filePath){
    if(!$fp=fopen($filePath,"r")){
      print "hiba";
      return false;
    }
    $config=NULL;
      while (!feof($fp)){
        $config.=fread($fp,1024);
      }
    if(!$config){
      return false;
    }
    $rows=split("\r\n|\n",$config);
      if(!is_array($rows)){
        return false;
      }
    print_r($rows);
    $result=false;
      foreach($rows as $k=>$e){
        if(preg_match("/^[a-z]/",$e)){
          $e=trim($e);
          $i=0;
            $varname=$value=NULL;
            while($i<strlen($e)){
              if($e[$i]==" " || $e[$i]=="\t"){
                break;
              }
              else {
                $varname.=$e[$i];
              }
              $i++;
            }
            $e=trim(substr($e,$i));
            $i=0;
            while($i<strlen($e)){
              if($e[$i]==" " || $e[$i]=="\t"){
                break;
              }
              else {
                $value.=$e[$i];
              }
              $i++;
            }
          $this->vars[$varname]=$value;
            $result=true;
        }
      }
    return $result;
  }
}
```

A.2. db package

A.2.1. CDBInterface

```
 include_once $CLASS_DIR.'exception/EDBException.php';
 /**
  * class CDBInterface
  * Az adatbázis kezelési felületet megvalósító osztály
  * @package default
  * @subpackage db
  * @file CDBInterface.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.21.
  */
class CDBInterface {
     /**
      * Az adatbázis neve.
      * @var string
      */
    private $dbName;
    /**
     * Az adatbázis felhasználó.
     * @var string
     */
    private $dbUser;
    /**
     * Az adatbázis jelszó.
     * @var string
     */
    private $dbPswd;
    /**
     * Az adtbázis szerver hosztneve vagy IP címe.
     * @var string
     */
    private $dbHostName;
    /**
     * Az adatbázis kapcsolat erőforrás.
     * @var resource
     */
    private $dbConn=NULL;
    /**
     * Az adatbázis interfész konstruktora.
     * @param string $dbHostName Az adatbázis szerver IP címe vagy hosztneve.
     * @param string $dbName Az adatbázis neve.
     * @param string $dbUser Az adatbázis felhasználói név.
     * @param string $dbPswd Az adatbázis jelszó.
     */
    public function __construct($dbHostName,$dbName,$dbUser,$dbPswd) {
        $this->dbHostName=$dbHostName;
            $this->dbName=$dbName;
                $this->dbUser=$dbUser;
                    $this->dbPswd=$dbPswd;
    }
    /**
     * Csatlakozás az adatbázishoz.
     * @throws EDBException Kapcsolódási hiba esetén váltódik ki.
     */
    public function connect () {
        if(!$this->dbConn=mysql_pconnect($this->dbHostName,$this->dbUser,
$this->dbPswd)){throw new EDBException("Can't open connection to $this->dbHostName
 database server! (".mysql_error().")",11);
        }
        if(!mysql_select_db($this->dbName,$this->dbConn)){
            throw new EDBException("Can't connect to $this->dbName database!
(".mysql_error().")",12);
        }
    }
    /**
     * Az sql select kérés eredményét adja vissza asszociatív tömb formájában.
     * @param string $selectQuery Az SQL select.
     * @return array|NULL Az eredmény rekordjainak tömbje.
     */
    public function queryToArray ($selectQuery) {
        if(!$this->dbConn){
            try{
                $this->connect();
            }
            catch (EDBException $e){
                unset($e);
                    return NULL;
            }
        }
        $result=NULL;
            if(($r=mysql_query($selectQuery,$this->dbConn))===false){
```

```
                throw new EDBException ("SQL error! (".mysql_error().")",13);
            }
            while ($row=mysql_fetch_assoc($r)){
                $result[]=$row;
            }
        return $result;
    }
    /**
     * SQL updatet hajt végre.
     * @param string $updateQuery Az SQL update.
     * @return int A frissített sorok száma.
     */
    public function update ($updateQuery){
        if(!$this->dbConn){
            try{
                $this->connect();
            }
            catch (EDBException $e){
                unset($e);
                    return NULL;
            }
        }
        $result=NULL;
        if(($r=mysql_query($updateQuery,$this->dbConn))===false){
                throw new EDBException ("SQL error! (".mysql_error().")",14);
            }
        return $r;
    }
    /**
     * Destruktor.
     * Lezárja az adtbázis kapcsolatot.
     */
    public function __destruct(){
        if($this->dbConn){
            mysql_close($this->dbConn);
        }
    }
}
?>
```

A.2.2. CNetworkDataTools

```
include_once ($CLASS_DIR."util/CMacUtil.php");
include_once ($CLASS_DIR."db/CDBInterface.php");
/**
 * class CNetworkDataTools
 * A térképezéshez szükséges adatokat szolgáltatja az adatbázisból. Illetve a
 * felpített fák afdatait frissíti az adatbázisban.
 * @package default
 * @subpackage db
 * @file CNetworkDataTools.php
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.04.21.
 */
class CNetworkDataTools {
 /**
  * Adatbázis interfész.
  * @var CDBInterface
  */
private $dbInterface=NULL;
/**
 * Konstruktor
 * @param CDBInterface $dbInterface Adatbázis interfész.
 */
 public function __construct(CDBInterface $dbInterface) {
 $this->dbInterface=$dbInterface;
  $this->dbInterface->connect();
}
/**
 * Megadja annak a Layer3-as switchnek a menedzsment VLAN-ban felvett MAC címét
 * amelyhez az adott switch tartozik.
 * @param int $cid A cisco switch adatbázis ID-ja.
 * @return string A keresett MAC cím.
 */
public function getRouterMacByCisco($cid) {
 if(!$this->dbInterface){
  return NULL;
 }
 $pid=NULL;
  while($cid){
   $query="select parentid,parenttype from ciscodev where id=$cid";
    if(!$r=$this->dbInterface->queryToArray($query)){
     return NULL;
    }
```

```
     if($r[0]["parenttype"]=="R"){
      if(!$r=$this->dbInterface->queryToArray("select mac from ciscorout where
id=".$r[0]["parentid"])){
       return NULL;
      }
      return $r[0]["mac"];
     }
     else $cid=$r[0]["parentid"];
    }
   return NULL;
 }
 /**
  * Az adott MAC című eszközök adatainak rekordajait adja vissza.
  * @param array A MAC címek listája.
  * @return array A switchek adatainak listája.
  */
 public function getSwitchDataByMacList($macList){
  if(!$this->dbInterface || !is_array($macList)){
   return NULL;
  }
  $query="select VP.id,IC.ip,E.mac,VP.melyeszkhez,VP.portra,VP.portrol from ipcim
IC inner join eszkozok E on IC.id=E.ip inner join vpeszkozok VP on E.id=VP.eszk
where E.mac in (";
   foreach ($macList as $k=>$e){
    $query.="'".CMacUtil::MacToDbFormat($e)."',";
   }
    $query=rtrim($query,",").")";
    print $query;
   if(!($r=$this->dbInterface->queryToArray($query))){
    return NULL;
   }
   return $r;
 }
 /**
  * Az adott cisco adott portjához rendelt eszközök listáját adja vissza.
  *(id,ip,mac,melyeszkhez,portrol,portra)
  * @param int $id
  * @param string $port
  * @return array A rekordokat tartalmazó tömb.
  */
 public function getSwitchDataListByCisco($id,$port=NULL){
  if(!$id || !$this->dbInterface) {
   return NULL;
  }
   $query="select VP.id,IC.ip,E.mac,VP.melyeszkhez,VP.portra,VP.portrol from
eszkozok E inner join ipcim IC on IC.id=E.ip inner join vpeszkozok VP on
E.id=VP.eszk where cid=$id ".($port?" and cport=$port":"");
    $result=$this->dbInterface->queryToArray($query);
  return $result;
 }
 /**
  * A switchek fájának megjelenítéséhez szükséges adatokat olvassa ki az
  * adatbázisból.
  * @param int $cid A cisco switch adatbázis kulcsa.
  * @param int $cport A cisco switch adott portja.
  * @param int $mgmtVlan A menedzsment VLAN azonosítója.
  * @return array Az switchek adatainak rekordjait tartalmazó tömb.
  */
 public function getSwitchDataListForDisplay($cid,$cport,$mgmtVlan){
  if(!$cid || !$cport || !$mgmtVlan || !$this->dbInterface) {
   return NULL;
  }
  $query="select VE.id,ET.tipus,E.mac,IC.ip,VL.vlanid,VE.melyeszkhez,VE.portrol as
uplink,VE.portra,
CONCAT(VU.utca,' ',VP.hsz,' ',VP.hszkieg,' ',VP.emelet) as cim, SS.id as nemok,
CD.hostname, E.cport
from vpeszkozok VE inner join eszkozok E on VE.eszk=E.id inner join eszkoztipus ET
on ET.id=E.tipusid
inner join ipcim IC on IC.id=E.ip inner join vlans VL on VL.vpeszk=VE.id inner
join ciscodev CD on CD.id=E.cid
inner join vp VP on VP.id=VE.vp inner join vputca VU on VU.id=VP.utca
left join servicestate SS on SS.vpeszkid=VE.id
where E.cid=$cid and E.cport=$cport and VL.vlanid<>1 and VL.vlanid<>$mgmtVlan;";
  $result=$this->dbInterface->queryToArray($query);
  return $result;
 }
 /**
  * A keresett cisco eszköz adatit tartalmazó rekordot adja vissza.
  * (id,ip)
  * @param int $id A cisco eszköz adatbázis azonosítója.
  * @return array Az adatokat tartalmazó rekord.
  */
 public function getCiscoDataById ($id){
  if(empty($id)){
   return NULL;
  }
  $query="select CD.id,IC.ip,CD.hostname from ciscodev CD inner join mgmtipcim IC
on CD.ip=IC.id where CD.id=$id";
   if(!$this->dbInterface || !($r=$this->dbInterface->queryToArray($query))){
```

```
    return NULL;
   }
   $result=$r[0];
    $result["mac"]="00:00:00:00:00:00";
  return $result;
 }
 /**
  * Adott Distribution L3 switchre kapcsolt Cisco switchek listáját adja vissza.
  * @param int $id A distribution eszköz adatbázis kulcsa.
  * @return array A cisco switchek adatainak listája.
  */
 public function getCiscoListByDrId($id){
  if(!$id){
   return NULL;
  }
   $query="select CD.id,IC.ip,CD.hostname from ciscodev CD inner join mgmtipcim IC
on CD.ip=IC.id where parenttype='S' and port='I' and parentid in (select id from
ciscodev where parenttype='R' and parentid=$id)";
    if(!($r=$this->dbInterface->queryToArray($query))){
     return NULL;
    }
   $result=NULL;
    foreach ($r as $k=>$e){
     $result[]=$e;
    }
   $found=true;
   while($found){
    $query="select CD.id,IC.ip,CD.hostname from ciscodev CD inner join mgmtipcim
IC on CD.ip=IC.id where parenttype='S' and port='I' and parentid in (";
    foreach($r as $k=>$e){
     $query.=$e["id"].",";
    }
    $query=rtrim($query,",").")";
     if(!($r=$this->dbInterface->queryToArray($query))|| !count($r)){
      break;
     }
     else {
      foreach($r as $k=>$e){
       $result[]=$e;
      }
     }
   }
  return $result;
 }
 public function getRouterMacById($routerId){
  $query="select mac from ciscorout where id=$routerId";
   if(!$r=$this->dbInterface->queryToArray($query)){
    return NULL;
   }
  return $r[0]["mac"];
 }
 /**
  * Frissíti az adott eszközök adatait az adatbázisban a felépített fa alapján.
  * @param array $switchDataList A switchek adatainak listája.
  * @param CSwitchTree $tree A felépített switch fa
  * @return int|boolean Az updatelt sorok darabszáma, vagy false.
  */
 public function updateSwitchListInDb (& $switchDataList,& $tree){
  if(!is_array($switchDataList) || !$tree instanceof INetworkTree){
   return false;
  }
  foreach($switchDataList as $k=>$e){
   if($host=$tree->getHostByMac($e["mac"])){
    $sqlQuery="update vpeszkozok set";
    if(($p=$host->getParent()) && ($pd=$this->searchElementByMac
($switchDataList,$p->getMac()))){
     $sqlQuery.=" melyeszkhez=".$pd["id"];
     $sqlQuery.=", portra=".$host->getParentPort();
    }
    else {
     $sqlQuery.=" melyeszkhez=NULL, portra=NULL";
    }
    $sqlQuery.=", portrol=".$host->getUplinkPort();
     $sqlQuery.=" where id=".$e["id"];
     if($this->dbInterface->update($sqlQuery)){
      $result++;
     }
   }
  }
  return $result;
 }
 /**
  * Az adatbázisból nyert switch listából kikeres egy rekordot a aswitch adatbázis
  * kulcsa alapján.
  * @param array $switchList A switchek adatainak rekordjait tartalmazó tömb.
  * @param int $id A keresett eszköz adatbázis kulcsa.
  * @return array A megtalált rekordot tartalmazó asszociatív tömb.
  */
 private function searchElementByID($switchList,$id){
```

```
  if(!is_array($switchList)){
   return NULL;
  }
  foreach($switchList as $k=>$e){
   if($e["id"]==$id){
    return $e;
   }
  }
  return NULL;
 }
 /**
  * Switch adat listából kikeres egy rekordot adott MAC cím alapján.
  * @param array $switchDataList A switch adatokat tartalmazó tömb.
  * @param string $mac A keresett bejegyzéshez tartozó MAC címe
  * @return array A megtalált rekord vagy NULL.
  */
 private function searchElementByMac($switchDataList,$mac){
  if(($mac=CMacUtil::validMac($mac)) && is_array($switchDataList)){
   foreach($switchDataList as $k=>$e){
    if(CMacUtil::equals($e["mac"],$mac)){
     return $e;
    }
   }
  }
  return NULL;
 }
 /**
  * Switch adatokat tartalmazó listából kikeresi a MAC címnek megfelelő eszköz
  * adatbázis kulcsát.
  * @param array $switchDataList A switchek listája amelyet az adatbázisból
  * nyertünk.
  * @param int $id A keresett eszköz adatbázis kulcsa.
  * @return string A megtalált MAC cím.
  */
 private function searchMacByID($switchDataList,$id){
  if(!is_array($switchList)){
   return NULL;
  }
  foreach($switchList as $k=>$e){
   if($e["id"]==$id){
    return $e["mac"];
   }
  }
  return NULL;
 }
 /**
  * Az adott HUB-ban lévő distribution eszközök adatait adja vissza.
  * @param int $hid A HUB adatbázis kulcsa.
  * @return array A rekordok listája.
  */
 public function getDrListByHubId($hid){
  if(!$hid){
   return NULL;
  }
  $query="select CR.id,CR.ip,CR.mac,H.nev as hub from ciscorout CR inner join hub
H on CR.hubid=H.id where (CR.parentid<>0 and CR.parentid is not NULL) and
CR.hubid=$hid";
   if(!$this->dbInterface || !($r=$this->dbInterface->queryToArray($query))){
    return NULL;
   }
   return $r;
 }
 /**
  * A hálózat HUB-jainak adatbázis kulcsait adja vissza.
  * @return array Az adatbázis kulcsok listája.
  */
 public function getHubIdList () {
  $query="select id from hub";
   if(!$this->dbInterface || !($r=$this->dbInterface->queryToArray($query))){
    return NULL;
   }
  return $r;
 }
}
```

A.3. network package

A.3.1. CArpTable

```
include_once $CLASS_DIR.'exception/ENoSuchPortException.php';
include_once $CLASS_DIR.'util/CMacUtil.php';
 /**
```

```
  * class CArpTable
  * Az MAC address táblát reprezentáló osztály.
  * Minden egyes {@link ASwitch} osztályt megvalósító konkrét switch példány
  * tartalmaz egy {@link CArpTable} példányt.
  * @see ASwitch
  * @package default
  * @subpackage network
  * @file CArpTable.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.03.10.
  */
class CArpTable{
  /**
   * Az MAC address táblát tartalmazó tömb.
   * A tömb indexei, az interfészek azonosítói, a tömb elemei pedig a MAC címeket
   * tartalmazó tömbök.
   * @var array
   */
  private $arpTable=array();
  /**
   * Konstruktor. A paraméterként kapott interfész lista alapján létrehozza az
   * interfészeket az ARP táblában.
   * @param array $interfaces Az interfészeket tartalmazó tömb.
   */
  public function __construct(& $interfaces){
    $this->setInterfaces($interfaces);
  }
  /**
   * Megkeresi, hogy a MAC cím melyik porton jelentkezik.
   * @param string A keresett MAC cím.
   * @return int A megtalált port.
   */
  public function getPortByMac($mac){
    if(is_array($this->arpTable) && ($mac=CMacUtil::validMac($mac))){
      foreach($this->arpTable as $port=>$macList){
        if(is_array($macList)){
          foreach ($macList as $k=>$e){
            if(CMacUtil::equals($mac,$e)){
              return $port;
            }
          }
        }
      }
    }
    return NULL;
  }
  /**
   * Megadja, hogy az adott porton hány MAC cím található
   * @param string $port Az interfész neve.
   * @return int A MAC címek darabszáma.
   * @throws ENoSuchPortException Ha nem található a keresett port az ARP táblában
   */
  public function getMacCountOnPort($port){
    if(is_array($this->arpTable)){
      if(!array_key_exists($port,$this->getArpData())){
        throw new ENoSuchPortException("Arp Table has not $port
interface",ENoSuchPortException::$kod);
      }
      if(is_array($this->arpTable[$port])){
        return count($this->arpTable[$port]);
      }
    }
    return 0;
  }
  /**
   * Egy adott porthoz tartozó MAC címek listáját adja vissza.
   * @param string $port A keresett port neve.
   * @return array A MAC címek listája.
   * @throws ENoSuchPortException Ha nem található a keresett port az ARP
   * táblában.
   */
  public function getMacListByPort($port){
    if(!array_key_exists($port,$this->arpTable)){
      throw new ENoSuchPortException("Arp Table has not $port
interface",ENoSuchPortException::$kod);
    }
    return $this->arpTable[$port];
  }
  /**
   * Hozzárendel egy MAC címet az ARP tábla egy interfészéhez.
   * @param string $mac A MAC cím.
   * @param int $port Az interfész indexe.
   * @throws ENoSuchPortException Ha nem található a keresett port az ARP táblában.
   */
  public function addMacToPort($mac,$port){
    if(!array_key_exists($port,$this->getArpData())){
      throw new ENoSuchPortException("Arp Table has not $port
```

```
interface",ENoSuchPortException::$kod);
    }
    if ($p=$this->getPortByMac($mac)){
      throw new EDoubleMacException("The $mac MAC already exists on $p port.",11);
    }
    $this->arpTable[$port][]=$mac;
  }
  /**
   * Adott MAC címet eltávolítja az ARP táblából
   * @param string $mac Az eltávolítandó MAC cím.
   * @return string Az eltávolítandó MAC cím, vagy NULL.
   */
  public function removeMac($mac){
    if(!($mac=CMacUtil::validMac($mac))){
      return NULL;
    }
    $result=NULL;
    if($p=$this->getPortByMac($mac)){
      $key=array_search($this->arpTable[$port]);
        $result=$this->arpTable[$port][$key];
          unset ($this->arpTable[$port][$key]);
    }
    return $result;
  }
  /**
   * Kiüríti az ARP Táblát
   */
  public function clearTable () {
    if(is_array($this->arpTable)){
      foreach($this->arpTable as $port =>$Maclist){
        unset($this->arpTable[$port]);
          $this->arpTable[$port]=array();
      }
    }
  }
  /**
   * A MAC address táblát tartalmazó tömbhöz ad hozzáférést.
   * @return array A táblát tartalmazó tömb. Indexei a switxh interfészeinek nevei.
   */
  protected function getArpData(){
    return $this->arpTable;
  }
  /**
   * Beállítja az interfészneveket a táblában.
   * @param array $interfaces Az interfész nevek tömbje.
   */
  public function setInterfaces($interfaces){
    if(is_array($interfaces)){
      $this->arpTable=NULL;
      foreach($interfaces as $k=>$e){
        $this->arpTable[$k]=array();
      }
    }
  }
  /**
   * Megadja, hogy van e bejegyzés a MAC address táblában.
   * @return boolean Ha üres akkor true, egyébként false.
   */
  public function isEmpty() {
    if(!is_array($this->arpTable)){
      return true;
    }
    foreach($this->arpTable as $port=>$e){
      try {
        if(!$this->getMacCountOnPort($port)){
          return false;
        }
      }
      catch (ENoSuchPortException $e){
        unset($e);
      }
    }
    return true;
  }
}
```

A.3.2. CHost

```
$PACKAGE=$CLASS_DIR."network/";
$EXCEPTION_DIR=$CLASS_DIR."exception/";
 include_once($PACKAGE."IHost.php");
 include_once($EXCEPTION_DIR."EHostDoesNotExistException.php");
 include_once($EXCEPTION_DIR."EInvalidMacException.php");
 include_once($EXCEPTION_DIR."EInvalidIPException.php");
 /**
  * class CHost
```

```
 * A hálózaton lévő hosztot megvalósító osztály.
 * @package default
 * @subpackage network
 * @file CHost.php
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.04.10.
 */
class CHost implements IHost {
  /**
   * Az eszköz IP címe.
   * @var string
   */
  private $ip=NULL;
 /**
   * Az eszköz MAC címe.
   * @var string
   */
  private $mac="00:00:00:00:00:00";
 /**
   * A szülő eszköz, amihez az eszköz kapcsolódik.
   * @var IHost
   */
  private $parent=NULL;
 /**
   * A szülő eszköz portja, amihez az eszköz kapcsolódik.
   * @var string
   */
  private $parentPort=NULL;
 /**
   * Konstruktor
   * @param array $hostData A hosz paramétereit tartalmazó tömb (MAC,IP,Szülő
   * eszköz, Szülő eszköz portja)
   * @throws EHostDoesNotExistException Kivételt dob, ha nincsenek adat
   * paraméterek, illetve a MAC cím adatparaméter hiányzik.
   */
  public function __construct($hostData){
    if(!is_array($hostData) || empty($hostData["mac"])){
      throw new EHostDoesNotExistException("The Host couldn't be created",1);
    }
    $this->mac=CMacUtil::validMac($hostData["mac"]);
      if(!empty($hostData["ip"])){
        $this->ip=$hostData["ip"];
      }
        if(!empty($hostData["parent"])){
          $this->parent=$hostData["parent"];
        }
          if(!empty($hostData["parentPort"])){
            $this->parentPort=$hostData["parentPort"];
          }
  }
  /**
   * Lekérdezi a MAC címet.
   * @return string A MAC cím.
   */
  public function getMac() {
    return $this->mac;
  }
  /**
   * Beállítja a MAC címet.
   * @param string $mac A beállítandó MAC cím.
   * @throws EInvalidMacException Ha hibás MAC címet tartalmaz a $mac paraméter.
   */
  protected function setMac($mac){
    if($m=CMacUtil::validMac($mac)) {
      $this->mac=$m;
    }
    else {
      throw new EInvalidMacException("Invalid MAC address: $mac",1);
    }
  }
  public function getIP() {
    return $this->ip;
  }
  /**
   * Beállítja az eszköz IP címét.
   * @param string $ip A beállítandó IP cím.
   * @throws EInvalidIPException Ha nem helyes IP cím van a $ip paraméterben.
   */
  public function setIP ($ip) {
    if(is_string($ip)){
      $minta="/^(([0-9]{1,3}.){3}[0-9]{1,3}){1}$/";
      if(preg_match($minta,$ip)){
        $this->ip=$ip;
      }
      else {
        throw new EInvalidIPException("$ip IP address is invalid",$code);
      }
```

```
    }
  }
  /**
   * Megadja az aktív eszköz szülőjét.
   * @return IHost A szülő eszköz;
   */
  public function getParent () {
    return $this->parent;
  }
  /**
   * Lekérdezi a zülő eszköz portját amihez a hoszt csatlakozik.
   * @return string A szülőeszköz portjának a neve.
   */
  public function getParentPort(){
    return $this->parentPort;
  }
  /**
   * Beállítja az Aktív eszköz szülőjét.
   * @param IHost $parent A szülő eszköz referenciája.
   */
  public function setParent (IHost $parent){
    if(!is_a($parent,"IHost")){

    }
    else {
      $this->parent=$parent;
    }
  }
  /**
   * Beállítja, hogy az aktív eszköz a szülő ezköz mely interfészéhez kapcsolódik.
   * @param string $port A szülő eszköz interfész azonosítója.
   */
  public function setParentPort($port){
    if(!$this->getParent()){
      throw new EHostDoesNotExistException("This element has not got any parent
element",$code);
    }
    $this->parentPort=$port;
  }
  /**
   * Megadja, hogy a host rendelkezik-e gyermek eszközzel.
   * @return boolean Mindig false.
   */
  public function hasChild () {
    return false;
  }
   /**
    * A hoszt adatait tartalmazó HTML string.
    * @return string HTML string.
    */
  public function toString () {
     $result="<div id=\"host\">\n";
       $result.="<div id=\"data\">\n";
          $result.="<div>ip:".$this->getIP()."</div>\n";
          $result.="<div>mac:".$this->getMac()."</div>\n";
        $result.="</div>\n";
     $result.="</div>\n";
     return $result;
   }
 }
```

A.3.3. CIPArpTable

```
 include_once $CLASS_DIR.'util/CMacUtil.php';
 /**
  * class CIPArpTable
  * Az IP-ARP tábla megvalósítása. A bejegyzések IP - MAC - VLAN kötéseket
  * tartalmazzák.
  * A <i>CSwitchL3</i> példányok rendelkeznek egy-egy {@link CIPArpTable}
  * példánnyal.
  * @see CSwitchL3cisco
  * @package default
  * @subpackage network
  * @file CIPArpTable.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.07.
  */
class CIPArpTable {
   /**
    * Az IP-ARP bejegyzések tömbje.
    * Egy bejegyzés IP címet, MAC címet, VLAN azonosítót tartalmaz asszociatív
    * tömb formában.
    * <code>
    *   array ("ip"=>"192.168.0.1","mac"=>"00:00:00:00:00:00","vlan"=>"1");
```

```
  * </code>
  * @var array
  */
private $ipArpTable=array();
/**
 * Konstruktor
 * @param array $entries
 */
public function __construct($entries=NULL){
  if(is_array($entries)){
    foreach($entries as $k=>$e){
      if(!empty($e["ip"]) && !empty($e["mac"]) && !empty($e["vlan"])){
        $this->addEntry($e["ip"],$e["mac"],$e["vlan"]);
      }
    }
  }
}
/**
 * Megkeresi a megadott MAC címhez tartozó IP címet.
 * @param string $mac A megadott MAC cím.
 * @return string A keresett IP cím.
 */
public function getIpByMac ($mac){
  if(!$mac=CMacUtil::validMac($mac) || !is_array($this->ipArpTable)){
    return NULL;
  }
  foreach($this->ipArpTable as $k=>$entry){
    if($entry["mac"]==$mac){
      return $entry["ip"];
    }
  }
  return NULL;
}
/**
 * Megkeresi az adott IP-hez tartozó MAC címet.
 * @param string $ip A megadott IP cím.
 * @return string A megtalált MAC cím vagy NULL.
 */
public function getMacByIp ($ip){
  if(!$this->isValidIp($ip) || !is_array($this->ipArpTable)){
    return NULL;
  }
  foreach($this->ipArpTable as $k=>$entry){
    if($entry["ip"]==$ip){
      return $entry["mac"];
    }
  }
  return NULL;
}
/**
 * Megvizsgálja az IP cím formai helyességét.
 * @param string $ip A megvizsgálandó IP cím.
 * @return boolean true ha helyes, egyébként false.
 */
protected function isValidIp($ip){
  if(count($tokens=split("\.",$ip))!=4){
    return false;
  }
  foreach ($tokens as $k=>$e){
    if(intval($e)>255 || intval($e)<0){
      return false;
    }
  }
  return true;
}
/**
 * Az adott VLAN-hoz tartozó bejegyzéseket adja vissza.
 * @param int $vlan A VLAN azonosító.
 * @return array A megtalált bejegyzések listája vagy NULL.
 */
public function getEntriesByVlan($vlan){
  if(!is_array($this->ipArpTable)){
    return NULL;
  }
  $result=NULL;
  foreach($this->ipArpTable as $k=>$entry){
    if($entry["vlan"]==$vlan){
      $result[$k]=$entry;
    }
  }
  return $result;
}
/**
 * Hozzáad egy bejegyzést az IP-ARP táblához
 * @param string $ip Az IP cím.
 * @param string $mac A MAC cím.
 * @param int $vlan A VLAN azonosító.
 * @return boolean A sikeresség.
 */
```

```
public function addEntry($ip,$mac,$vlan){
  if(!($this->isValidIP($ip)) || !($mac=CMacUtil::validMac($mac)) || !$vlan){
    return false;
  }
  if(!($entry=$this->getEntryByMac($mac)) || $entry["vlan"]!=$vlan){
    $this->ipArpTable[]=array("mac"=>$mac,"ip"=>$ip,"vlan"=>$vlan);
      return true;
  }
  // kivétel ??
  return false;
}
/**
 * Megkeres egy bejegyzést a megadott MAC cím alapján.
 * @param string $mac A keresett MAC cím.
 * @return array A megtalált bejegyzés vagy NULL.
 */
public function getEntryByMac ($mac){
  if(!is_array($this->ipArpTable)){
    return NULL;
  }
  foreach($this->ipArpTable as $k=>$e){
    if($e["mac"]==$mac){
      return $e;
    }
  }
  return NULL;
}
/**
 * IP cím alapján megkeres egy bejegyzést.
 * @param string $ip A keresett IP cím.
 * @return array A megtalált bejegyzés vagy NULL.
 */
public function getEntryByIp($ip){
  if(!is_array($this->ipArpTable)){
    return NULL;
  }
  foreach($this->ipArpTable as $k=>$e){
    if($e["ip"]==$ip){
      return $e;
    }
  }
  return NULL;
}
/**
 * Eltávolítja az adott MAC címre vonatkozó bejegyzést.
 * @param string $mac A keresett MAC cím.
 * @return array Az eltávolított bejegyzés vagy NULL.
 */
public function removeEntryByMac($mac){
  if(!is_array($this->ipArpTable)){
    return NULL;
  }
  foreach($this->ipArpTable as $k=>$e){
    if($e["mac"]==$mac){
      unset($this->ipArpTable[$k]);
        return $e;
    }
  }
  return NULL;
}
/**
 * Eltávolítja az adott IP-hez tartozó bejegyzést.
 * @param string $ip A keresett IP cím.
 * @return array Az eltávolított bejegyzés vagy NULL.
 */
public function removeEntryByIp($ip){
  if(!is_array($this->ipArpTable)){
    return NULL;
  }
  foreach($this->ipArpTable as $k=>$e){
    if($e["ip"]==$ip){
      unset($this->ipArpTable[$k]);
        return $e;
    }
  }
  return NULL;
}
/**
 * Eltávolítja az adott VLAN-hoz tartozó bejegyzédeket
 * @param int $vlan A vlan azonosítója.
 * @return array Az eltávolított bejegyzések listája vagy NULL.
 */
public function removeEntriesByVlan($vlan){
  if(!is_array($this->ipArpTable)){
    return NULL;
  }
  $result=NULL;
  foreach($this->ipArpTable as $k=>$e){
    if($e["vlan"]==$vlan){
```

```
        $result[$k]=$e;
        unset($this->ipArpTable[$k]);
      }
    }
    return $result;
  }
  /**
   * IP címet általános formátumra alakítja.
   * @param string $ip Az átalakítandó IP cím.
   * @return string A kész IP cím, vagy NULL.
   */
  public static function validIP($ip){
    if(!$ip){
      return NULL;
    }
    $minta="[\.|\ |:|\-]";
      $tokens=split($minta,trim($ip));
        if(count($tokens)!=4){
          return NULL;
        }
      $result=NULL;
        foreach($tokens as $k=>$e){
          $t=intval($e);
          if($t>255 || $t<0){
            return NULL;
          }
          $result.="$t.";
        }
    return rtrim($result,".");
  }
 }
```

A.3.4. CSwitchDbTree

```
 include_once($CLASS_DIR."db/CNetworkDataTools.php");
 /**
  * class CSwitchDbTree
  * Switchek fája, amely az adatbázisból kapott adatok alapján all össze.
  * @file CSwitchDbTree.php
  * @package default
  * @subpackage network
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.26.
  */
class CSwitchDbTree {
  /**
   * A switchek adatrekordjait tartalmazó tömb.
   * @var array
   */
   private  $tree=NULL;
  /**
   * Adatok kezeléséhez szükséges felület
   * @see CNetworkDataTool
   * @var CNetworkDataTools
   */
  private $dataTools=NULL;
  /**
   * A memedzsment vLAN azonosítója.
   * @var int
   */
  private $mgmtVlan=NULL;
  /**
   * Konstruktor
   * @param string $hostname Az adatbázis szerver hosztneve vagy IP címe.
   * @param string $dbname Az adatbázis neve.
   * @param string $user Az adatbázis felhasználói név.
   * @param string $passwd Az adatbázis jelszó.
   * @param int $mgmtVlan A menedzsment VLAN azonosítója.
   */
  public function __construct($hostname,$dbname,$user,$passwd,$mgmtVlan){
     $this->dataTools=new CNetworkDataTools(new CDBInterface
($hostname,$dbname,$user,$passwd));
      $this->mgmtVlan=$mgmtVlan;
  }
  /**
   * Felépíti az adott cisco eszköz adott portjához csatlakozó eszközök fáját.
   * @param int $cid A cisco switch adatbázis azonosítója.
   * @param int $cport A cisco switch megfelelő portja.
   * @return A sikeresség.
   */
  public function buildTree($cid,$cport){
    if(!$this->dataTools || !($r=$this->dataTools->getSwitchDataListForDisplay
($cid,$cport,$this->mgmtVlan))){
      return false;
```

```
    }
    foreach($r as $k=>$e){
      $this->tree[$e["id"]]=$e;
      $this->tree[$e["id"]]["trunks"]=array();
    }
    $nemVegtelen=count($this->tree);
    foreach($this->tree as $k=>$e){
      foreach($this->tree as $parentIndex => $parent){
        if($k!=$parentIndex && $e["melyeszkhez"]==$parentIndex){
          if(!$this->hasDoubleChild($parentIndex,$e["portra"])){
            $this->tree[$parentIndex]["trunks"][$e["portra"]]=$k;
          }
          else {
            $this->tree[$e]["portra"]=NULL;
            $this->tree[$e]["melyeszkhez"]=NULL;
          }
        }
      }
    }
    return true;
  }
  /**
   * Megadja, hogy egy adott switch adott portjára redundánsan két eszköz
   * csatlakozik-e.
   * @param int $parentId A szülő switch adatbázis indexe.
   * @param int $parentPort A szülő switch vizsgált portja.
   * @return boolean Ha duplán csatlakoznak eszközök akkor true, egyébként false.
   */
  private function hasDoubleChild($parentId,$parentPort){
    if(!is_array($this->tree)){
      return false;
    }
    $found=0;
    foreach($this->tree as $k=>$e){
      if($e["melyeszkhez"]==$parentId && $e["portra"]==$parentPort){
        ++$found;
      }
    }
    return $found>1?true:false;
  }
 /**
  *  Az eszközfa HTML kóddá alakítása
  * @return string Az eszköz fa HTML formátumban
  */
  public function toString(){
    $result="";
    foreach($this->tree as $k=>$e){
      if(!$e["melyeszkhez"]){
        $result.="<div id=\"kontener\">\n\t<ul id=\"fa\">\n";
          $result.=$this->eszkozToString($e)."\t</ul>\n</div>\n";
      }
    }
    return $result;
  }
 /**
  * Egy eszköz
  * @param array $e Az eszköz adatait tartalmazó tömb.
  * @param int $szint A fa szintjét jelöli. (A forrás behúzásához kell);
  * @param boolean $uccso Azt jelöli, hogy ez a fa adott ágán az utolsó elem.
  * @return string Az eszközt reprezentáló HTML kód
  */
  protected function eszkozToString(array & $e,$szint=0,$uccso=null){
    $tab=$this->tab($szint);
      $result="$tab<li id=\"eszkoz\"";
      if($e["melyeszkhez"]){
        if($uccso){
          $result.=" class=\"uccso\"";
        }
        else {
          $result.=" class=\"gyermek\"";
        }
      }
      $result.= ">\n";
      // eszköz sor
      $result.="$tab\t<div id=\"esor\">\n";
    if($e["melyeszkhez"]){
          // port szám
          $result.="$tab\t\t<div id=\"portnum\">".$e["portra"]." <img
src=\"balnyil.jpg\" alt=\"&lt;-\" > ".$e["uplink"].".</div>\n";
        }
      if(!empty($e["focus"])){
          $result.="<a name=\"focus\"></a>\n";
        }
      $result.="$tab\t\t<div id=\"sp\">\n$tab\t\t\t<div id=\"eszkadatok\" ";
        if($e["nemok"]){
          $result.="class=\"nemok\"";
        }
        if(!empty($e["focus"])){
          $result.="style=\"border:3px double #6099cc\" ";
```

```
        }
        $result.=">\n";
        //típus
          $result.="$tab\t\t\t\t<div id=\"tipus\">".$e["tipus"]."</div>\n";
        //uplink
          if(!$e["melyeszkhez"] && $e["uplink"]){
            $result.="$tab\t\t\t\t<div id=\"uplink\">uplink: ".$e["uplink"].
"</div>\n";
          }
        //mac
          $result.="$tab\t\t\t\t<div id=\"mac\">".$e["mac"]."</div>\n";
        //ip
          $result.="$tab\t\t\t\t<div id=\"ip\">".$e["ip"]."</div>\n";
        //cím
          $result.="$tab\t\t\t\t<div id=\"cim\">".$e["cim"]."</div>\n";
        //vlan
          $result.="$tab\t\t\t\t<div id=\"vlan\">".$e["vlanid"]." VLAN</div>\n";
      $result.="$tab\t\t\t</div>\n$tab\t\t</div>\n";

      $result.="$tab\t</div>\n";
      if(is_array($e["trunks"])){
        $result.="$tab\t<ul id=\"gyermekek\">\n";
        $u=count($e["trunks"]);
        for($i=1;$i<9 && $u;$i++){
          if(array_key_exists($i,$e["trunks"])){
            if(!(--$u)){
             $uccso=true;
            }
            $result.= $this->eszkozToString $this->tree[$e["trunks"][$i]],
$szint+2,$uccso);
          }
        }
        $result.="$tab\t</ul>\n";
      }
      $result.="$tab</li>\n";
    return $result;
  }
 /**
  *  Megadott darabszámú tabulátort gyárt a forráskód formázásához.
  * @param int $tabs A tabulátorok darabszáma.
  * @return string A tabulátorokat tartalmazó string.
  */
  function tab($tabs){
    $result="\t\t";
      for($i=0;$i<$tabs;$i++){
        $result.="\t";
      }
    return $result;
  }
 }
```

A.3.5. CSwitchFactory

```
 $PCKG="network/";
 include_once($CLASS_DIR.$PCKG."IHostFactory.php");
 include_once($CLASS_DIR.$PCKG."CHost.php");
 include_once($CLASS_DIR.$PCKG."active_elements/CSwitchCisco.php");
 include_once($CLASS_DIR.$PCKG."active_elements/CSwitchSwh.php");
 include_once($CLASS_DIR.$PCKG."active_elements/CSwitchZte.php");
 include_once($CLASS_DIR.$PCKG."active_elements/CSwitchHuawei.php");
 include_once($CLASS_DIR.$PCKG."active_elements/CMediaGateway.php");
 /**
  * class CSwitchFactory
  * Switcheket előállító osztály.
  * A konstruktorban vagy a setElementTypes mezódus segítségével fel kell
  * paraméterezni, hogy mely
  * MAC típushoz milyen osztály tartozik. Ez alapján példányosítja a megfelelő
  * osztályt.
  * <code>
  * $types=array(array("mac"=>"00:0e:8","className"=>"CSwitchCisco"),
array("mac"=>"00:1c","className"=>"CSwitchCisco"),
array("mac"=>"00:0f:e2","className"=>"CSwitchHuawei"));
  * $factory=new CSwitchFactory($switchTypes);
  * $switch=$factory->createHost($switchData);
  * </code>
  * @todo A legyártandó switcheknek megfelelő osztályokat mindig be kell
  * includolni!
  * @package default
  * @subpackage network
  * @file CSwitchFactory.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.10.
  */
 class CSwitchFactory implements IHostFactory {
```

```
  /**
   * A switch típusokra vonatkozó adatokat tartalmazó tömb.
   * A tömb index a MAC cím gyártóspecifikus része, az adatrész pedig az osztály
   * neve amit példányosítani kell.
   * <code>
   * array(array("mac"=>"00:10:1","className"=>"CswitchDummy"));
   * </code>
   * @var array
   */
   protected static $elementTypes=NULL;
   /**
    * A konstruktor.
    * @param array $elementTypes Az egyes switch típusokhoz tarozó bejegyzések
    * listája. {@link $elementTypes}
    */
  public function __construct($elementTypes=NULL){
    print_r($elementTypes);
    $this->setHostTypes($elementTypes);
  }
  /**
    * Legyárt egy switchet a MAC címe alapján
    * @param array $switchData A switch adatait tartalmazó tömb.
    * @return IHost A legyártott switch.
    * @throws
    */
  public static function createHost ($switchData){
    if(!is_array($switchData)||!is_array(CSwitchFactory::$elementTypes)){
      return NULL;
    }
    $result=NULL;
    if(!empty($switchData["mac"])){
      // kivétel keletkezhet!!
      if($type=CSwitchFactory::getHostClassByMac(CMacUtil::validMac
($switchData["mac"]))){
        $result=new $type["className"] ($switchData);
      }
      else {
        $result = new CHost($switchData);
      }
    }
    return $result;
  }
  /**
   * Hozzáférést biztosít a hoszt típusokat tartalmazó tömbhöz.
   * @return array A hoszt típusokat tartalmazó tömb.
   */
  public static function getHostTypes (){
    return CSwitchFactory::$elementTypes;
  }
  /**
   * Beállítja az eszköz típusokat.
   * @param array $elementTypes Az eszköz típusokra vonatkozó adatokat tartalmazó
   * tömb.
   */
  public static function setHostTypes ($elementTypes){
    if(is_array($elementTypes)){
      CSwitchFactory::$elementTypes=$elementTypes;
    }
  }
  /**
   * Megadja a MAC cím alapján a hosztot reprezentáló osztály nevét.
   * @param string $mac A hoszt MAC címe, ami alapján a típusa beazonosítható.
   * @return string A megfelelő osztály neve, vagy NULL.
   */
  public static function getHostClassByMac($mac){
    foreach(CSwitchFactory::$elementTypes as $k=>$e){
      $kp=substr_count($e["mac"],":");
        $digits=strlen($e["mac"])-$kp;
      if(CMacUtil::getVendor($mac,$digits)==$e["mac"]){
        return $e;
      }
    }
    return NULL;
  }
 }
```

A.3.6. CSwitchTree

```
require_once $CLASS_DIR.'network/INetworkTree.php';
require_once $CLASS_DIR.'util/CNetworkTreeIterator.php';
/**
 * Konstans ami a maximális az egy porton lévő maximális MAC cím darabszámot adja
 * meg.
 */
define ("MIN_MAC",15000);
/**
```

```
 * class CSwitchTree
 * Az adott hálózati szegmensben lévő switchek kapcsolódási fáját reprezentálja.
 * @file CSwitchTree.php
 * @package default
 * @subpackage network
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.04.14.
 * <code>
 *   $tree=new CSwitchTree ($hostDataList,$switchFactory,$routerMac);
 *     $tree->buildTree();
 *       $tree->toString();
 * </code>
 */
class  CSwitchTree implements INetworkTree {
  /**
   * A switcheket legyártó osztály példánya.
   * @var CSwitchFactory
   */
  protected $hostFactory;
  /**
   * A fa elemeit, részfáit tartalmazó lista
   * @var array
   */
  protected $tree=NULL;
  /**
   * Az adott hálózati szegmenset kiszolgáló router megfelelő VLAN-ban felvett MAC
   * címe.
   * @var string
   */
  protected $routerMac;
  /**
   * Enter description here...
   * @var unknown_type
   */
  private $macList=NULL;
  /**
   * A fa felépítéséhez szükséges segédváltozó.
   * @var int
   */
  private $minMac=MIN_MAC;
  /**
   * A fa felépítéséhez szükséges segédváltozó.
   * @var string
   */
  private $parentPort=0;
  /**
   * Konstruktor.
   * @param array $hostDataList A fába rendezendő eszközök adatainak, vagy
   * maguknak a példányoknak a listája.
   * @param CSwitchFactory $hostFactory A switcheket gyártó osztály egy példánya.
   * @param string $routerMac Az adott hálózati szegmensben lévő router megfelelő
   * VLAN-hoz tartozó MAC címe.
   */
  public function __construct($hostDataList,$hostFactory,$routerMac){
    $this->routerMac=CMacUtil::validMac($routerMac);
        $this->setHostFactory($hostFactory);
          $this->makeMacList($hostDataList);
            $this->createHosts($hostDataList);
    }
  /**
   * A gyártó osztály példányát beállító metódus.
   * @param CSwitchFactory $hostFactory
   */
  public function setHostFactory ($hostFactory){
    if($hostFactory instanceof CSwitchFactory ) {
      $this->hostFactory=$hostFactory;
    }
    else {
      $this->hostFactory=NULL;
    }
  }
  /**
   * Az adott hálózati szegmenset kiszolgáló router menedzsment VLAN-ban felvett
   * MAC címét beállító metódus.
   * @param string $mac A MAC cím.
   */
  public function setRouterMac ($mac) {
    $this->routerMac=CMacUtil::validMac($mac);
  }
  /**
   * @param array $hostDataList
   * @return array
   */
  private function makeMacList ($hostDataList) {
    if(!is_array($hostDataList)){
      return false;
    }
```

```
    foreach($hostDataList as $k=>$e){
      if(!empty($e["mac"])){
        $this->macList[]=CMacUtil::validMac($e["mac"]);
      }
    }
    return count($this->macList);
  }
  /**
   * Legyártja az részfában lévő eszközöket és hozzáfűzi őket a $tree listához.
   * Ha a listában host adatok vannak akkor a $hostFactory-t használja, ha pedig
   * kész IHost példányok akkor hozzáfűzi őket a listához.
   * @param array $hostDataList A hosztokat vagy adataikat tartalmazó tömb.
   * @return int A listába behelyezett eszközök számával tér vissza.
   */
  private function createHosts ($hostDataList){
    if(!is_array($hostDataList)){
      return 0;
    }
    $result=0;
    foreach ($hostDataList as $k=>$e){
      if($e instanceof IHost){
        $this->tree[$e->getMac()]=$e;
          ++$result;
      }
      else {
        if($h=$this->getFactory()->createHost($e)){
          if($h instanceof ASwitch ){
            try {
              $h->setUplinkPort($h->getPortByMac($this->routerMac));
            }
            catch (ENoSuchPortException $e) {
              unset($e);
            }
          }
          $this->tree[$h->getMac()]=$h;
          ++$result;
        }
      }
    }
    return $result;
  }
  /**
   * A gyártó osztályhoz nyújt hozzáférést.
   * @return CSwitchFactory A switcheket gyártó osztály vagy NULL;
   */
  public function getFactory() {
    return $this->hostFactory;
  }
  /**
   * A fa felépítése
   *
   */
  public function buildTree(){
    if(!$this->tree || !$this->macList){
      return false;
    }
    $this->macList[]=$this->routerMac;
    $nemVegtelen=count($this->tree);
    while(count($this->tree)>1 && $nemVegtelen){
      IHost & $host=array_shift($this->tree);
      $this->minMac=MIN_MAC;
      $this->parentPort=0;
      IHost & $tempParent=NULL;
      foreach($this->tree as $k=>$e){
        if($r=$this->searchParent($host->getMac(),$e)){
          $tempParent=$r;
        }
      }
      if($tempParent){
        try {
          if($this->parentPort!=$tempParent->getUplinkPort()){
            $tempParent->addHostToPort($host,$this->parentPort);
            $host->setParent($tempParent);
            $host->setParentPort($this->parentPort);
          }
          else {
            array_push($this->tree,$host);
          }
        }
        catch(Exception $e){
          print $e->getMessage();
          unset($e);
          $tempHost=$tempParent->getHostByPort($this->parentPort);
          if(($p=$tempHost->getPortByMac($host->getMac()) &&
$p!=$tempHost->getUplinkPort())){
            $tempParent->getArpTable()->addMacToPort(sprintf
("0000.0000.%04X",$nemVegtelen),$this->parentPort);
            $nemVegtelen++;
            array_unshift($host,$this->tree);
```

```
          }
          else {
            array_push($this->tree,$host);
          }
        }
      }
      else {
        array_push($this->tree,$host);
      }
      --$nemVegtelen;
    }
  }
  /**
   * Megkeresi az adott MAC címmel rendelkező hoszt szülő eszközét.
   * @param string $mac A az eszköz MAC címe aminek a szülő eszközét keressük
   * @param IHost $root A részfa gyökere, ahonnan a rekurzív keresés indul.
   * @return ASwitch A megtalált szülő eszköz vagy NULL.
   */
  public function searchParent ($mac,IHost & $root){
    $result=NULL;
    if($root instanceof ASwitch ){
      if(($p=$root->getPortByMac($mac))!=NULL &&
($mm=$root->getMacCountOnPort($p))<$this->minMac){
        $this->minMac=$mm;
        $this->parentPort=$p;
        $result=$root;
      }
      if($root->hasChild()){
        foreach($root->getInterfaces() as $port=>$host){
          if($host && $r=$this->searchParent($mac,$host)){
            $result=$r;
          }
        }
      }
    }
    return  $result;
  }
  /**
   * Megállapítja, a fa szerkezet helyességét.
   * @return boolean A fa helyessége.
   */
  public function isValidTree() {
    if(!is_array($this->tree) || count($this->tree)!=1) {
      return false;
    }
    return $this->checkValid($this->tree[0]);
  }
  /**
   * Segéd metódus az isValidTree() metódushoz.
   * Rekurzívan bejérja a fát és megvizsgálja, hogy minden switchnek meg van-e az
   * uplink portja, és a hosztoknak van-e szülőjük.
   * @param IHost $host A részfa gyökere.
   * @return boolean A helyesség.
   */
  private function checkValid($host){
    if($host instanceof ASwitch){
      if(!$host->getUplinkPort()){
        return false;
      }
      if($host->hasChild()){
        foreach ($host->getInterfaces() as $port=>$e){
          if($e && !$this->checkValid($e)){
            return false;
          }
        }
      }
    }
    else if(!($host instanceof IHost) || !$host->getParent()){
      return false;
    }
    return true;
  }
  /**
   * A részfákat tartalmazó tömbhöz ad hozzáférést.
   * @return array Az elemeket tartalmazó tömb.
   */
  public function getTree() {
    return $this->tree;
  }
  /**
   * MAC cím alapján kikeres egy eszközt a fából.
   * @param string $mac A keresett eszköz MAC címe.
   * @return IHost A megtalált eszköz vagy NULL;
   */
  public function getHostByMac($mac){
    $it=$this->iterator();
    while($it->hasNext()){
      $host=$it->next();
      if(CMacUtil::equals($host->getMac(),$mac)){
```

```
        unset($it);
        return $host;
      }
    }
    unset($it);
    return NULL;
  }
  /**
   * A fa bejárását teszi lehetővé.
   * @return CNetworkTreeIterator A fát bejáró iterátor.
   */
  public function iterator() {
    return new CNetworkTreeIterator($this);
  }
  /**
   * Az eszközök fájának HTML formátumú reprezentációja.
   * @return string A HTML forrás.
   */
  public function toString(){
    $result="<div id=\"switch_tree\">\n";
    if(is_array($this->getTree())){
      foreach($this->getTree() as $k=>$e){
        if($e instanceof IHost){
          $result.=$e->toString();
        }
      }
    }
    $result.="</div>\n";
    return $result;
  }
}
```

A.3.7. IHost

```
$PACKAGE=$CLASS_DIR."network/";
/**
 * interface IHoszt
 * A hálózatban lévő hosztot leíró interfész.
 * @package default
 * @subpackage network
 * @file IHost.php
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.03.19.
 */
 interface IHost {
  /**
   * Visszadaja a hoszt MAC címét.
   * @return string A hoszt MAC címe.
   */
 public function getMac();
 /**
  * Megadja a hoszt IP címét.
  * @return string A hoszt IP címe.
  */
 public function getIP();
 /**
  * Beállítja a hoszt IP címét
  * @param string $ip A beállítandó IP cím
  */
 public function setIP($ip);
 /**
  * Megadja a hoszt szülő eszközét
  * @return IHost
  */
 public function getParent();
 /**
  * Beállítja a hoszt szülőeszközét
  * @param IHost $parent
  */
 public function setParent(IHost $parent);
 /**
  * Megadja a hoszt szülőeszközének portját, amihez kapcsolódik
  * @return string
  */
 public function getParentPort();
 /**
  * Beállítja, hogy a hoszt a szülő ezköz melyik interfészére kapcsolódik.
  * @param string $port A szülő eszköz interfészének az azonosítója.
  */
 public function setParentPort($port);
 /**
  * A hoszt adatait tartalmazó szöveget adja vissza
  * @return string A hosztot reprezentáló string.
  */
```

```
  public function toString ();
}
```

A.3.8. IHostFactory

```
/**
 * interface HostFactory
 * A különböző hosztokat gyártó osztályok interfésze.
 * A konstruktorban paraméterként meg kell admia egy tömböt, amely a különböző
 * MAC típusokhoz tartozó osztályneveket tartalmazza.
 * Valamint az ismert osztályokat be kell includolni a Factory osztályba.
 * @package default
 * @subpackage network
 * @file IHostFactory.php
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.04.14.
 */
interface IHostFactory {
   /**
    * Előállít egy Host példányt az adatai alapján.
    * A pédánynak megfelelő osztályt a MAC címe alapján választja ki. Ha nem talál
    * megfelelőt akkor CHost példányt hoz létre.
    * @param array $hostData A host adatait tartalmazó tömb.
    * @return IHost A legyártott host vagy NULL;
    */
  public static function createHost ($hostData);
 /**
    * Hozzáférést nyújt a Host típusokat mac címmel azonosító adattaghoz.
    * @return array A host típusokat tartalmazó tömb.
    */
  public static function getHostTypes ();
  /**
   * Beállítja a Host típusokat MAC címmel azonosító adatokat tartalmazó
   * attributumot.
   * @param array $hostTypes A típusokat megadó tömb.
   */
  public static function setHostTypes ($hostTypes);
  /**
   * A MAC cím alapján megadja a megfelelő osztályt.
   * @param string $mac A MAC cím.s
   * @return string A megtalált osztály neve.
   */
  public static function getHostClassByMac ($mac);
}
```

A.3.9. InetworkTree

```
/**
 * interface INetworkTree
 * A hálózat Fa struktúráját és annak felépítését reprezentáló interfész.
 * @package default
 * @subpackage network
 * @file INetworkTree.php
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.04.14.
 */
interface INetworkTree {
 /**
  * Konstruktor
  * @param array $hostDataList Az adott csomóponthoz tartozó hosztok adat
  * rekordjainak listája.
  * @param IHostFactory $hostFactory A hosztokat legyártó osztály egy példánya.
  * @param string $routerMac Az adott hálózati csomópont átjárójának menedzsment
  * VLAN-ban felvett MAC címe.
  */
  public function __construct($hostDataList,$hostFactory,$routerMac);
  /**
   * Felépítia fát.
   */
  public function buildTree();
  /**
   * Beállítja a fában lévő eszközöket legyártó osztályt.
   * @param IHostFactory $hostFactory Egy eszköz gyártó osztály példány.
   */
  public function setHostFactory ($hostFactory);
  /**
   * A fa felépítéséhez szükséges router MAC címét állítja be.
   * @param string $mac A router interfészének MAC címe amelyhez az adott
```

```
     * eszközök tartoznak.
     */
    public function setRouterMac($mac);
    /**
     * Megállapítja, hogy minden eszköz része-e a fának.
     */
    public function isValidTree();
    /**
     * Megkeresi az adott MAC címmel rendelkező hoszt szülő eszközét.
     * @param string $mac A az eszköz MAC címe aminek a szülő eszközét keressük
     * @param IHost $root A részfa gyökere, ahonnan a rekurzív keresés indul. Ha
     * NULL akkor a fa gyökeréből indul a keresés.
     * @return IHost A megtalált szülő eszköz vagy NULL.
     */
    public function searchParent ($mac,IHost & $root);
    /**
     * Megkeres egy hosztot a fában a MAC címe alapján
     * @param string $mac A keresett hoszt MAC címe.
     * @return IHost A megtalált hoszt vagy NULL.
     */
    public function getHostByMac($mac);
    /**
     * Hozzáférést nyújt az hosztok fájához
     * @return array A hosztok fáját tartalmazó lista. Azért lista, mertha nem
     * egyértelmű a fa akkor több gyökér elem lehetséges.
     */
    public function getTree();
    /**
     * A hosztokat legyártó osztály példányhoz nyújt hozzáférést.
     * @return IHostFactory A gyártó osztály példány.
     */
    public function getFactory ();
   /**
     * A Fa struktúrát megjelenítő HTML formátumú string.
     */
    public function toString();
}
```

A.4. active_elements package

A.4.1. AActiveElements

```
$PACKAGE=$CLASS_DIR."network/";
$EXCEPTION_DIR=$CLASS_DIR."exception/";
include_once($PACKAGE."IHost.php");
include_once($EXCEPTION_DIR."EHostDoesNotExistException.php");
/**
 * abstract class AActiveElement
 * Az aktív eszközök ős absztrakt osztálya. Ebből kell származtatni minden
 * switchet, kodest, egyebet
 * @file AActiveElements
 * @package network
 * @subpackage active_elements
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.03.19.
 */
abstract class AActiveElement implements IHost {
 /**
  * Az eszköz IP címe.
  * @var string
  */
  private $ip=NULL;
 /**
  * Az eszköz MAC címe.
  * @var string
  */
  private $mac="00:00:00:00:00:00";
 /**
  * A szülő eszköz, amihez az eszköz kapcsolódik.
  * @var IHost
  */
  private $parent=NULL;
 /**
  * A szülő eszköz portja, amuhez az eszköz kapcsolódik.
  * @var int
  */
  private $parentPort=0;
  public function getMac() {
   return $this->mac;
 }
```

```
  /**
   * Beállítja a MAC címet.
   * @param string $mac A beállítandó MAC cím.
   * @throws EInvalidMacException Ha hibás MAC címet tartalmaz a $mac paraméter.
   */
  protected function setMac($mac){
    if($m=CMacUtil::validMac($mac)) {
      $this->mac=$m;
    }
    else {
      throw new EInvalidMacException("Invalid MAC address: $mac",1);
    }
  }
  public function getIP() {
    return $this->ip;
  }
  /**
   * Beállítja az eszköz IP címét.
   * @param string $ip A beállítandó IP cím.
   * @throws EInvalidIPException Ha nem helyes IP cím van a $ip paraméterben.
   */
  public function setIP ($ip) {
    if(is_string($ip)){
      $minta="/^((([0-9]{1,3}.){3}[0-9]{1,3}){1}$/";
      if(preg_match($minta,$ip)){
        $this->ip=$ip;
      }
      else {
        throw new EInvalidIPException("$ip IP address is invalid",$code);
      }
    }
  }
  /**
   * Megadja az aktív eszköz szülőjét.
   * @return IHost A szülő eszköz;
   */
  public function getParent () {
    return $this->parent;
  }
  public function getParentPort(){
    return $this->parentPort;
  }
  /**
   * Beállítja az Aktív eszköz szülőjét.
   * @param IHost $parent A szülő eszköz referenciája.
   */
  public function setParent (IHost $parent){
    if(!is_a($parent,"IHost")){
     ;  }
    else {
      $this->parent=$parent;
    }
  }
  /**
   * Beállítja, hogy az aktív eszköz a szülő ezköz mely interfészéhez kapcsolódik.
   * @param int $port A szülő eszköz interfész azonosítója.
   */
  public function setParentPort($port){
    if(!$this->getParent()){
      throw new EHostDoesNotExistException("This element has not got any parent
element",$code);
    }
    $this->parentPort=$port;
  }
  /**
   * Megadja, hogy az aktív eszközre van e kapcsolva gyermek eszköz.
   * @return boolean
   */
  public abstract function hasChild();
  /**
   * Az aktív eszköz adatait tartalmazó stringet adja meg HTML formátumban.
   * @return $string Az aktív eszközt reprezentáló HTML formátumú string.
   */
  public function toString () {
    $result="<div id=\"active_element\">\n";
    $result.="</div>\n";
  }
}
```

A.4.2. ASwitch

```
$PACKAGE=$CLASS_DIR."network/active_elements/";
$EXCEPTION_DIR=$CLASS_DIR."exception/";
include_once($PACKAGE."AActiveElement.php");
include_once($PACKAGE."CArpTable.php");
include_once($CLASS_DIR."util/CSNMPInterface.php");
```

```
 include_once($CLASS_DIR."util/CTelnetInterface.php");
 include_once($EXCEPTION_DIR."ENoSuchPortException.php");
 include_once($EXCEPTION_DIR."EDoubleMacException.php");
 include_once($EXCEPTION_DIR."EInvalidPortException.php");
 include_once($EXCEPTION_DIR."EInvalidIPException.php");
 include_once($EXCEPTION_DIR."ENoSuchVlanException.php");
 include_once($EXCEPTION_DIR."EInvalidParameterException.php");
 /**
  * abstarct class ASwitch
  * A switchek absztrakt ősosztálya. Minden switchet ebből az osztályból kell
  * származtatni.
  * @package network
  * @subpackage active_elements
  * @file ASwitch.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.03.21.
  */
  abstract class ASwitch extends AActiveElement implements IHost {
  /**
   * A switch ARP táblája.
   * @var CArpTable
   */
   private $arpTable=NULL;
   /**
    * Az eszköz portjai. A tömb indexei az interfész azonosítók.
    * A tömb elemei pedig a portokra csatlakoztatott {@link IHost} példányok.
    * @var array
    */
  private $interfaces=NULL;
  /**
   * A trunk portok listája.
   * @var array
   */
  private $trunkPorts=array();
  /**
   * A switchen keresendő MAC címek listája.
   * @var array
   */
  private $macList=NULL;
  /**
   * A switch adatbázis kulcsa.
   * @var int
   */
  private $id=NULL;
  /**
   * A switch típusa
   * @var string
   */
  private $type=NULL;
  /**
   * Az SNMP kezelést megvalósító osztály példánya.
   * @var CSNMPInterface
   */
  private $snmpInterface=NULL;
  /**
   * Telnet hozzáférést biztosító osztály egy példánya.
   * @var unknown_type
   */
  private $telnetInterface=NULL;
  /**
   * Annak a portnak az azonosítója amellyel a szülő eszközhöz csatlakozik.
   * @var string
   */
  private $uplinkPort=NULL;
  /**
   * Konstruktor
   * @param array $switchData A switch adatai tartalmazó tömb.
   * @throws EInvalidParameterException Ha a kötelező paraméterek nincsenek
   * rendben.
   * @todo A példányosításkor figyelni kell a paraméter helyességre ({@param
   * $switchData},MAC cím), különben kivétel {@link EInvalidParameterException}
   * váltódik ki amit le kell kezelni!
   */
  public function __construct($switchData){
    // paraméter ellenőrzés
    if(!is_array($switchData)){
      throw new EInvalidParameterException("The switchData parameter is not an
array",1);
    }
    // MAC cím ellenőrzés
    if($mac=CMacUtil::validMac($switchData["mac"])){
      $this->setMac($mac);
    }
    else {
      throw new EInvalidParameterException("Invalid MAC address in switchData
parameter",1);
    }
```

```
    // IP cím ellenőrzés
    if(array_key_exists("ip",$switchData)){
      $this->setIP($switchData["ip"]);
    }
    // TELNET
      $this->telnetInterface=new CTelnetInterface($this->getIP());
        if($switchData["telnetLogin"]){
          $this->telnetLogin=$switchData["telnetLogin"];
        }
        if($switchData["telnetPassword"]){
          $this->telnetPassword=$switchData["telnetPassword"];
        }
    //   SNMP
      //print_r($switchData);
      if(!empty($switchData["snmpParams"]) &&
is_array($switchData["snmpParams"])){
        $sp=&$switchData["snmpParams"];
          //print_r($sp);
          $this->setSnmpInterface(new
CSNMPInterface($sp["roCommunity"],$sp["rwCommunity"],$sp["version"]));
      }
      else if(!empty($switchData["snmpParams"]) && $switchData["snmpParams"]
instanceof CSNMPInterface ){
        $this->setSnmpInterface($switchData["snmpParams"]);
      }
    // portok
      if(array_key_exists("interfaces",$switchData) &&
is_array($switchData["interfaces"])){
        // paraméter
        $this->setInterfaces(& $switchData["interfaces"]);
      }
      else if ($this->getSnmpInterface()){
        // SNMP

        $this->setInterfaces($this->readInterfaces());
      }
      else if(array_key_exists("portCount",$switchData)){
        // port szám
        for($i=1;$i<=$switchData["portCount"];$i++){
          $this->interfaces[$i]=NULL;
        }
      }
      else {
        $this->setInterfaces(array(1));
        // egyik sem
      }
    // eszköz típus
      $this->type=$this->readType();
    // ARP tábla
      $this->arpTable=new CArpTable($this->interfaces);
    // A keresett MAC címek
      if(array_key_exists("macList",$switchData)){
        $this->setMacList($switchData["macList"]);
      }
    // Adatbázis kulcs
      if(array_key_exists("id",$switchData)){
        $this->setId($switchData["id"]);
      }
    // TRUNK portok listája
      if(array_key_exists("trunkPorts",$switchData)){
      $tp=&$switchData["trunkPorts"];
        if(is_array($tp)){
          $this->trunkPorts=$tp;
        }
    }
  }
  /**
   * A switch portjait állítja be.
   * @param array $interfaces
   */
  public function setInterfaces($interfaces) {
    if(!$interfaces && !is_array($interfaces)){
      return NULL;
    }
    $this->interfaces=array();
      foreach($interfaces as $k=>$e){
        $this->interfaces[$e]=NULL;
      }
  }
  /**
   * hozzáférést biztosít a switch portjainak listájához.
   * @return array A portok listája.
   */
  public function getInterfaces(){
    return $this->interfaces;
  }
  /**
   * Hozzáférést biztosít az ARP táblához
   * @return CArpTable A switch ARP táblájának referenciája
```

```
   */
  public function getArpTable () {
     return $this->arpTable;
  }
  /**
   * Megkeres egy MAC címet az ARP táblában.
   * @param string $mac A keresett MAC cím.
   * @return string Annak az interfésznek az azonosítója, amelyen a MAC cím
   * jelentkezik vagy NULL.
   */
  public function getPortByMac ($mac) {
    return $this->arpTable->getPortByMac($mac);
  }
  /**
   * Magadja, hogy hány darab MAC cím van az adott interfészen.
   * @param int $port Az interfész azonosítója.
   * @return int A MAC címek darabszáma.
   * @throws ENoSuchPortException Ha nem található a keresett port az ARP
   * táblában.
   * @see CArpTable::getMacCountOnPort
   */
  public function getMacCountOnPort ($port) {
    return $this->arpTable->getMacCountOnPort($port);
  }
  /**
   * Visszadja az adott interfészen lévő hosztot.
   * @param string $port Az interfész azonosítója.
   * @return IHost Az interfészre kapcsolt hoszt, vagy NULL.
   * @throws ENoSuchPortException Kivételt dobja, ha nem létezik a $port
   * azonosítójú interfész.
   */
  public function getHostByPort ($port) {
    if(!array_key_exists($port,$this->interfaces)){
      throw new ENoSuchPortException("The $port interface doesn't exist",$code);
    }
    return $this->interfaces[$port];
  }
  /**
   * A switch adott portjához rendel egy hosztot.
   * Ha $checkArp értéke true, akkor egyeztet, hogy az ARP táblában a $port-nak
   * megfelelő interfészhez van e rendelve a hoszt MAC címe.
   * Ha a $checkArp értéke false akkor úgy is hozzáadható a hosz az adott porthoz,
   * hogy az nem szerepel, vagy másik porthoz van rendelve az ARP táblában.
   * @param IHost $host A hoszt.
   * @param string $port Az interfész amire a hoszt kapcsolódik.
   * @param boolean $checkArp Ha TRUE akkor a metódus ellenőrzi a hosztot az ARP
   * táblában.
   * @throws ENoSuchPortException Akkor váltódik ki, ha nemlétező interfészre
   * hivatkozik a $port paraméter.
   * @throws EDoubleMacException Akkor váltódik  ki, ha a $host már hozzá lett
   * rendelve valamelyik interfészhez.
   * @throws EInvalidPortException Akkor váltódik ki, ha a hoszt MAC címe az ARP
   * táblában nem ahhoz az interfészhez van rendelve, amelyre a $port paraméter
   * mutat.
   *
   */
  public function addHostToPort(IHost $host,$port,$checkArp=true){
    if($h=$this->getHostByPort($port)){ // ha már van a porton eszköz
      throw new EDoubleMacException("There is a host with ".$h->getMac()." MAC
address on $port",$code);
    }
    else if ($checkArp){
      if(($p=$this->arpTable->getPortByMac($host->getMac()))){
        if($p!=$port){ // nem ezen a porton van a MAC
          throw new EInvalidPortException($host->getMac()." MAC address is not on
$port. It's on the $p port.",$code);
        }
      }
       else { // a MAC nincs benne az ARP táblábans
         throw new EInvalidPortException($host->getMac()." MAC address is not in
the ARP table",$code);
       }
    }
    $this->interfaces[$port]=$host;
      $host->setParent($this);
        $host->setParentPort($port);
  }
  /**
   * Eltávolít egy hosztott az adott interfészről.
   * @param string $port Az interfész azonosítója.
   * @return IHost Az eltávolított hoszt.
   * @throws ENoSuchPortException Akkor váltódik ki, ha nemlétező interfészre
   * hivatkozik a $port paraméter.
   */
    public function removeHostFromPort($port){
    if($temp=$this->getHostByPort($port)){
      $this->interfaces[$port]=NULL;
        return $temp;
    }
```

```
}
/**
 * Visszadja a trunk portok listáját.
 * @return array A trunk portok listája.
 */
public function getTrunkPorts(){
  return $this->trunkPorts;
}
/**
 * A switchen keresett MAC címek listájához nyújt hozzáférést.
 * @return array A MAC címek listája.
 */
public function getMacList(){
  return $this->macList;
}
/**
 * Bellítja az eszközön keresett MAC címek listáját.
 * @param array & $macList
 */
protected function setMacList(array & $macList){
  $this->macList=$macList;
}
/**
 * Visszaadja az adott porthoz bejegyzett MAC címek listáját.
 * @param string $port A port azonosítója.
 * @return array A MAC címek listája vagy NULL.
 * @throws ENoSuchPortException Kivétel váltódik ki, ha nemlétező interfészre
 * hivatkoztunk.
 */
public function getMacListOnPort ($port){
  if(!$at=$this->getArpTable()){
    return NULL;
  }
  return $at->getMacListByPort($port);
}
/**
 * Beállítja az SNMP interfészt.
 * @param CSNMPInterface $snmpInterface Az SNMP kapcsolat példánya.
 */
public function setSnmpInterface(CSNMPInterface $snmpInterface){
  $this->snmpInterface=$snmpInterface;
}
/**
 * Visszadaja az SNMP interfészt.
 * @return CSNMPInterface Az SNMP interfész példány.
 */
public function getSnmpInterface() {
  return $this->snmpInterface;
}
/**
 * Hozzáférést ad a TELNET interfészhez
 * @return CTelnetInterface A telnet interfész.
 */
protected function & getTelnetInterface() {
  return $this->telnetInterface;
}
  /**
 * A telnet interfészben tárolt login nevet adja vissza.
 * @return string Telnet login név.
 */
 protected function getTelnetLogin(){
  return $this->telnetLogin;
}
/**
 * A telnet interfészben tárolt jelszót adja vissza.
 * @return string Telnet jelszó.
 */
protected function getTelnetPassword(){
  return $this->telnetPassword;
}
/**
 * Az ARP táblát tölti fel MAC címekkel.
 * @param array $vlans A vlan azonosítókat tartalmazó tömb.
 */
protected function loadArp($vlans){
  if(!is_array($vlans)){
    // nem tömb
    return NULL;
  }
  foreach($vlans as $k=>$e){
    $l=$this->readMacListByVlan($e);
      if(is_array($l)){
        //print_r($l);
        foreach($l as $mac=>$port){
          try {
            $this->getArpTable()->addMacToPort($mac,$port);
          }
          catch (ENoSuchPortException $e){
          // Nincs ilyen port
```

```
            unset($e);
          }
          catch (EDoubleMacException $e){
          // kétszer szerepel a MAC
            unset($e);
          }
        }
      }
    }
  }
  /**
   * Kiolvassa az eszközből a trunk portok listáját.
   * @return array A trunk portok listája.
   */
  protected abstract function readTrunkPorts();
  /**
   * Kiolvassa az eszközből, hogy az adott MAC cím melyik porton jelentkezik.
   * @param string $mac A keresett MAC cím.
   * @return int Az interfész száma, vagy 0.
   */
  protected abstract function readPortByMac($mac);
  /**
   * Kiolvassa a teljes ARP táblát.
   */
  protected abstract function readArpTable();
  /**
   * Az adott VLAN-ból jövő MAC címeket olvassa ki az eszközből.
   * @param int $vlan A VLAN azonosítója.
   * @return array A MAC címek listáját tartalmazó tömb. A tömb indexei a MAC
   * címek,az értékek, pedig az interfészek, amelyen a MAC cím jelentkezik.
   * @throws ENoSuchVlanException Ha nincs az eszközön az adott VLAN.
   */
  protected abstract function readMacListByVlan ($vlan);
  /**
   * Kiolvassa az switch fizikai interfészeit SNMP segítségével.
   * @return array Az interfészek neveit tartalmazó tömb.
   */
  protected abstract function readInterfaces();
  /**
   * SNMP segítségével kiolvassa a switch típusát.
   */
  protected abstract function readType();
  /**
   * Megadja, hogy a switch rendelkezik e legalább egy gyermek eszközzel.
   * @return boolean Ha van alárendelve hoszt akkor true, egyébként false.
   */
  public function hasChild() {
    if(!is_array($this->interfaces)){
      return false;
    }
    foreach($this->interfaces as $k => $e){
      if(is_a($e,"IHost")||is_a($e,"ASwitch")){
        return true;
      }
    }
    return false;
  }
  /**
   * Beállítja, hogy a switch melyik portjával kapcsolódik a szülő eszközhöz.
   * @param string $port A fizikai interfész azonosítója.
   * @throws ENoSuchPortException
   */
  public function setUplinkPort($port){
    if(!is_array($this->interfaces) ||
!array_key_exists($port,$this->interfaces)){
      // kivétel
      throw new ENoSuchPortException("Switch [".$this->getMac()."] has not
interface $port to set uplink port.",1);
    }
    $this->uplinkPort=$port;
  }
   /**
    * Visszaadja hogy a switch melyik portján keresztül kapcsolódik a szülő
   * eszközhöz.
    *
    * @return string A port azonosítója.
    */
  public function getUplinkPort() {
     return $this->uplinkPort;
   }
   /**
    * A switch adatait reprezentáló szöveget adja vissza HTML formátumban.
    * Amennyiben van switchhez kapcsolt eszköz, annak az adatait is hozzáfűzi.
    * @return string Az adatokat tartalmazó  HTML formátumú string.
    */
    public function toString () {
      $result="<div id=\"switch\">\n";
        $result.="<div id=\"data\">\n";
          $result.="<div>type:".$this->getType()."</div>\n";
```

```
          $result.="<div>ip:".$this->getIP()."</div>\n";
          $result.="<div>mac:".$this->getMac()."</div>\n";
          $result.="<div>uplink:".$this->getUplinkPort()."</div>\n";
        $result.="</div>\n";
        if($this->hasChild()){
          $result.="<ul>\n";
            foreach($this->getInterfaces() as $k=>$e){
              if($e instanceof IHost){
                $result.="<li>$k port \n";
                $result.=$e->toString();
                $result.="</li>\n";
              }
            }
          $result.="</ul>\n";
        }
      $result.="</div>\n";
      return $result;
    }
    /**
     * A switch típus leírásához nyújt hozzáférést.
     * @return string A switch típus leírását tartalmazó string.
     */
    public function getType(){
      return $this->type;
    }
    /**
    * Lekérdezi a switch adatbázis kulcsát.
     * @return int A switch adatbázis kulcsa.
     */
    public function getId(){
      return $this->id;
    }
    /**
     * Beállítja a switch adatbázis kulcsát.
     * @param int $id A switch adatbázis kulcsa.
     */
    public function setId($id){
      $this->id=$id;
    }
  }
```

A.4.3. CSwitchCisco

```
 include_once ($CLASS_DIR.$PACKAGE."ASwitch.php");
 /**
  * class CSwitchCisco
  * A Cisco switchek osztálya
  * @package network
  * @subpackage active_elements
  * @file CSwitchCisco.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.03.28.
  */
 class CSwitchCisco extends ASwitch {
   /**
    * Konstruktor
    * Ha a paraméter tömb tartalmaz VLAN információt akkor már konstruktor időben
    * feltölti a MAC address táblát.
    * @param array $switchData A switch adatait tartalmazó asszociatív tömb.
    */
   public function __construct($switchData){
     parent::__construct($switchData);
       if(array_key_exists("vlans",$switchData)){
         $vl=is_array($switchData["vlans"])?
$switchData["vlans"]:array($switchData["vlans"]);
          $this->loadArp($vl);
       }
   }
  /**
   * Az adott VLAN-ból jövő MAC címeket olvassa ki az eszközből.
   * @param int $vlan A VLAN azonosítója.
   * @return array A MAC címek listáját tartalmazó tömb. A tömb indexei a MAC
   * címek,az értékek, pedig az interfészek, amelyen a MAC cím jelentkezik.
   * @throws ENoSuchVlanException Ha nincs az eszközön az adott VLAN.
   */
  protected function readMacListByVlan ($vlan){
    $result=NULL;
    if(!($ti=$this->getTelnetInterface())){
      return NULL;
    }
    try{
      $ti->connect();
        $ti->setBufferSize(96000);
          $ti->setWindow(80,30000);
```

```
            $ti->readToPrompt("Username: ");
               $ti->send($this->getTelnetLogin()."\n");
                   $ti->readToPrompt("Password: ");
                     $ti->send($this->getTelnetPassword()."\n");
                       $ti->readToPrompt("#");
                         $ti->send("show mac-address-table vlan $vlan\n");
                           $enters=4;
                             $list="";
                               $minta="/More-- $/";
                               while($enters){
                                 $ti->readToPrompt("#",$minta)
                                if(preg_match("/(#)$/",($list.=$ti->toString()))){
                                       break;
                                    }
                                    $ti->send(" ");
                                }
                           $ti->send(" ");
                               $ti->send("exit\n");
    }
    catch (ETelnetConnectException $e){
        $ti->disconnect();
      return NULL;
    }
    catch (EBufferOverflowException $e){
        $ti->disconnect();
      return NULL;
    }
    $ti->disconnect();
        $ti->clearBuffer();
    return  $this->textToMacListByVlan($list,$vlan);
  }
  /**
   * A telneten keresztül kapott szöveges fájlt, ami tartalmazza a MAC listát,
   * tömbbé alakítja.
   * @param string $str A MAC listát tartalmazó string.
   * @param int $vlan A vlan azonosító.
   * @return array A MAC címeket és hozzájuk tartozó portokat tartalmazó tömb.
   */
  protected function textToMacListByVlan(& $str,$vlan){
    if(!$str || !$vlan){
      return NULL;
    }
    $result=array();
      $rows=split("\n|\r\n",$str);
        foreach($rows as $k=>$e){
          $row=trim($e);
            $minta="/$vlan(\t| )+([0-9a-f]{4}.[0-9a-f]{4}.[0-9a-f]{4})
.+(Fa0\/[0-9]+|Gi0\/[0-9]+)/";
            if(preg_match($minta,$row,$tokens)){
              if($mac=CMacUtil::validMac($tokens[2])){
                $result[$mac]=$tokens[3];
              }
            }
        }
    return $result;
  }
  /**
   * Kiolvassa az eszközből a trunk portok listáját.
   * @return array A trunk portok listája.
   */
  protected function readTrunkPorts(){
    return NULL;
  }
  /**
   * Kiolvassa az eszközből, hogy az adott MAC cím melyik porton jelentkezik.
   * @param string $mac A keresett MAC cím.
   * @return int Az interfész száma, vagy 0.
   */
  protected function readPortByMac($mac){
    $result=NULL;
    if(!($ti=$this->getTelnetInterface())||
!($mac=$this->macToCiscoFormat($mac))){
      return NULL;
    }
    try{
      $ti->connect();
        $ti->setBufferSize(48000);
          $ti->setWindow(80,100);
            $ti->readToPrompt("Username: ");
              $ti->send($this->getTelnetLogin()."\n");
                $ti->readToPrompt("Password: ");
                  $ti->send($this->getTelnetPassword()."\n");
                    $ti->readToPrompt("#");
                      $ti->send("show mac-address-table address $mac\n");
                        $ti->readToPrompt("#");
                          $list=$ti->toString();
                            $ti->send("exit\n");
    }
    catch (ETelnetConnectException $e){
```

```
        $ti->disconnect();
      return NULL;
    }
    catch (EBufferOverflowException $e){
        $ti->disconnect();
      return NULL;
    }
    $ti->disconnect();
      $str=$ti->toString();
        $ti->clearBuffer();
    return $this->textToPort($str);
  }
  protected function textToPort (& $str){
    if(!$str){
      return NULL;
    }
    $rows=split("\n|\r\n",$str);
      foreach($rows as $k=>$e){
          $row=trim($e);
            $tokens=split("\t|\ ",$row);
              if(!empty($tokens[13]) && (substr($tokens[13],0,2)=="Gi" ||
substr($tokens[13],0,2)=="Fa")){
                return $tokens[13];
              }
        }
    return NULL;
  }
  /**
   * Kiolvassa a teljes ARP táblát.
   */
  protected function readArpTable(){
    return NULL;
  }
  /**
   * A fizikai interfészek listáját olvassa ki az eszközből SNMP segítségével.
   * @return array Az interfészek listáját tartalmazó tömb.
   */
  protected function readInterfaces(){
    if(!$this->getSnmpInterface()){
      return NULL;
    }
    $mib=".iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr";
      $si=$this->getSnmpInterface();
      try {
        $il=$si->walk($this->getIp(),$mib);
      }
      catch (ESNMPException $e){
        print $e->getMessage();
        unset($e);
        return NULL;
      }
      $result=NULL;
      for ($i=0;$i<count($il);$i++){
        $e=intval(substr($e,8));
        if(substr($il[$i],8,5)=="FastE"){
          $result[]="Fa".substr($il[$i],20);
        }
        else if (substr($il[$i],8,8)=="GigabitE"){
          $result[]="Gi".substr($il[$i],23);
        }
      }
      return $result;
  }
  /**
   * Az adott vlan-hoz tartozó MAC címek listáját adja vissza.
   * @param int $vlan A vlan azonosító
   * @return array A mac címeket tartalmazó lista.
   */
  public function getMacListByVlan($vlan){
    return $this->readMacListByVlan($vlan);
  }
  /**
   * MAC címet a cisco eszközöknek megfelelő formátumúra alakít (hhhh.hhhh.hhhh).
   * @param string $mac Az átalakítandó MAC cím.
   * @return string A cisco eszközöknek megfelelő formátumú MAC cím, vagy NULL.
   */
   public function macToCiscoFormat($mac){
     if(!($mac=CMacUtil::validMac($mac))){
       return NULL;
     }
     $tokens=split(":",$mac);
       $result="";
       for($i=0;$i<count($tokens);$i++){
         $result.=$i%2?$tokens[$i].":":$tokens[$i];
       }
     return rtrim($result,":");
   }
   /**
    * SNMP segítségével kiolvassa az eszköz típusát.
```

```
    * @return string A Cisco switch típus leírása.
    */
   protected function readType(){
      if(!($si=$this->getSnmpInterface())){
        return NULL;
      }
      $mib="SNMPv2-SMI::mib-2.47.1.1.1.1.13";
        try {
          $descr=$si->walk($this->getIP(),$mib);
        }
        catch (ESNMPException $e){
          print $e->getMessage();
          unset($e);
          return NULL;
        }
          $descr=substr($descr[0],8);
            $tokens=split(" ",$descr);
      return "Cisco ".trim($tokens[0],"\"");
  }
   /**
    * A cisco adott VLAN interfészéhez tartozó MAC címet adja meg.
    * @param int $vlan A VLAN azonosítója.
    * @return string A VLAN-hoz tartozó MAC cím.
    */
  public function getOwnMacByVlan($vlan){
    if(!($si=$this->getSnmpInterface()) || !$vlan){
      return NULL;
    }
    $mib="IF-MIB::ifPhysAddress.$vlan";
      try{
        $result=$si->get($this->getIP(),$mib);
      }
      catch(ESNMPException $e){
        unset($e);
        return NULL;
      }
      if($result){
        return CMacUtil::validMac(substr($result,8));
      }
      return NULL;
  }
   /**
    * Az adott porton jelentkező MAC címek listáját adja meg.
    * @param string $port A port azonosítója.
    * @return array A MAC címek listáját tartalmazó tömb.
    */
  public function getMacListOnPort ($port){
    if(is_integer($port)){
      $i=1;
      foreach($this->getInterfaces() as $k=>$e){
        if($i==$port){
          $port=$k;
          break;
        }
        ++$i;
      }
    }
    if(!$at=$this->getArpTable()){
      return NULL;
    }
    return $at->getMacListByPort($port);
  }
}
```

A.4.4. CSwitchHuawei

```
 $PACKAGE=$CLASS_DIR."network/active_elements/";
 $EXCEPTION_DIR=$CLASS_DIR."exception/";
 include_once($PACKAGE."ASwitch.php");
 /**
  * class CSwitchHuawei
  * A Huawei switchet reprezentáló osztály.
  * @package network
  * @subpackage active_elements
  * @file CSwitchHuawei.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.03.24.
  */
class  CSwitchHuawei extends ASwitch {
   /**
    * Konstruktor
    * Ha a paraméter tömb tartalmaz VLAN információt akkor már konstruktor időben
   * feltölti a MAC address táblát.
    * @param array $switchData A switch adatait tartalmazó tömb.
```

```
    */
  public function __construct(array & $switchData){
    parent::__construct($switchData);
      if(array_key_exists("vlans",$switchData)){
         $vl=is_array($switchData["vlans"])? switchData["vlans"]:
array($switchData["vlans"]);
          $this->loadArp($vl);
       }
  }
  /**
   * A trönk portokat olvassa ki a switchből
   *
   * @return array A TRUNK portok listája.
   */
  protected function readTrunkPorts(){
    return NULL;
  }
  /**
   * SNMP-én keresztül kiolvassa a switchből, hogy az adott MAC cím melyik porton
   * jelentkezik.
   * @param string $mac A keresett MAC cím.
   * @return int A megtalált port azonosítója.
   */
  protected function readPortByMac($mac){
    if(!($si=$this->getSnmpInterface())){
      return NULL;
    }
    if(!is_array($vlans=$this->readVlans())){
      return NULL;
    }
      foreach ($vlans as $k=>$e){
        $mib="enterprises.2011.2.23.1.3.2.1.2.$e.".$this->macToSnmpFormat($mac);
        try {
          if($result=$si->get($this->getIP(),$mib)){
            $tokens=split(":",$result);
              if(count($tokens)==2){
                return trim($tokens[1]);
              }
          }
        }
        catch (ESNMPException $e){
          unset($e);
        }
      }
    return NULL;
  }
  /**
   * Beolvassa a teljes ARP táblát.
   */
  protected function readArpTable(){
    ;
  }

  /**
   * Az eszközből kiolvassa az adott VLAN-ban jelentkező MAC címeket.
   * Telnet interfész segítségével történik a kiolvasás.
   * @param int $vlan A VLAN azonosítója
   * @return array A mac címek és portok listája.
   */
  protected function readMacListByVlan($vlan){
    $result=NULL;
    if(!($ti=$this->getTelnetInterface())){
      return NULL;
    }
    try{
      $ti->connect();
        $ti->setBufferSize(48000);
          $ti->setWindow(80,10000);
              $ti->readToPrompt("Password:");
                $ti->send($this->getTelnetPassword()."\n");
                  $ti->readToPrompt("<Quidway>");
                    $ti->send("super\n");
                      $ti->readToPrompt("Password:");
                        $ti->send($this->getTelnetPassword()."\n");
                          $ti->readToPrompt("<Quidway>");
                            $ti->send("display mac-address vlan $vlan\n");
                              $minta="/(<Quidway>|More ----)$/";
                                $enters=4;
                                  $list="";
                                while($enters){
                                  $ti->readToPrompt("<Quidway>",$minta);
                                     if(preg_match("/<Quidway>$/",
($list.=$ti->toString()))){
                                       break;
                                     }
                                  $ti->send(" ");
                                }
                              $ti->send("quit\n");
    }
```

```
    catch (ETelnetConnectException $e){
        $ti->disconnect();
      return NULL;
    }
    catch (EBufferOverflowException $e){
        $ti->disconnect();
      return NULL;
    }
    $ti->disconnect();
        $ti->clearBuffer();
      return $this->textToMacListByVlan($list,$vlan);
  }
  /**
   * Kigyűjti az eszköz portjainak listáját SNMP segítségével.
   * @return array A fizikai interfészek listája.
   */
  protected function readInterfaces(){
    if(!$this->getSnmpInterface()){
      return NULL;
    }
    $mib="IF-MIB::ifType";
      $si=$this->getSnmpInterface();
      try {
        $il=$si->walk($this->getIP(),$mib);
      }
      catch (ESNMPException $e){
        unset($e);
        return NULL;
      }
      $result=NULL;
      $ifNUmber=0;
      foreach ($il as $k => $e){
        $tokens=split(":",$e);
        if(count($tokens)==2 && trim($tokens[1])=="ethernetCsmacd(6)"){
          $result[]=(++$ifNUmber);
        }
      }
      return $result;
  }
  /**
   * A Telneten keresztül kiolvasott listából készíti el az adott VLAN-hoz tartozó
   * MAC címek listáját .
   * @param string $str A listát tartalmazó string.
   * @param int $vlan A keresett VLAN azonosítója.
   * @return array A MAC címek és portok listája.
   */
   protected function textToMacListByVlan(& $str,$vlan){
    if(!$str || !$vlan){
      return NULL;
    }
    $result=array();
      $rows=split("\n|\r\n",$str);
        foreach($rows as $k=>$e){
          $row=trim($e);
            $tokens=split("\t|\ ",$row);
              $minta="/(([0-9a-f]{4})-){2}[0-9a-f]{4}\ +($vlan)/";
              if(preg_match($minta,$row,$tokens)){
                if(($mac=CMacUtil::validMac(substr($tokens[0],0,14))) &&
preg_match("/Ethernet0\/[0-9]+/",$row,$ptokens)){
                  $result[$mac]=substr($ptokens[0],10);
                }
              }
        }
    return $result;
  }
  /**
   * A switchből kiolvassa a létrehozott VLAN-okat SNMP segítségével.
   * @return array A VLAN-ok listája.
   */
  public function readVlans(){
    if(!$this->getSnmpInterface()){
      return NULL;
    }
    $mib="SNMPv2-SMI::enterprises.2011.2.23.1.2.1.1.1.1";
    $si=$this->getSnmpInterface();
      try {
        $vl=$si->walk($this->getIP(),$mib);
      }
      catch (ESNMPException $e){
        unset($e);
        return NULL;
      }
      if(!is_array($vl)){
        return NULL;
      }
        $result=NULL;
          foreach ($vl as $k=>$e){
            if(count($r=split(":",$e))==2){
              $result[]=intval(trim($r[1]));
```

```
            }
          }
      return $result;
    }
    /**
     * A MAC címet az SNMP kérésekhez szükséges formátumra konvertála
     * (255.255.255.255.255.255)
     * @param string $mac A konvertálandó MAC cím.
     * @return string Az SNMP kérésnek megfelelő MAC cím.
     */
    public function macToSnmpFormat($mac){
      if(!($mac=CMacUtil::validMac($mac))){
         return NULL;
       }
       $tokens=split(":",$mac);
       $result=NULL;
         foreach ($tokens as $k=>$e){
           $result.=hexdec($e).".";
         }
       return rtrim($result,".");
    }
    /**
     * A MAC címet a Huaweinek megfelelő formátumúra konvertálja a telnet
     * kérésekhez.
     * (HH-HH-HH)
     * @param string $mac A konvertálandó MAC cím.
     * @return string A Huawei switch telnethez megfelelő formátumú MAC cím.
     */
     public function macToHuaweiFormat($mac){
       if(!($mac=CMacUtil::validMac($mac))){
         return NULL;
       }
       $tokens=split(":",$mac);
         $result="";
         for($i=0;$i<count($tokens);$i++){
           $result.=$i%2?$tokens[$i]."-":$tokens[$i];
         }
       return rtrim($result,"-");
     }
     /**
      * Visszadja Annak a portnak az azonosítóját, amelyen az adott MAC cím
     * szerepel.
      * @param string $mac A keresett MAC cím.
      * @return int A port azonosítója.
      */
     public function getPortByMac($mac){
       if(!$this->getArpTable() ||
!($result=$this->getArpTable()->getPortByMac($mac))){
         return $this->readPortByMac($mac);
       }
       return $result;
     }
      /**
       * A switch típusát olvassa ki SNMP segítségével.
       *
       * @return string A típus információt tartalmazó string.
       */
     protected function readType(){
        if(!($si=$this->getSnmpInterface())){
          return NULL;
        }
        $mib="SNMPv2-MIB::sysDescr.0";
          try {
            $descr=$si->get($this->getIP(),$mib);
          }
          catch (ESNMPException $e){
            unset($e);
            return NULL;
          }
              $result="Huawei";
              $minta="/Quidway ([\-A-Za-z0-9])+/";
                if(preg_match($minta,$descr,$type)){
                  $result.=" ".substr($type[0],8);
                }
       return $result;
    }
}
```

A.4.5. CSwitchL3Cisco

```
$PACKAGE="network/active_elements/";
include_once ($CLASS_DIR.$PACKAGE."CSwitchCisco.php");
include_once ($CLASS_DIR."network/CIPArpTable.php");
/**
 * class CSwitchL3Cisco
 * Cisco Layer 3-as switchet megvalósító osztály
```

```
 * @package network
 * @subpackage active_elements
 * @file CSwitchL3Cisco.php
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.04.07.
 */
class CSwitchL3Cisco extends CSwitchCisco {
  /**
   * Az IP-ARP tábla
   * @var CIPArpTable
   */
  private $ipArpTable=NULL;
  /**
   * Konstruktor.
   * @param array $switchData A routerre vonatkozó adatok tömbje.
   * @see CSwitchCisco::__construct($switchData);
   */
  public function __construct($switchData){
    CSwitchCisco::__construct($switchData);
    $this->loadIPArp(false);
  }
  /**
   * Feltölti az IP-ARP táblát a routerből SNMP-én keresztül.
   * @return boolean A sikeresség.
   */
  protected function loadIPArp($telnet=false){
    if(!$this->getSnmpInterface() && !$telnet){
      return false;
    }
    if(!$this->ipArpTable){
      $this->ipArpTable=new CIPArpTable();
    }
    $result=false;
    if(!$telnet){
      $mib=".iso.org.dod.internet.mgmt.mib-2.at.atTable.atEntry.atIfIndex";
      try {
            if(!is_array($vlans=$this->getSnmpInterface()->walk($this->getIP()
,$mib))){
        return false;
       }
       $mib=".iso.org.dod.internet.mgmt.mib-2.at.atTable.atEntry.atPhysAddress";
       if(!is_array($macs=$this->getSnmpInterface()->walk($this->getIP(),$mib))){
         return false;
       }
       $mib=".iso.org.dod.internet.mgmt.mib-2.at.atTable.atEntry.atNetAddress";
       if(!is_array($ips=$this->getSnmpInterface()->walk($this->getIP(),$mib))){
         return false;
       }
      }
      catch (ESNMPException $e){
        unset($e);
        return false;
      }
      for($i=0;$i<count($vlans);$i++){
     if($this->ipArpTable->addEntry(CSwitchL3Cisco::netAddressToIP
(substr($ips[$i],17)),trim(substr($macs[$i],12,17)),substr($vlans[$i],9))){
          $result=true;
        }
      }
    }
    else {
      $result=NULL;
      if(!($ti=$this->getTelnetInterface())){
        return false;
      }
      try{
        $ti->connect();
        $ti->setBufferSize(48000);
        $ti->setWindow(80,100);
        $ti->readToPrompt("Username: ");
        $ti->send($this->getTelnetLogin()."\n");
        $ti->readToPrompt("Password: ");
        $ti->send($this->getTelnetPassword()."\n");
        $ti->readToPrompt("#");
        $ti->send("show ip arp\n");
        $enters=4;
          $list="";
            $minta="/More-- $/";
              while($enters){
                $ti->readToPrompt("#",$minta);
                  if(preg_match("/(#)$/",($list.=$ti->toString()))){
                    break;
                  }
                $ti->send(" ");
              }
        $ti->send(" ");
        $ti->send("exit\n");
```

```
      }
      catch (ETelnetConnectException $e){
        $ti->disconnect();
        return NULL;
      }
      catch (EBufferOverflowException $e){
        $ti->disconnect();
        return NULL;
      }
      $ti->disconnect();
      $ti->clearBuffer();
      $result=$this->textToEntries($list);
    }
    return $result;
  }
  /**
   * A telneten keresztül beolvasott IP-ARP bejegyzéseket szúrja be az IP-ARP
   * táblába.
   * @param string $str A telneten keresztüli beolvasás eredmény szövege.
   * @return boolean A sikeresség.
   */
  private function textToEntries ($str) {
    if(!$str){
      return false;
    }
    $rows=split("/\r\n|\n/",$str);
    $result=false;
      if(is_array($rows)){
        foreach($rows as $k=>$e){
          if(preg_match_all("/Internet.*\n/",$e,$matches)){
            foreach($matches[0] as $m=>$entry){
              $found=NULL;
                $ip=$mac=$vlan=NULL;
              if(!preg_match("/[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}/",
$entry,$found)){
                continue;
              }
              else {
                $ip=$found[0];
              }
              if(!preg_match("/[0-9a-f]{4}.[0-9a-f]{4}.[0-9a-f]{4}/",
$entry,$found)){
                continue;
              }
              else {
                $mac=$found[0];
              }
              if(!preg_match("/Vlan[0-9]{1,}/",$entry,$found)){
                continue;
              }
              else {
                $vlan=substr($found[0],4);
              }
              if($this->ipArpTable->addEntry(trim($ip),trim($mac),trim($vlan))){
                $result=true;
              }
            }
          }
        }
      }
    return $result;
  }
  /**
   * Az SNMP olvasás során kapott hexa IP címformátumot átalakítja normál
   * decimális IP formátumra
   * @param string $addr A hexa formátumú IP cím.
   * @return string A decimális formátumú IP cím.
   */
  protected static function netAddressToIP($addr){
    if(!$addr){
      return NULL;
    }
    $tokens=split(":",$addr);
    $result=NULL;
    foreach($tokens as $k=>$e){
      $result.=hexdec($e).".";
    }
    return rtrim($result,".");
  }
  /**
   * A MAC cím alapján kikeres egy IP címet az IP-ARP táblából
   * @param string $mac A keresett IP-hez tartozó MAC cím.
   * @return string A megtalált IP vagy NULL.
   */
  public function getIpByMac($mac){
    if(!$this->ipArpTable){
      return NULL;
    }
    return $this->ipArpTable->getIpByMac($mac);
```

```
  }
  /**
   * Kikeresi az adott IP címhez bejegyzett MAC címet az IP-ARP táblából.
   * @param string $ip Az IP cím amihez keressük a MAC címet.
   * @return string A megtalált MAC cím vagy NULL.
   */
  public function getMacByIp($ip){
    if(!$this->ipArpTable){
      return NULL;
    }
    return $this->ipArpTable->getMacByIp($ip);
  }
  /**
   * Az adott VLAN-hoz tartozó IP-ARP bejegyzések listáját adja meg
   * @param int $vlan A VLAN azonosító.
   * @return array A bejegyzések listája vagy NULL.
   */
  public function getIpEntriesByVlan($vlan){
    if(!$this->ipArpTable){
      return NULL;
    }
    return $this->ipArpTable->getEntriesByVlan($vlan);
  }
  /**
   * Az IP-ARP táblából eltávolít egy bejegyzést a MAC cím alapján.
   * @param string $mac A keresett MAC cím.
   * @return array A bejegyzést tartalmazó asszociatív tömb.
   */
  public function removeIpEntryByMac($mac){
    if(!$this->ipArpTable){
      return NULL;
    }
    return $this->ipArpTable->removeEntryByMac($mac);
  }
  /**
   * Megkeres egy IP-ARP tábla bejegyzést a megadott MAC cím alapján.
   * @param string $mac A keresett MAC cím.
   * @return arrayA bejegyzést tartalmazó asszociatív tömb.
   */
  public function getIPEntryByMac($mac){
    return $this->ipArpTable->getEntryByMac($mac);
  }
  /**
   * Beolvassa az adott VLAN-hoz tartozó IP-ARP bejegyzéseket.
   * @deprecated Eltávolítva! Mindig false-t ad!
   * @param int $vlan A VLAN azonosítója.
   * @return boolean a sikeresség.
   */
  public function readIPArpByVlan($vlan){
    if(!$this->getTelnetInterface()){
      return false;
    }
    return false;
  }
}
```

A.4.6. CSwitchSwh

```
 $PACKAGE="network/active_elements/";
 include_once ($CLASS_DIR.$PACKAGE."ASwitch.php");
 /**
  * class CSwitchSwh
  * AZ SWH gyártmányú switcheket, illetve médiakonvertereket reprezentáló osztály.
  * @package network
  * @subpackage active_elements
  * @file CSwitchSwh.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.08.
  */
 class CSwitchSwh extends ASwitch {
   /**
    * Konstruktor
    * Ha a paraméter tömb tartalmaz VLAN információt akkor már konstruktor időben
   * feltölti a MAC address táblát.
    * @param array $switchData A switch adatait tartalmazó asszociatív tömb.
    */
    public function __construct(array & $switchData){
    parent::__construct($switchData);
      if(array_key_exists("vlans",$switchData)){
         $vl=is_array($switchData["vlans"])?
$switchData["vlans"]:array($switchData["vlans"]);
           $this->loadArp($vl);
       }
  }
```

```
  /**
   * Kiolvassa az adott VLAN-ra beállított portok listáját.
   * @param int $vlan A VLAN azonosítója.
   * @return array Az eredményt tartalmazó tömb.
   */
  protected function readPortsByVlan($vlan){
     if(!$this->getSnmpInterface()){
      return NULL;
    }
    $mib="SNMPv2-SMI::enterprises.9304.100.2109.4.4.2.2.2.1.2";
      $si=$this->getSnmpInterface();
      try {
        $il=$si->walk($this->getIP(),$mib);
      }
      catch (ESNMPException $e){
        unset($e);
        return NULL;
      }
      $result=NULL;
        if(is_array($il)){
          foreach($il as $k=>$e){
            $vl=split(":",$e);
            if($vlan==trim($vl[1])){
              $result[]=$k+1;
            }
          }
        }
    return $result;
   }
/* === implementált metódusok === */
   /**
   * Kiolvassa az eszközből a trunk portok listáját.
   * @return array A trunk portok listája.
   */
  protected function readTrunkPorts(){
  }
  /**
   * Kiolvassa az eszközből, hogy az adott MAC cím melyik porton jelentkezik.
   * @param string $mac A keresett MAC cím.
   * @return int Az interfész száma, vagy 0.
   */
  protected function readPortByMac($mac){
    if(!$this->getArpTable()->isEmpty()){
      return $this->getArpTable()->getPortByMac($mac);
    }
  }
  /**
   * Kiolvassa a teljes ARP táblát.
   */
  protected function readArpTable(){
  }
  /**
   * Az adott VLAN-ból jövő MAC címeket olvassa ki az eszközből.
   * @param int $vlan A VLAN azonosítója.
   * @return array A MAC címek listáját tartalmazó tömb. A tömb indexei a MAC
   * címek,az értékek, pedig az interfészek, amelyen a MAC cím jelentkezik.
   * @throws ENoSuchVlanException Ha nincs az eszközön az adott VLAN.
   */
  protected function readMacListByVlan ($vlan){
    if(!($pl=$this->readPortsByVlan($vlan))){
      return NULL;
    }
    // MAC címek
    $mib1="SNMPv2-SMI::enterprises.9304.100.2109.5.7.3.1.2";
    // portok
    $mib2="SNMPv2-SMI::enterprises.9304.100.2109.5.7.3.1.3";
      $si=$this->getSnmpInterface();
      try {
        $ml=$si->walk($this->getIP(),$mib1);
        $il=$si->walk($this->getIP(),$mib2);
      }
      catch (ESNMPException $e){
        unset($e);
        return NULL;
      }
        if(!is_array($ml) || !is_array($il)){
          return NULL;
        }
          $result=NULL;
          foreach($il as $k=>$e){
            if(in_array(($port=substr($e,9)+1),$pl)){
                $mac=trim(substr(($ml[$k]),12));
                $result[CMacUtil::validMac($mac)]=$port;
              }
            }
    return $result;
  }
  /**
   * Kiolvassa az switch fizikai interfészeit SNMP segítségével.
```

```
   * @return array Az interfészek neveit tartalmazó tömb.
   */
  protected function readInterfaces(){
    if(!$this->getSnmpInterface()){
      return NULL;
    }
    $mib="IF-MIB::ifIndex";
      $si=$this->getSnmpInterface();
      try {
        $il=$si->walk($this->getIP(),$mib);
      }
      catch (ESNMPException $e){
        unset($e);
        return NULL;
      }
      foreach ($il as $k => $e){
        $tokens=split(":",$e);
          $result[]=trim($tokens[1]);
      }
      return $result;
  }
   /**
    * SNMP segítségével kiolvassa az eszköz típusát.
    * @return string A switch típus leírása.
    */
  protected function readType(){
      if(!($si=$this->getSnmpInterface())){
        return NULL;
      }
      $mib="SNMPv2-MIB::sysDescr.0";
        try {
          $descr=$si->get($this->getIP(),$mib);
        }
        catch (ESNMPException $e){
          unset($e);
          return NULL;
        }
          $descr=substr($descr,8);
            $tokens=split(" ",$descr);
      return trim($tokens[0]);
  }
 }
```

A.4.7. CSwitchZte

```
 $PACKAGE="network/active_elements/";
 include_once ($CLASS_DIR.$PACKAGE."ASwitch.php");
 /**
  * class CSwitchZte
  * A ZTE gyártmányú switcheket reprezentáló ostály.
  * @package network
  * @subpackage active_elements
  * @file CSwitchZte.php
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.08.
  */
class CSwitchZte extends ASwitch{
   /**
    * Konstruktor
    * Ha a paraméter tömb tartalmaz VLAN információt akkor már konstruktor időben
   * feltölti a MAC address táblát.
    * @param array $switchData A switch adatait tartalmazó asszociatív tömb.
    */
   public function __construct(array & $switchData){
    parent::__construct($switchData);
      if(array_key_exists("vlans",$switchData)){
         $vl=is_array($switchData["vlans"])?
$switchData["vlans"]:array($switchData["vlans"]);
          $this->loadArp($vl);
       }
  }
 /**
   * A Telneten keresztül kiolvasott listából készíti el az adott VLAN-hoz tartozó
   * MAC címek listáját .
   * @param string $str A listát tartalmazó string.
   * @param int $vlan A keresett VLAN azonosítója.
   * @return array A MAC címek és portok listája.
   */
   protected function textToMacListByVlan(& $str,$vlan){
    if(!$str || !$vlan){
      return NULL;
    }
    $result=NULL;
      $rows=split("\n|\r\n",$str);
```

```
        foreach($rows as $k=>$e){
          $row=trim($e);
            if(preg_match("/dynamic$/",$row)){
              $m=substr($row,0,17);
              if(($mac=CMacUtil::validMac($m))){
                $minta="/$m\ +$vlan\ +[1-9]{1}/";
                  if(preg_match($minta,$row,$tokens)){
                    $result[$mac]=$port=$tokens[0][strlen($tokens[0])-1];
                  }
              }

            }
        }
    return $result;
  }
/* === implementált metódusok === */
   /**
    * Kiolvassa az eszközből a trunk portok listáját.
    * @return array A trunk portok listája.
    */
  protected function readTrunkPorts(){
  }
  /**
   * Kiolvassa az eszközből, hogy az adott MAC cím melyik porton jelentkezik.
   * @param string $mac A keresett MAC cím.
   * @return int Az interfész száma, vagy 0.
   */
  protected function readPortByMac($mac){
  }
  /**
   * Kiolvassa a teljes ARP táblát.
   */
  protected function readArpTable(){
  }
  /**
   * Az adott VLAN-ból jövő MAC címeket olvassa ki az eszközből.
   * @param int $vlan A VLAN azonosítója.
   * @return array A MAC címek listáját tartalmazó tömb. A tömb indexei a MAC
   * címek,az értékek, pedig az interfészek, amelyen a MAC cím jelentkezik.
   * @throws ENoSuchVlanException Ha nincs az eszközön az adott VLAN.
   */
  protected function readMacListByVlan ($vlan){
    $result=NULL;
    if(!($ti=$this->getTelnetInterface())){
      return NULL;
    }
    try{
      $enters=3;
      $minta="/(zte\>|break -----)$/";
      $ti->connect();
        $ti->setBufferSize(48000);
            $ti->readToPrompt("login:");
              $ti->send($this->getTelnetLogin()."\n");
                $ti->readToPrompt("password:");
                  $ti->send($this->getTelnetPassword()."\n");
                    $ti->readToPrompt("zte>");
                      $ti->send("show fdb vlan $vlan\n");
                        while($enters){
                          $ti->readToPrompt("zte>",$minta);
                            if(preg_match("/zte>$/",($list.=$ti->toString()))){
                              break;
                            }
                            $ti->send(" ");
                        }
                          $ti->send("exit\n");
    }
    catch (ETelnetConnectException $e){
        $ti->disconnect();
      return NULL;
    }
    catch (EBufferOverflowException $e){
        $ti->disconnect();
      return NULL;
    }
    $ti->disconnect();
      $ti->clearBuffer();
    return $this->textToMacListByVlan($list,$vlan);
  }
  /**
   * Kiolvassa az switch fizikai interfészeit SNMP segítségével.
   * @return array Az interfészek neveit tartalmazó tömb.
   */
  protected function readInterfaces(){
    if(!$this->getSnmpInterface()){
      return NULL;
    }
    $mib="IF-MIB::ifIndex";
      $si=$this->getSnmpInterface();
      try {
```

```
        $il=$si->walk($this->getIP(),$mib);
      }
      catch (ESNMPException $e){
        unset($e);
        return NULL;
      }
      foreach ($il as $k => $e){
        $tokens=split(":",$e);
          $result[]=trim($tokens[1]);
      }
      return $result;
  }
  /**
    * SNMP segítségével kiolvassa az eszköz típusát.
    * @return string A switch típus leírása.
    */
   protected function readType(){
      if(!($si=$this->getSnmpInterface())){
        return NULL;
      }
      $mib="SNMPv2-SMI::enterprises.3902.15.2.2.1.6.0";
        try {
          $descr=$si->get($this->getIP(),$mib);
        }
        catch (ESNMPException $e){
          unset($e);
          return NULL;
        }
      return "Zte ".trim(substr($descr,8),"\"");
  }
}
```

A.5. util package

A.5.1. CMacUtil

```
/**
 * class CMacUtil
 * A MAC cím formázását, formátum ellenőrzését végző segéd osztály.
 * @file CMacUtil.php
 * @package default
 * @subpackage util
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.03.20.
 */
class CMacUtil {
  /**
   * Több féle MAC formátumot alakít át 00:00:00:00:00:00 formátumra.
   * @param string $mac
   * @return string A formázott MAC cím.
   */
 public static function validMac($mac){
   // típosok: 00.00.00.00.00.00,0000.0000.0000,00:00:00:00:00:00,0:0:0:0:0:0,
   // 00-00-00-00-00-00,0000-0000-0000,
   $result=NULL;
   $mac=strtolower($mac);
   $minta="[\.|:| |-]";
   $tokens=split($minta,$mac);
   if (count ($tokens) == 3){
     $minta="/^[0-9a-f]{4}$/";
     foreach($tokens as $k=>$e){
       if(!preg_match($minta,$e)){
         return NULL;
       }
       $result.=substr($e,0,2).":".substr($e,2,2).":";
     }
   }
   else if(count($tokens)==6){
     $minta="/^[0-9a-f]{1,2}$/";
     foreach($tokens as $k=>$e){
       if(!preg_match($minta,$e)){
         return NULL;
       }
       $result.=sprintf("%02s",$e).":";
     }
   }
   return rtrim($result,":");
 }
 /**
  * A MAC cím gyártóspecifikus részét adja vissza
```

```
   * @param string $mac
   * @param int $digits Megadja, hogy hány hexa jegyet vegyen figyelembe.
   * @return string A MAC cím gyártóspecifikus részét tartalmazó string.
   */
  public static function getVendor($mac,$digits=6){
    $vmac="";
    if(!($vmac=CMacUtil::validMac($mac))){
      return NULL;
    }
      $len=$digits+($digits/2-0.5);
    return substr($vmac,0,$len);
  }
  /**
   * A MAC címet az adatbázisban tárolt string formátumra konvertálja.
   * @param string $mac A konvertálandó MAC cím.
   * @return string Az adatbázisnak megfelelő formátumú MAC cím.
   */
  public static function MacToDbFormat ($mac) {
    if(!($mac=CMacUtil::validMac($mac))){
      return NULL;
    }
    $tokens=split(":",$mac);
    $result="";
      foreach($tokens as $k=>$e){
        $result.=dechex(hexdec($e)).":";
      }
      return rtrim($result,":");
  }
  /**
   * Összehasonlít két MAC címet.
   * Az összehasonlításhoz különböző formátumú MAC címeket is megadhatunk
   * paraméterként.
   * @param string $mac1 Az egyik összehasonlítandó MAC cím.
   * @param string $mac2 A másik összehasonlítandó MAC cím.
   * @return boolean Az egyezőség vizsgálat eredménye.
   */
  public static function equals($mac1,$mac2){
    if(($mac1=CMacUtil::validMac($mac1)) && ($mac2=CMacUtil::validMac($mac2))){
      if($mac1==$mac2){
        return true;
      }
    }
    return false;
  }
}
```

A.5.2. CNetworkTreeIterator

```
 include_once $CLASS_DIR."network/INetworkTree.php";
 /**
  * class CNetworkTreeIterator
  * Egy hálózati fát bejáró iterátor. Felsorolja a fában lévő összes elemet.
  * @file CNetworkTreeIterator
  * @package default
  * @subpackage util
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.04.17.
  */
 class CNetworkTreeIterator{
  /**
   * A fa eszközeit tartalmazó sor.
   * @var array
   */
   private $data=NULL;
  /**
   * A rekurzív bejárást megvalósító verem.
   * @var array
   */
   private $stack=NULL;
  /**
   * Konstruktor
   * @param INetworkTree $tree A hálózati fa amit be kell járni.
   */
   public function __construct (& $tree) {
     if($tree instanceof INetworkTree){
        $this->data=$tree->getTree();
    }
     $this->stack=array();
       array_push($this->stack,array_shift($this->data));
   }
   /**
    * Megadja, hogy vannak-e még elemek a fában.
    * @return boolean Ha van még elem akkor true;
    */
```

```
   public function hasNext() {
    return count($this->stack)?true:false;
  }
  /**
   * A hálózati fa következő elemét adja meg.
   * @return IHost A hálózati fa egy eleme.
   */
  public function next() {
    $result=array_shift($this->stack);
      if($result instanceof ASwitch){
        if($result->hasChild()){
          foreach($result->getInterfaces() as $k=>$e){
            if($e instanceof IHost) {
              array_push($this->stack,$e);
            }
          }
        }
      }
      else if (!count($this->interfaces && count($this->data))) {
        array_push($this->stack,array_shift($this->data));
      }
    return $result;
  }
 }
```

A.5.3. CTelnetInterface

```
 $EXCEPTION_DIR=$CLASS_DIR."exception/";
 include_once ($EXCEPTION_DIR."ETelnetConnectException.php");
 include_once ($EXCEPTION_DIR."EBufferOverflowException.php");
 /**
  * @global Előre definiált konstansok a telnet kommunikációhoz.
  */
 /**
 * End of subnegotiation parameters.
 */
define ("SE",chr(0xf0));
/**
 * No operation.
 */
define ("NOP",chr(0xf1));
/**
 *   The data stream portion of a Synch. This should always be accompanied by a
 * TCP Urgent notification.
 */
define ("DATA_MARK",chr(0xf2));
/**
 * NVT character BRK.
 */
define ("BRK",chr(0xf3));
/**
 * Interrupt Process.
 */
define ("IP",chr(0xf4));
/**
 * Abort output
 */
define ("AO",chr(0xf5));
/**
 * Are You There ?
 */
define ("AYT",chr(0xf6));
/**
 * Erase character
 */
define ("EC",chr(0xf7));
/**
 * Erase Line
 */
define ("EL",chr(0xf8));
/**
 * Go Ahead signal.
 */
define ("GA",chr(0xf9));
/**
 * Indicates that what follows is subnegotiation of the indicated option.
 */
define ("SB",chr(0xfa));
/**
 * WILL. Indicates the desire to begin performing, or confirmation that you are
 * now performing, the indicated option.
 */
define ("WILL",chr(0xfb));
/**
 * WILL NOT. Indicates the refusal to perform, or continue performing, the
 * indicated option.
```

```
 */
define ("WONT",chr(0xfc));
/**
 * DO. Indicates the request that the other party perform, or confirmation that
 * you are expecting the other party to perform, the indicated option.
 */
define ("DOO",chr(0xfd));
/**
 * DON'T. Indicates the demand that the other party stop performing, or
 * confirmation that you are no longer expecting the other party to perform, the
 * indicated option.
 */
define ("DONT",chr(0xfe));
/**
 * Data Byte 255.
 */
define ("IAC",chr(0xff));
// egyéb parancsok
define ("NAWS",chr(0x1f));
 /**
  * class CTelnetInterface
  * Telnet kapcsolat létesítéséhez nyújt felületet
  * @file CTelnetInterface.php
  * @package default
  * @subpackage util
  * @author Lefi
  * @version 1.0
  * @copyright Copyright &copy; 2008, Lefi
  * Created on 2008.03.25.
  */
 class  CTelnetInterface {
  private $buffer=NULL;
  private $bufferSize=1024;
  private $prompt="";
  private $timeOut=NULL;
  private $hostname=NULL;
  private $port=23;
  private $socket=NULL;
  private $errornum=NULL;
  private $errormsg="";
  private $login=NULL;
  private $pswd=NULL;
  /**
   * Konstruktor
   * @param string $hostname Az elérni kívánt hosztnév vagy IP cím.
   * @param string $login A telnet login név.
   * @param string $pswd A telnet jelszó.
   * @param string $prompt A prompt amit kapnunk kell.
   * @param int $timeOut Időtúllépés
   */
  public function __construct($hostname,$login=NULL,$pswd=NULL,$prompt="",
$timeOut=10){
    $this->hostname=$hostname;
      $this->prompt=$prompt;
        $this->timeOut=$timeOut;
          $this->login=$login;
            $this->pswd=$pswd;
  }
  /**
   * Beállítja a promptot
   * @param string $prompt A prompt.
   */
  public function setPrompt($prompt){
    $this->prompt=$prompt;
  }
  /**
   * Megnyitja a Telnet kapcsolatot.
   * @return boolean True, ha ok.
   * @throws ETelnetConnectException Kivételt dob, ha hiba történt.
   */
  public function connect(){
    if(!$this->isConnected()){
      if(!$this->socket=fsockopen($this->hostname,$this->port,$this->errornum,
$this->errormsg,$this->timeOut)){
        throw new ETelnetConnectException($this->errormsg,$this->errornum);
      }
    }
    return true;
  }
  /**
   * Lezárja a Telnet kapcsolatot.
   * @return boolean A sikeresség.
   */
  public function disconnect() {
    if($this->socket){
      if(!fclose($this->socket)){
        $this->errormsg="Telnet connenction error. Can't close connection to
$this->hostname host";
          $this->errornum=1;
```

```
          throw new ETelnetConnectException($this->errormsg,$this->errornum);
    }
    $this->socket=NULL;
      $this->clearError();
        return true;
  }
  $this->errorstr="";
    $this->errornum=1;
  return false;
}
/**
 * A megadott promptig olvas
 * @param string $prompt A prompt amíg az olvasás folyik
 * @param string $regexp Reguláris kifejezés ami promptként illeszkedik a
 * beolvasott szövegre.
 * @return string A beolvasott szöveg.
 * @throws ETelnetConnectException Ha kapcsolati hiba van.
 * @throws EBufferOverflowException Ha buffer méret túllépés van.
 */
public function readToPrompt($prompt=NULL,$regexp=NULL){
  if(!$prompt){
    if(!$this->prompt){
    }
    else {
      $prompt=$this->prompt;
    }
  }
  if(!$this->socket){ // Ha nem tud kapcsolódni, kivétellel elszáll
    $this->connect();
  }
  $end=false;
    $this->buffer="";
      $n=0;
      $loop=0;
    while(!$end && !feof($this->socket)){
      $c=fgetc($this->socket);
      if($c==IAC){
        if($this->negotiation()){
          $c="";
          ++$loop;
          print"$loop<br>";flush();
          if($loop>10){
            throw new ETelnetConnectException("Negotiation faild! LOOP",1);
          }
          else {
            continue;
          }
        }
        throw new ETelnetConnectException("Negotiation faild",1);
      }
      else {
        $loop=0;
      }
      $this->buffer.=$c;
     if(strlen($this->buffer)>=strlen($prompt)){
       if(substr($this->buffer,strlen($this->buffer)-strlen($prompt))==$prompt){
          break;
        }
      }
      if($regexp){
        if(preg_match($regexp,$this->buffer)){
          break;
        }
      }
    if(++$n>$this->bufferSize) { // puffer méret túllépés
      $this->errormsg="Not enough size in Telnet buffer";
        $this->errornum=1;
      throw new EBufferOverflowException($this->errormsg,$this->errornum);
    }
  }
  return true;
}
/**
 * Stringet ír a telnet csatornára.
 * @param string $str A csatornára írandó string.
 * @return boolean Az írás sikeressége.
 */
public function send($str){
  if(!$this->socket){ // Ha nem tud kapcsolódni, kivétellel elszáll
    $this->connect();
  }
  return fwrite($this->socket,$str);
}
/**
 * Képernyőméret kérése a távoli telnet végponttól.
 * @param int $width A kért szélesség karakterekben.
 * @param int $height A kért magasság sorokban.
 * @return string A méretkérés során keletkezett input.
 */
```

```
  public function setWindow($width,$height){
    if(!$this->socket){ // Ha nem tud kapcsolódni, kivétellel elszáll
      $this->connect();

    }
    fwrite($this->socket,IAC.WILL.NAWS);
      $str=fread($this->socket,$this->bufferSize);
        if(strstr($str,DOO.NAWS)){
          $width=dechex($width);
            if(($l=strlen($width))<=2){
              $wh=0;
              $wl=hexdec($width);
            }
            else {
              $wl=hexdec(substr($width,$l-2,2));
              $wh=$hexdec(substr($width,0,$l-2));
            }
          $height=dechex($height);
            if(($l=strlen($height))<=2){
              $hh=0;
              $hl=hexdec($height);
            }
            else {
              $hl=hexdec(substr($height,$l-2,2));
              $hh=hexdec(substr($height,0,$l-2));
            }
          fwrite($this->socket,
IAC.SB.NAWS.chr($wh).chr($wl).chr($hh).chr($hl).IAC.SE);
        }
      return $str;
  }
  /**
   * szöveg karaktereit hexadecimális értékké konvertálja vesszővel elválasztva
   * @param string $str Szöveg.
   * @return string A hexa számokat tartalmazó karakterlánc.
   */
  public function toHex($str){
    $result="";
    for($i=0;$i<strlen($str);$i++){
      $result.=bin2hex($str[$i]).",";
    }
    return rtrim($result,",");
  }
  /**
   * A távoli végponttól jövő kontroll kérések nyugtázása.
   * @return boolean A sikeresség.
   */
   public function negotiation(){
    if(!$this->socket){
      return false;
    }
    if(($c=fgetc($this->socket))!=IAC){
      $opt=fgetc($this->socket);
        if($c==DOO || $c==DONT){
          fwrite($this->socket,IAC.WONT.$opt);
        }
        if($c==WILL || $c==WONT){
          fwrite($this->socket,IAC.DONT.$opt);
        }
        else {
        }

    }
    else {
      // hiba! Nem megfelelő karakter
      return false;
    }
    return true;
  }
  /**
   * Lekérdezi a telnet puffer méretét.
   * @return int A puffer mérete karakterekben.
   */
  public function getBufferSize(){
    return $this->bufferSize;
  }
  /**
   * Meghatározza a telnet puffer méretét.
   * @param int $size A méret karakterekben
   */
  public function setBufferSize($size){
    if($size<0){
      return;
    }
    $this->bufferSize=$size;
  }
  /**
   * Törli a hiba üzenetet, és kódot.
   */
```

```
 public function isConnected(){
   if($this->socket){
     return true;
   }
   return false;
 }
 /**
  * A telnet puffer tartalmát törli.
  */
 public function clearBuffer(){
   $this->buffer=NULL;
 }
 /**
  * A hibaüzenet tartalmát törli.
  */
 private function clearError(){
   $this->errormsg="";
     $this->errornum=NULL;
 }
  /**
   * A telnet puffer tartalmát adja vissza.
   * @return string A telnet puffer tartalma.
   */
 public function toString(){
    return $this->buffer;
  }
}
```

A.5.4. CSNMPInterface

```
/**
 * @see ESNMPException
 */
include_once ($CLASS_DIR."exception/ESNMPException.php");
/**
 * class CSNMPInterface
 * SNMP műveleteket nyújtó felület.
 * @file CSNMPInterface.php
 * @package default
 * @subpackage util
 * @author Lefi
 * @version 1.0
 * @copyright Copyright &copy; 2008, Lefi
 * Created on 2008.03.24.
 */
class CSNMPInterface {
 /**
  * Read Only community
  * @var string
  */
  private $roCommunity=NULL;
  /**
   * Read and Write community
   * @var string
   */
 private $rwCommunity=NULL;
 /**
  * SNMP verzió
  * @var string
  */
 private $version="2c";
 /**
  * SNMP kérések időkorlátja másodpercben.
  * @var int
  */
 private $timeout=100000;
 /**
  * Az újrapróbálkozások száma hiba esetén.
  * @var int
  */
 private $retries=3;
 /**
  * Konstruktor
  * @param string $roCommunity Read Only community
  * @param string $rwCommunity Read and Write community
  * @param string $version SNMP verzió.
  */
 public function __construct($roCommunity,$rwCommunity,$version){
   $this->roCommunity=$roCommunity;
     $this->rwCommunity=$rwCommunity;
       $this->version=$version;
 }
 /**
  * SNMP get protokoll funkció. MIB változó olvasása.
  * @param string $hostname A távoli eszköz hosztneve, vagy IP címe.
  * @param string $mib Az olvasandó MIB OID-ja.
```

```
   * @return string Az érték <típus>: <érték> formában.
   * @throws ESNMPException Hiba esetén kivételt dob.
   */
  public function get($hostname,$mib){
    $community=$this->roCommunity?$this->roCommunity:$this->rwCommunity;
    if(!$result=snmpget ($hostname,$community,$mib,
$this->timeout,$this->retries)){
        throw new ESNMPException("SNMP Error. I can't get result from $hostname
reading $mib object.",1);
    }
    return $result;
  }
  /**
   * MIB változót állít be.
   * @param string $hostname A távoli eszköz hosztneve, vagy IP címe.
   * @param string $mib Az olvasandó MIB OID-ja.
   * @param string $type Az érték típusa.
   * @param string $value A változó értéke.
   * @return boolean A sikeresség.
   * @throws ESNMPException Hiba esetén kivételt dob.
   */
  public function set($hostname,$mib,$type,$value){
    if(!snmpset($hostname,$this->rwCommunity,$mib,$type,$value,$this->timeout)){
      throw new ESNMPException("SNMP Error. I can't write data into $mib object of
$hostname.",1);
    }
    return true;
  }
  /**
   * SNMP lista bejárás.
   * @param string $hostname A távoli eszköz hosztneve, vagy IP címe.
   * @param string $mib Az olvasandó MIB OID-ja.
   * @return array A bejárt lista rekordjai.
   * @throws ESNMPException Hiba esetén kivételt dob.
   */
  public function walk($hostname,$mib){
    $community=$this->roCommunity?$this->roCommunity:$this->rwCommunity;
    if(!is_array($result=snmpwalk($hostname,$community,$mib,$this->timeout,
$this->retries))){
      throw new ESNMPException("SNMP ERROR. I can't walk $mib objects on $hostname
agent.",1);
    }
    return $result;
  }
  /**
   * Lekérdezi az Időkorlátot.
   * @return int Az időkorlát másodpercekben.
   */
  public function getTimeOut(){
    return $this->timeout;
  }
  /**
   * Lekérdezi az újrapróbálkozások számát.
   * @return int Az újrapróbálkozások száma.
   */
  public function getRetries(){
    return $this->retries;
  }
  /**
   * Beállítja az Időkorlátot.
   * @param int Az időkorlát másodpercekben.
   */
  public function setTimeOut($timeOut){
    $this->timeout=$timeOut;
  }
  /**
   * Beállítja az újrapróbálkozások számát.
   * @param int Az újrapróbálkozások száma.
   */
  public function setReries($retries){
    $this->retries=$retries;
  }
 }
```

Irodalomjegyzék

[1] Matt Zandstra. Tanuljuk meg a PHP5 használatát 24 óra alatt. Kiskapu Kft., Budapest, 2004.

[2] Wendell Odom. CCNA INTRO Exam Certification Guide. Cisco Press, Indianapolis, 2004.

[3] Wendell Odom. CCNA ICND Exam Certification Guide. Cisco Press, Indianapolis, 2004.

[4] Cisco MIB browser: http://tools.cisco.com/Support/SNMP/do/BrowseOID.do

[5] Online MIB térkép: http://www.carsten.familie-doh.de/mibtree/root.html

[6] PHP manual: http://www.php.net/manual/en/index.php

[7] Részletes áttekintés az SNMP protokollról: http://www.tcpipguide.com/free/t_TCPIPNetworkManagementFrameworkandProtocolsSNMPand.htm

[8] SNMP Wiki: http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol