AUTOMATIZÁLÁSI ÉS
KOMMUNIKÁCIÓ-TECHNOLÓGIAI TANSZÉK

Miskolci Egyetem

Gépészmérnöki és Informatikai Kar

SNMP hálózat-felügyelet

Szakdolgozat

Készítette: Tompa Tamás

Neptun-kód: LJQHYK

Cím: Miskolc, Szentgyörgy út

Tartalomjegyzék

1. Bevezetés

A kezdetben, mikor megjelentek a számítógépes hálózatok, csak egyetemi oktatók, kutatók használták, és a biztonsági illetve az egyéb hálózati problémákkal kapcsolatos kérdéseknek ekkor még nem szenteltek sok figyelmet. Azután néhány évvel, megjelent az internet, melyet már nem csak hozzáértő emberek kezdtek el használni a szolgáltatásai, lehetőségei miatt, így az interneten lévő számítógépek száma rohamosan növekedett. Ma már az internet árának rohamos csökkenése lehetővé tette, hogy a hétköznapi háztartásokban is használójává váljunk, így egyre jobban növekszik a globális számítógépes-hálózat, az internet.

A számítógép hálózatok növekvő mérete miatt egyre nehezebb a számítógépes hálózatok karbantartása, működésének ellenőrzése, illetve minél nagyobb a hálózat annál nagyobb feladat hárul rá, annál nehezebb gond esetén megtalálni a hiba okát, és annál nagyobb a veszteség is. A különböző hálózat felügyelő rendszerek segítségével lehetőség nyílik a számítógépes hálózat működésének ellenőrzősére, megfigyelésére. Az ilyen felügyeleti rendszerek lehetnek szoftveresek, illetve hardveresek is, ma a piacon többféle hálózat-felügyeleti rendszer is létezik. A hálózat megfigyelése növeli a rendelkezésre állóságot, csökkenti a hibaelhárítás idejét, költségét, illetve a megfigyelés során rendellenes működéseket felismerhetünk, kiszűrhetünk. Segítségükkel megfigyelhetjük a hálózat elérhetőségét, késleltetését, terheltségét, a kapcsolatok minőségét és terheltségét. A hálózat menedzselés alapvető célja, hogy számítógéppel helyettesítse az ember erőforrásokat, egy biztonságosabb számítógép-hálózat érdekében.

A számítógépes hálózatokban megjelenhetnek illetéktelen behatolók, akik nem kívánt tevékenységeket folytathatnak, és attól függően, hogy mekkora sikerrel járnak számukra nem publikus adatokhoz juthatnak. A dolgozatom során megoldandó feladat célja, hogy az ilyen illetéktelen behatolókat kitiltsa a számítógép-hálózatból, illetve a nem kívánt állomás helyét azonosítsa.

A dolgozatom témáját az Automatizálási és Kommunikáció-technológiai Tanszék által kiírt témák közül választottam. Azért eset a választásom e téma mellett, mert mindig is érdekelt a számítógép-hálózatok működése, felépítése és a dolgozathoz kapcsolódó feladat megoldása során mélyebben betekintést nyerhetek a számítógép-hálózatok és hálózatban jelenlévő hálózati eszközök működésébe.

Dolgozatom célja a számítógépes hálózat-felügyelet, ezen belül az SNMP hálózat-felügyelet témakörének tanulmányozása, illetve annak a feladatnak a megvalósítása, hogy a számítógép-hálózatban adott állomás helye meghatározható és szükség esetén kitiltható legyen.

2. A hálózat-felügyelet fogalma, feladata, elemei

[1][2]

Egy számítógépes rendszer minél több embert szolgál ki, annál nagyobb feladat hárul rá, annál nagyobb gondot jelent, ha a számítógépes-hálózatban lévő valamely eszköz „fel mondja a szolgálatot”, azaz elromlik. Ha a számítógépes hálózatban valamilyen hiba következik be, akkor általában hiba megoldása, felderítése jóval több időt vehet igénybe, mint maga a hiba megelőzése. Ezen kívül a számítógépes hálózatban egyéb berendezések is működnek, ilyenek például a router-ek, hub-ok, switch-ek, melyek el is romlanak tovább növelve a hiba lehetőségét, ezért a mai korszerű informatikai és távközlési rendszerek nélkülözhetetlen részei a felügyeleti rendszerek.

A hálózat-felügyeleti rendszerek, eszközök, módszerek, segítségével csökkenthető a hiba okozta kár. Ezen rendszerek feladata hasonlít a monitoring, illetve a mérés-adatgyűjtő rendszerek működéséhez, hiszen folyamatosan figyelik egy rendszer adott paramétereit. Lehet egy rendszer tökéletesen működő, a lehető legautomatizáltabb, figyelheti a legapróbb rezdüléseket is, az egész mit sem ér a rendszergazda, a rendszer-adminisztrátor nélkül, aki amikor már minden automatizmus kimerült, tanácsot ad és intézkedik. A hálózat-felügyelet élén kezdetben a rendszergazdák álltak, ők oldották meg a hálózatban keletkezett hibákat. Ma sincs ez másképp, de ma már a rendszerek jóval bonyolultabbak, jóval nagyobbak.

A hálózat-felügyelet lényege úgy fogalmazható meg, hogy azon erőforrások irányítása, koordinálása, felügyelése, melyek a hálózati kommunikáció lebonyolításáért felelősek, illetve feladata, hogy úgy irányítson egy összetett hálózatot, hogy közben maximalizálja annak hatékonyságát. Célja, hogy biztosítsa a hálózat egészének folyamatos és az elvárásoknak megfelelő működését. A hálózat-felügyelet tárgya lehet számítógép hálózat, adatátviteli hálózat és kommunikációs hálózat is.

A hálózat-felügyeleti rendszerekben lévő eszközök valamilyen modellt valósítanak meg. A modellben felügyelt objektumok vannak, amibe minden beletartozik, amit működés közben valamilyen módon meg tudunk figyelni. Maga a felügyelet folyamata a felügyelő állomáson zajlik, ahol egy magas szintű alkalmazás fut, amely összegyűjti, raktározza, elemzi a hálózat működését jellemző adatokat, a berendezések állapotát mutató

paraméterektől az adatforgalom számáig. A felügyelő állomást ágensek, azaz az adatgyűjtő programok látják el információval. A felügyelő programban az adatgyűjtés könnyen megvalósítható, hiszen az adatcsomagok mindenhová eljutnak, így lehetséges az adatforgalom közvetlen megfigyelése. Az összegyűlt adatok elemezhetők, a hálózat működését jellemző statisztika állítható elő belőlük. Ezeket a statisztikákat különféle grafikonokon szokás megjeleníteni, de használhatók ezek a statisztikák - vagy akár az egyedi változóértékek - riasztásra is.

Mint minden másrendszer esetében, így a hálózat-felügyelő rendszerekkel szemben is vannak különböző követelmények, elvárások, melyek a következők:

- hálózat elérhetőségének figyelése,
- válaszidők figyelése,
- biztonsági funkciók nyújtása,
- hálózati forgalom irányítása,
- különböző helyreállítási lehetőségek.

2.1 A hálózat-felügyelet elemei

Ezek a következők:

- megfigyelés
- megjelenítés
- naplózás
- riasztás

Megfigyelés
[3]
A megfigyelés lehetővé teszi a rendelkezésre állás növelését, a hibaelhárítás idejének és költségének csökkenését, a rendellenes működés felismerését és kiszűrését. A megfigyelhetőek az eszközök esetében az elérhetőség, a késleltetés, a terheltség, kapcsolatok esetében pedig a kapcsolat terheltsége, minősége. A megfigyelést lehetővé tévő eszközök az ICMP figyelés, TCP és UDP port-figyelés, az SNMP, stb.

Megjelenítés
[3][4]
Segítségével ábrázolható a hálózat felépítése, jól áttekinthető hálózatrajz hozható létre, amely dokumentálja az épület alaprajzán a számítógépes hálózat kábeleit, végpontjait, elosztóit, router-eit, swich-eit és internet csatlakozási pontjait. Kiválóan használható, ha fejlesztést tervezünk a hálózatban, hiszen ennek segítségével megtudhatjuk, hogy pontosan milyen berendezések vannak a számítógép-hálózatban. Egyszerűen áttekinthető benne a teljes épület kábelezése és nem kell szétbontani az összes kábelcsatornát, álmennyezetet, hogy megtudjuk, hol futnak a kábelek és így rengeteg munkát és időt spórolhatunk meg.

Naplózás
[3]
Segítségével nyilvántarthatóak a hálózat működése során keletkező rendellenességek, hibák, a hálózat terheltsége, melyek könnyebb értelmezés érdekében grafikonon is megjeleníthetőek.

Riasztás
[3]
Valamilyen váratlanul bekövetkező, hiba, rendellenes működés vizuálisan történő megjelenítését jelenti, tájékoztatja a rendszergazdát a hibáról. Különböző küszöbértékek határozhatóak meg, melyeket, ha átlép a rendszer, akkor riasztást, vagy akár műveletet vált ki.

2.2 Hálózat-felügyeleti modellek
[5][6]

Több szervezet is dolgozott ki hálózat-felügyeleti módszertanokat, modelleket, protokollokat, ezen szervezetek a következők:

- International Organization for Standardization (ISO)
- International Telecommunication Union (ITU, régebben CCITT)
- Internet Engineering Task Force (IETF)

Ezen három szervezet közül az ISO volt a legelső, aki az OSI (Open Systems Interconnection) program részeként kifejlesztett egy architektúrát a hálózatmenedzsmentre. Az 1980-as években több javaslat is született a felügyeleti architektúrákra vonatkozóan. Az

ISO tervei között szerepelt még a távközlő hálózatokra vonatkozó menedzsment szabványok létrehozása is, mely szabványok kidolgozása a CCITT feladata volt. A következőkben az internet fejlődésével egyre inkább nőtt az igény egy jól működő hálózat-felügyeleti protokoll iránt.

A hálózatmenedzsment alapját az alábbi három modell alkotja:

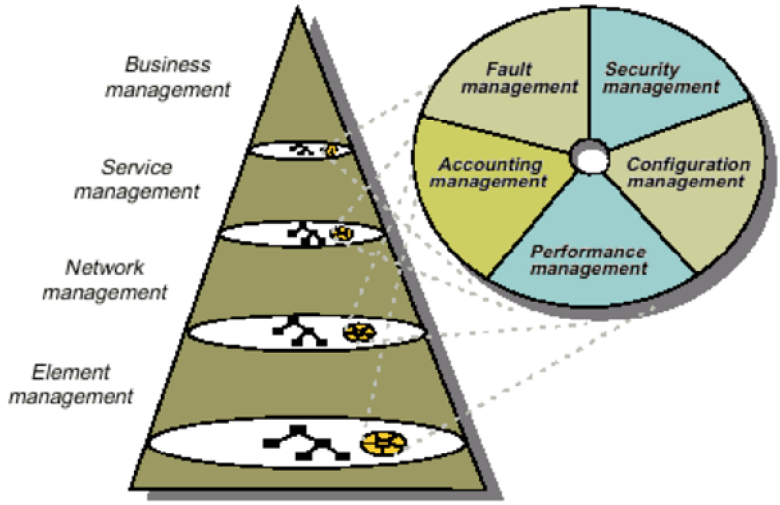
1. Az ISO 7894-4 szabvány, amely a hálózatmenedzsmentet funkciók szerint osztályozza

2. OSI referencia modell, amely a hálózati funkciókat rétegekre osztja

3. Menedzser állomás/agent modell, amely biztosítja a hálózati egységek irányítását, ellenőrzését és monitorozását.

2.2.1 Funkciók szerinti osztályozási modell
[5][6]

Az ISO (International Organization for Standardization) a hálózatmenedzsmentet 5 funkció szerinti részterületre osztotta, melyet a következő ábra (1. ábra) szemléltet:



1. ábra A hálózatmenedzsment részterületei

Ezen funkciók a következők:

- Fault management (hibamenedzsment)
- Security management (biztonságmenedzsment)
- Configuration management (konfiguráció menedzsment)
- Performance management (teljesítmény menedzsment)

- Accounting management (elszámolás kezelés)

Hibamenedzsment
[5][6]

A hálózati hibák felderítéséért és megoldásáért felelős. Ez a menedzsment egyik legfontosabb eleme, mert a hibák a hálózat rendellenes működését eredményezhetik, ami következtében a hálózat használhatatlanná válhat. Célja a hibák izolálása, naplózása, az érintett személyek értesítése, hibák automatikus javítása. Előfeltétele a hálózat dokumentálása.
Mivel a hibák leállást, vagy elfogadhatatlan hálózati működést is eredményezhetnek, ezért talán a legtöbb figyelmet a hibamenedzsmentre fordítanak az ISO szerinti hálózatmenedzsment rendszerekben. A hiba hatékony menedzsmentje azért is fontos, mert az kézzelfogható, számszerűsíthető veszteséget okozhat. További feladata a hibák által érintett területek meghatározása.
A hibamenedzsment során gyakran kell megoldani felhasználókkal kapcsolatos problémákat, ezért feladatai közé tartozik ezen problémák megelőzése is.
Például egy hálózati egység újrabootolásakor üzenetet küldhet, ami lehet egy sürgős e-mail, vagy jelentkezhet akár egy jelzésként is a hálózatmenedzser képernyőjén. Ha ezek a reboot üzenetek gyakoriak az adott egységtől, akkor az valószínűleg egy közelgő hibát jelez.
Másik példa, ha egy port nagyon magas átviteli hibaaránnyal dolgozik. Ekkor ez a magas hibaarány dominószerűen további hibákat okozhat, amelyek hatására egy protokoll állandóan újrakonfigurálja magát.

Biztonságmenedzsment
[5][6]

Feladata a hálózati adatforgalomhoz történő illetéktelen hozzáférést megakadályozni, a megtörtént illetéktelen hozzáférést felszámolni, a hálózati hozzáférést korlátozni a helyi szabályozások alapján, illetve a hálózat működésképtelenné tételét megakadályozni. Egy ilyen biztonság menedzsment rendszer kezelheti például az erőforrásokhoz való hozzáféréseket és megtilthatja annak elérését.
Szükséges egy biztonságpolitika megalkotása, és a kísérő szabályzatok meghozatala. A hálózatok biztonsági szintjét az USA Védelmi Minisztériuma által definiált TCSEC (Trusted Computer System Evaluation Criteria) szintek szerint osztályozhatjuk. A szintek

A-tól D-ig terjednek, ahol „A” jelenti a legmagasabb biztonsági szintet, „D” pedig a legalacsonyabbat.
„A”-ba a kritikus rendszerek, „B”-be a kiemelt rendszerek, „C”-be a normál rendszerek tartoznak, a „D”-be pedig a minimális védelem tartozik.
Például a jelszavak állományának bővítése, illetve karbantartása is a biztonság menedzsment feladata, illetve feladatai közé tartozik még a LAN és az internet közötti tűzfal felállítása is.

Konfiguráció menedzsment
[5][6]

A konfiguráció menedzsment a hálózatmenedzsment legfontosabb területe, mivel ha a hálózat nincs megfelelően konfigurálva, akkor előfordulhat, hogy egyáltalán nem is működik, váratlanul meghibásodhat, vagy egyszerűen nem optimálisan működik, tehát nem ad megfelelő teljesítményt. Célja a hálózati- és rendszereszközök konfigurációinak szisztematikus kezelése, amely következtében folyamatosan nyomon követhető a hálózat üzemeltetéséből adódó hardver és szoftver konfigurációk változása. Adatokat szolgáltat az aktuális konfigurációs részletekről illetve nyilvántartja a hálózat változásait. Funkciói közé tartozik még a készletnyilvántartás, a kábelezési rendszer nyilvántartása és a topológiai változások nyomon követése is. A fontos információk egy adatbázisban helyezkedhetnek el, amely információk jellemzik a csomópontokat, és értékük mindig aktuális. Ez az adatbázis általában a következő adatokat tartalmazza: kapcsolóelemek adatai, erőforrás kihasználtság, hőmérsékleti adatok, a forgalomirányítók adatai, stb. Egy probléma felmerülésekor ez az adatbázis segíthet a hiba okának felderítésében.
Példa lehet, amikor frissítünk egy szoftvert, majd a frissítés következtében valamelyik szolgáltatás leáll.

Teljesítmény menedzsment
[5][6]

A teljesítmény menedzselés során figyeljük a hálózat kihasználtságát, a végpontok közti válaszok időtartamát. A hálózat kihasználtsága azon hálózati egység függvénye, amely az adattovábbítást végzi. A végpontok közti válaszok időtartama az, az idő, amely ahhoz szükséges, hogy az egyik pontból az adat a hálózatban kiválasztott másik ponthoz érjen.

A teljesítmény menedzsment három részfeladatot lát el. Ezek a követezőek: az adatok kinyerése a hálózatból, a nyers adatok feldolgozása, illetve a riasztás vagy egy művelet végrehajtása.
Feladatai közé tartozik a hálózat teljesítményére vonatkozó adatok meghatározása, teljesítménymutatók definiálása, melyek következtében megpróbálunk a múltban bekövetkezett eseményekből a jövőbeli eseményekre következtetni. Fontos a küszöbértékek meghatározása, melyeknek átlépése riasztást, vagy valamilyen műveltet vált ki. Ezen küszöbértékek finomhangolásának elengedhetetlen eszköze a paraméterek mérése.
Ilyen mutatók például az átviteli kapacitás, keretméret, erőforrás használat, válaszidő, stb.

Elszámolás kezelés
[5][6]

Az elszámolás kezelés gyakori neve még a nyilvántartás menedzsment. Fő célja a hálózati erőforrások felhasználásának nyomon követése, naplózása, könyvelése. Feladatai közé tartozik a szoftverek szolgáltatta adatok alapján a hálózati erőforrásokat fokozottabban igénylő felhasználók kiszűrése és részükre adott esetben nagyobb sávszélesség biztosítása, vagy esetleg éppen ellenkezőleg, amennyiben tevékenységük felesleges a hálózati politika szerint, abban az esetben a hozzáférés leszűkítése. Ezzel csökkenthetőek a hálózati problémák és igazságosan eloszthatóak az erőforrások a felhasználók és a szolgáltatások között. Hasonlóan a teljesítmény menedzsmenthez, a nyilvántartás menedzsmentnél is fontos a hálózat terheltségének mérése, figyelése. Ezek alapján meghatározhatók bizonyos felhasználói szokások, melyek alapján például a kvótákat a megfelelő szintre be lehet állítani az igazságos erőforrás-kiosztás érdekében.. Ez jelentősen befolyásolja az egyes szolgáltatások által igényelt sávszélességet. A hálózat erőforrás felhasználásainak megfigyelésével folyamatosan finomítani lehet a hálózat teljesítményét.

2.2.2 OSI referencia modell
[7]

Az OSI referencia modell szerint egy hálózatot 7 rétegre osztunk. Az egyes rétegek megnevezése a következő:

1. Fizikai
2. Adatkapcsolati
3. Hálózati
4. Szállítási
5. Viszony
6. Megjelenítési
7. Alkalmazási

Fizikai réteg

Fő feladata a bitek kommunikációs csatornán való továbbítása. Ez a réteg határozza meg az eszközökkel kapcsolatos specifikációkat, beleértve a kábelek, aljzatok lábkiosztását, és a használt feszültségszinteket is.

Adatkapcsolati réteg

Feladata, hogy biztosítsa azokat az eljárásokat, amelyek lehetővé teszi, hogy adatokat lehessen továbbítani két hálózati elem között, illetve jelzi és ha van rá lehetőség, akkor korrigálja a fizikai szinten létrejött hibákat. Ebben a rétegben működnek különböző hálózati eszközök, ilyenek a switch-ek és a bridge-ek is.

Hálózati réteg

Ez a réteg biztosítsa, hogy a váltakozó hosszúságú adatok eljussanak a küldőtől a címzetthez, oly módon, hogy az adatok továbbítása akár több hálózaton keresztül is történhet. Megvalósítja az útvonalválasztást és a hibaellenőrzési funkciókat, ebben a rétegben a működnek a router-ek.

Szállítási réteg

Feladata, hogy biztosítsa és ellenőrizze egy adott kapcsolat megbízhatóságát. Nyomon követi az adatcsomagokat és hiba esetén gondoskodik a csomagok újraküldéséről. A réteg által tartalmazott legismertebb protokoll a TCP és az UDP.

Viszonyréteg

Feladata, hogy megvalósítsa a végfelhasználói alkalmazások közötti kommunikáció menedzselésére alkalmas mechanizmust. Ilyen mechanizmus lehet a duplex és a fél-duplex összeköttetés.

Megjelenítési réteg

Az alkalmazási réteg számára biztosítja, hogy az adatok a végfelhasználó rendszerének megfelelő formában álljanak elő. Megvalósítja a kódolást, dekódolást, tömörítést és az egyszerűbb adatkezeléseket, például, hogy egy adott kódolású szöveges fájlt egy más kódolású szöveges állománnyá konvertál.

Alkalmazási réteg

A réteg szolgáltatásai az alkalmazások közötti kommunikációt támogatják, protokolljain keresztül az alkalmazások képesek egyeztetni az adatok formátumáról biztonsági, szinkronizációs és egyéb hálózati igényekről. Az ismertebb alkalmazási réteg szintű protokollok a HTTP, az SMTP, és az FTP.

2.2.3 A menedzser állomás/agent modell

[5]

A menedzser állomás grafikus vagy szöveges képet nyújt a hálózatról, az ágensek által kapott vagy küldött adatokra alapozva. Az ágensek információt adnak a hálózati egységekről, és lehetőséget biztosítanak annak konfigurálására. A menedzserállomás egy ablakot nyit a hálózatra, amelyen keresztül vezérelhető, megfigyelhető a teljes hálózat, az ágensek pedig a hálózati egységekhez biztosítanak hozzáférést. Az ágens egy interfészt biztosít bizonyos számlálók állapotának nyomon követésére és a paraméterek megváltoztatására, vagy az események nyomon követésére hiba esetén. Az ágens hálózati egységre specifikált, általában minden hálózati eszköz saját interfésszel és saját menedzselhető attribútummal rendelkezik. A hálózati állomások és az ágensek kombinációja lehetővé teszi a hálózat vezérlését.

2.3 Az SNMP

[5][6][8][9]

Története

Az SNMP a Simple Network Management Protocol, azaz az egyszerű hálózat menedzsment protokoll rövidítése, és a TCP/IP család részét képezi. Az IETF fejlesztette ki, létrehozásakor azt tűzték ki célul, hogy lehetőséget biztosítson arra, hogy a számítógép-hálózatot távolról lehessen menedzselni egyes adatok lekérdezésével, beállításával illetve, hogy a hálózati egyedeket, amelyek lehetnek akár nyomtatók, szerverek, switch-ek, router-ek, bárhonnan ellenőrizni, felügyelni, és szükség esetén konfigurálni lehessen. Tehát a hálózatra csatlakoztatott eszközök vezérlését és adatinak lekérdezését valósítja meg. Egy egyszerű kérdezz-felelek protokollnak tekinthető, ahol a hálózati menedzselő rendszeren futó alkalmazások folyamatosan vagy egy előre meghatározott időközönként lekérdezik a felügyeleti eszközökhöz rendelhető változókat, amelyek majd valamilyen választ adnak az esetleges további feldolgozás céljából.

1989-ben került a TCP/IP hálózatok standardjai közé majd később világszinten is elfogadottá vált a TCP/IP hálózatok Interneten keresztül történő felügyeletére. A protokoll részletes leírását az 1065-ös, 1166-os, 1167-es, 1441-es, 1452-es, a 3411-es és a 3418-as RFC-k tartalmazzák.

1991-ben bővült ki a Távoli Monitorozás (Remote Monitoring - RMON) implementálásával. Az RMON kibővíti az SNMP lehetőségeit a menedzselésében.

1993-ban megjelent meg az SNMPv2. A v2 néhány újabb funkcióval kibővíti az SNMP-t, valamint definiálja annak használatát OSI alapú rendszerekben.

Az SNMPv3 a jelenleg hivatalos SNMP szabvány, a régebbieket az IETF elavultnak nyilvánította. A SNMP-vel menedzselhető eszközök általában több SNMP verziót is támogatnak.

Működése

Működése alapvetően kliens-szerver kapcsolatra épül. A menedzselhető eszközön fut egy szerver, amely többnyire a 161 és 162-es portokon figyeli a kéréseket. A kéréseket a menedzselő állomás, vagy is a kliens küldi. Fontos fogalom az agent (ágens), amely a jelzéseket küldi a menedzser állomás felé. Az összegyűjtött adatok bizonyos idő elteltével, vagy meghatározott időközönként továbbítódnak a menedzser állomás felé. Az agent-ek

nem képesek egyben több adatot tárolni és továbbítani, ezért a menedzser állomás minden adatlekérdezés során hálózati forgalmat generál, amely jelentősen leterheli a hálózatot.

Elemei

Az SNMP modell a következő elemekből épül fel:

- manager
- agent
- managed nodes
- managemenet information base

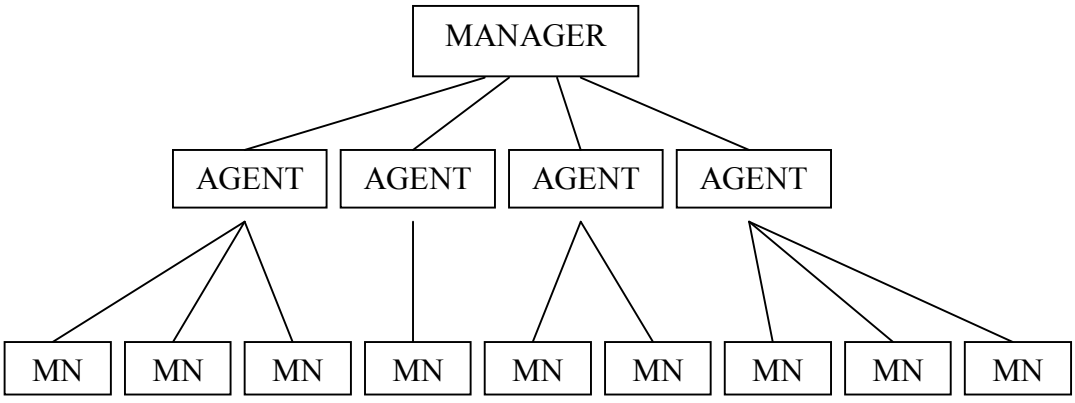
A manager a hálózati menedzser állomás, az információkat gyűjti és tárolja.

Az ágensek, vagyis az ügynök programok a csomópontokról gyűjtenek információkat, és azokat egy helyi adatbázisban tárolják. Minden csomópont csak önmagáról szolgáltat információt. Az ügynök programok a felügyeleti állomással az SNMP protokoll segítségével kommunikálnak.

A managed nodes, vagyis a felügyelt csomópont a felügyeleti rendszer egyik fontos szereplője. Felügyelt csomópont lehet a hálózatban minden olyan eszköz, amely képes SNMP ügynök futtatására. Napjainkban ezek lehetnek gazdagépek, switch-ek, router-ek stb. Minden csomópont csak önmagáról szolgáltat információt.

A management information base, vagyis a MIB, egy virtuális hiearchikus adatbázis, amely a csomópontra vonatkozó adatokat tárolja és szolgáltatja.

A következő ábra az SNMP modell felépítését szemlélteti:



2. ábra Az SNMP modell felépítése

Működési módozatok

A kommunikációs folyamat egyik módja, mikor az ügynökök a felügyelt állomás kéréseire válaszolnak, ebben az esetben lehetséges kérések ismertek. Egy másik mód, mikor az ügynök kezdeményezi az adatátvitelt SNMP trap segítségével. Ez általában akkor történik, mikor az ügynök kritikus paramétereket észlel a rendszerben és lehetőséget ad arra, hogy nemkívánatos eseményekkel kapcsolatban információhoz lehessen jutni. A harmadik mód esetében a csomópont nem képes SNMP ügynök futtatására, ekkor az ügynök egy másik eszközön fut, amely közvetít a felügyeleti- és a céleszköz között. Ezt a harmadik féle működési módszert neveik SNMP proxy ágensnek.

Üzenet típusok

Alapvetően UDP csomagok segítségével kommunikál. A protokoll első változatát még ma is alkalmazzák, melynek az üzenet típusai a következők:

Üzenet típus	*Magyarázat*
SET	egy objektumnak értékadás, az eszköz tulajdonságának a megváltoztatására szolgál
GET	egy adott információ lekérése
GETNEXT	a MIB fában a következő változó értékének a lekérdezése
SET, GET SESPONSE	válasz a SET, GET, és GETNEXT kérésekre
TRAP	az ügynök által a hálózati eszközről küldött információ, akkor jön létre, ha a menedzselt eszközt figyelmeztetés küldésére állították be

Az SNMP első változata az úgynevezett community string-ekre épül, amely az SNMP csomag fejrészében található. Segítségükkel az objektumokhoz való hozzáférési jogosultságot lehet befolyásolni. Az adott jogosultság olvashatóságot és írhatóságot befolyásoló lehet. Olvasási jogosultság esetén az adott objektumot csak olvasni lehet, míg írási jogosultság esetén írni és olvasni is lehetséges. Ezek a sztringek a hálózaton kódolás nélkül közlekednek.

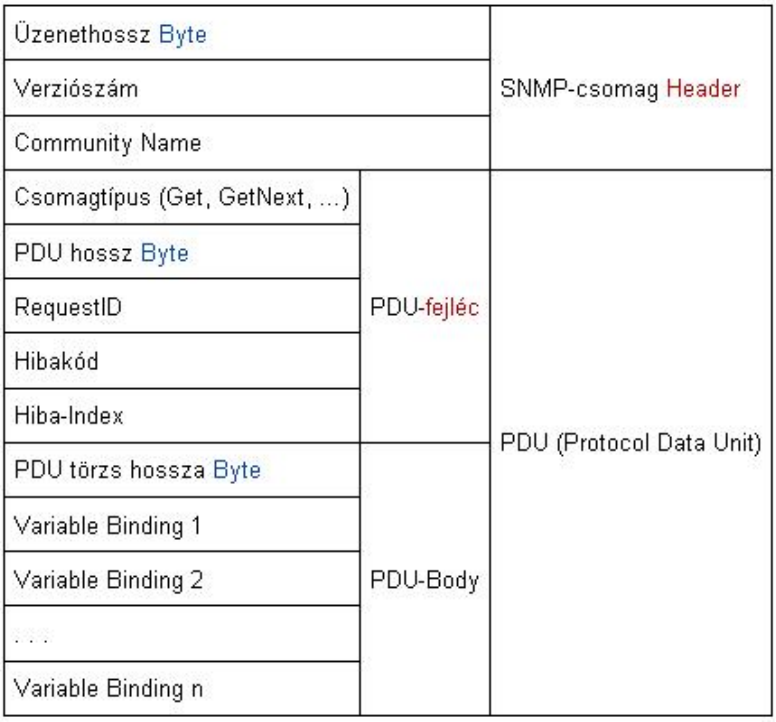
Az SNMP protokoll működéséhez szükséges, hogy az objektumok gyártótól független módon legyen leírva, továbbítva és az adatok szabványos kódolás segítségével továbbítódjanak. Erre a célra hozták létre a szabványos objektum definíciós nyelvet, az ASN 1.-et (Abstract Syntax Notation One).

Az SNMP a következő ASN 1.-beli adattípusokat engedélyezi:

Adattípus	*Magyarázat*
INTEGER	tetszőleges hosszúságú egész
BIT STRING	nulla vagy több bitből álló fűzés
OCTET STRING	Nulla vagy több előjel nélküli bájtból álló fűzér
NULL	helyfoglaló
OBJECT IDENTIFIER	objektumra való hivatkozást tesz lehetővé

SNMP csomagfelépítés

A következő 3. ábra az SNMP csomag felépítését szemlélteti.



3. ábra Az SNMP csomag felépítése

Látható, hogy az SNMP csomagot részben PDU csomag alkotja. A csomag fejrészét a csomag hozza bájtokban, az SNMP verziószáma, és a community jogosultság alkotja. A PDU fejlécében a PDU fejléc hossza bájtokban, a kérés azonosító, a hibakód, és a hiba*index található. A PDU törzsét a PDU törzs hossza bájtokban, és a „Variable Binding”, amely a MIB változót határozza meg.

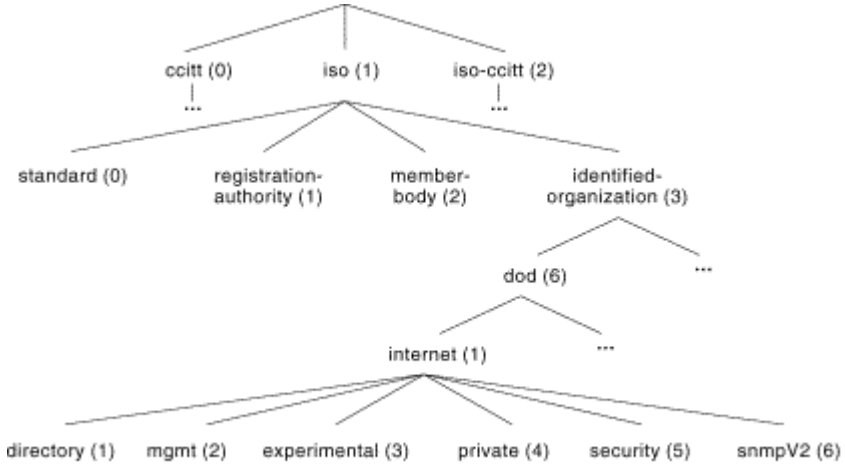
MIB (Management Information Base)
[5][6][8][10][11]

Az SNMP protokollt kifejezetten bővíthetőre tervezték. Ezt a MIB-ek (management information base: menedzsment információk csoportja) segítségével valósítják meg. Egy MIB bizonyos eszközök által használt speciális tulajdonságokat ír le, tehát olyan információk gyűjteménye, amelyek leírnak egy SNMP-vel menedzselhető entitást. Továbbá egy adott ügynök vagy menedzser felügyelete alá tartozó objektumokra vonatkozó információk összessége és az azok által alkotott struktúrák leírására használt állományformátum.

Egy fa struktúrájú hiearchiát alkot, melynek egyes csomópontjaira, vagy elemeire számláncokkal, vagy ezt helyettesítő nevekkel hivatkozhatunk.
A felügyeleti információk fa struktúráját, a MIB felépítését illetve szabályait az SMI (Structure of Management Information – felügyeleti adatok struktúrája) írja le. Az SMI megadja azon információk alaptípusait, melyekkel az SNMP tevékenykedhet, egy vázat szolgáltat a menedzselhető adatokra, de nem határozza meg a menedzselhető eszközöket. Minden fontos adatot tartalmaz, ami a menedzselés szempontjából fontos lehet, és az egyes eszközök felügyelete ezekből az adatokból építkezik. Ilyen adatok az objektum neve (name), adattípusa (datatype), leírása (description), hozzáférése (object acces), és az objektum azonosító (object identifier).

Sokféle MIB létezik, vannak gyártó-specifikusak, és általánosak is. A MIB-I volt az első SNMP MIB, amelyet elfogadtak általánosnak, ez csak korlátozott számú, elsősorban az IP hálózat közötti útválasztó változókkal foglalkozik. Később a MIB-II-vel pár fontosnak tartott elem került még bele a standard SNMP MIB-be. A MIB-II kiterjesztette a MIB-I lehetőségeit egy sor átviteli közegtípusra, és hálózati eszközre, széles körben elterjedt az interneten belül, szinte az összes hálózati eszköz támogatja.

A MIB struktúra felépítését a következő 4. ábra szemlélteti:



4. ábra A MIB struktúra részlete

A 4. ábrán a MIB-nek az a részfája látható, ami fontos a hálózati eszközök menedzseléséhez. A fa felső szintjén a szabványosító szervezetek vannak. Az ISO (1) csomópont alatt van az azonosított szervezetek (3) csomópont, és ezen belül a DOD (Department of Defense), a DOD alatt pedig az internet (1) található. A nevek mellé írt számok a nevet helyettesítik, de a nevek helyett pontokkal elválasztott, számokból álló sorozattal is meg adhatjuk az elem helyét a fában. A programok számára ez egyszerűbb, mint a nevekkel történő hivatkozás. A neveket is használhatjuk, akkor, ha pontosan ismerjük az írásmódjukat.

Minden egyes MIB-nek saját egyedi object azonosítója van (OID - Object Identifier), ez az object azonosító ponttal elválasztott egész számokból áll (0...255), amely az objektum pozícióját jelzik a teljes menedzsment fán belül. Például az Internet ág OID-je: 1.3.6.1., ezzel teljesen egyenértékű az iso.org.dod.internet megadás.

A teljes MIB fa tovább bővíthető újabb MIB-ekkel, mivel a meghatározott csoportokhoz végtelen számú alcsoport csatolható.

A következő táblázat néhány csoport elemet mutat be.

MIB	*Általa közölt Információ*
System	Az állomás vagy az eszköz operációs rendszere
interfaces	Hálózati interfész
at	címfordítás
ip	Internet protokoll szoftver

3. A feladat megoldásának lehetőségei

Ebben a fejezetben a feladat megoldásához rendelkezésre álló technikákat, technológiákat, fejlesztőeszközöket ismertetem.

3.1 A feladat

Ahhoz, hogy a feladat megoldását el tudjam kezdeni, illetve ezzel kapcsolatban információkat gyűjtsek, lehetőségeket mérlegeljek, el kell merengeni magán a feladaton, feltétlenül szükséges, a feladatot megértése.

A probléma, hogy számítógép-hálózatban jelen lehetnek olyan számítógépek, melyek tevékenykedése káros hatással lehet a hálózat működésére, illetve céljuk lehet bizonyos adatok, információk megszerzése, melyek számukra nem feltétlenül publikusak. Ezt a problémát úgy lehet orvosolni, hogy a nem kívánt állomást kitiltjuk a hálózatból, erre kell valamilyen megoldást találni. A megoldás célja, hogy megvalósítsa azt, hogy ezeket a nem kívánt állomásokat ki tiltsa a hálózatból, oly módon, hogy ha az adott állomás a hálózat más részén is csatlakozni próbál, akkor ezt ne tehesse meg, tehát legyen „kizárva” az egész hálózatból. Ez a kitiltás lehet időleges is, tehát nem biztos, hogy örökre ki kell tiltani, a kizárás vonatkozhat egy adott időszakra is. A problémának a megoldása a számítógépes-hálózatban lévő L2 szinten működő hálózati switch-eket menedzselésével valósulhat meg. Tehát a feladat elsődleges célja, hogy ezt a problémát egy rendszer vagy egy program segítségével megoldja.

3.2 Megoldási lehetőségek

A megoldásra több lehetőség is kínálkozik. Léteznek különböző hálózat-felügyeleti rendszerek a piacon, melyek segítségével a hálózat menedzselhető, illetve néhány programozási nyelvnek van hálózat-menedzselő csomagja, ilyen például a C és a Java programozási nyelv is. A következő alfejezetben ezeket, a megoldási módozatokat fogom ismertetni, majd mérlegem az egyes lehetőségeket.

A Nagios rendszer, mint megoldási lehetőség
[6][12]

A Nagios egy ingyenes és nyílt forráskódú rendszer, amely a hálózati eszközök, szolgáltatások, állapotellenőrzésére illetve a begyűjtött adatok elemzésére szolgál. Eredetileg Linux platformra készült, de kompatibilis az összes Unix rendszerrel. A telepítés után, a http://127.0.0.1/nagios3 címen érhető el a web-es felülete, amelyen keresztül felügyelni lehet a hálózati szolgáltatásokat, eszközöket, illetve konfigurálás is végezhető.

A Nagios rendszer egyik erőssége a plugin architektúra, ami a konfigurálhatóságot rugalmasabbá teszi. Ennek az architektúrának a lényege, hogy funkcionálisan két részre osztja fel a programcsomagot, ez a két rész a központi rész, és a külső programok. Teljes mértékben a modulonként viselkedő plugin-okra támaszkodik, amelyek az ellenőrzés folyamatát végzik, összegyűjtik a szükséges információkat, elemzik azokat, majd az eredményt átadják a Nagios központi részének. Az eredmények kiértékelés után a központi rész végrehajtja az előre meghatározott tevékenységeket. Ezek a tevékenységek a konfigurációs adatbázisban történnek megadásra.

A teljes rendszer működésének alapfeltétele, hogy a felügyelendő eszközökön aktívan működjenek ágens jellegű programok. Ennek egyik megvalósítása, ha minden felügyelt gépre SNMP-t telepítünk. Linux alapú állomások esetén jó választás lehet a NET-SNMP.

A másik lehetőség az NRPE (Nagios Remote Plugin Executor), amely egy közbenső réteg és önállóan is futhat egy felügyeleti gépen, feladata összegyűjteni a hoszt gép szükséges adatait. A központi gépen beállítjuk a Nagios-t, hogy időközönként forduljon a felügyelt hoszthoz, aminek következtében a megadott kérések alapján információkat kap. Beállítható, hogy mely IP címekről fogadjon el információkat. Az NRPE rendelkezik Windows-os verzióval is (Winrpe), így alkalmas bármely Windows-os gép felügyeletére. Ezzel a módszerrel megoldható egy tűzfal mögötti hálózat monitorozása, úgy, hogy a tűzfalon nyitunk egy portot a felügyelt gépek felé.

Egy harmadik lehetőség az NSCA program, amely abban különbözik az NRPE-től, hogy itt a kliens rész az, amely a konfigurációs fájl szerint összegyűjti az adatokat és elküldi a központi gépnek, ahol az NSCA feldolgozza és továbbítja a Nagios részére.

Tehát ahhoz, hogy a feltelepített Nagios-t használni is tudjuk, további beállításokra van szükség. A beállításokat konfigurációs fájlokban kell megadni, ezekben a fájlokban

kerülnek megadásra, hogy milyen eszközöket illetve szolgáltatásokat felügyeljen a rendszer, és hogy ezeket mely felügyelt gépen/gépeken valósítsa meg.

Lehetőség van a Nagios rendszerhez plugin-t készíteni, ha a rendszer adottságai nem megfelelőek, segítségével van lehetőség a rendszer tudásának a bővítésére. Ahogyan fentebb említettem a pluginok az ellenőrzés folyamatát végzik, összegyűjtik a szükséges információkat, elemzik azokat, majd az eredményt átadják a Nagios központi részének. A plugin-okat különböző programozási nyelveken van lehetőség írni. Ilyen eszköz lehet a C, a Python és a Perl programnyelv. A plugin fejlesztése során fel lehet használni már kész, meglévő modulokat is.
Olyan plugint, amely kifejezetten a feladatra megoldást ad nem találtam készen, így egy új modult lenne szükség fejleszteni, majd fejlesztés után beintegrálni a Nagios rendszerbe.

A rendszer előnyei:

- nyílt forráskódú
- rengeteg funkció
- látványos grafikus felület
- plugin architektúra

A rendszer hátárnyai:

- bonyolult telepítés és konfigurálás
- nehéz bővíthetőség

A CiscoWorks rendszer, mint megoldási lehetőség
[6]

A CiscoWorks a Cisco hálózati felügyeleti rendszere, amely különálló, egyedileg telepíthető modulokból épül fel. Ez a szoftvercsomag a fentebb bemutatott Nagios rendszerrel ellentétben nem ingyenes, így, különböző irodalmakból tájékozódtam a rendszer felépítésével, használatával, működésével kapcsolatban.

A rendszer moduljai:

- Resource Manager Essentials (RME): a Cisco eszközök adminisztrációs és felügyeleti funkcióját valósítja meg, amely magába foglalja a hálózati változások követését, és az eszközök nyilvántartását.

- CiscoView (CV): SNMP alapú eszközmenedzsment szoftver, amely grafikus felületű, és lehetőséget biztosít a Cisco eszközökből felépített hálózat valósidejű figyelésére és alapszintű beavatkozások elvégzésére, illetve biztosítja az adott eszköz élethű megjelenítését az eszköz előlapjával, illetve hátlapjával együtt.

- Campus Manager (CM): feladata a fizikai és logikai kapcsolatok grafikus megjelenítése, csomagok útirányának elemzése, felhasználók követése.

- Device Fault Manager (DFM): az eszközök állapotfigyelését végzi, figyelmeztet a hibákra, akár még a bekövetkezésük előtt, illetve együtt tud működni más hálózatmenedzsment rendszerekkel.

- Acces Control List Manager (ACLM): az eszközök által támogatott IP, IPX és MAC alapú kapcsolat forgalomszűrési beállításait végzi el.

- Internet Performance Monitor (IPM): a hálózati kapcsolatok periodikus ellenőrzését, a végpontok közötti útvonalak felderítését, valamint a válaszidők mérését végzi.

- Quality of Service Policy Manager (QPM): felületet biztosít a QoS policy-k (Quality of Service irányelvek) definiálásához és alkalmazásához.
- VPN Management Solution: VPN (Virtual Private Network – virtuális magánhálózat) környezet menedzsmentjét biztosítja, feladata a VPN hálózatok konfigurációja, implementálása.

A rendszernek van több modulja is, amely alkalmas lehet a feladat megvalósítására, ezek a RME és a CV, melyek segítségével Cisco eszközöket illetve SNMP-t támogató eszközöket lehet valós időben felügyelni és menedzselni. A rendszer ára meglehetősen borsos, így ezt a megoldást elvetettem.

Előnyök:

- rengeteg funkció
- a hálózat valósidejű menedzselhetősége

Hátrány:

- nem ingyenes, nagyon drága

Saját alkalmazás készítésének mérlegelése

A fentebb tárgyalt hálózat-felügyeleti rendszerek segítségével megoldható a probléma, de az egyik nehezen bővíthető (plugin-ok), illetve új plugin írása nehézkes, a másik pedig nagyon drága.

A Nagios plugin készítéséhez szükséges programozói ismeret, főként a C nyelv mély ismerete. Az egyik megoldási lehetőség a Nagios modul készítése, de léteznek egyéb más programozási nyelvek is, melyeknek van SNMP és hálózat-menedzselési csomagja, ezért ezek a megoldási lehetőségek is szóba jöhetnek. Ilyen egyéb más programozási nyelv lehet a Java vagy a C nyelv.

A C nyelv és az SNMP
[13]

A C nyelvhez implementáltak SNMP függvénykönyvtárat, amely segítségével az SNMP-t támogató eszközökhöz alkalmazás készíthető.

C nyelvű SNMP-n alapuló alkalmazás készítésének lépései:

1. SNMP kapcsolat létrehozása
2. tulajdonságok definiálása a kapcsolathoz
3. Ha szükséges, akkor a MIB fa további MIB-ekkel való kibővítése
4. Primary Data Unit (PDU) létrehozása
5. OID-k belecsomagolása a PDU-ba
6. Kérés küldése, és várakozás a válaszra
7. válasz azonosítása és a visszatérési érték ellenőrzése
8. PDU felszabadítása
9. kapcsolat lezárása

A C nyelv hátrányai közé sorolható a pointerek megléte, amely egy olyan változó fajta ami egy adott memóriarekeszre mutat. Alkalmazása nehezen érhetővé, és bonyolulttá teszi az adott alkalmazást.

A Java nyelv és az SNMP
[14][15]

A Java nyelvhez készült legismertebb SNMP csomag megnevezése az SNMP4J, amely szabadon használható. Támogatja a SNMPv3-at MD5 és SHA autentikációval, a GETBULK MIB lekérdezést, amely segítségével a MIB fa csomópontja alatt lévő MIB-ek lekérdezhetőek, a log-olást a Log4J csomag használatával, és a többszálúságot. Az 1.4.1-es és attól későbbi java verziókkal használható.
A Java programozási nyelv előnye, hogy a C nyelvvel ellentétben nem tartalmaz mutatókat, viszont ellenben vannak más egyéb jellegzetességei (kontruktorok, osztályok, interfészek, öröklődés, stb.). Java program készítéséhez, egy egész más szemléletre van szükség, mint a fentebb említett C nyelv esetében. Fő eltérés, hogy a Java nyelv objektum-orientált, amely objektum-orientáltság a valós világ modellezésén alapul, illetve az osztályok megléte, amelyeket új objektumok definiálására lehet felhasználni. Itt az elsődleges cél nem a műveltek megalkotása, hanem az egymással kapcsolatban álló programegységek kapcsolatainak a megtervezése.

4. A feladat megoldásához felhasznált eszközök

Ebben a fejezetben a megoldás kiválasztásának szempontjait, a feladat megoldásához ténylegesen felhasznált eszközöket, az eszközök használatához szükséges információkat, telepítéseket, beállításokat fogom ismertetni.

4.1 Megoldási lehetőség kiválasztása

Az előző fejezetben felsorolt megoldási lehetőségek közül a saját alkalmazás készítését választom. Oka, hogy a fentebb említett felügyeleti-rendszerek bonyolultak, működésük, felépítésük megértéséhez mélyebb ismeretekre és rengeteg időre van szükség illetve a bővíthetőségük nehézkes. Ahhoz, hogy a fentebb említett rendszerek valamelyikét bővíteni lehessen, programozásra van szükség, egy olyan modult kell készíteni, amely az adott problémát megoldja. Ehhez feltétlenül szükséges az adott rendszer felépítésének tanulmányozása, megértése, amely sok időt igényel. Egy már kész rendszer felépítését számomra sokkal nehezebb megérteni, mint ha készíteni egy saját alkalmazást, amely megvalósítja a feladatot, többek között ezért választom a saját alkalmazás készítését, mint megoldási lehetőséget.

4.2 Programozási nyelv kiválasztása

Az alkalmazás elkészítéséhez a Java programozási nyelvet választom, melyhez létezik nagyméretű SNMP csomag, illetve ez a programozási nyelv áll hozzám közel és a program készítése kapcsán szeretném mélyebben elsajátítani e programozási nyelv rejtelmeit, használatát.

4.3 Fejlesztői környezet kiválasztása

[16][17][18]

Az alkalmazás fejlesztéséhez több fejlesztői környezet is rendelkezése áll melyek segítségével a szoftverfejlesztés során felhasznált programozási nyelven, vagy nyelveken létrehozott forráskódot futtatni illetve tesztelni lehet. A fejlesztői környezet csak a fejlesztést végző személy vagy személyek számítógépén kell, hogy rendelkezésre álljon, a kész program futtatásához nincs rá szükség, ahhoz csak a futtatási környezet szükséges.

Ilyen elterjedtebb szoftver többek között az Eclipse és a NetBeans. Mind a két szoftver nagyon hasonló, de kis eltérések mutatkoznak a használhatóságukban.
A NetBeans nyílt forráskódú, platformfüggetlen szoftver, melyet az Oracle fejlesztett ki. A program grafikus fejlesztőfelületet kínál a különböző alkalmazások, appletek készítéséhez, amelynek segítségével könnyebben, gyorsabban tudjuk fejleszteni saját programjainkat. Előnye, hogy sok funkcióval rendelkezik, illetve van beépített grafikus editora, amely segítséget felhasználói felületek tervezésére, elkészítésére. Hátránya, hogy az általa generált forráskódban nem lehet változtatásokat végezni.
Az Eclpise szintén nyílt forráskódú és platformfüggetlen szoftverrendszer, melyet eredetileg az IBM fejlesztett ki. Szintén sok funkcióval és szolgáltatással rendelkezik, melyek plugin-okba vannak szervezve eltérve a legtöbb fejlesztői környezettől, ahol a funkciók a rendszer forráskódjába vannak építve. Megfelelő plugin-ok telepítésével kiterjeszthető úgy, hogy a Javán kívül más programnyelveket például a C vagy a Perl nyelvet is támogassa.
Előnye, hogy az általa generált forráskód tetszőlegesen módosítható, nem úgy mint a NetBeans esetében, ezért a fejlesztéshez az Eclipse fejlesztői környezetet választom, melyet a következő honlapról töltöttem le: http://www.eclipse.org/downloads/.

4.4 UML modellező kiválasztása

[26][27]

A feladat része nemcsak az alkalmazás implementálása, hanem annak a megtervezése is. Egy jól elkészített alkalmazás terv segítségével az implementáció gyorsabban megvalósulhat meg.
A tervezéshez többféle modellező eszközt is lehet alkalmazni melyek segítségével elkészíthetőek az alkalmazás felépítését szemléltető diagrammok. Erre a legmegfelelőbb modellező nyelv az UML, amely egy általános célú modellező nyelv és segítségével szöveges illetve grafikus modellek is készíthetőek. Az UML több diagramtípust is definiál ilyen az osztálydiagram, a csomagdiagram, az aktivitásdiagram stb.
Az egyik legelterjedtebb UML modellező eszköz a StarUML, amely egy nyílt forráskódú szoftver, támogatja az UML 2.0 diagramtípusait és egy könnyen kezelhető vizuális szerkesztő felületet biztosít a diagramok elkészítéséhez. Segítségével az elkészült diagramok JPEG formátumban ki is exportálhatóak illetve biztosít kódgenerálási funkciót.
A választásom azért erre az UML modellező szoftverre esik, mert ingyenes, könnyű a használata, tartalmaz kódgenerálási funkciót, és a kész diagramok JPEG formátumban

elmenthetőek, amely nagy segítség az adott diagram képként való dokumentumba illesztéséhez.

4.5 A fejlesztéshez használt hardverelemek

[19][20][21]

Ahhoz, hogy a programot el tudjam készíteni és, hogy menetközben tesztelni tudjam az alkalmazás egyes részeit szükséges egy SNMP támogató eszköz. Ezek az eszközök meglehetősen drágák, ezért kölcsön kaptam egy switch-et, amely nagy segítésül szolgál a fejlesztés alatt. Azért switch-et mert, azokat az információkat, hogy mely állomások mely eszköz portokon találhatóak a CAM tábla tartalmazza, ezt pedig a switch állítja elő.

A switch egy olyan hálózati eszköz, ami a rá csatlakoztatott számítógépek közötti hálózati kommunikációt irányítja és az OSI modell 2. rétegében, az adatkapcsolati rétegben helyezkedik el. Működése hasonló a hub-ok működéséhez, de annyival különbözik a hub-októl, hogy a MAC cím vizsgálatával képes megállapítani, hogy az adott keretet mely portjára kell továbbítania, míg a hub az egyik csatlakozóján érkező adatot az összes többi csatlakozójára továbbítja. A switch-ek feladata a hálózati csomagokban lévő MAC címek megállítása, a portok és a MAC címek egymáshoz való társítása, adatütközés elkerülése, illetve típustól függően egyéb szolgáltatásaik is lehetnek, ilyen például az SNMP rendszeren keresztüli port- és eszközfigyelés és egyéb biztonsági beállítások (pl.: port security).

A kölcsön kapott eszköz egy Cisco Catalyst 3550 típusú 24 portos, SNMP-t támogató switch. Az eszköz szolgáltatása igen széles, rendelkezik 24 darab 10/100-as és 2 darab 1000BASE-X porttal, Cisco IOS operációs rendszerrel, biztonsági beállításokkal, felhasználó-, és eszköz autentikácival, ACL (Acces Contor List) utasításlista fogadásával, illetve menedzselhető SNMP-n keresztül és támogatja a Cisco, az RMON és a Bridge MIB-eket is.

Az alábbi képen az eszköz látható:

5. ábra A Cisco Catalyst 3550 típusú switch

Az eszköz konkrét konfigurálásáról, és a konfiguráláshoz szükséges beállításokról majd a későbbi 5. fejezetben írok részletesen.

A switch konfigurálásáról
[22][23]

A konfigurációs parancsokat az adott eszközön futó IOS (Internetworking Operating System) dolgozza fel és hajta végre. Az IOS egy egységes parancssori konfigurációs felületet biztosít több üzemmóddal. Alapvetően kétféle üzemmód van, melyek a következők:
- user EXEC
- Privileged EXEC

A user EXEC a switch alapüzemmódja , segítségével a terminál beállításokat lehet megváltoztatni, rendszerinformációkat lehet lekérdezni, és teszteket lehet futtatni.

A Privileged EXEC üzemmód segítségével van lehetőség a konfigurálásra, ez egy privilegizált üzemmód, a user EXEC mód kibővítése a konfigurációs utasítások végrehajtásának lehetőségével. Az enable utasítás után érhető el, ebbew a módba való belépéshez szükséges a parancssori jelszó megadása.

További üzemmódok:

- Global
- VLAN database
- Interface configuration
- Line configuration

A Global a privilegizált üzemmódba való belépés után érhető el a global utasítás kiadásával, segítségével a switch egészére jellemző utasításokat lehet kiadni.
A VLAN database a szintén a privilegizált üzemmódba való belépés után érhető el, amely segítségével az egyes VLAN-okra vonatkozó beállításokat lehet megadni.
Az Interface configuration üzemmódhoz szintén szükséges a privilegizálrt üzemmódba való belépés, segítségével a switch interfészt lehet konfigurálni.
A Line configuration üzemmód segítségével a terminál paramétereket lehet megváltoztatni, a line vty vagy a line console parancs kiadása után.

Néhány parancs:

Művelet	Parancs
Lekérdezés	show, show config, show running-config, show startup-config, show vlan
Csatlakozás más eszközhöz	telnet <a másik eszköz hálózati címe>
Konfigurálás	Configure, configure terminal, configure network

Egy hálózatba csatlakoztatott új switch esetén a következő teendőket fontos megtenni:

- A switch nevének beállítása
- A felügyeleti célokra használt IP cím beállítása
- Alapértelmezett átjáró megadása
- Virtuális terminál interfész felügyeleti hozzáférésének beállítása
- Biztonsági beállítások megadása
- Amennyiben szükséges a switch portjaihoz tartozó hozzáférés beállítása

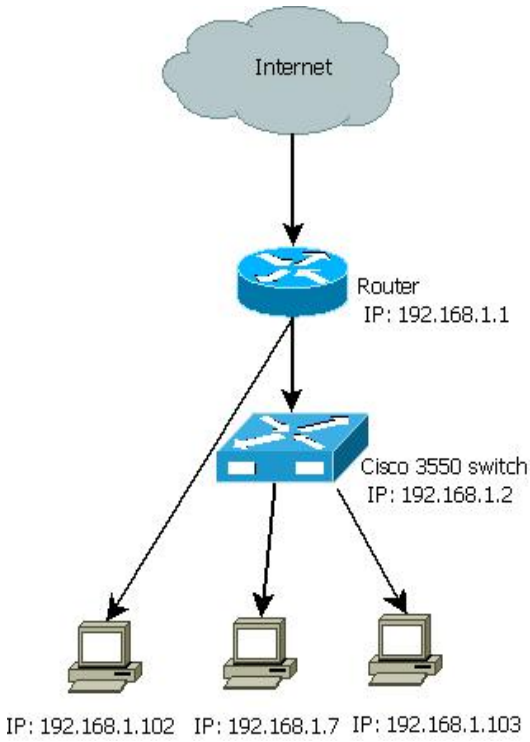
5. A fejlesztés előkészületei

Ebben a fejezetben a fejlesztéshez szükséges előkészületeket ismertetem. A fejlesztés megkezdése előtt szükséges a hálózat összeállítása és a megfelelő eszközök konfigurálása, mert kell, hogy legyen egy teszthálózat melyet felügyelni és melyben állomásokat kitiltani, engedélyezni lehet.

5.1 A hálózat összeállítása

Összeállítottam a kölcsön kapott switch segítségével egy teszthálózatot, amellyel nyomon követhetem az alkalmazás egyes részeinek a működését majd a végleges alkalmazást is tesztelhetem.

A következő 6. ábra ezen hálózat felépítését szemlélteti:



6. ábra A felépített teszthálózat

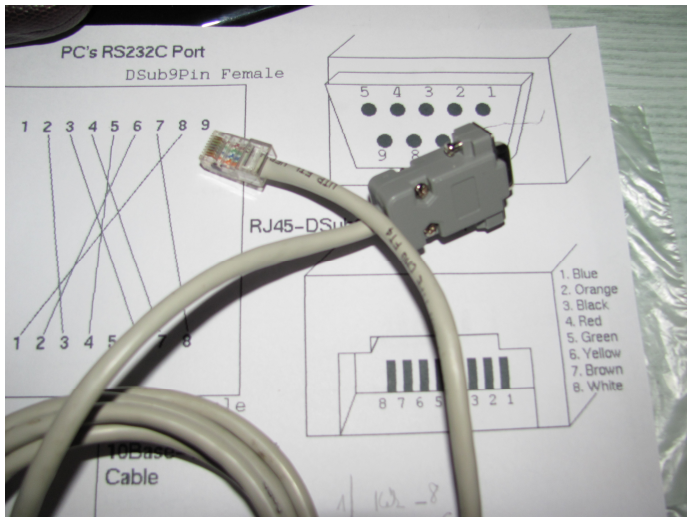
A hálózatban a forgalomirányítást a 192.168.1.1 IP című router végzi, melyre csatlakozik a Cisco 3550 switch illetve a 192.168.1.102 IP című állomás. A router az IP címeket DHCP segítségével osztja ki, melynél beállítottam, hogy az adott MAC című állomások mindig a hozzájuk rendelt IP címet kapják. Szándékosan van az egyik állomás közvetlenül a routerre, a másik kettő pedig a switchre csatlakoztatva. Az elrendezés célja, hogy

tanulmányozhassam a CAM táblában szereplő információkat a hálózat switchen kívüli részéről.

5.2 A switch konfigurálása

A fentebb lévő ábrán szereplő hálózat működéséhez szükséges, hogy a switch megfelelő módon legyen konfigurálva. Az eszköz konfigurálásához szükség van egy konzol kábelre, és egy terminál emulációs szoftverre. Én a Windows XP-ben megtalálható Hyper Terminal-t használtam, mint terminál emulációs szoftvert, amely a Start menü->Kellékek->Kommunikáció->Hyper Terminal helyen indítható el. A felületén kiadott parancsok a konzol kábelen keresztül jutnak el a switch-hez, amely kábel egyik vége a számítógép RS-232 portjára, míg a másik vége a switch console portjára csatlakozik. Ilyen kábel nem állt rendelkezésemre (korábban még nem foglalkoztam ezen eszközök konfigurálásával) így az Interneten megtalálható információk alapján készítettem egyet. A kábel bekötési rajza nem túl bonyolult, nem tartalmaz semmilyen áramköri alkatrészt, így az elkésztése nem igényel nagyobb elektronikai ismeretet.

Az elkészült konzol kábel a következő 7. ábrán látható:



7. ábra Az elkészült konzol kábel

A kábel elkészítése után következhet a tényleges konfiguráció, melyhez elő kell készíteni a HyperTerminal-t.

Ehhez a következő beállításokat kell megadnia Hyper Terminal ablakában a [24]-es számú forrás alapján:

- Bit/másodperc: 9600

- Adatbitek: 8
- Paritás: nincs
- Stopbitek: 1
- Átvitelvezérlés: nincs

A fentebb említett beállítások megvalósítása után következik a tényleges konfiguráció, melyhez a [22] és [23] forrásokat használtam segítségül. A konfigurálás megkezdéséhez szükséges a HyperTerminal elindítása.

Eszköznév beállítása

Első lépésben a switch nevét kell beállítani. Ehhez a privilegizált üzemmódba kell lépni az enbale paranccsal, majd meg kell adni a parancssori jelszót. Ez után a configure terminal utasítás kiadása után van lehetőség az eszköz konfigurálására, a nevét a hostname <eszköznév> utasítás segítségével van lehetőség megadni. Az alábbi sorok a név beállítását szemléltetik:

```
Switch>enable
Switch# configure terminal
Switch(config)# hostname Cisco3550
Cisco3550(config)# exit
```

VLAN beállítása

Az éppen aktuális vlan beállításokat a show vlan utasítás segítségével lehet lekérdezni, melynek kimenete a következő:

```
VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                 Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                 Fa0/13, Fa0/14, Fa0/15, Fa0/16
                                                 Fa0/17, Fa0/18, Fa0/19, Fa0/20
                                                 Fa0/21, Fa0/22, Fa0/23, Fa0/24
                                                 Gi0/1, Gi0/2
2    VLAN2                             active
3    VNET                              active
4    GNET                              active
1002 fddi-default                      active
1003 token-ring-default                active
1004 fddinet-default                   active
1005 trnet-default                     active

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
```

```
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1    enet  100001     1500  -      -      -        -    -        0      0
2    enet  100002     1500  -      -      -        -    -        0      0
3    enet  100003     1500  -      -      -        -    -        0      0
4    enet  100004     1500  -      -      -        -    -        0      0
1002 fddi  101002     1500  -      -      -        -    -        0      0
1003 tr    101003     1500  -      -      -        -    -        0      0
1004 fdnet 101004     1500  -      -      -        ieee -        0      0
1005 trnet 101005     1500  -      -      -        ibm  -        0      0

Remote SPAN VLANs
------------------------------------------------------------------------------

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
```

A vlan beállítási információk alapján az összes port az alapértelmezett VLAN1-ba tartozik. Ez így meg is felel, de szükséges hozzá felügyeleti IP cím és alapértelmezett átjáró, rendelése.

A VLAN1-hez felügyeleti IP cím rendelése

Erre azért van szükség, hogy az eszköz menedzselhető legyen az adott VLAN-ból. Ehhez be kell lépni a konfigurációs módba a configure terminal utasítással, majd ezt követően van lehetőség a VLAN-ok beállítására.

A következő sorok a beállítás menetét szemléltetik.

```
Cisco3550(config)#int vlan 1
Cisco3550(config-if)#ip address 192.168.1.2 255.255.255.0
Cisco3550(config-if)# exit
```

A VLAN1-hez alapértelmezett átjáró rendelése

Az alapértelmezett átjáró jelen esetben a 192.168.1.1 IP című router**.** A beállíátshoz szintén szükség van a konfigurációs módba való belépéshez a configure terminal parancs segítségével.

A következő sorok a beállítás menetét szemléltetik.

```
Cisco3550(config)# ip default-gateway 192.168.1.1
Cisco3550(config-if)# exit
```

Virtuális terminálhoz jelszó rendelése

Ehhez azért van szükség, hogy az eszköz web-es felületén be lehessen jelentkezni. A következő utasítások a beállítást szemléltetik.

```
Cisco3550(config)# line vty 0 15
Cisco3550 (config-line)# password Cisco3550
Cisco3550 (config-line)# login
```

Az SNMP beállítása

Következő lépés az SNMP beállítása. A [25] szerint az eszközön alapértelmezett engedélyezve van az SNMP. További teendő beállítani az SNMP változók engedélyét, szükség, hogy olvasni illetve írni is lehessen az SNMP változókat. A beállításhoz szükséges a konfigurációs módba való belépés a configure terminal utasítással. Ennek a beállítását a következő utasítások szemléltetik.

```
Cisco3550# configure terminal
Cisco3550(config)# snmp-server community public RW
Cisco3550(config) # snmp-server community private RW
```

A beállítások mentése

Ahhoz, hogy áramszünet, vagy kikapcsolás esetén se vesszenek el a végrehajtott beállítások, ehhez szükséges menteni őket.

A folyamatot a következő sorok szemléltetik:

```
Cisco3550#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
```

A konfiguráció ellenőrzése

Ezt a show runnig-config paranccsal lehet lekérdezni privilegizált üzemmódban.

6. Az alkalmazás tervezése

Ebben a fejezetben az alkalmazás tervezésének menetét és a tervezés során készült diagramokat dokumentálom.

Az alkalmazás célja, hogy segítségével felügyelhető legyen egy SNMP alapú számítógép-hálózat, illetve egy hálózatban lévő állomás helye meghatározható legyen és, hogy egyéb funkciókat például az állomás kitiltását megvalósítsa és ehhez egy kényelmesen kezelhető grafikus felhasználói felütet biztosítson.
Mivel az alkalmazást Java nyelven készítem így nagy valószínűséggel több osztályból, csomagból fog állni, melyeket az átláthatóság érdekében célszerű alrendszerekbe foglalni. Az alrendszerekbe foglalást csomagok segítésével fogom megvalósítani, ahol minden egyes csomag egy alrendszert fog jelölni.

Elvárások az alkalmazással szemben

- bejelentkezési felület biztosítása
- a hálózatban lévő állmások megjelenítése
- a hálózat figyelése és a gyűjtött információk alapján az alkalmazás adatinak frissítése
- adott állomás/állomások kitiltása a hálózatból
- kitiltott állomás/állomások engedélyezése
- a kitiltás/engedélyezés naplózása
- folyamatos hálózatfigyelés
- mindehhez grafikus felhasználói felület biztosítása

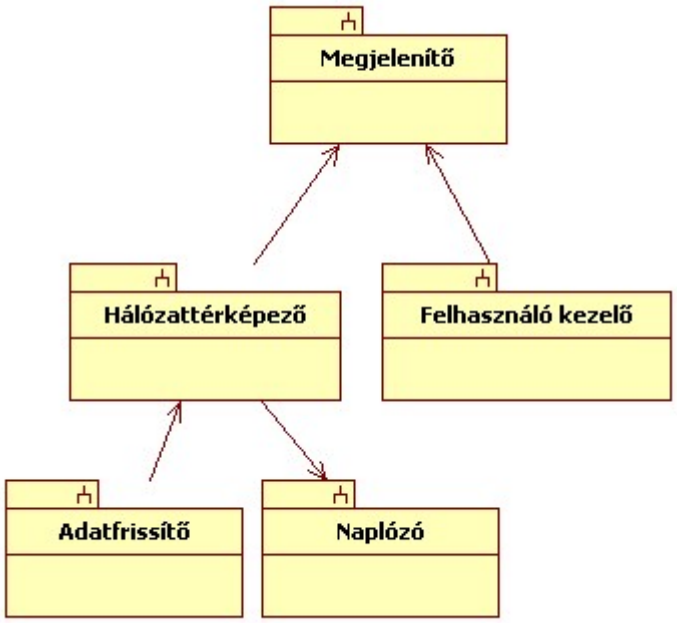
Az említett elvárásokból következtetés a szükséges alrendszerekre:

- Megjelenítései alrendszer
    - A hálózatban lévő állomások megjelenítése
    - Grafikus felhasználói felület biztosítása
- Hálózattérképező alrendszer
    - a hálózatban lévő állmások megjelenítéséhez szükséges adatok előállítása
    - folyamatos hálózatfigyelés

- Adatfrissítési alrendszer
    - a hálózat figyelése és a gyűjtött információk alapján az alkalmazás adatinak frissítése
- Naplózási alrendszer
    - a kitiltás/engedélyezés naplózása
- Felhasználó kezelési alrendszer
    - Bejelentkezési felület biztosítása

6.1 A rendszer felépítésének kezdeti terve

A következő 9. ábra a rendszert alkotó alrendszereket és azok kapcsolatát szemlélteti:



8. ábra A rendszer alrendszerei és azok kapcsolata

Megjelenítési alrendszer

Célja az adatok és a rendszerfunkciók grafikus felhasználói felületen való megjelenítése. Az adatokat a vele kapcsolatban álló alrendszerek továbbítják majd számára.

Hálózattérképező alrendszer

Célja, hogy feltérképezze az adott hálózati eszközön lévő állomásokat. Ezt majd az eszköz által előállított CAM tábla segítségével fogja megvalósítani, az általa előállított információk alapján dolgozik majd az egész alkalmazás.

Felhasználó kezelő alrendszer

Azért van rá szükség, mert fontos az, hogy ne tudjon akárki hálózatban lévő állomásokat kitiltani. Ez majd egy bejelentkezési felület segítségével fog megvalósulni, amelyen keresztül lehet majd megadni a felhasználói nevet és a hozzá tartozó jelszót. Ezen alrendszer kezeli majd a felhasználói információkat tartalmazó állományt.

Naplózó alrendszer

Célja, hogy naplózza a rendszerben lezajló tevékenységeket, eseményeket. A naplózást és a naplózott adatok állományba való tárolását fogja majd megvalósítani.

Adatfrissítő alrendszer

Célja a hálózattérképező alrendszer által gyűjtött adatok frissítése. Azért szükséges, mert a hálózatban lévő állomások száma folyamatosan változhat és célszerű a változásokat bizonyos időközönként megjeleníteni a felhasználói felületen.

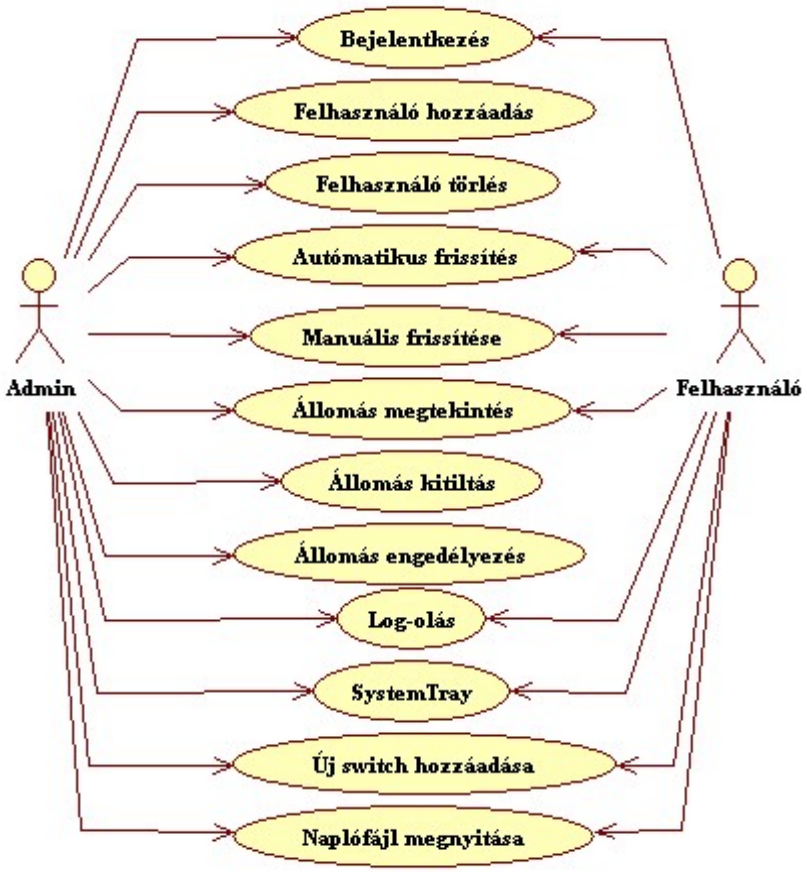
Az alrendszerek közvetett vagy közvetlen módon a megjelenítési alrendszernek szállítanak információkat, melyeket az majd megjelenít. A hálózattérképező alrendszer a hálózatban lévő gépekről, a felhasználó kezelő alrendszer a rendszerben lévő felhasználókkal kapcsolatos információkat továbbítja. Az adatfrissítő alrendszer a hálózattérképező alrendszer számára a hálózatban végbemenő változásokról továbbít információt, a naplózó alrendszer pedig a hálózattérképező alrendszertől kapott információkat tárolja, naplózza.

6.2 A rendszer funkciói

Az alkalmazás funkcióit use case diagram segítségével szemléltetem melynek célja, hogy a szoftverfejlesztés első fázisában rögzítse a szoftverrel szemben támasztott követelményeket illetve, hogy kompletten leírja a rendszer funkcióit. A diagram tulajdonsága, hogy szemléletes és könnyen áttekinthető, elemei pedig az aktorok, a use case-ek, és a relációk. Az aktorok egy szerepkört reprezentálnak, tulajdonképpen azok a felhasználók, akik a rendszert használják. A use case-ek viselkedési minták, magát a funkciót jelölik, a relációk pedig a use case-ek és az aktorok közötti kapcsolatokat írják le. A funkciók magyarázatában szereplő prekondíciók azon feltételek felsorolását jelentik, melyeknek teljesülniük kell ahhoz, hogy a use case által jelzett tevékenység elkezdődjön, a

post kondíciók pedig azoknak a leírása, hogy milyen állapot következik be a use case végén. A szokásos működés azon eseményeket írja le, melyek a use case „szokásos” körülmények közötti működését jellemzik. A kivételes esetek olyan lehetséges eseményeket jelölnek, amelyek váratlanul következhetnek be. [28][29]

Az elkészült use case diagramot a következő 9. ábra szemlélteti.



9. ábra Az rendszer funkciói

Aktorok

Admin: a rendszer adminisztrátora, tulajdonképpen ő is felhasználó csak több joggal rendelkezik. Lehetősége van a rendszerhez felhasználókat hozzáadni, törölni illetve neki van lehetősége állomásokat kitiltani, engedélyezni a hálózatban.

Felhasználó: kevesebb jogosultsággal rendelkezik, mint az adminisztrátor. Az átlag felhasználónak nincs lehetősége újabb felhasználókat hozzáadni vagy törölni a rendszerből illetve az állomásokat sem tudja kitiltani, engedélyezni csak megnézheti, hogy állomások vannak a hálózatban.

Use case-ek

Bejelentkezés: célja, hogy megvalósítsa a rendszerbe való bejelentkezést ezzel különbséget téve az admin és az átlagfelhasználó között. Ezen a bejelentkező felületen van lehetőség az SNMP-t támogató eszköz IP címének a megadására is.

- Prekondíció: feltétele a program elindítása
- Post kondíció: a bejelentkezett felhasználó/admin eléri az adott funkciókat

Szokásos működés: a felhasználói név, jelszó és az eszköz IP címének a megadása után jelentkezik be a programba

- Kivételes esetek: hibás felhasználói név, hibás jelszó vagy hibás IP cím megadása esetén a program előugró ablak formájában tájékoztatja a felhasználót a keletkezett hibáról.

Felhasználó hozzáadás: ezen funkció segítéségével van lehetőség újabb felhasználókat hozzáadni a rendszerhez.

- Prekondíció: feltétele a programba adminként való bejelentkezés
- Post kondíció: az admin újabb felhasználókat adhat hozzá a rendszerhez

Szokásos működés: adminként való bejelentkezés után egy menüpontban elérve ezt a funkciót lehetőség van a felhasználói név és a hozzá tartozó jelszó megadásával hozzáadni az adott felhasználót a rendszerhez.

- Kivételes esetek: ha a felhasználói információkat tartalmazó fájl sérült vagy nem elérhető, akkor erről egy felugró ablak tájékoztatja a felhasználót.

Felhasználó törlés: ezen funkció segítéségével van lehetőség felhasználókat törölni a rendszerből

- Prekondíció: feltétele a programba adminként való bejelentkezés
- Post kondíció: az admin a megadott felhasználót törölheti a rendszerből
- Szokásos működés: adminként való bejelentkezés után egy menüpontban elérve ezt a funkciót lehetőség van a felhasználói név megadásával törölni az adott felhasználót. Ha z adott felhasználó több jelszóval is rendelkezik, akkor a hozzá tartozó összes jelszó is törlődik.

- Kivételes esetek: ha a felhasználói információkat tartalmazó fájl sérült, nem elérhető, vagy nem létezik a törölni kívánt felhasználó akkor erről a program egy felugró ablak tájékoztat.

Automatikus frissítés: ezen funkció segítéségével van lehetőség a hálózatban bekövetkező változások megjelenítésére. Ha egy állomás eltűnik vagy éppen megjelenik a hálózatban akkor ezen funkció bekapcsolása után a változás a rendszerben a megadott időközönként látszódni fog.

- Prekondíció: feltétele a programba való bejelentkezés
- Post kondíció: a hálózatban bekövetkező változások a megadott időközönként látszódni fognak a rendszerben
- Szokásos működés: egy menüpontból elérve ezt a funkciót van lehetőség megadni a frissítés végrehajtásának időközét. Ha megad a felhasználó 1 másodperces időközt, akkor a változások másodpercenként fognak megjelenni a programban. Ezt a funkciót a program elindítása egyszer van lehetőség bekapcsolni, és onnantól kezdve a program bezárásáig működni fog illetve letiltódik a manuális frissítési funkció.
- Kivételes esetek: ha az adatok valamilyen hiba folytán nem frissíthetőek, akkor azt egy előugró ablak formájában közli a program.

Manuális frissítés: ezen funkció segítéségével van lehetőség a hálózatban bekövetkező változások adott gomb megnyomása után történő megjelenítésére. Ha egy állomás eltűnik vagy éppen megjelenik a hálózatban akkor a funkció megvalósítását tartalmazó gomb megnyomása után a változás a rendszerben a látszódni fog.

- Prekondíció: feltétele a programba való bejelentkezés
- Post kondíció: a hálózatban bekövetkező változások az adott gomb megnyomása után látszódni fognak a rendszerben
- Szokásos működés: egy gomb segítségével elérve ezt a funkciót van lehetőség frissíteni a változásokat. Ez a funkció nem elérhető ha az automatikus frissítés be van kapcsolva .
- Kivételes esetek: ha az adatok valamilyen hiba folytán nem frissíthetőek, akkor azt egy előugró ablak formájában közli a program.

Állomás megtekintés: ezen funkció segítéségével van lehetőség a hálózatban lévő állomások megtekintésére. Az állomások adatai egy táblázatban szerepelnek.

- Prekondíció: feltétele a programba való bejelentkezés
- Post kondíció: a hálózatban lévő állomások adatai megjelennek egy táblázatban
- Szokásos működés: a programba való bejelentkezés után a táblázat automatikusan feltöltődik a hálózatban lévő állomások adataival.
- Kivételes esetek: ha az adatok táblázatba való feltöltése során hiba történt, akkor azt egy előugró ablak formájában közli a program.

Állomás kitiltás: segítségével van lehetőség a táblázatban kijelölt állomás kitiltására

- Prekondíció: feltétele a programba adminként való bejelentkezés
- Post kondíció: a kijelölt kitiltandó állomás adatai megjelennek a táblázat alatt szereplő rublikákban.
- Szokásos működés: a táblázatban kijelölt állomás adatai kiíródnak a táblázat alatt lévő rublikákba, majd a kitiltás gomb megnyomása után az adott állomás kitiltódik a hálózatból, és a táblázat adatainak frissítése után el is tűnik a táblázatból, majd hozzáadódik a kitiltott gépeket tartalmazó táblázathoz.
- Kivételes esetek: ha a kitiltandó állomás kitiltása nem sikerült, akkor erről tájékoztat a program.

Állomás engedélyezés: segítségével van lehetőség a kitiltott állomás engedélyezésére

- Prekondíció: feltétele a programba adminként való bejelentkezés és hogy, az adott állomás ki legyen tiltva
- Post kondíció: a kitiltott állomásokat tartalmazó táblázatban kijelölt állomás engedélyeződik, és így újra csatlakozhat a hálózathoz
- Szokásos működés: a kitiltott állomásokat tartalmazó táblázatban a kijelölt állomás az engedélyező gomb megnyomása után engedélyeződik és így újra kapcsolódhat a hálózathoz
- Kivételes esetek: ha a kitiltott állomás engedélyezése nem sikerült, akkor erről tájékoztat a program.

Log-olás: segítségével valósul meg az állomás kitiltás vagy engedélyezés naplózása

- Prekondíció: feltétele a programba adminként való bejelentkezés
- Post kondíció: a kitiltás vagy engedélyezés hatására a kitiltott vagy engedélyezett állomás adatai a naplófájlba íródnak
- Szokásos működés: az állomások adatait tartalmazó táblázatban a kijelölt gép adatai illetve a végbement esemény (kitiltás vagy engedélyezés) a kitiltás vagy engedélyezés gomb megnyomására a naplófájlba íródnak
- Kivételes esetek: ha a naplófájl létrehozása vagy az adatok naplófájlhoz írása nem sikerült, akkor erről tájékoztat a program

System tray: segítségével a program az „x” gombra való kattintás után tálcán lévő óra mellé helyeződik, majd onnan újra megnyitható. Erre azért van szükség, hogy az alkalmazás folyamatosan figyelje a számítógép-hálózatot

- Prekondíció: feltétele a programba való bejelentkezés
- Post kondíció: a program a tálcán lévő óra mellé helyeződik, majd szükség esetén újra megnyílik
- Szokásos működés: a program ablakában lévő „x” gombra való kattintás után, nem lép ki az alkalmazásból, hanem egy ikon formájában a tálcán lévő óra mellé helyeződik, majd szükség esetén az ikonra való dupla kattintás esetén újra megnyílik.
- Kivételes esetek: ha az operációs rendszer nem támogatja a system tray funkciót, akkor ezt a program közli a felhasználóval

Új switch hozzáadása: ezen funkció segítéségével van lehetőség a bejelentkezéskor megadott switch-en kívül, új switch-eket megadni.

- Prekondíció: feltétele a programba való bejelentkezés
- Post kondíció: az újonnan hozzáadott switch/switch-eken lévő állomások adatai hozzáadódnak programhoz
- Szokásos működés: egy menüpontból elérve ezt a funkcióit van lehetőség megadni a felügyelni kíván újabb switch-ek IP címét
- Kivételes esetek: ha az adott switch nem elérhető vagy a megadott IP cím formátuma hibás, akkor ezt közli a program a felhasználóval

Naplófájl megnyitása: segítségével a rendszerben létrejött naplófájl tartalma nyitható meg

- Prekondíció: feltétele a programba való bejelentkezés és a naplófájl létezése
- Post kondíció: a létező naplófájl tartalma megnyílik az alapértelmezett böngészőben
- Szokásos működés: egy menüpont segítségével a naplófájl tartalma kiíródik egy webes böngészőben
- Kivételes esetek: ha az adott naplófájl nem létezik, nincs a helyén, vagy hibás akkor ezt jelzi a program a felhasználó számára

6.3 A rendszer részletesebb felépítése

Ebben az alfejezetben a mellékletben található 1.ábra segítségével magyarázom az egyes alrendszerekben lévő osztályok működését, célját.

Megjelenítési alrendszer

Az alrendszer az ábrán lévő osztályokat tartalmazza, melyek célja, hogy előállítsák a program felhasználói felületét és megvalósítsák az ahhoz tartozó funkciókat.

A Login osztály célja, hogy előállítsa a bejelentkezési felületet és a hozzá kapcsolódó funkciókat és az ellenőrzött adatbevitelt.

A Foablak osztály célja, hogy előállítsa a program főablakát és az abban lévő felhasználói felületi elemeket és a hozzá tartozó funkciókat.

A Felhasznalo_hozzaad osztály célja, hogy megvalósítsa a felhasználói felület azon részét, amely segítségével újabb felhasználókat van lehetőség a rendszerhez adni, ezt egy úgynevezett InternalFrame (belső ablak) segítségével valósítja meg.

A Felhasznalo_torol osztály célja a rendszerben szereplő felhasználók törléséhez felhasználói felület biztosítása és a törlési funkciók megvalósítása, ezt egy InternalFrame segítségével valósítja meg.

Az Uj_eszkoz osztály célja, hogy segítségével a hálózatban szereplő újabb SNMP-re képes switch-eket lehessen a programhoz hozzáadni felügyeleti célból, ezt egy InternalFrame segítségével valósítja meg.

Az Adatfrissit osztály valósítja meg a program főablakában lévő, a felügyelt hálózatról információkat tartalmazó komponens frissítését. Ez azt jelenti, hogy ha a felügyelt

hálózatban valamilyen változás következik be (pl. állomás eltűni, vagy csatlakozik), akkor ez a változás megjelenik a programban is.
A Kitiltott_gepek osztály valósítja meg a kitiltott gépek újabb ablakban való megjelenítését és engedélyezését egy InternalFrame segítségével.

A hálózattérképező alrendszer
Az alrendszer célja, hogy előállítsa a program működéséhez szükséges adatokat. Ez valójában a CAM tábla előállítását jelenti az adott hálózati switch-re.

A Cam_tabla osztály valósítja meg a CAM táblában szereplő egyes állomások adattagjainak az előállítását.
A Cam_tabla_tarol osztály célja, hogy a Cam_tabla osztály segítéségével megvalósuljon a tábla adatainak a tárolása, illetve a tároláshoz szükséges metódusokat tartalmazza.
A Cam_tabla_eloallit osztály valósítja meg a CAM tábla tényleges előállítását, az előállításhoz szükséges metódusokat és az adott switch port letiltását illetve engedélyezését.

Felhasználó kezelő alrendszer
Az alrendszer célja, hogy megvalósítsa a rendszerbe való bejelentkezéshez szükséges felhasználói adatok tárolását.

A Felhasznalo osztály célja, hogy biztosítsa az egyes felhasználók tárolásához szükséges adattagokat.
A Felhasznalo_tarol osztály valósítja meg a felhasználók adatainak adatszerkezetben való tárolását és a tároláshoz szükséges metódusokat.

Adatfrissítő alrendszer
Célja hogy megvalósítsa az Utemezo osztály segítségével az automatikus frissítési funkciót, vagyis azt, hogy a megadott időközönként automatikusan újrarajzolódjanak a program által figyelt switch-hez tartozó adatok.

Naplózó alrendszer

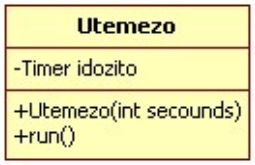
Célja, hogy megvalósítsa az egyes funkciók végrehajtáskor bekövetkezett események (pl. az állomás kitiltás vagy engedélyezés) naplózását.

A HTMLFormatter osztály célja, hogy megadja a naplózott adatok fájlba írásához szükséges HTML dokumentum felépítését.
A sajatLogger osztály valósítja meg a naplózott adatok HTML dokumentumba való írását.

6.4. Az alrendszerek osztálydiagramjai

Ebben az alfejezetben az egyes alrendszerek osztályait szemléltetem osztálydiagram segítségével és részletezem az egyes osztályokban lévő metódusok működését, célját.

Az adatfrissítő alrendszer osztálydiagramja



10.ábra Az adatfrissítő alrendszer osztálydiagramja

Ez az alrendszer egyetlen osztályból áll, amely egyetlen adattaggal az idozito-vel rendelkezik. Az osztály konstruktorában megadott adat a frissítési időközt jelöli másodpercben. Az run() metódus a törzsében megadott kódot az osztály konstruktorában megadott időközönként lefuttatja.
A run() metódus törzsében megadott kód a program főablakában lévő adatokat tartalmazó táblázatot frissíti. Ezt úgy valósítja meg, hogy lekérdezi a táblázat frissítése előtt és a frissítés után is a táblázat sorainak a számát és e függvényében hajt végre utasításokat. Ha az éppen kijelölt sor indexe a frissítés után érvénytelen (azaz -1), akkor az adott sor már nincs a táblázatban, ez esetben törlődik a sor kijelölése, így nem lesz olyan sor kijelölve, amely már nem is létezik. Ha a táblázat frissítése után a még a frissítés előtt kijelölt sor indexe nem érvénytelen (nem -1) lesz, akkor az adott sor a frissítés után is a táblázatban maradt így a kijelölést vissza lehet rá állítani. Ez megoldja azt a problémát, hogy a táblázat

frissítése után a még a frissítés előtt kijelölt soron a kijelölés meg fog maradni, nem fog eltűnni.

A felhasználó kezelő alrendszer osztálydiagramja

11.ábra A felhasználó kezelő alrendszer osztálydiagramja

Az alrendszert két osztály, a Felhasznalo és a Felhasznalo_tarol alkotják.
A Felhasznalo osztály két adattaggal, a felhasználó névvel és a felhasználói névhez tartozó jelszóval rendelkezik.

Az osztály metódusai:

- getFelhNev(): visszaadja a felhasználói név adattagot
- getFelhJelszo():visszaadja a felhasználói jelszó adattagot
- toString(): automatikusan String-é konvertálva visszaadja a felhasználói nevet és jelszót felhasználói név: megadott név jelszó: megadott jelszó alakban.
- hashCode(): az objektum hash kódját állítja elő
- equals(Object obj): akkor ad vissza igazat, hogy ha két felhasználó objektum megegyezik, egyébként hamisat ad
- compareTo(felhasznalo felhasznaloMasik): a paraméterként megadott objektummal összehasonlítja az aktuális objektumot, és annak a függvényében add vissza 1-et, -

1-et, vagy 0-át hogy megegyezik e a két objektum. A felhasználók név szerinti rendezéséhez szükséges.

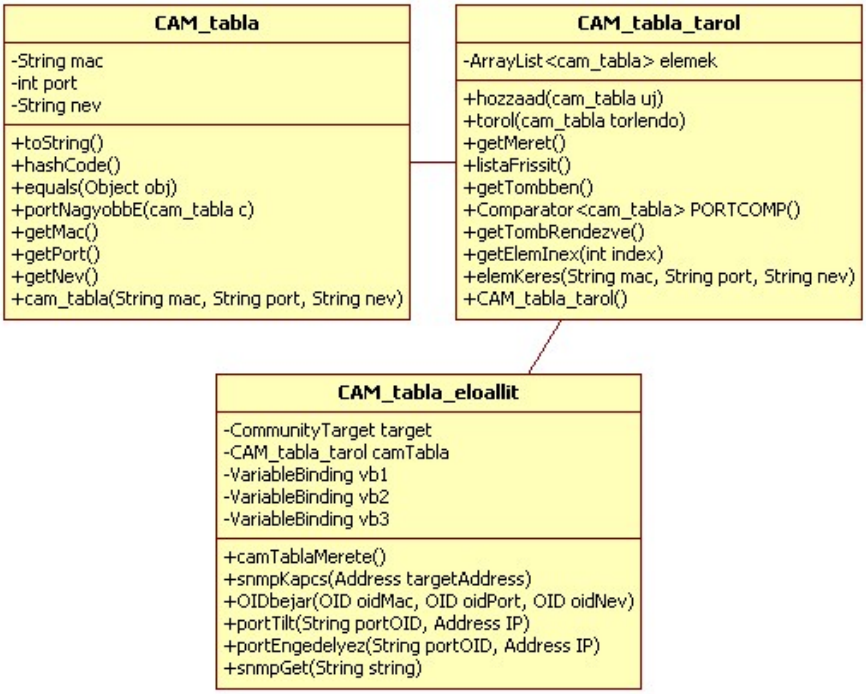
- felhasznalo(String nev, String jelszo): konstruktor

A Felhasznalo_tarol osztály egyetlen adataggal, a tároláshoz szükséges adatszerkezettel rendelkezik.

Metódusai:

- felhasznaloTarol(): konstruktor
- hozzaad(felhasznalo uj): az ArrayList-hez hozzáadja a paraméterként megadott felhasználót
- torol(felhasznalo torlendo): az ArrayList-ből törli a paraméterként megadott felhasználót
- meret(): visszadja az ArrayList méretét
- getTombben(): visszadja az ArrayList elemeit tömbben
- getTombRendezve(): visszadja az ArrayList elemeit tömbben felhasználói név szerint rendezve
- getFelhasznalo(int index): vissadja a pereméterben megadott indexű felhasználót
- elemKeres(String nev, String jelszo): visszadja a paraméterben megadott adatokkal rendelkező felhasználót
- elemKeresVane(String nev, String jelszo): visszaadja, hogy létezik-e a paraméterben megadott adatú felhasználó

A hálózattérképező alrendszer osztálydiagramja



12.ábra A hálózattérképező alrendszer osztálydiagramja

Az alrendszert három osztály a CAM_tabla, a CAM_tabla_tarol és a CAM_tabla_eloallit alkotja. A CAM_tabla osztály három adattaggal, a CAM táblában szereplő adatokkal, az állomás MAC címével, a switch portjával és a switch nevével rendelkezik, a CAM_tabla_tarol osztály, pedig az adatok tárolásához szükséges adattaggal, az elemek-kel rendelkezik.

A CAM_tabla osztály metódusai:

- cam_tabla(String mac, String port, String nev): kontruktor
- toString(): automatikusan String-é konvertálja az osztály adattagjait MAC cím port név alakban
- hashCode() : az objektum hash kódját állítja elő
- equals(Object obj): akkor ad vissza igazat, hogy ha két CAM_tabla objektum megegyezik, egyébként hamisat ad
- portNagyobbE(cam_tabla c): ha az aktuális példány port száma a nagyobb mint a paraméterként kapott példányé, akkor 1-et, ha fordítva igaz, akkor -1-et, egyébként 0-át ad vissza
- getMac(): visszadja az osztály mac adattagját
- getPort(): visszadja az osztály port adattagját

- getNev(): visszadja az osztály nev adattagját

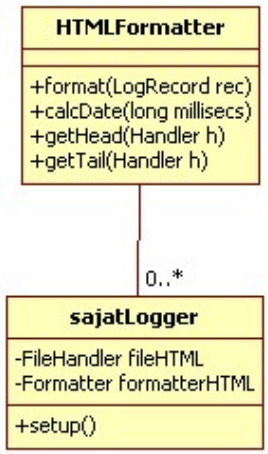
A CAM_tabla_tarol osztály metódusai:

- CAM_tabla_tarol(): konstruktor
- hozzaad(cam_tabla uj): uj elem listához való hozzáadását valósítja meg
- torol(cam_tabla torlendo): a torlendo elem listából való törlését hajtja végre
- getMeret(): a lista méretét adja vissza
- listaFrissit(): igazából a lista összes elemét törli
- getTombben(): a lista adatait adja vissza tömbben
- Comparator<cam_tabla> PORTCOMP: a port szerinti rendezéshez szükséges komparátor
- getTombRendezve(): a lista adatait adja vissza tömbben, de rendezve
- getElemInex(int index)： visszaadja a pereméterként megadott indexű elemet
- elemKeres(String mac, String port, String nev): visszaadja, hogy a paraméterként megkapott adatokkal rendelkező elem hol van listában, ha nincs benne, akkor null-t ad vissza

A CAM_tabla_eloallit osztály metódusai:

- camTablaMerete(): a CAM tábla méretét adja vissza
- snmpKapcs(Address targetAddress): a switch-hez való kapcsolódást és a szükséges adatok switch-ből való SNMP-n keresztüli lekérdezését valósítja meg
- OIDbejar(OID oidMac, OID oidPort, OID oidNev): a paraméterként megkapott oid-ket járja be GETBULK mechanizmus segítségével
- portTilt(String portOID, Address IP): letiltja a paraméterként megkapott switch portot a szintén paraméterként megkapott IP címen
- portEngedelyez(String portOID, Address IP): engedélyezi a paraméterként megkapott switch portot a szintén paraméterként megkapott IP címen
- snmpGet(String string): visszaadja a paraméterként String-ben megkapott SNMP változó értékét

A naplózó alrendszer osztálydiagramja



13.ábra A naplózó alrendszer osztálydiagramja

Az alrendszer két osztályból, a HTMLFormatter és a sajatLogger-ből áll. A sajatLogger osztály két adattaggal, a naplózott adatok HTML fájlba írásához szükséges fájlkezelővel és a HTML dokumentum felépítését megadó formatter-rel rendelkezik. A HTML fájl a naplózott adatokat egy táblázatban jeleníti meg, a naplózás dátumának és a naplózott esemény adatainak segítségével.

A HTMLFormatter osztály metódusai:

- format(LogRecord rec): a táblázat előállításához szükséges HTML tag-eket állítja elő
- String calcDate(long millisecs): a táblázatban szereplő naplózás dátumának formátumát állítja be
- getHead(Handler h): a HTML dokumntum fejrészét (head) adja vissza String-ben
- getTail(Handler h) : a HTML dokumnetum záró karaktereit adja állítja elő

A sajatLogger osztály metódusai:

- setup(): a log fájl létrehozásához és a fájl szerkezetének beállításához szükséges kódot tartalmazza

A megjelenítő alrendszer osztálydiagramja

Az alrendszer osztálydiagramját terjedelmes mérete miatt az 1. melléklet 2. ábrája tartalmazza, melyen látható, hogy az alrendszer 7 osztályból áll, melyek célja, hogy megvalósítsák a grafikus felhasználói felületet és a felületen lévő elemekhez funkciókat társítsanak.

Az alrendszert a Login, a Foablak, a Felhasznalo_torol, a Felhasznalo_hozzaad, az Adatfrissit, a Kitiltott_gepek és az Uj_eszkoz osztályok alkotják.

A Login osztály metódusai:

- `adminE()`: azt adja vissza, hogy a felhasználó admin-knét van-e bejelentkezve a rendszerbe, és attól függően tilt le egyes funkciókat a jogosultság korlátozás miatt
- `getEszkozIP()`: visszaadja a bejelentkezési felületen megadott switch IP címét
- `IPcimE(String IPcim)`: azt vizsgálja, hogy a bejelentkezési felületen megadott IP cím formátuma helyese-e és ettől függően igazat vagy hamisat a visszatérési értéke
- `beolvas(File bemenet)`: a felhasználói információkat tartalmazó fájl beolvasást valósítja meg
- `main(String[] args)`: főprogram
- `login()`: konstruktor
- `actionPerformed(ActionEvent e)`: a bejelentkezési felületen lévő felhasználói felületeti elemek eseménykezelő metódusa

A Felhasznalo_hozzaad osztály metódusai:

- felhHozzaAd(): konstruktor
- fajlbaIr(String beFajl, String nev, String jelszo): az új felhasználó hozzáadását megvalósító felületen megadott adatokat írja a felhasználói információkat tartalmazó fájlba
- actionPerformed(ActionEvent e): az új felhasználó hozzáadási felületen lévő grafikus elemek eseménykezelő metódusa

A Felhasznalo_torol osztály metódusai:

- felhTorol(): konstruktor
- sortTorol(String fajl, String torlendoSor): a felhasználó törlő grafikus felületen megadott felahsználót törli a rendszerből, oly módon, hogy létreho zegy átmeneti fájl, melybe beleírja az eredeti fájl összes sorát kivéve a törlendő sort, majd az így létrejött átmeneti fájlt átnevezi az eredeti fájllá. A metódus paraméterben megkapja az információkat tartalmazó fájlt, és a törlendő sor egy adatát

- actionPerformed(ActionEvent e): a felhasználó törlési felületen lévő grafikus elemek eseménykezelő metódusa

A Kitiltott_gepek osztály metódusai:

- tablahozAd(String macCim, String port, String nev, String ip): a kitiltott gépek felhasználói felületén lévő táblázathoz adja hozzá a kitiltott állomás adatait
- kitiltottGepek(): konstruktor
- actionPerformed(ActionEvent e): a kitiltott gépek felületen lévő grafikus elemek eseménykezelő metódusa
- valueChanged(ListSelectionEvent e): a kitiltott gépek felületen lévő táblázat eménykezelő metódusa

Az Adatfrissit osztály metódusai:

- utemezesVanE(): a visszatérési értéke attól függően igaz vagy hamis, hogy be van-e kapcsolva az automatikus frissítés a programban
- getFrssitadat(): az automatikus frissítés felhasználó által megadott adatát adja vissza integer-ré konvertálva
- adatFrisstesBeallit(): konstruktor
- actionPerformed(ActionEvent e): az automatikus táblázatfrissítés felületen lévő grafikus elemek eseménykezelő metódusa
- szamE(String szoveg): azt vizsgálja, hogy a frissítési paraméterként megadott adat helyes formátumú-e (csak pozitív egész szám lehet)

Az Uj_eszkoz osztály metódusai:

- ujEszkoz(): konstruktor
- ipCimE(String IPcim): azt vizsgálja, hogy a grafikus felületen megadott új eszköz IP címe helyes formátumú-e
- getEszkIP(): a grafikus felületen megadott eszköz vagy eszközök IP címét adja vissza tömbben, abból a célból, hogy a táblázat frissítésnél a tömbben lévő IP címek mindegyékére megvalósuljon az adatok újbóli lekérdezése
- actionPerformed(ActionEvent e): az osztály felületén lévő grafikus elemek eseménykezelő metódusa

A Foablak osztály metódusai:

- main(String[] args): főprogram
- foablak(): konstruktor, az egyes felhasználói felületi elemek létrehozását, és az elemkehez való eseményfigyelők társítását valósítja meg
- getFrame(): a főablak frame-jét adja vissza, más osztályokból való főablakon történő felugró ablakok megjelenítéséhez szükséges
- tablazatFelepit(): a program főablakában található adatokat tartalmazó táblázatot hozza létre, és állítja be a táblázat tulajdonságait
- getTablazat(): a létrehozott táblázatot adja vissza, a táblázat más osztályokból való lekérdezése miatt szükséges
- kitiltottGepMac(): a kitiltott állomás MAC címét adja vissza
- kitiltottGepPort(): a kitiltott állomás switch portját adja vissza
- setTarget(Address cim): beállítja az SNMP kapcsolat létrehozásához szükséges IP címet
- getTarget(): visszaadja az aktuális meglévő SNMP kapcsolat által használt IP címet
- kitiltVanE(): igazat ad vissza, ha volt állomás kitiltva a programban, a kitiltott állomások újbóli csatlakozásának a figyelése miatt szükséges, ne figyelje a program az újonnan kapcsolódott állomásokat kitiltás céljából, ha még nem is volt kitiltva állomás
- tablahozAd(String macCim, String port, String nev): a paraméterben megkapott adatokat adja hozzá a táblázathoz
- tablazatFrissites(): a táblázat adatainak frissítését valósítja meg, azért van rá szükség, mert ha a hálózatban valamely állomás csatlakozott vagy eltűnt, akkor e szerint a táblázatban lévő adatok is frissüljenek
- systemTray(): azt a funkciót valósítja meg, hogy az alkalmazás képes legyen a tálcán lévő óra mellé „ülni”, onnan folyamatosan figyelve az általa felügyelt hálózatot
- valueChanged(ListSelectionEvent e): a táblázatban bekövetkező változások eseményfigyelő metódusa
- gombTilt(): a jogosultságok beállításához szükséges gombok letiltását valósítja meg
- fajlbaIrKitiltott(String beFajl, String mac, String port, String nev, String ip): a kitiltott állomások adatinak adatállományban való tárolását valósítja a már egyszer kitiltott állomások újbóli csatlakozási kísérletének kizárása céljából

- automataPortLetiltas(): a már egyszer kitiltott állomások újbóli csatlakozása esetén ismételten kitiltja azokat és az újonnan letiltott portok adatát a kitiltott állomásokat tartalmazó adatállományba írja és naplózza az eseményt
- actionPerformed(ActionEvent e): az osztály felületén lévő grafikus elemek eseménykezelő metódusa

7. Az elkészült alkalmazás

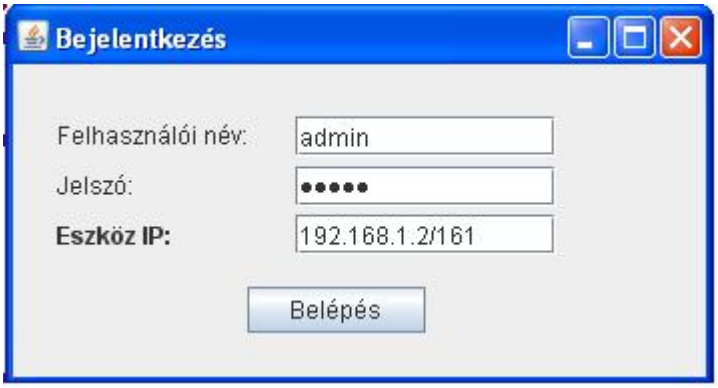
Ebben a fejezetben az elkészült alkalmazás használatához szükséges feltételeket ismertetem és a programról működés közben készült képek segítségével pedig szemléltetem funkcióit.

7.1 A használat feltételei

Az elkészült alkalmazás használatához szükséges, hogy az általa felügyelt hálózatban működjön legalább 1 darab SNMP-t támogató switch, amely kezeli a bridge-mib-eket és az alkalmazást futtató számítógépen, pedig legalább 6-os verziójú java futtató környezet (JRE: Java Runtime Environment) legyen telepítve.
A bridge-mib-ek az alkalmazás működésének alapját képező CAM tábla előállításához, a legalább 6-os verziójú JRE pedig az alkalmazásban lévő egyes funkciók például a system tray vagy a keletkezett naplófájl alapértelmezett böngészővel való megnyitásához szükségesek.

7.2 Az alkalmazás működés közben

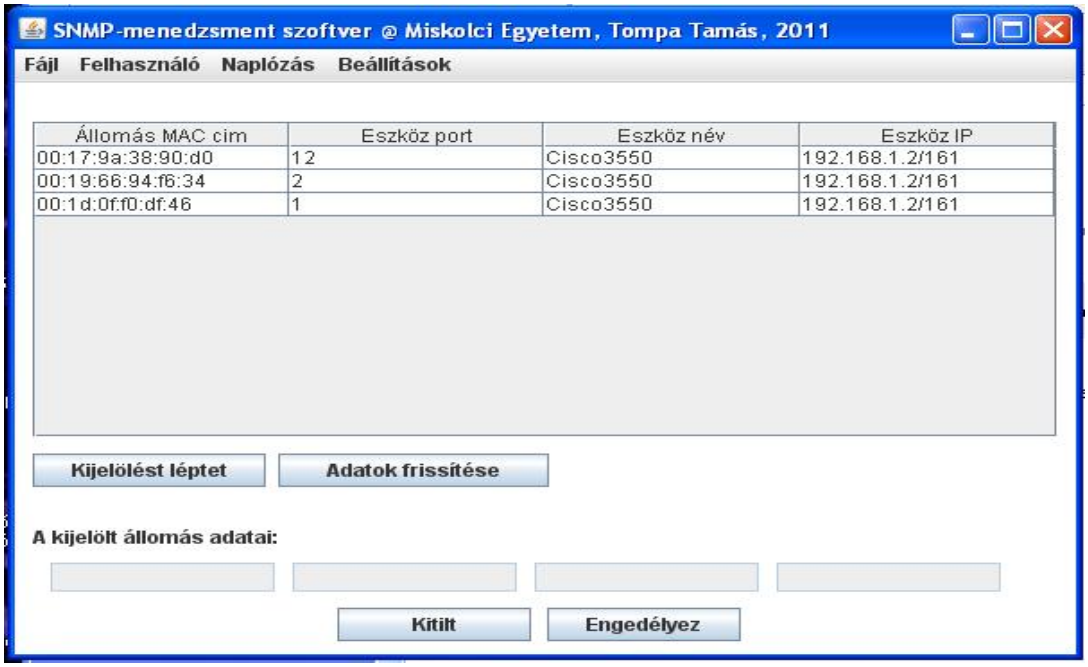
Első lépés az alkalmazás elindítása, amely után megjelenik a bejelentkezési felülete, melyen meg kell adni a felhasználói nevet, a hozzá tartozó jelszót és egy SNMP-re képes switch IP címét. A bejelentkezéshez az „admin” felhasználói nevet, a hozzá tartozó „admin” jelszót és a teszt hálózatban lévő Cisco 3550 típusú switch IP címét adtam meg.
Ez a következő 14. ábrán látható.



14. ábra Az alkalmazás bejelentkező felülete

Ha a bejelentkezésnél megadott adatok valamelyike nem fele meg az adott formátumnak, vagy az adott eszköz nem elérhető, akkor ezt a program közli a felhasználóval majd lehetőség van az adatok újbóli megadására.

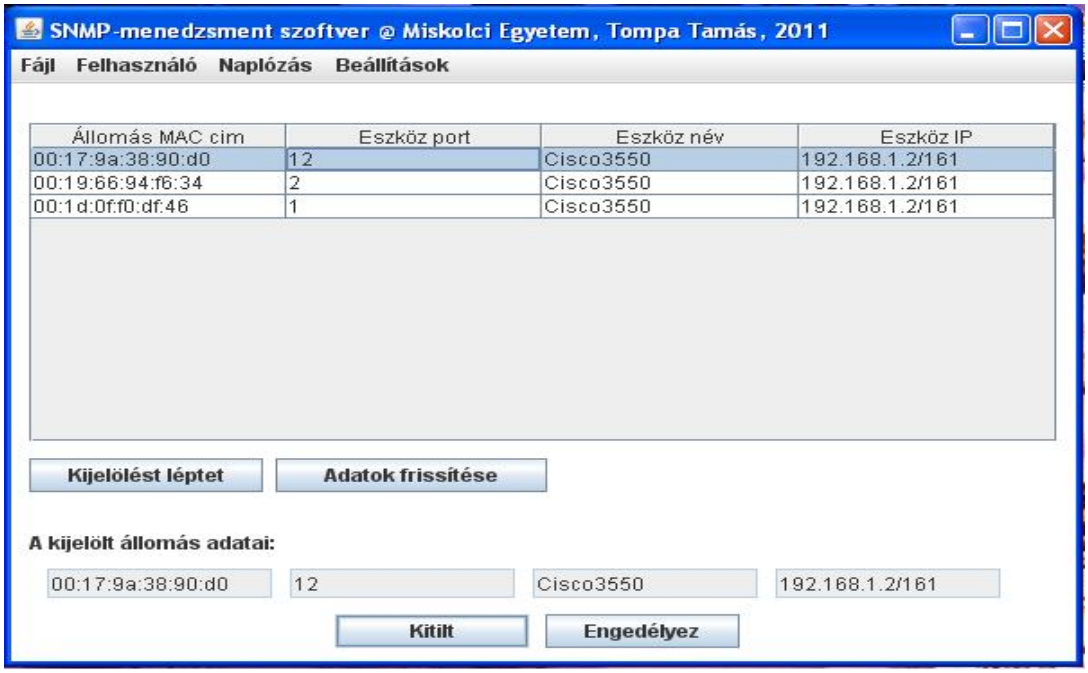
A sikeres bejelentkezést követően nyílik meg a program fő felülete, melyet a következő 15. ábra szemléltet.



15. ábra A program bejelentkezés utáni felülete

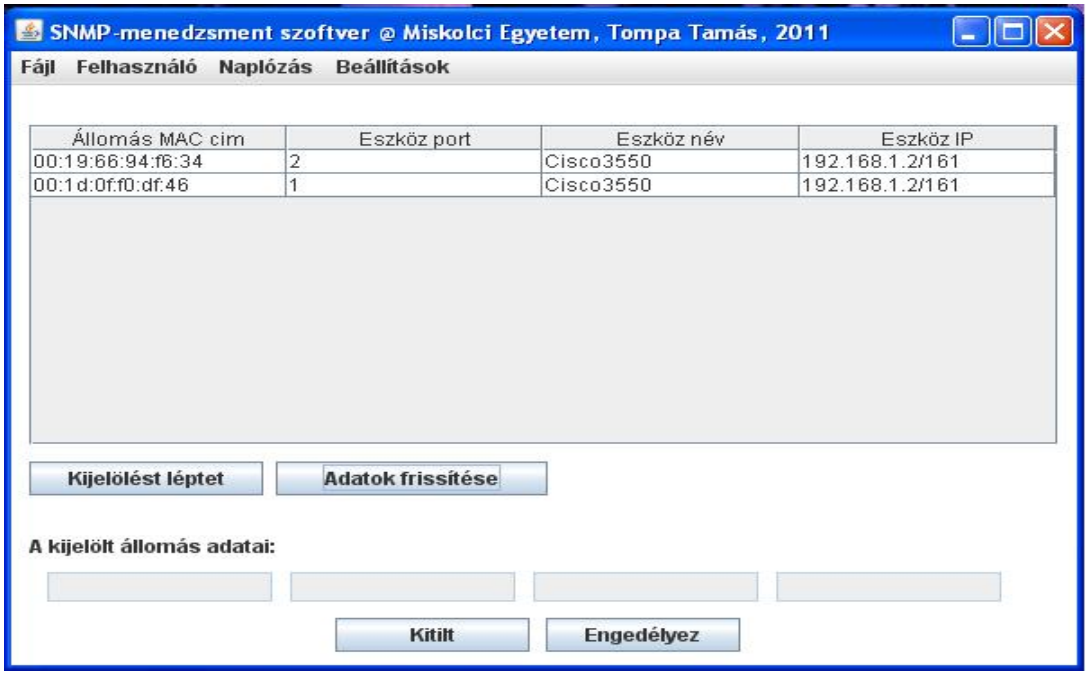
A táblázatban szereplő adatok azt mutatják meg, hogy mely MAC című állomás mely nevű, IP című switch, mely portjára van csatlakozva.

Az adott állomás kitiltására a táblázatban való megfelelő sorra való kattintás segítségével van lehetőség, mely a 16. ábrán látható.



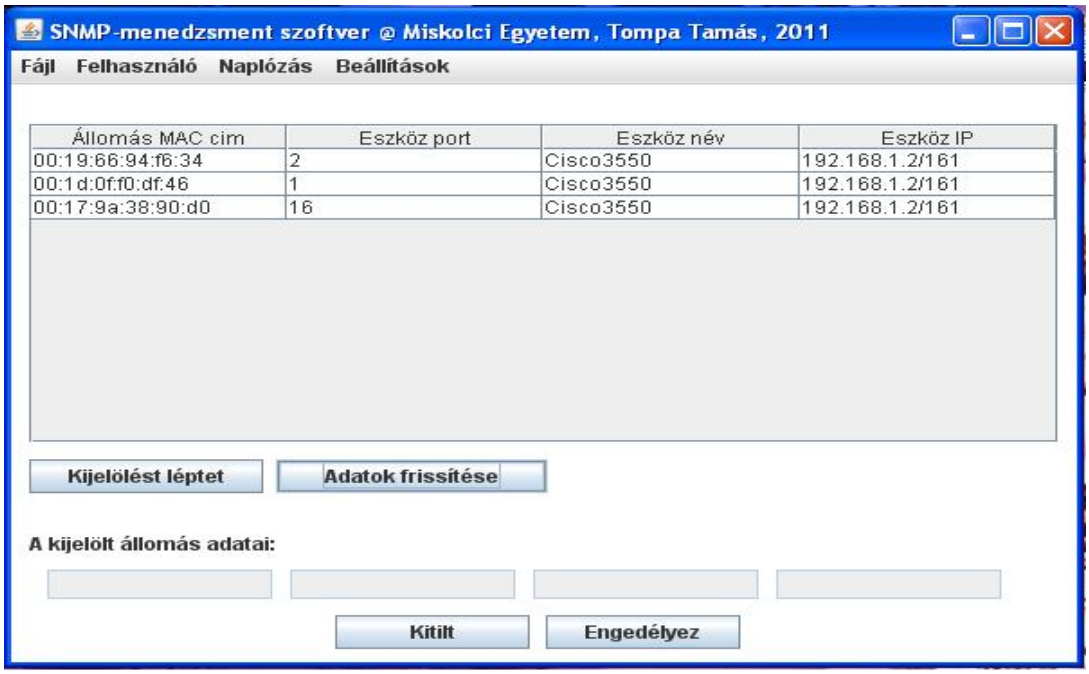
16. ábra A kijelölt állomás kitiltása

A „Kitilt” gombra való kattintás után a táblázatban kijelölt állomás kizáródik a switch-ről, majd a táblázat adatinak frissítése után el is tűnik a táblázatból. Az adatok frissítését az „Adatok frissítése” gomb segítéségével valósította meg. A folyamatot a 17. ábra szemlélteti.



17. ábra A kitiltott állomás eltűnt a táblázatból

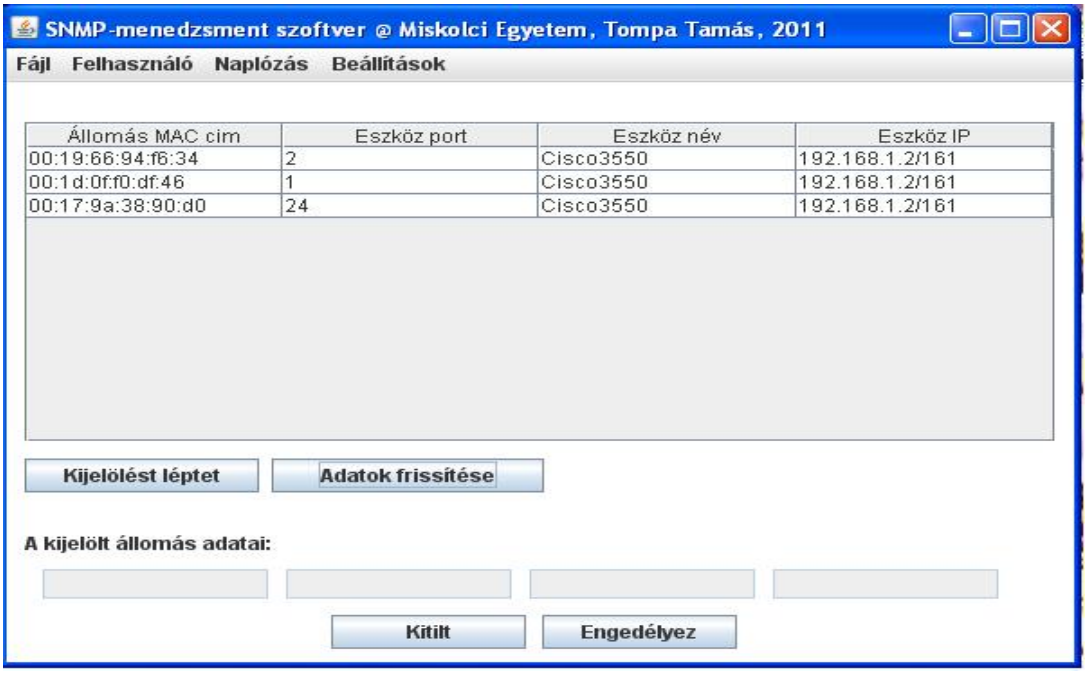
Ezt követően, ha újból csatlakozni kíván a kizárt állomás, akkor a program automatikusan kitiltja.
A következő 18. ábrán az látható, hogy az előbb kitiltott állomás újból csatlakozott a switch 16-os portjára.



18. ábra Az kizárt állomás csatlakozott a switch 16-os portjára

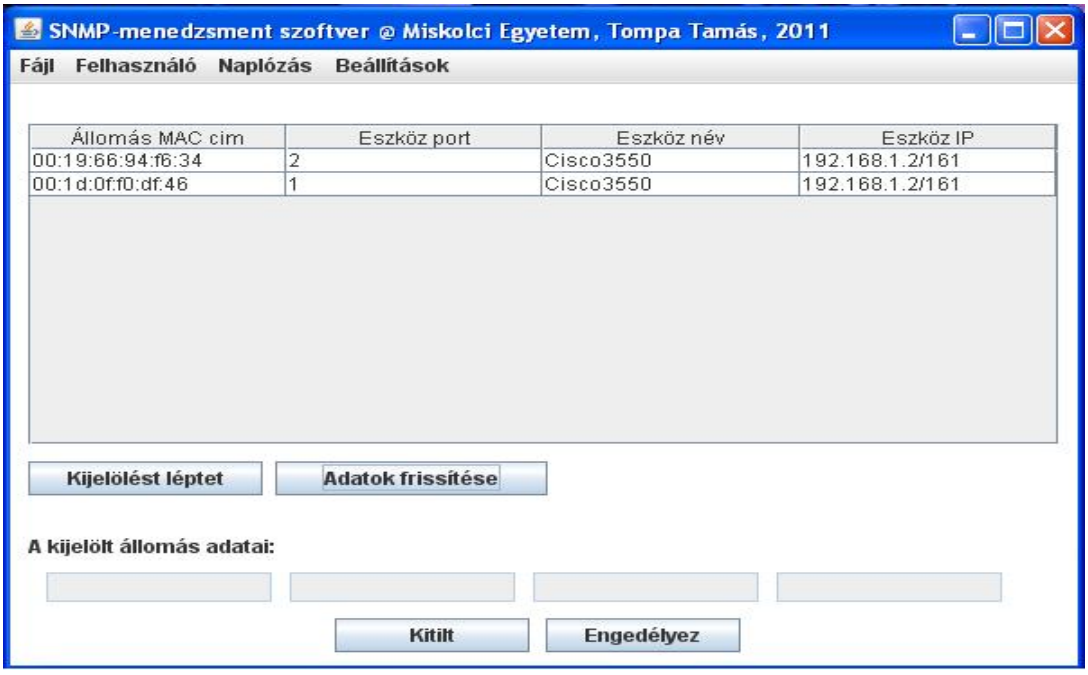
A táblázat adatainak frissítése után a program automatikusan ismét kitiltja a már egyszer kizárt állomást, és az állomás a táblázat adatainak frissítése után ismét eltűnik a táblázatból.

A következő képen az látható, hogy a 16-os switch portról a program által automatikusan kitiltott állomás ismét csatlakozott a switch 24-es portjára.



19. ábra A kitiltott állomás csatlakozott a switch 24-es portjára

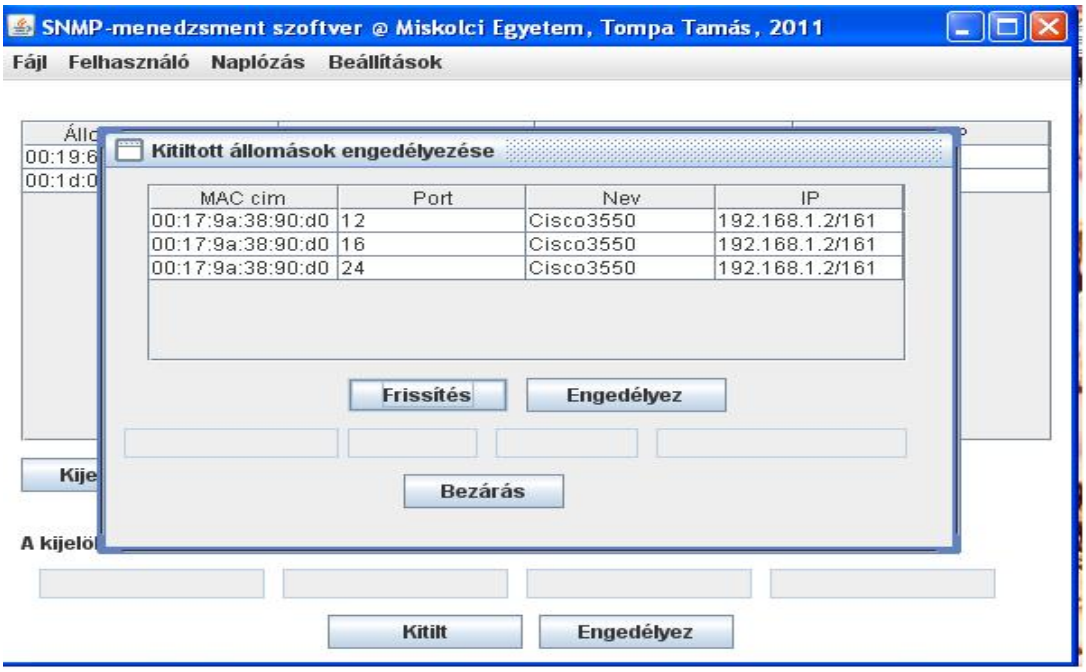
A táblázat adatainak frissítése után a program ismét automatikusan letiltja a kizárt MAC című állomás switch portját, azaz jelen esetben a 24-es portot és az állomás eltűnik a táblázatból. A folyamat a következő 20- ábrán látható.



20. ábra Az automatikusan kitiltott állomás eltűnt a táblázatból

A kitiltott állomások engedélyezésére a „Beállítások” menüpontban lévő ”Kitiltott állomások” menüpont segítségével van lehetőség. Itt megjelenik a kitiltott állomások

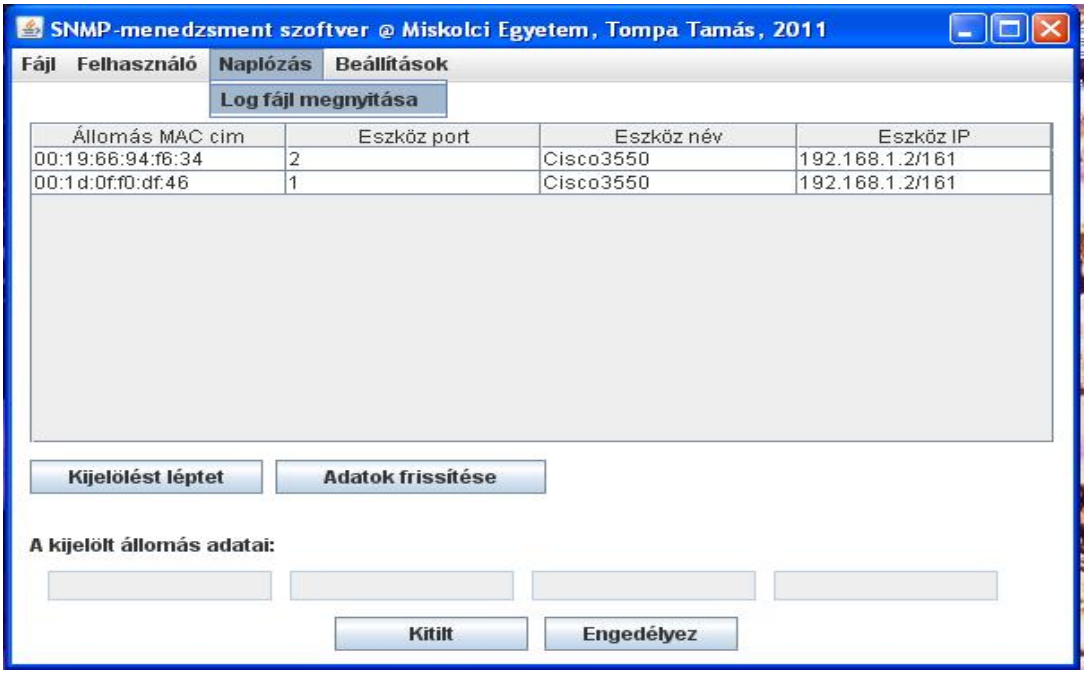
adatait tartalmazó táblázat a táblázat alatt lévő „Frissítés” gomb megnyomása után. Ez a következő 21. ábrán látható.



21. ábra A kitiltott állomások engedélyezése

Az engedélyez gombra való kattintás után van lehetőség a táblázatban kijelölt állomás engedélyezésére, amelyek a program főablakában lévő táblázat adatainak frissítése után megjelennek a táblázatban.

A végrehajtott művelteket, a kitiltást és az engedélyezést a program naplózta. A naplófájl megtekintésére a „Naplózás” menüpontban lévő „Log fájl megnyitása” almenüpont segítségével van lehetőség, mely a következő 22. ábrán látható.



22. ábra A keletkezett naplófájl megtekintésének menüpontja

A menüpontra való kattintást követően megjelenik a keletkezett html naplófájl, amelyet az alapértelmezett böngésző jelenít meg. A keletkezett naplófájl tartalma a következő 23. ábrán látható.

Mon Nov 21 15:20:54 CET 2011

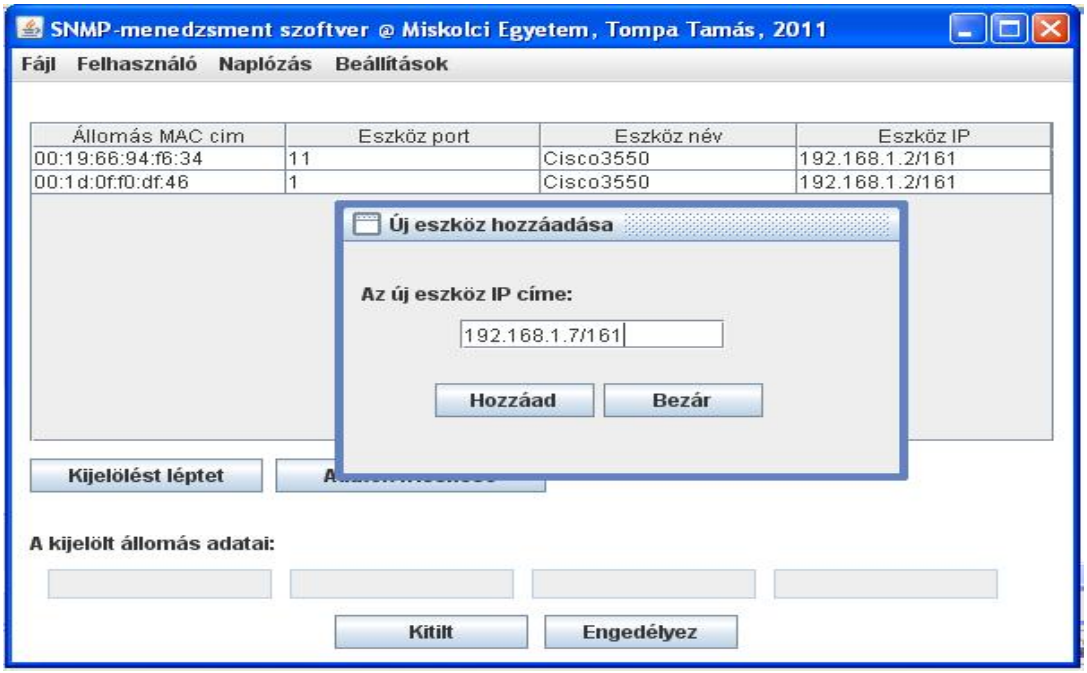
Type	Log uzenet
INFO	nov. 21,2011 15:20 Mac:00:17:9a:38:90:d0 Port:12 Nev:Cisco3550 IP:192.168.1.2/161 állomás kitiltva
INFO	nov. 21,2011 15:21 Mac:00:17:9a:38:90:d0 Port:16 Nev:Cisco3550 IP:192.168.1.2/161 állomás kitiltva az ALKALMAZÁS által automatikusan
INFO	nov. 21,2011 15:22 Mac:00:17:9a:38:90:d0 Port:24 Nev:Cisco3550 IP:192.168.1.2/161 állomás kitiltva az ALKALMAZÁS által automatikusan
INFO	nov. 21,2011 15:22 Mac:00:17:9a:38:90:d0 Port:12 Nev:Cisco3550 IP:192.168.1.2/161 állomás engedélyezve
INFO	nov. 21,2011 15:22 Mac:00:17:9a:38:90:d0 Port:16 Nev:Cisco3550 IP:192.168.1.2/161 állomás engedélyezve
INFO	nov. 21,2011 15:22 Mac:00:17:9a:38:90:d0 Port:24 Nev:Cisco3550 IP:192.168.1.2/161 állomás engedélyezve

23. ábra A létrejött naplófájl

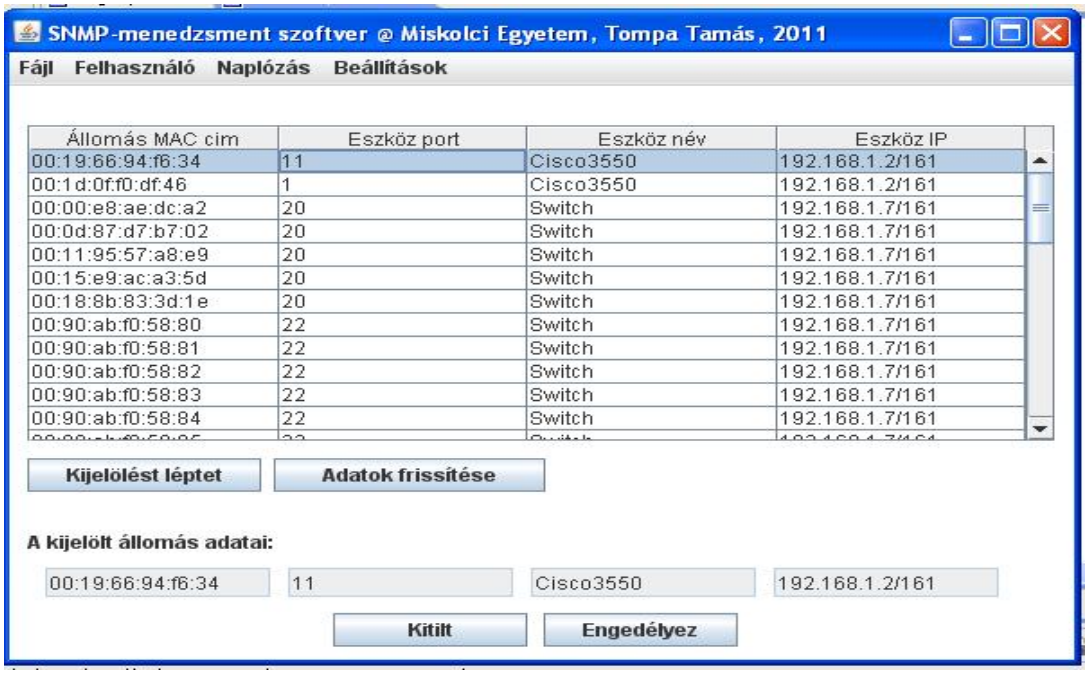
A naplófájlban látható, hogy Cisco3550 nevű switch 12-es portján lévő 00:17:9a:38:90:d0 MAC című állomás ki lett tiltva 15:20-kor, majd ezt követően az állomás csatlakozott a switch 16-os portjára 15:21-kor, a 24-es portjára 15:22-kor, melyeket az alkalmazás automatikusan tiltott le. Majd 15:22-kor lettek engedélyezve a switch letiltott portjai.

A program további funkciói

A programhoz lehetőség van a bejelentkezéskor megadott switch-en kívül más újabb switch-eket is hozzáadni felügyeleti célból. Ezt a „Beállítások” menüpontban lévő „Új eszköz” almenüpont segítségével van lehetőség megvalósítani, ahol az újabb eszközök IP címét kéri a program. Fontos, hogy az új eszközöket mindenképp úgy kell hozzáadni a programhoz, hogy egyesével egymás után beírjuk az IP címüket, majd ha végeztünk az eszközök megadásával utána zárjuk be az ablakot, az ablak bezárása után nincs lehetőség újabb eszközök hozzáadására. A folyamatot a következő 24. és 25. ábrák szemléltetik.



24. ábra Új switch hozzáadása

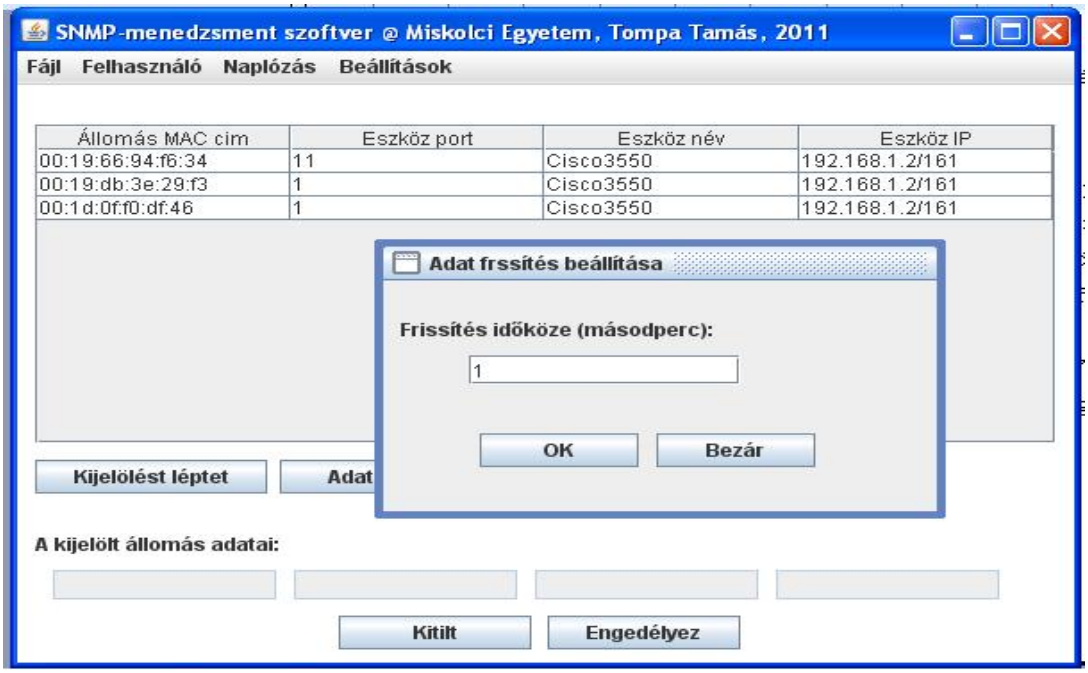


25.ábra A programhoz hozzáadott switch-ek

A táblázatban látható a bejelentkezéskor megadott Cisco 3550 típusú switch-en lévő, és a hozzáadott 192.168.1.7/161 IP című switch-en lévő állomások. Az utóbbi IP című és Swtich nevű switch-et az iReasoning SNMP Agent Simulator nevezetű program demó verziójának segítségével hoztam létre, a programot a következő internetes elérhetőségről töltöttem le: http://ireasoning.com/download.shtml

A programhoz újabb felhasználók hozzáadására vagy már meglévő felhasználók törlésére a „Felhasználó" menüpontban lévő „Hozzáad" és „Töröl" almenüpontokkal van lehetőség megvalósítani. Új felhasználó hozzáadását a felhasználó nevének és a házzá tartozó jelszavának megadásával lehet megvalósítani, meglévő felhasználó törlését pedig a törölni kívánt felhasználó felhasználói nevének megadásával lehet végrehajtani. Törlés estében az adott felhasználóhoz tartozó összes jelszó törlődik, az új felhasználó hozzáadása pedig nem történik meg, ha a rendszer már létezik a hozzáadni kívánt nevű és jelszavú felhasználó. A program alapértelmezett felhasználói neve és a hozzá tartozó jelszó az „admin".

A táblázatban lévő adatok automatikus frissítését beállítani a „Beállítások" menüpontban lévő „Adat frissítés" almenüponttal van lehetőség. Itt a frissítés gyakoriságának időközét kell megadni másodpercben. Ha a felhasználó hibás frissítési adatot (nem számot) ad meg, akkor azt közli a program. A folyamat a következő 26. ábrán látható.



26. ábra Az automatikus adatfrissítés beállítása

Az ábrán látható, hogy a megadott adat frissítési időköz 1 másodperc, azaz a program 1 másodpercenként figyeli az általa felügyelt switch-en bekövetkező változásokat.

8. Irodalomjegyzék

[1] http://old.hungary.com/telecomputer/4_06/1_2k.htm

[2] Varga Tamás - Nagysebességű hálózati technikák, 2000
www.aut.bmf.hu/konya/okt/netint_www/halmanag.pdf

[3] http://www.canet.hu/download.php?view.46

[4] http://rendszergazdak.blog.hu/2009/04/06/mire_jo_a_halozati_rajz

[5] Szállási Tibor, Suba Attila – Számítógép-hálózatok menedzselése
http://ebookz.hu/ebook.php?azon=5807f0

[6] Pletl Szilveszter – Számítógép-hálózatok menedzselése
www.isst.rs/exams/download/oper/no-render/file/HaloMenJegyzet.pdf

[7] http://hu.wikipedia.org/wiki/OSI_modell

[8] http://hu.wikipedia.org/wiki/SNMP

[9] http://szabilinux.hu/snmp/index.html

[10] http://e-oktat.pmmf.hu/snmp2

[11]http://nik.uniobuda.hu/schubert.tamas/H%E1l%F3zati%20szolg%E1ltat%E1sok%20terv/HSZ-H%E1l%F3zatmenedzsment.pdf

[12] http://nagiosplug.sourceforge.net/developer-guidelines.html

[13] http://www.cuddletech.com/articles/snmp/node23.html

[14] http://www.snmp4j.org/index.html

[15] http://hu.wikipedia.org/wiki/Objektumorient%C3%A1lt_programoz%C3%A1s

[16] http://hu.wikipedia.org/wiki/Fejleszt%C5%91i_k%C3%B6rnyezet

[17] http://hu.wikipedia.org/wiki/NetBeans

[18] http://hu.wikipedia.org/wiki/Eclipse

[19] http://hu.wikipedia.org/wiki/Switch_%28informatika%29

[20] http://hu.wikipedia.org/wiki/Hub_%28h%C3%A1l%C3%B3zat%29

[21]http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps646/product_data_sheet09186a00800913d7.html

[22] https://www.tilb.sze.hu/tilb/targyak/NGB_TA024_1/krp-1f.pdf

[23] www.szgti.bmf.hu/~zbalogh/nik/szgh_gyak/switch_alap_konfig.doc

[24]http://www.cisco.com/en/US/tech/tk801/tk36/technologies_tech_note09186a0080094465.shtml

[25]http://www.cisco.com/en/US/docs/switches/lan/catalyst3550/software/release/12.1_8_ea1/configuration/guide/swsnmp.html

[26] http://hu.wikipedia.org/wiki/Unified_Modeling_Language

[27] http://en.wikipedia.org/wiki/StarUML

[28] http://dcs.vein.hu/goth/oktatas/he.pdf

[29] http://users.iit.uni-miskolc.hu/~mileff/szt/srs.html

Képek, ábrák forrása:

1. ábra: http://comgt.com/lib/management/tmnlayers-ensb.png

2. ábra: [5] Szállási Tibor, Suba Attila – Számítógép-hálózatok menedzselése
http://ebookz.hu/ebook.php?azon=5807f0 - 48. oldal

3. ábra: http://hu.wikipedia.org/wiki/SNMP

4. ábra: http://www.cisco.com/en/US/i/000001-100000/20001-25000/24001-24500/24187.jpg

A program készítése során felhasznált kódrészletek forrása:

[1] http://www.vogella.de/articles/Logging/article.html

[2] http://www.javadb.com/remove-a-line-from-a-text-file

Linkek utoljára ellenőrizve: *2011.11.22.*

9. Összefoglalás

Dolgozatom készítése során tanulmányoztam a hálózat-felügyelet és ezen belül az SNMP hálózat-felügyelet témakörét, bemutatásra kerültek a hálózat-felügyelet céljai, funkciói, feladatai, az SNMP hálózat-felügyeleti protokoll illetve készítettem egy saját SNMP-n alapuló hálózat-felügyeleti szoftvert. A dolgozat témájával kapcsolatos információkról főként az Interneten tájékozódtam, jórészt magyar nyelvű szakirodalom felhasználásával. A felhasznált irodalmak között szerepelnek elektronikus formátumú könyvek, főiskolai illetve egyetemi jegyzetek, előadásanyagok, különböző rövidebb ismertetők, leírások, és a feladat megoldásához felhasznált programozási nyelvvel foglalkozó Internetes tartalmak.

A dolgozat készítése alatt mélyebb ismereteket szereztem az SNMP-ről, mint egy a gyakorlatban igen elterjedt hálózatmenedzselési protokollról, tanulmányoztam a Management Information Base felépítését, illetve a feladat megoldásának lehetőségeit. A megoldási lehetőségek között szereplő hálózat-felügyeleti rendszert otthoni környezetben ki is próbáltam, illetve mélyebb ismereteket szereztem az alkalmazás készítése során felhasznált Java programozási nyelvről.

Az elkészített alkalmazás lehetőség ad egy SNMP-t támogató hálózati eszköz menedzselésére. Segítségével nyomon követhetőek az egyes hálózati eszközökön lévő állomások, az adott nem kívánt állomás kitiltható az eszközről, illetve egy külön funkcióban valósul meg a kitiltott állomások megjelenítése. Az egyes események, például állomások kitiltása és engedélyezése naplózásra kerülnek a rendszerben, mely segítségével nyomon követhetőek a rendszerben végrehajtott tevékenységek. A program a kényelmesebb használhatóság érdekében rendelkezik grafikus felhasználói felülettel, melyen végrehajthatóak az egyes funkciók.

Továbbfejlesztési lehetőségként a program finomítását, és egy olyan funkcióval való kiegészítését javaslom, amely lehetőséget ad arra, hogy a program által figyelt számítógép-hálózatban, ha valamely állomás nem kívánt tevékenységet folytat például port scanner-rel vagy shnifferel, akkor az automatikusan kerüljön kizárásra a számítógép-hálózatból.

A dolgozat írása közben támadtak nehézségek, ilyen nehézségek voltak a feladat megoldására való technikák, lehetőségek felkutatása illetve a megoldási lehetőségként szánt alkalmazás elkészítése Java nyelven, amely komplexitása miatt okozott bonyodalmakat.

A dolgozat készítése alatt sok hasznos új információt szereztem a számítógép-hálózatok működéséről, a hálózatban lévő eszközök konfigurálásáról és a Java programozási nyelvről.

10. Summary

I studied the topic of network control, as well as the field of SNMP network control in my thesis. I explained the aims, functions, tasks of network control, the SNMP network protocol and I created a network control software based on SNMP. I gathered the relevant information mainly from Hungarian resources from the Internet. The list of my resources includes electronic books, university and college notes, lecture summaries, different short guide-books, descriptions, and Internet contents on the programming language I used for solving the task.

When I did this research, I deepened my knowledge about SNMP, which is a widely used network management protocol; I examined the structure of the Management Information Base, and the possible solutions for the task. I tried the network control system, which was one of the possible solutions, in my home environment, and I enriched my knowledge about the Java programming language, which I used during the process of creating my application.

The application makes it possible to manage a network appliance which supports SNMP. With the help of it, stations on particular network appliances can easily be monitored, a station can be banned from an appliance, and the visualization of the banned stations is done via a separate function. Different actions, like banning and licensing stations, are listed in the system, so these can easily be traced back. The program has a graphic application system in which the particular functions can be executed.

For further development purposes, I recommend the refinement of the program and its completion with a function which makes it possible to automatically ban undesirable actions of a station with a port-scanner or shniffer in the computer system monitored by the program.

Complications aroused during my research, like finding the proper technique to solve the task, exploring the possibilities and creating my solution in Java language. The latter was complicated because of its complexity.

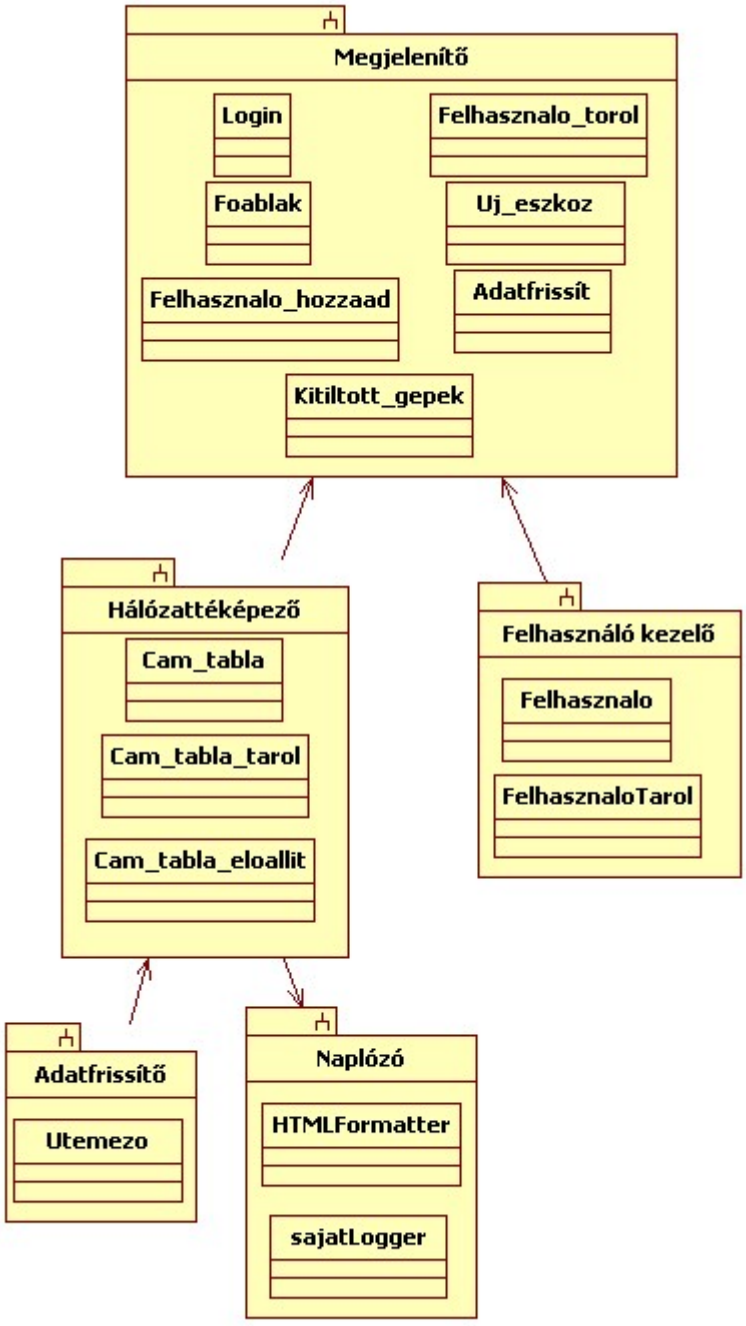
Because of my investigation, I have picked up numerous useful information on the working of computer networks, the configuration of the network appliances, and the Java programming language.

Köszönetnyilvánítás

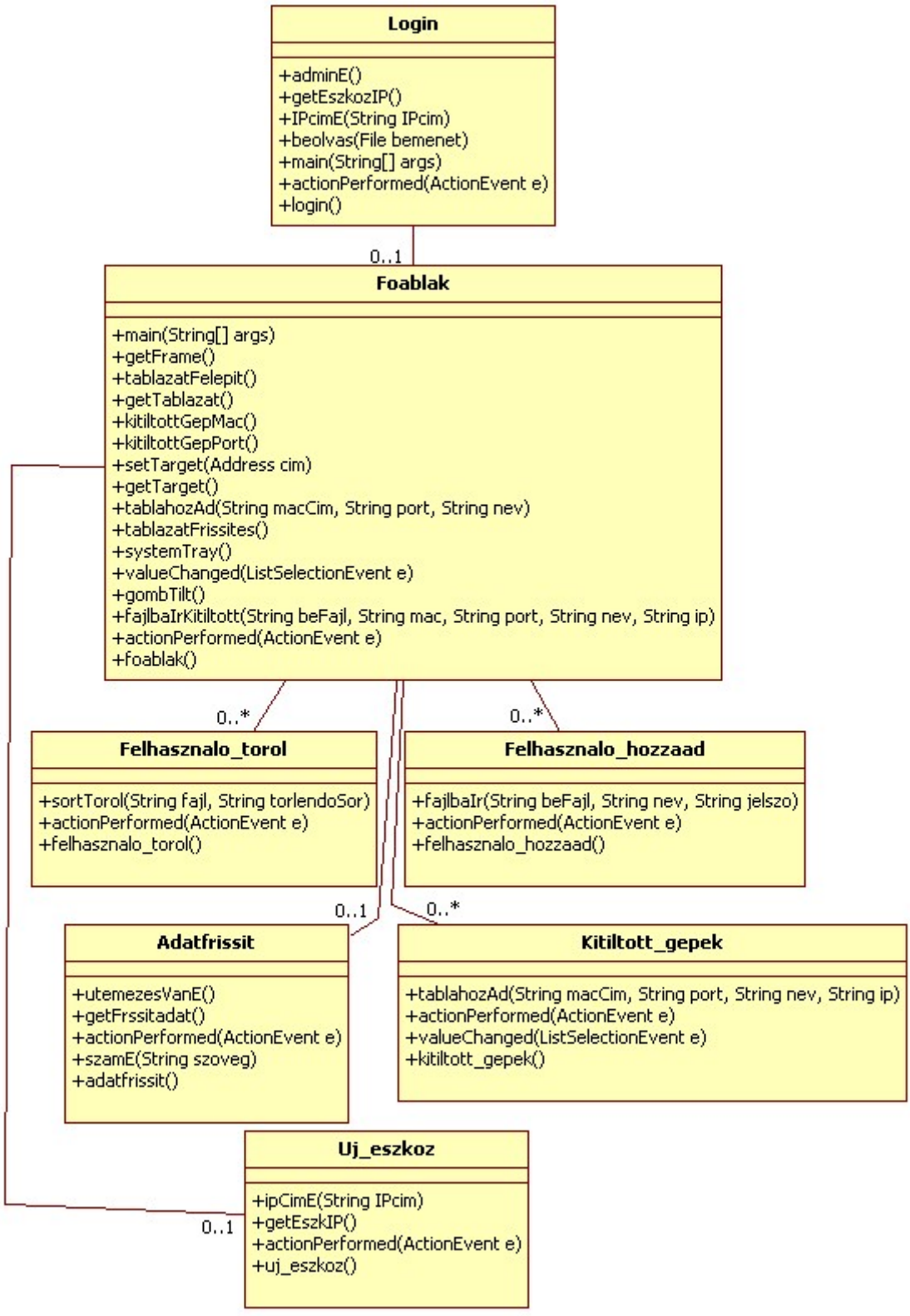
Köszönöm Dr. Kovács Szilveszter konulensemnek, hogy lehetőséget biztosított dolgozatom elkészítéséhez, köszönöm segítőkész támogatását, hasznos tanácsait, illetve, hogy tanácsaival és iránymutatásaival hozzájárult szakmai fejlődésemhez.

11. Nyomtatott mellékletek

1. Melléklet



1. ábra A rendszer részletesebb felépítése



2. ábra A megjelenítő alrendszer osztálydiagramja

2. Melléklet

A cam_tabla osztály forráskódja

```
package CAM_tabla;


public class cam_tabla {
	private String mac;			/* allomas MAC cim */
	private int port;			/* switch port */
	private String nev;			/* switch neve */


	public cam_tabla(String mac, String port, String nev) {
		this.mac = mac;
		this.port = Integer.parseInt(port);
		this.nev = nev;
	}


	public String toString() {
		return "MAC cim:" + this.mac + "  Port:" + this.port + "
Nev:" + this.nev;
	}


	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((mac == null) ? 0 :
mac.hashCode());
		result = prime * result + port;
		return result;
	}


	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		cam_tabla other = (cam_tabla) obj;
		if (mac == null) {
			if (other.mac != null)
				return false;
		} else if (!mac.equals(other.mac))
			return false;
		if (port != other.port)
			return false;
		return true;
	}


	public int portNagyobbE(cam_tabla c) {
		if(this.port > c.port)
			return 1;
		if(this.port < c.port)
```

```
                return -1;

            return 0;
        }


        public String getMac() {
            return this.mac;
        }


        public int getPort() {
            return this.port;
        }


        public String getNev() {
            return this.nev;
        }
}
```

A CAM_tabla_tarol osztály forráskódja

```
package CAM_tabla;

import java.util.*;

public class CAM_tabla_tarol {
        ArrayList<cam_tabla> elemek;


        public CAM_tabla_tarol() {
            this.elemek = new ArrayList<cam_tabla>();
        }


        public void hozzaad(cam_tabla uj) {
            if(uj == null)
                return;

            if(this.elemek.contains(uj))
                return;

            this.elemek.add(uj);
        }


        public void torol(cam_tabla torlendo) {
            if(torlendo == null)
                return;

            if(!this.elemek.contains(torlendo))
                return;

            this.elemek.remove(torlendo);
        }
```

```
	public int getMeret() {
		return this.elemek.size();
	}


	public void listaFrissit() {
		this.elemek.clear();
	}


	public cam_tabla[] getTombben() {
		cam_tabla[] tomb = new cam_tabla[this.elemek.size()];


		return this.elemek.toArray(tomb);
	}

	public Comparator<cam_tabla> PORTCOMP = new Comparator<cam_tabla>()
{
		@Override
		public int compare(cam_tabla o1, cam_tabla o2) {
			return o1.portNagyobbE(o2);
		}
	};



	/* port szerinti novekvo sorrend */
	public cam_tabla[] getTombRendezve() {
		cam_tabla[] tomb = new cam_tabla[this.elemek.size()];

		this.elemek.toArray(tomb);
		Arrays.sort(tomb,PORTCOMP);

		return tomb;
	}


	/* adott indexu elem visszaad */
	public cam_tabla getElemInex(int index) {
		return this.elemek.get(index);
	}


	/* adott MAC cimu elem keresese */
	public cam_tabla elemKeres(String mac, String port, String nev) {
		cam_tabla keresendo = new cam_tabla(mac, port, nev);
		int hol = this.elemek.indexOf(keresendo);

		if(hol == -1)
			return null;

		return this.elemek.get(hol);
	}
}
```

A cam_tabla_eloallit osztály forráskódja

```
package CAM_tabla;

import java.io.*;

import javax.swing.JOptionPane;
import org.snmp4j.*;
import org.snmp4j.event.*;
import org.snmp4j.smi.*;
import org.snmp4j.transport.*;
import GUI.foablak;



public class cam_tabla_eloallit {

      private static CommunityTarget target = new CommunityTarget();
      private static CAM_tabla_tarol camTabla = new CAM_tabla_tarol();

      private static VariableBinding vb1;
    private static VariableBinding vb2;
    private static VariableBinding vb3;



      public static int camTablaMerete() {
            return camTabla.getMeret();
      }


      public static void snmpKapcs(Address targetAddress) throws
IOException {
            /* target definicio */
            target.setAddress(targetAddress);
            target.setCommunity(new OctetString("public"));
            target.setTimeout(1000);
            target.setVersion(0);
            target.setRetries(1);


            /* az arraylsit elemeinek frissitese */
            camTabla.listaFrissit();


            /* transport mapping definicio */
            try {
            TransportMapping transport = new
DefaultUdpTransportMapping();
            transport.listen();

            } catch(IOException ex) {
                  System.out.println("IO hiba: " + ex.getMessage());
            }


            OIDbejar(new OID(".1.3.6.1.2.1.17.4.3.1.1"), new
OID(".1.3.6.1.2.1.17.4.3.1.2"), new OID(".1.3.6.1.2.1.1.5.0"));

      }
      /* OID bejarasa */
```

```
      public static void OIDbejar(OID oidMac, OID oidPort, OID oidNev)
throws IOException {
            PDU requestPDUMac = new PDU();
            PDU requestPDUPort = new PDU();
            PDU requestPDUNev = new PDU();


          requestPDUMac.add(new VariableBinding(oidMac));
          requestPDUMac.setType(PDU.GETNEXT);

          requestPDUPort.add(new VariableBinding(oidPort));
          requestPDUPort.setType(PDU.GETNEXT);

          requestPDUNev.add(new VariableBinding(oidNev));
          requestPDUNev.setType(PDU.GET);


          try
          {
            TransportMapping transport = new
DefaultUdpTransportMapping();
            Snmp snmp = new Snmp(transport);
            transport.listen();


            boolean finished = false;
            while (!finished) {
              vb1 = null;
              vb2 = null;
              vb3 = null;


                  @SuppressWarnings("deprecation")
                  PDU responsePDUMac =  snmp.sendPDU(requestPDUMac,
target);
                  PDU responsePDUPort =  snmp.sendPDU(requestPDUPort,
target);
                  PDU responsePDULan =  snmp.sendPDU(requestPDUNev,
target);


              if (responsePDUMac != null) {
                vb1 = responsePDUMac.get(0);
                vb2 = responsePDUPort.get(0);
                vb3 = responsePDULan.get(0);
              }



              if (responsePDUMac == null) {

      JOptionPane.showMessageDialog(foablak.getFrame(),"Az eszköz nem
elérhető!", "Hiba!", JOptionPane.WARNING_MESSAGE);
                  finished = true;
                  return;
                } else if (responsePDUMac.getErrorStatus() != 0) {
                    JOptionPane.showMessageDialog(foablak.getFrame(),
responsePDUMac.getErrorStatusText().toString(), "Hiba!",
JOptionPane.WARNING_MESSAGE);
                  finished = true;
                } else if (vb1.getOid() == null) {
```

```
                    JOptionPane.showMessageDialog(foablak.getFrame(),
"vb.getOid() == null", "Hiba!", JOptionPane.WARNING_MESSAGE);
                    finished = true;
                } else if (vb1.getOid().size() < oidMac.size()) {
                    JOptionPane.showMessageDialog(foablak.getFrame(),
"vb.getOid().size() < targetOID.size()", "Hiba!",
JOptionPane.WARNING_MESSAGE);
                    finished = true;
                } else if
(oidMac.leftMostCompare(oidMac.size(),vb1.getOid()) != 0) {
                  // System.out.println("targetOID.leftMostCompare() !=
0)");
                  finished = true;
                } else if
(Null.isExceptionSyntax(vb1.getVariable().getSyntax())) {
                    JOptionPane.showMessageDialog(foablak.getFrame(),
"Null.isExceptionSyntax(vb.getVariable().getSyntax())", "Hiba!",
JOptionPane.WARNING_MESSAGE);
                    finished = true;
                } else if (vb1.getOid().compareTo(oidMac) <= 0) {
                    JOptionPane.showMessageDialog(foablak.getFrame(), "A
visszakapott MIB változó hibás", "Hiba!", JOptionPane.WARNING_MESSAGE);
                    JOptionPane.showMessageDialog(foablak.getFrame(),
vb1.toString() + " <= "+oidMac, "Hiba!", JOptionPane.WARNING_MESSAGE);
                    finished = true;
                }


              else {
                camTabla.hozzaad(new
cam_tabla(vb1.getVariable().toString(),vb2.getVariable().toString(),vb3.g
etVariable().toString()));



                /* foablak tablazatahoz adasa */

foablak.tablahozAd(vb1.getVariable().toString(),vb2.getVariable().toStrin
g(),vb3.getVariable().toString());



                requestPDUMac.setRequestID(new Integer32(0));
                requestPDUMac.set(0, vb1);

                requestPDUPort.setRequestID(new Integer32(0));
                requestPDUPort.set(0, vb2);

                requestPDUNev.setRequestID(new Integer32(0));
                requestPDUNev.set(0, vb3);
              }
            }
            snmp.close();

          }
          catch (IOException e) {
            System.out.println("OID bejar hiba: " + e.getMessage());
          }
      }
```

```
      /* port letiltasa */
      public static void portTilt(String portOID, Address IP) throws
IOException {

          PDU pduTilt = new PDU();
          OID oid = new OID(portOID);
            /* oid */
          Variable var = new Integer32(2);
      /* ertek (2:tilt) */
          VariableBinding varBind = new VariableBinding(oid, var);
          pduTilt.add(varBind);
          pduTilt.setType(PDU.SET);
          pduTilt.setRequestID(new Integer32(2));

            TransportMapping transport = new
DefaultUdpTransportMapping();
            transport.listen();

            Snmp snmp = new Snmp(transport);

            target.setAddress(IP);
            ResponseEvent respEv = snmp.send(pduTilt, target);
      }



      /* port engedelyezese */
      public static void portEngedelyez(String portOID, Address IP)
throws IOException {

          PDU pduEnged = new PDU();
          OID oid = new OID(portOID);
            /* oid */
          Variable var = new Integer32(1);
      /* ertek (1:enged.) */
          VariableBinding varBind = new VariableBinding(oid, var);
          pduEnged.add(varBind);
          pduEnged.setType(PDU.SET);
          pduEnged.setRequestID(new Integer32(2));

            TransportMapping transport = new
DefaultUdpTransportMapping();
            transport.listen();

            Snmp snmp = new Snmp(transport);

            target.setAddress(IP);
            ResponseEvent respEv = snmp.send(pduEnged, target);
      }



      /* SNMP lekerdezes tetsz. OID-re */
      public static String snmpGet(String string) throws IOException {

            String nev = null;
            PDU pdu = new PDU();
            pdu.setType(PDU.GET);
            pdu.add(new VariableBinding(new OID(string)));
```

```
            TransportMapping transport = new
DefaultUdpTransportMapping();
            transport.listen();


            try {
                  Snmp snmp = new Snmp(transport);
                  ResponseEvent respEv = snmp.send(pdu, target);

                  PDU response = respEv.getResponse();
                  nev = response.get(0).getVariable().toString();

            } catch(Exception e) {
                  System.out.println("Hiba: " + e.getMessage());
            }
            return nev;
      }

}
```

A HTMLFormatter osztály forráskódja

```
/* kod forrasa: http://www.vogella.de/articles/Logging/article.html */

package Log;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Formatter;
import java.util.logging.Handler;
import java.util.logging.Level;
import java.util.logging.LogRecord;


public class HTMLFormatter extends Formatter
{
      // This method is called for every log records
      public String format(LogRecord rec) {
            StringBuffer buf = new StringBuffer(1000);
            // Bold any levels >= WARNING
            buf.append("<tr>");
            buf.append("<td>");

            if (rec.getLevel().intValue() >= Level.WARNING.intValue()) {
                  buf.append("<b>");
                  buf.append(rec.getLevel());
                  buf.append("</b>");
            } else {
                  buf.append(rec.getLevel());
            }
            buf.append("</td>");
            buf.append("<td>");
            buf.append(calcDate(rec.getMillis()));
            buf.append(' ');
            buf.append(formatMessage(rec));
```

```
            buf.append('\n');
            buf.append("<td>");
            buf.append("</tr>\n");
            return buf.toString();
      }


      private String calcDate(long millisecs) {
            SimpleDateFormat date_format = new SimpleDateFormat("MMM
dd,yyyy HH:mm");
            Date resultdate = new Date(millisecs);
            return date_format.format(resultdate);
      }

      // This method is called just after the handler using this
      // formatter is created
      public String getHead(Handler h) {
            return "<HTML>\n<HEAD>\n" + (new Date()) +
"\n</HEAD>\n<BODY>\n<PRE>\n"
                        + "<table border>\n  "
                        + "<tr><th>Type</th><th>Log uzenet</th></tr>\n";
      }

      // This method is called just after the handler using this
      // formatter is closed
      public String getTail(Handler h) {
            return "</table>\n  </PRE></BODY>\n</HTML>\n";
      }
}
```

A sajatLogger osztály forráskódja

```
/* kod forrasa: http://www.vogella.de/articles/Logging/article.html */


package Log;

import java.io.IOException;
import java.util.logging.FileHandler;
import java.util.logging.Formatter;

import java.util.logging.Logger;



public class sajatLogger {
      private static FileHandler fileHTML;
      private static Formatter formatterHTML;



      public static void setup() throws IOException {

            // Create Logger
            Logger logger = Logger.getLogger("");
            fileHTML = new FileHandler("Log.html");
```

```
            // Create HTML Formatter
            formatterHTML = new HTMLFormatter();
            fileHTML.setFormatter(formatterHTML);
            logger.addHandler(fileHTML);

      }
}
```

A felhasznalo osztály forráskódja

```
package user;

public class felhasznalo implements Comparable<felhasznalo> {
      private String nev;
      private String jelszo;


      public felhasznalo(String nev, String jelszo) {
            this.nev = nev;
            this.jelszo = jelszo;
      }


      public String getFelhNev() {
            return this.nev;
      }


      public String getJelszo() {
            return this.jelszo;
      }


      public String toString() {
            return "Felhasználó: " + this.nev + "  jelszo: " +
this.jelszo;
      }


      @Override
      public int hashCode() {
            final int prime = 31;
            int result = 1;
            result = prime * result + ((jelszo == null) ? 0 :
jelszo.hashCode());
            result = prime * result + ((nev == null) ? 0 :
nev.hashCode());
            return result;
      }


      @Override
      public boolean equals(Object obj) {
            if (this == obj)
                  return true;
            if (obj == null)
                  return false;
            if (getClass() != obj.getClass())
                  return false;
            felhasznalo other = (felhasznalo) obj;
```

```
            if (jelszo == null) {
                  if (other.jelszo != null)
                        return false;
            } else if (!jelszo.equals(other.jelszo))
                  return false;
            if (nev == null) {
                  if (other.nev != null)
                        return false;
            } else if (!nev.equals(other.nev))
                  return false;
            return true;
      }


      @Override
      public int compareTo(felhasznalo felhMasik) {
            return this.nev.compareTo(felhMasik.nev);
      }

}
```

A felhasznaloTarol osztály forráskódja

```
package user;

import java.util.*;

public class felhasznaloTarol {
      private ArrayList<felhasznalo> felhasznalok;


      public felhasznaloTarol() {
            this.felhasznalok = new ArrayList<felhasznalo>();
      }


      public void hozzaad(felhasznalo uj) {
            if(uj == null)
                  return;

            if(this.felhasznalok.contains(uj))
                  return;

            this.felhasznalok.add(uj);
      }


      public void torol(felhasznalo torlendo) {
            if(torlendo == null)
                  return;

            if(!this.felhasznalok.contains(torlendo))
                  return;

            this.felhasznalok.remove(torlendo);
      }


      public int meret() {
```

```
            return this.felhasznalok.size();
      }


      public felhasznalo[] getTombben() {
            felhasznalo[] tomb = new
felhasznalo[this.felhasznalok.size()];


            return this.felhasznalok.toArray(tomb);
      }



      /* nev szerinti sorrendben */
      public felhasznalo[] getTombRendezve() {
            felhasznalo[] tomb = new
felhasznalo[this.felhasznalok.size()];

            this.felhasznalok.toArray(tomb);
            Arrays.sort(tomb);

            return tomb;
      }


      /* adott indexu elem visszaad */
      public felhasznalo getFelhasznalo(int index) {
            return this.felhasznalok.get(index);
      }


      /* adott felhasznalo keresese */
      public felhasznalo elemKeres(String nev, String jelszo) {
            felhasznalo keresendo = new felhasznalo(nev, jelszo);
            int hol = this.felhasznalok.indexOf(keresendo);

            if(hol == -1)
                  return null;

            return this.felhasznalok.get(hol);
      }




      /* adott felhasznalo keresese vane ilyen */
      public boolean elemKeresVane(String nev, String jelszo) {
            felhasznalo keresendo = new felhasznalo(nev, jelszo);
            int hol = this.felhasznalok.indexOf(keresendo);

            if(hol == -1)
                  return false;

            return true;
      }

}
```

Az Utemezo osztály forráskódja

```
package Utemezo;


import java.awt.Toolkit;
import java.util.Timer;
import java.util.TimerTask;

import GUI.foablak;


public class Utemezo extends TimerTask {
	private Timer idozito;


	public Utemezo(int secounds) {
		Toolkit.getDefaultToolkit();
		this.idozito = new Timer();
		idozito.schedule(this, 0, secounds*500);

	}


	@Override
	public void run() {
			if(true) {

				int kijSorIndex =
foablak.getTablazat().getSelectedRow();
				int osszesSorFrissElott =
foablak.getTablazat().getRowCount();


				foablak.tablazatFrissites();
				int osszesSorFrissUtan =
foablak.getTablazat().getRowCount();


				if(kijSorIndex >= osszesSorFrissUtan) {
					foablak.getTablazat().clearSelection();
				} else if(osszesSorFrissUtan <
osszesSorFrissElott || kijSorIndex == -1){
					foablak.getTablazat().clearSelection();
				} else if(kijSorIndex != -1) {

	foablak.getTablazat().setRowSelectionInterval(kijSorIndex,
kijSorIndex);
				}

				if(foablak.kitiltVanE() == true)
					foablak.automataPortLetiltas();

			}
	}



}
```

Az **adatFrisstesBeallit osztály forráskódja**

```
package GUI;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JInternalFrame;
import javax.swing.JFormattedTextField;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JOptionPane;

import Utemezo.Utemezo;



public class adatFrisstesBeallit extends JInternalFrame implements
ActionListener {
	private static final long serialVersionUID = 1L;
	private JFormattedTextField formattedTextFieldFrissit;
	private JButton btnBezar;

	private static String frissitesAdat = "0";
	private static JButton btnOk;
	private static boolean utemezoVane = false; /* be van-e kapcsolva
az itemezo */




	/* Van-e utemezo beallitva */
	public static boolean utemezesVanE() {
		return utemezoVane;
	}



	/* frissitesi parameter visszaadasa */
	public static int getFrssitadat() {
		return Integer.parseInt(frissitesAdat);
	}



	public adatFrisstesBeallit() {
		setTitle("Adat frss\u00EDt\u00E9s
be\u00E1ll\u00EDt\u00E1sa");
		setBounds(100, 100, 320, 190);
		getContentPane().setLayout(null);

		formattedTextFieldFrissit = new JFormattedTextField();
		formattedTextFieldFrissit.setToolTipText("Itt adhatja meg
m\u00E1sodpercben, hogy milyen id\u0151k\u00F6z\u00F6nk\u00E9nt
friss\u00FCljenek a tabl\u00E1zat adatai.");
		formattedTextFieldFrissit.setBounds(48, 51, 152, 20);
		getContentPane().add(formattedTextFieldFrissit);
```

```
            JLabel lblFrisstsIdkze = new JLabel("Friss\u00EDt\u00E9s
id\u0151k\u00F6ze (m\u00E1sodperc):");
            lblFrisstsIdkze.setBounds(10, 26, 190, 14);
            getContentPane().add(lblFrisstsIdkze);

            btnOk = new JButton("OK");
            btnOk.setActionCommand("OKGomb");
            btnOk.setBounds(54, 104, 89, 23);
            getContentPane().add(btnOk);

            btnBezar = new JButton("Bez\u00E1r");
            btnBezar.setActionCommand("bezarGomb");
            btnBezar.setBounds(153, 104, 89, 23);
            getContentPane().add(btnBezar);

            this.btnBezar.addActionListener(this);
            btnOk.addActionListener(this);


            /* ha van utemezes beallitva, akkor igaz */
          /*if(!frissitesAdat.isEmpty()) {
                utemezoVane = true;
            }*/
      }




      @Override
      public void actionPerformed(ActionEvent e) {

            /* bezaras */
            if(e.getActionCommand().equals("bezarGomb")) {

                  /* ha nincs benne semmi, akkor nel egyen az utemezo
igaz */
                  if(frissitesAdat.isEmpty()) {
                        utemezoVane = false;
                  } else if(!szamE(frissitesAdat)) {
                        utemezoVane = false;
                  }

      ((JInternalFrame)getRootPane().getParent()).doDefaultCloseAction();
            }




            /* adat kiolvasas */
            if(e.getActionCommand().equals("OKGomb")) {
                  frissitesAdat =
this.formattedTextFieldFrissit.getText();

                  if(!szamE(frissitesAdat)) {
                        JOptionPane.showMessageDialog(this,"Hibás a
megadott adat!", "Hiba", JOptionPane.WARNING_MESSAGE);
                        utemezoVane = false;
                        return;
```

```
                  }


                  /* ha nem ures es szam, akkor utemezo futtatasa */
                  if(!frissitesAdat.isEmpty() && szamE(frissitesAdat)) {

                        new Utemezo(getFrssitadat());       /* utemezo
elinditasa */
                        utemezoVane = true;
                        foablak.gombTilt();
                        this.formattedTextFieldFrissit.setText("");

      ((JInternalFrame)getRootPane().getParent()).doDefaultCloseAction();
                  }
            }

      }




      /* csak szam-e a megkapott adat */
      public static boolean szamE(String szoveg) {
            for(char betu : szoveg.toCharArray()) {
                  if(!Character.isDigit(betu)) {
                        return false;
                  }
            }
            return true;
      }

}
```

A felhHozzaAd osztály forráskódja

```
package GUI;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;

import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.JButton;


public class felhHozzaAd extends JInternalFrame implements ActionListener
{
      private static final long serialVersionUID = 1L;
      private JTextField textFieldFelhNev;
      private JPasswordField passwordFieldFelhJelszo;
      private static felhHozzaAd frame;
```

```
	private JButton btnHozzad;
	private JButton btnBezar;
	private String felhNev;
	private String felhJelszo;



	public felhHozzaAd() {
		setTitle("Felhaszn\u00E1l\u00F3 hozz\u00E1ad\u00E1sa");
		setBounds(100, 100, 320, 190);
		getContentPane().setLayout(null);


		JLabel lblFelhasznliNv = new JLabel("Felhaszn\u00E1l\u00F3i
n\u00E9v:");
		lblFelhasznliNv.setBounds(10, 48, 106, 14);
		getContentPane().add(lblFelhasznliNv);

		JLabel lblJelsz = new JLabel("Jelsz\u00F3:");
		lblJelsz.setBounds(10, 79, 106, 14);
		getContentPane().add(lblJelsz);

		textFieldFelhNev = new JTextField();
		textFieldFelhNev.setBounds(145, 45, 150, 20);
		getContentPane().add(textFieldFelhNev);
		textFieldFelhNev.setColumns(10);

		passwordFieldFelhJelszo = new JPasswordField();
		passwordFieldFelhJelszo.setBounds(145, 76, 150, 20);
		getContentPane().add(passwordFieldFelhJelszo);
		passwordFieldFelhJelszo.setColumns(10);

		btnHozzad = new JButton("Hozz\u00E1ad");
		btnHozzad.setBounds(53, 120, 89, 23);
		btnHozzad.setActionCommand("HozzaadGomb");
		getContentPane().add(btnHozzad);

		btnBezar = new JButton("Bez\u00E1r");
		btnBezar.setBounds(152, 120, 89, 23);
		btnBezar.setActionCommand("BezarGomb");
		getContentPane().add(btnBezar);



	   this.btnBezar.addActionListener(this);
	   this.btnHozzad.addActionListener(this);
	}



	/* felhasznalo fajlba irasa */
	public static void fajlbaIr(String beFajl, String nev, String
jelszo) throws IOException{
		BufferedWriter bw = null;
		bw = new BufferedWriter(new FileWriter(beFajl, true));
		BufferedReader br = null;
		br = new BufferedReader(new FileReader(beFajl));
```

```
            /* ha mar van, akkor nem adja hozza */
            String line = null;
          while ((line = br.readLine()) != null) {
            if (line.trim().contains(nev) &&
line.trim().contains(jelszo)) {
                  JOptionPane.showMessageDialog(frame,"Ilyen felhasználó
már létezik!", "Hiba", JOptionPane.WARNING_MESSAGE);
                        return;
               }
          }


            bw.append(nev + "|" + jelszo + "\r\n");         /* hozzafuz */
            bw.flush();
      }




      @Override
      public void actionPerformed(ActionEvent e) {

            /* bezaras */
            if(e.getActionCommand().equals("BezarGomb")) {

      ((JInternalFrame)getRootPane().getParent()).doDefaultCloseAction();
            }


            /* felhasznalo hozzaad gomb */
            if(e.getActionCommand().equals("HozzaadGomb")) {
                  this.felhNev = this.textFieldFelhNev.getText();
                  this.felhJelszo = new
String(this.passwordFieldFelhJelszo.getPassword());


                  /* adatbevitel vizsgalat */
                  if(felhNev.isEmpty()&& felhJelszo.isEmpty()) {
                        JOptionPane.showMessageDialog(frame,"Nem adott
meg felhasználói nevet és jelszót sem!", "Hiba",
JOptionPane.ERROR_MESSAGE);
                        return;
                  } else if(felhNev.isEmpty()) {
                        JOptionPane.showMessageDialog(frame,"Nem adott
meg felhasználói nevet!", "Hiba", JOptionPane.ERROR_MESSAGE);
                        return;
                  } else if(felhJelszo.isEmpty()) {
                        JOptionPane.showMessageDialog(frame,"Nem adott
meg jelszót!", "Hiba", JOptionPane.ERROR_MESSAGE);
                        return;
                  }



                  try {
                        fajlbaIr("felh.auth", this.felhNev,
this.felhJelszo);
```

```
                        this.textFieldFelhNev.setText("");
                        this.passwordFieldFelhJelszo.setText("");
                    } catch (IOException e1) {
                        JOptionPane.showMessageDialog(frame,"I/O hiba!",
"Hiba", JOptionPane.ERROR_MESSAGE);
                    }
                }

        }
}
```

A felhTorol osztály forráskódja

```
package GUI;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JButton;

public class felhTorol extends JInternalFrame implements ActionListener {
      private static final long serialVersionUID = 1L;
      private JTextField textFieldFelhNev;
      private JButton btnTorles;
      private JButton btnBezras;
      private static felhTorol frame;



      public felhTorol() {
            setTitle("Felhaszn\u00E1l\u00F3 t\u00F6rl\u00E9se");
            setBounds(100, 100, 320, 190);
            getContentPane().setLayout(null);

            JLabel lblFelhasznlNeve = new JLabel("Felhaszn\u00E1l\u00F3
neve:");
            lblFelhasznlNeve.setBounds(10, 50, 127, 14);
            getContentPane().add(lblFelhasznlNeve);

            textFieldFelhNev = new JTextField();
            textFieldFelhNev.setBounds(116, 47, 150, 20);
            getContentPane().add(textFieldFelhNev);
            textFieldFelhNev.setColumns(10);

            btnTorles = new JButton("T\u00F6rl\u00E9s");
            btnTorles.setActionCommand("TorlesGomb");
            btnTorles.setBounds(48, 96, 89, 23);
```

```
            getContentPane().add(btnTorles);

            btnBezras = new JButton("Bez\u00E1r\u00E1s");
            btnBezras.setActionCommand("BezarasGomb");
            btnBezras.setBounds(150, 96, 89, 23);
            getContentPane().add(btnBezras);

            this.btnTorles.addActionListener(this);
            this.btnBezras.addActionListener(this);
      }



      /* sort torlo metodus */
      /* kod forrasa: http://www.javadb.com/remove-a-line-from-a-text-
file */
      public static void sortTorol(String fajl, String torlendoSor) {
            try {
                  File beFile = new File(fajl);

                  if (!beFile.isFile()) {
                        JOptionPane.showMessageDialog(frame,"A fájl nem
létezik vagy hibás!", "Hiba!", JOptionPane.ERROR_MESSAGE);
                    return;
                  }


                  /* atmenetei fajl */
                  File tempFile = new File(beFile.getAbsolutePath() +
".tmp");

                  BufferedReader br = new BufferedReader(new
FileReader(beFile));
                  PrintWriter pw = new PrintWriter(new
FileWriter(tempFile));

                  String line = null;



                  /* az eredti fajl tartalamanak irasa az atmeneti fajlba
a torlendo sor kivetelevel */
                  while ((line = br.readLine()) != null) {
                    if (!line.trim().contains(torlendoSor)) {
                        pw.println(line);
                        pw.flush();
                    }
                  }
                  pw.close();
                  br.close();


                  /* eredeti fajl torlese */
                  if (!beFile.delete()) {
                        JOptionPane.showMessageDialog(frame,"A fájl
törlése nem sikerült!", "Hiba!", JOptionPane.ERROR_MESSAGE);
                    return;
                  }

                  /* az atmeneti fajl atnevezese az eredetire */
```

```
                    if (!tempFile.renameTo(beFile))
                                JOptionPane.showMessageDialog(frame,"A fáj
átnevezése nem sikerült!", "Hiba!", JOptionPane.ERROR_MESSAGE);


                } catch (FileNotFoundException ex) {
                        JOptionPane.showMessageDialog(frame,"Hiba!",
ex.getMessage(), JOptionPane.ERROR_MESSAGE);

                } catch (IOException ex) {
                        JOptionPane.showMessageDialog(frame,"Hiba!",
ex.getMessage(), JOptionPane.ERROR_MESSAGE);

                }
        }




      @Override
      public void actionPerformed(ActionEvent e) {
            String torlendo = this.textFieldFelhNev.getText();


            if(e.getActionCommand().equals("BezarasGomb")) {

      ((JInternalFrame)getRootPane().getParent()).doDefaultCloseAction();
            }



            if(e.getActionCommand().equals("TorlesGomb")) {
                  if(torlendo.isEmpty()) {
                        JOptionPane.showMessageDialog(this,"Nem adott meg
felhasználói nevet!", "Hiba!", JOptionPane.WARNING_MESSAGE);
                        return;
                  }

                  if(torlendo.contains("admin")) {
                        JOptionPane.showMessageDialog(this,"Az admin nem
törölhető!", "Hiba!",  JOptionPane.WARNING_MESSAGE);
                        return;
                  }

                  sortTorol("felh.auth", torlendo);
                  this.textFieldFelhNev.setText("");
            }



      }
}
```

A foablak osztály forráskódja

```
/* Foablak, foprogram */

package GUI;



import java.awt.AWTException;
import java.awt.Desktop;
import java.awt.EventQueue;
import java.awt.Image;
import java.awt.MenuItem;
import java.awt.PopupMenu;
import java.awt.SystemTray;
import java.awt.Toolkit;
import java.awt.TrayIcon;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import javax.swing.table.TableRowSorter;
import javax.swing.JDesktopPane;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JMenu;
import javax.swing.JOptionPane;
import javax.swing.ListSelectionModel;
import java.awt.event.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.JTable;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.JButton;
import org.snmp4j.smi.*;

import CAM_tabla.*;
import Log.proba;
import Log.sajatLogger;
```

```
public class foablak extends JFrame implements ActionListener,
ListSelectionListener {
	private static final long serialVersionUID = 1L;

	private final static Logger LOGGER =
Logger.getLogger(proba.class.getName());
	private File logFajl;
	private static boolean kitiltasVanE = false;



	/* GUI elemek, valtozok */
	private static JFrame frame;
	private JPanel contentPane;
	private JMenuBar menuBar;
	private JMenuItem mntmKilepes;
	private JMenu mnFajl;
	private JMenuItem mntmFelhHozzad;
	private JMenuItem mntmFelhTorol;
	private static JMenu mnFelhasznalok;
	private JScrollPane scrollPane;
	private JTextField textFieldMac;
	private JTextField textFieldPort;
	private JTextField textFieldNev;
	private JButton btnKitilt;
	private JButton btnEngedelyez;
	private JButton btnKijelolesLeptetes;
	private static JTable tablazat;
	private static TableModel tablaModel;
	private static JDesktopPane desktop;

	/* SNMP valtozok */
	private static String eszkIP = login.getEszkozIP();
	private static Address targetAddress; /* eszkoz IP/SNMP port */
	private static JButton btnAdatFrissit;



	private String[][] sor = new String[][] { };
	private String[] oszlop = new String[] { "MAC cim", "Port", "VLAN"
};
	private static JMenuItem mntmAdatFrissites;
	private JMenu mnBeallitasok;
	private JMenuItem mntmKitiltottAllomasok;
	private static String kitiltottMac;
	private static String kitiltottPort;
	private static String kitiltottIP;
	private static String kitiltottNev;
	private JMenuItem mntmUjEszkoz;
	private JTextField textFieldIP;
	private JMenu mnNaplozas;
	private JMenuItem mntmLogFajlMegnyitasa;



	/* Windows-os kinezet beallitas */
	/*{
		try {
```

```
      javax.swing.UIManager.setLookAndFeel("com.sun.java.swing.plaf.windo
ws.WindowsLookAndFeel");
            } catch(Exception e) {
                  e.printStackTrace();
            }
      }*/



      /* Foprogram */
      public static void main(String[] args) throws IOException {
            EventQueue.invokeLater(new Runnable() {
                  public void run() {
                        try {
                              frame = new foablak();
                              frame.setVisible(true);
                        } catch (Exception e) {
                              e.printStackTrace();
                        }
                  }
            });
      } /* main vege */



      public foablak() {
            setTitle("SNMP-menedzsment szoftver @ Miskolci Egyetem, Tompa
Tam\u00E1s, 2011");
            setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
            setBounds(100, 100, 600, 450);

            desktop = new JDesktopPane();
            desktop.setLayout(null);
            contentPane = new JPanel();
            contentPane.setLayout(null);
            setContentPane(desktop);

            menuBar = new JMenuBar();
            setJMenuBar(menuBar);

            mnFajl = new JMenu("F\u00E1jl");
            mnFajl.setActionCommand("FajlMenu");
            menuBar.add(mnFajl);

            mntmKilepes = new JMenuItem("Kil\u00E9p\u00E9s");
            mntmKilepes.setActionCommand("Kilepes");

            mnFajl.add(mntmKilepes);

            mnFelhasznalok = new JMenu("Felhaszn\u00E1l\u00F3");
            mnFelhasznalok.setActionCommand("Felhasznalo");
            menuBar.add(mnFelhasznalok);

            mntmFelhHozzad = new JMenuItem("Hozz\u00E1ad");
            mntmFelhHozzad.setToolTipText("Itt adhat hozz\u00E1 \u00FAj
felhaszn\u00E1l\u00F3t");
            mntmFelhHozzad.setActionCommand("FelhHozzaad");
            mnFelhasznalok.add(mntmFelhHozzad);
```

```
            mntmFelhTorol = new JMenuItem("T\u00F6r\u00F6l");
            mntmFelhTorol.setActionCommand("FelhTorol");
            mnFelhasznalok.add(mntmFelhTorol);

            mnNaplozas = new JMenu("Napl\u00F3z\u00E1s");
            menuBar.add(mnNaplozas);

            mntmLogFajlMegnyitasa = new JMenuItem("Log f\u00E1jl
megnyit\u00E1sa");
            mntmLogFajlMegnyitasa.setActionCommand("logMegnyit");
            mnNaplozas.add(mntmLogFajlMegnyitasa);
            this.mntmLogFajlMegnyitasa.addActionListener(this);

            mnBeallitasok = new JMenu("Be\u00E1ll\u00EDt\u00E1sok");
            mnBeallitasok.setActionCommand("BeallitasokMenu");
            menuBar.add(mnBeallitasok);


            mntmAdatFrissites = new JMenuItem("Adat
friss\u00EDt\u00E9s");
            mntmAdatFrissites.setActionCommand("adatFrissites");
            mnBeallitasok.add(mntmAdatFrissites);

            mntmKitiltottAllomasok = new JMenuItem("Kitiltott
\u00E1llom\u00E1sok");

      mntmKitiltottAllomasok.setActionCommand("kitiltottAllomasok");
            mnBeallitasok.add(mntmKitiltottAllomasok);

            mntmUjEszkoz = new JMenuItem("\u00DAj eszk\u00F6z");
            mntmUjEszkoz.setActionCommand("ujEszkoz");
            mnBeallitasok.add(mntmUjEszkoz);

            textFieldPort = new JTextField();
            textFieldPort.setEditable(false);
            textFieldPort.setBounds(155, 330, 125, 20);
            desktop.add(textFieldPort);
            textFieldPort.setColumns(10);

            textFieldNev = new JTextField();
            textFieldNev.setEditable(false);
            textFieldNev.setBounds(290, 330, 125, 20);
            desktop.add(textFieldNev);
            textFieldNev.setColumns(10);

            textFieldMac = new JTextField();
            textFieldMac.setEditable(false);
            textFieldMac.setBounds(20, 330, 125, 20);
            desktop.add(textFieldMac);
            textFieldMac.setColumns(10);

            btnKitilt = new JButton("Kitilt");
            btnKitilt.setActionCommand("KitiltGomb");
            btnKitilt.setBounds(180, 361, 108, 23);
            desktop.add(btnKitilt);

            btnEngedelyez = new JButton("Enged\u00E9lyez");
            btnEngedelyez.setActionCommand("EngedelyezGomb");
            btnEngedelyez.setBounds(297, 361, 108, 23);
            desktop.add(btnEngedelyez);
```

```
            btnKijelolesLeptetes = new JButton("Kijel\u00F6l\u00E9st
l\u00E9ptet");
            btnKijelolesLeptetes.setActionCommand("kijelolesLeptetGomb");
            btnKijelolesLeptetes.setBounds(10, 255, 130, 23);
            desktop.add(btnKijelolesLeptetes);


            JLabel lblKijeloltAllomas = new JLabel("A kijel\u00F6lt
\u00E1llom\u00E1s adatai:");
            lblKijeloltAllomas.setBounds(10, 305, 143, 14);
            desktop.add(lblKijeloltAllomas);

            btnAdatFrissit = new JButton("Adatok friss\u00EDt\u00E9se");
            btnAdatFrissit.setActionCommand("adatFrissitGomb");
            btnAdatFrissit.setBounds(147, 255, 151, 23);
            desktop.add(btnAdatFrissit);



            /* ha nem admin, akkor tilt */
            if(!login.adminE()) {
                  mnFelhasznalok.setEnabled(false);
                  this.btnEngedelyez.setEnabled(false);
                  this.btnKitilt.setEnabled(false);
                  this.mntmKitiltottAllomasok.setEnabled(false);
            }
            this.mntmUjEszkoz.setEnabled(true);


            tablazatFelepit();
            systemTray();


            try {
                  targetAddress = new UdpAddress(eszkIP);
                  cam_tabla_eloallit.snmpKapcs(targetAddress);
            } catch (IOException e) {
                  JOptionPane.showMessageDialog(this,"Kapcsolódási
hiba!", "Hiba", JOptionPane.WARNING_MESSAGE);
            }


            this.mntmKilepes.addActionListener(this);
            this.mntmFelhHozzad.addActionListener(this);
            this.mntmFelhTorol.addActionListener(this);
            mntmAdatFrissites.addActionListener(this);
            this.btnKijelolesLeptetes.addActionListener(this);
            tablazat.getSelectionModel().addListSelectionListener(this);

            this.btnEngedelyez.addActionListener(this);
            this.btnKitilt.addActionListener(this);
            btnAdatFrissit.addActionListener(this);
            this.mntmKitiltottAllomasok.addActionListener(this);
            this.mntmUjEszkoz.addActionListener(this);
      }



      public static JFrame getFrame() {
```

```
            return frame;
      }


      /* tablazat es scrollPane letrehozasa */
      public void tablazatFelepit() {
            scrollPane = new JScrollPane();
            scrollPane.setAutoscrolls(true);
            scrollPane.setBounds(10, 27, 572, 217);
            desktop.add(scrollPane);

          tablazat = new JTable();
          tablazat.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            scrollPane.setViewportView(tablazat);
            tablazat.setName("");


            sor = new String[][] { };
            oszlop = new String[] { "Állomás MAC cim", "Eszköz port",
"Eszköz név", "Eszköz IP" };




            /* tablamodel, editalas letiltas */
            tablaModel = new DefaultTableModel(sor, oszlop) {
                  private static final long serialVersionUID = 1L;

                  @Override
                  public boolean isCellEditable(int row, int column) {
                        return false;
                  }
            };


            /* oszlop mozgatas leiltas */
            tablazat.getTableHeader().setReorderingAllowed(false);


      tablazat.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
            tablazat.setRowSorter(new
TableRowSorter<TableModel>(tablaModel));
            tablazat.setAutoCreateRowSorter(true);
            tablazat.setModel(tablaModel);

            textFieldIP = new JTextField();
            textFieldIP.setEditable(false);
            textFieldIP.setBounds(425, 330, 125, 20);
            desktop.add(textFieldIP);
            textFieldIP.setColumns(10);


      }



      /* tablazat visszaadasa */
      public static JTable getTablazat() {
```

```
            return tablazat;
      }


      /* kitiltott gep mac cime */
      public static String kitiltottGepMac() {
            return kitiltottMac;
      }

      /* kitiltott gep switch portja */
      public static String kitiltottGepPort() {
            return kitiltottPort;
      }


      public static void setTarget(Address cim) {
            targetAddress = cim;
      }

      public static Address getTarget() {
            return targetAddress;
      }



      public static boolean kitiltVanE() {
            return kitiltasVanE;
      }


      /* adatok tablazathoz adasa */
      public static void tablahozAd(String macCim, String port, String
nev) {

            if (macCim.isEmpty() || port.isEmpty()|| nev.isEmpty()) {
                  return;
            }

            DefaultTableModel model =
(DefaultTableModel)tablazat.getModel();


            /* ket ugyanolyan MAC-et nem ad hozza */
            boolean vanEMar  = false;
            for (int j = 0; j < model.getRowCount(); j++) {
                  if (macCim.equals(model.getValueAt(j, 0)) ) {
                        vanEMar = true;
                        break;
                  }
            }



            if (!vanEMar) {
                  model.addRow(new String[] {macCim, port, nev,
targetAddress.toString()});
            }


            /*Address[] iptomb = ujEszkoz.getEszkIP();
            for(Address ipCim : iptomb) {
```

```
                  if(ipCim != null)
                        model.addRow(new String[] {macCim, port, nev,
ipCim.toString()});
            }*/


      }




      /* tablazat adatinak frissitese */
      public static void tablazatFrissites() {

            try {
                  /* ha egy allomas csatlakozik */

                  Address[] IPTomb = ujEszkoz.getEszkIP();

                  cam_tabla_eloallit.snmpKapcs(targetAddress);


                  if(ujEszkoz.getEszkIP() != null) {
                        for(int i=0;i<IPTomb.length;i++) {
                              if(IPTomb[i] != null)

      cam_tabla_eloallit.snmpKapcs(IPTomb[i]);
                        }
                  }




                  if(cam_tabla_eloallit.camTablaMerete() >
tablazat.getRowCount())
                        ((AbstractTableModel)
tablaModel).fireTableDataChanged();


                  /* ha egy allomas eltunt */
                  if(cam_tabla_eloallit.camTablaMerete() <
tablazat.getRowCount()) {

                        /*!!!*/
                        int rowCount = tablaModel.getRowCount();
                          for(int i=0;i<rowCount;i++ ){
                                ((DefaultTableModel)
tablaModel).removeRow(0);
                          }

                        ((DefaultTableModel)
tablaModel).getDataVector().removeAllElements();


                        cam_tabla_eloallit.snmpKapcs(targetAddress);

                        if(ujEszkoz.getEszkIP() != null) {
                              for(int i=0;i<IPTomb.length;i++) {
```

```
                                    if(IPTomb[i] != null)

      cam_tabla_eloallit.snmpKapcs(IPTomb[i]);
                                }

                            }

                            tablazat.repaint();
                        }
                } catch (IOException ex) {
                    JOptionPane.showMessageDialog(frame,"A táblázat
adatinak a frissítése nem sikerült!", "Hiba", JOptionPane.ERROR_MESSAGE);
                }
        }




        /* Ertesitesi terulet funkcio */
        public void systemTray() {

                final TrayIcon trayIcon;

                if (SystemTray.isSupported()) {
                    SystemTray tray = SystemTray.getSystemTray();
                    Image kep =
Toolkit.getDefaultToolkit().getImage("switch.gif");

                    /* eger esemenyfigyelo */
                    MouseListener mouseListener = new MouseListener() {
                        public void mouseClicked(MouseEvent e) {

                        }

                            @Override
                            public void mouseEntered(MouseEvent arg0) {

                            }

                            @Override
                            public void mouseExited(MouseEvent e) {

                            }

                            @Override
                            public void mousePressed(MouseEvent e) {
                                    int kattintas = e.getClickCount();

                                    if(kattintas == 2 ) {

foablak.getFrames()[1].setVisible(true); /* foablakot megjeleniti */
                                    }
                            }

                            @Override
                            public void mouseReleased(MouseEvent e) {

                            }
                    };
```

```
                /* kilepes esemeny */
                ActionListener kilepesListener = new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        System.exit(0);
                    }
                };

                /* teljes meret esemeny */
                ActionListener teljesMeretListener = new ActionListener()
{
                    public void actionPerformed(ActionEvent e) {
                         foablak.getFrames()[1].setVisible(true);
                    }
                };


                PopupMenu popup = new PopupMenu();
                MenuItem teljesMeret = new MenuItem("Teljes méret");
                teljesMeret.addActionListener(teljesMeretListener);
                popup.add(teljesMeret);

                MenuItem kilepes = new MenuItem("Kilépés");
                kilepes.addActionListener(kilepesListener);
                popup.add(kilepes);


                trayIcon = new TrayIcon(kep, "SNMP-menedzsment", popup);

                trayIcon.setImageAutoSize(true);
                trayIcon.addMouseListener(mouseListener);

                try {
                    tray.add(trayIcon);
                } catch (AWTException e) {
                        JOptionPane.showMessageDialog(this,"Az ikont nem
sikerült hozzáadni!", "Hiba", JOptionPane.WARNING_MESSAGE);
                }
            } else {
                  JOptionPane.showMessageDialog(this,"Az Ön rendszere nem
támogtaja az értesítési terület funkciót!", "Hiba",
JOptionPane.WARNING_MESSAGE);
            }
      }




      /* tablazatban kijelolt sor adatainak kiirasa */
      /* akkor hivodik meg, ha a kijelolt sorban az ertek valtozott */
      @Override
      public void valueChanged(ListSelectionEvent e) {
            int kijeloltSorIndexe = tablazat.getSelectedRow();


            if(kijeloltSorIndexe != -1) {
                  String Mac = tablazat.getValueAt(kijeloltSorIndexe,
0).toString();
```

```
                String Port = tablazat.getValueAt(kijeloltSorIndexe,
1).toString();
                String Nev = tablazat.getValueAt(kijeloltSorIndexe,
2).toString();
                String IP = tablazat.getValueAt(kijeloltSorIndexe,
3).toString();
                this.textFieldMac.setText(Mac);
                this.textFieldPort.setText(Port);
                this.textFieldNev.setText(Nev);
                this.textFieldIP.setText(IP);
          } else if(kijeloltSorIndexe == -1) {
                this.textFieldMac.setText("");
                this.textFieldPort.setText("");
                this.textFieldNev.setText("");
                this.textFieldIP.setText("");
          }
     }



     /* Frissites menu és manual frissites gombok letiltasa */
     public static void gombTilt() {
          mntmAdatFrissites.setEnabled(false);
          btnAdatFrissit.setEnabled(false);
     }




     /* kitiltott gep fajlba irasa */
     public static void fajlbaIrKitiltott(String beFajl, String mac,
String port, String nev, String ip) throws IOException{
          BufferedWriter bw = null;
          bw = new BufferedWriter(new FileWriter(beFajl, true));

          bw.append(mac + "|" + port + "|" + nev + "|" + ip + "\r\n");
          /* hozzafuz */
          bw.flush();
     }



     /* automata gepkitiltas */
     public static void automataPortLetiltas() {
          int tablaMeret = tablaModel.getRowCount();

          for(int i=0;i<tablaMeret;i++) {
                String port;
                String portOID;
                Address ip;
                String nev;
                String mac;
                BufferedReader in = null;

                try {
                     String sor;
                     in = new BufferedReader(new FileReader(new
File("kitiltott.txt")));
```

```
                        while((sor = in.readLine()) != null) {
                            Scanner sc = new Scanner(sor);
                            sc.useDelimiter("\\|");
                            mac = sc.next();


                            if (mac.equals(tablaModel.getValueAt(i,
0))) {
                                  port = tablaModel.getValueAt(i,
1).toString();
                                  ip = new
UdpAddress(tablaModel.getValueAt(i, 3).toString());
                                  portOID = ".1.3.6.1.2.1.2.2.1.7." +
port;
                                  nev = tablaModel.getValueAt(i,
2).toString();


                                  cam_tabla_eloallit.portTilt(portOID,
ip);
                                  fajlbaIrKitiltott("kitiltott.txt",
mac, port, nev, ip.toString());

                                  LOGGER.log(Level.INFO, "Mac:" + mac +
" Port:" + port + " Nev:" + nev + " IP:" + ip +  "  állomás kitiltva az
ALKALMAZÁS által automatikusan");
                                  break;
                            }

                        }
                  } catch(FileNotFoundException ex) {

      JOptionPane.showMessageDialog(foablak.getFrame(),"Nem található az
információkat tartalmazó fájl!", "Hiba", JOptionPane.ERROR_MESSAGE);
                  } catch(IOException ex) {

      JOptionPane.showMessageDialog(foablak.getFrame(),"I/O hiba!",
"Hiba", JOptionPane.ERROR_MESSAGE);
                  } finally {
                        if(in != null)
                            try {
                                  in.close();
                            } catch (IOException e1) {
                                  e1.printStackTrace();
                            }
                  }
            }/* for vege */
      }




      /* esemenykezeles */
      @Override
      public void actionPerformed(ActionEvent e) {
```

```
            /* Kiliepes */
            if(e.getActionCommand().equals("Kilepes")) {
                  System.exit(EXIT_ON_CLOSE);
            }


            /* Felhasznalo hozzaadasa */
            if(e.getActionCommand().equals("FelhHozzaad")) {
                  felhHozzaAd felhAdFrame = new felhHozzaAd();

                setContentPane(desktop);
                desktop.add(felhAdFrame);

                felhAdFrame.setBounds(250, 100, 320, 190);;

                felhAdFrame.setVisible(true);
            }




            /* Felhasznalo torlese */
            if(e.getActionCommand().equals("FelhTorol")) {
                  felhTorol felhTorolFrame = new felhTorol();

                setContentPane(desktop);
                desktop.add(felhTorolFrame);

                felhTorolFrame.setBounds(250, 100, 320, 190);;

                felhTorolFrame.setVisible(true);
            }



            /* Adat frissites beallitasa */
            if(e.getActionCommand().equals("adatFrissites")) {
                  adatFrisstesBeallit adatFrsiistFrame = new
adatFrisstesBeallit();

                setContentPane(desktop);
                desktop.add(adatFrsiistFrame);


                adatFrsiistFrame.setBounds(250, 100, 320, 190);;
                adatFrsiistFrame.setVisible(true);
            }



            /* uj eszkoz hozzaadasa */
            if(e.getActionCommand().equals("ujEszkoz")) {
                  ujEszkoz ujEszkozFrame = new ujEszkoz();

                setContentPane(desktop);
                desktop.add(ujEszkozFrame);

                ujEszkozFrame.setBounds(180, 80, 320, 190);

                ujEszkozFrame.setVisible(true);
```

```
                this.mntmUjEszkoz.setEnabled(false);
            }



            /* kijeloles leptetese */
            if(e.getActionCommand().equals("kijelolesLeptetGomb")) {
                  int kijeloltIndexe = -1;

                  if (tablazat.getSelectedRow() !=
tablazat.getRowCount()-1) {
                        kijeloltIndexe = tablazat.getSelectedRow() + 1;
                  }

                  if (kijeloltIndexe == -1) {
                        tablazat.clearSelection();
                  } else {
                        tablazat.setRowSelectionInterval(kijeloltIndexe,
kijeloltIndexe);
                  }
            }



            /* Tablazat adatainak frissitese */
            if(e.getActionCommand().equals("adatFrissitGomb")) {
                  int kijSorIndex = tablazat.getSelectedRow();
                  int osszesSorFrissElott = tablazat.getRowCount();

                  tablazatFrissites();
                  int osszesSorFrissUtan = tablazat.getRowCount();

                  /* ha van sor kijelolve es ez a kiesett eppen eltunt
gep sora */
                  if(kijSorIndex >= osszesSorFrissUtan) {
                        tablazat.clearSelection();
                  } else if(osszesSorFrissUtan < osszesSorFrissElott ||
kijSorIndex == -1){
                        tablazat.clearSelection();
                  } else if(kijSorIndex != -1) {
                        tablazat.setRowSelectionInterval(kijSorIndex,
kijSorIndex);
                  }


                  /* ha volt kitiltva allomas,
                   * akkor figyeli csatlakozik-e ujbol,
                   * ha igen, akkor kitiltja az adott porton*/
                  if(kitiltasVanE == true)
                        automataPortLetiltas();
            }



            /* gepek engedelyezese */
            if(e.getActionCommand().equals("kitiltottAllomasok") ||
e.getActionCommand().equals("EngedelyezGomb")) {
```

```
                kitiltottGepek kitiltottGepekFrame = new
kitiltottGepek();

                setContentPane(desktop);
                desktop.add(kitiltottGepekFrame);

                kitiltottGepekFrame.setBounds(180, 80, 479, 289);


                kitiltottGepekFrame.setVisible(true);
            }


            /* Log fájl megnyitása */
            if(e.getActionCommand().equals("logMegnyit")) {
                  Desktop desktop =  Desktop.getDesktop();
                  logFajl = new File("Log.html");


                  if(!logFajl.isFile()) {
                        JOptionPane.showMessageDialog(this,"A fájl nem
található vagy sérült!", "Hiba!", JOptionPane.WARNING_MESSAGE);
                        return;
                  }


                  if (desktop.isSupported(Desktop.Action.BROWSE)) {
                        try {
                              desktop.open(logFajl);
                        } catch (IOException e1) {
                              JOptionPane.showMessageDialog(this,"A fájl
megnyitása sikertelen!", "Hiba!", JOptionPane.WARNING_MESSAGE);
                        }
                  } else {
                        JOptionPane.showMessageDialog(this,"Az operációs
rendszere nem támogatja ezt a funkciót!", "Hiba!",
JOptionPane.WARNING_MESSAGE);
                  }
            }



            /* gepek letiltas */
            if(e.getActionCommand().equals("KitiltGomb")) {
                  String port = this.textFieldPort.getText();
                  String portOID = ".1.3.6.1.2.1.2.2.1.7." + port;
                  String IP = this.textFieldIP.getText();

                  if(!port.isEmpty()) {
                        try {
                              kitiltottMac = this.textFieldMac.getText();
                              kitiltottPort = port;
                              kitiltottNev = this.textFieldNev.getText();
                              kitiltottIP = this.textFieldIP.getText();

                              fajlbaIrKitiltott("kitiltott.txt",
kitiltottMac, kitiltottPort, kitiltottNev, kitiltottIP);
                              kitiltasVanE = true;

                              sajatLogger.setup();
```

```
                                LOGGER.log(Level.INFO, "Mac:" +
kitiltottMac + " Port:" + kitiltottPort + " Nev:" + kitiltottNev + " IP:"
+ kitiltottIP +  "  állomás kitiltva");

                                cam_tabla_eloallit.portTilt(portOID, new
UdpAddress(IP));
                           } catch (IOException e1) {
                                JOptionPane.showMessageDialog(this,"Az
állomás kitiltása sikertelen!", "Hiba!", JOptionPane.WARNING_MESSAGE);
                           }
                     }
               }




      }
} /* osztaly vege */
```

A kitiltottGepek osztály forráskódja

```
package GUI;


import java.awt.event.*;
import java.io.*;
import java.util.Scanner;
import java.util.logging.*;

import javax.swing.*;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import javax.swing.table.TableRowSorter;


import CAM_tabla.cam_tabla_eloallit;
import Log.*;

import javax.swing.JTextField;

import org.snmp4j.smi.UdpAddress;

public class kitiltottGepek extends JInternalFrame implements
ActionListener, ListSelectionListener {
      private static final long serialVersionUID = 1L;
      private static JTable tablazatKitilt;
      private String[][] sor = new String[][] { };
      private String[] oszlop = new String[] { "MAC", "Port", "Nev",
"IP"};
      private static TableModel tablaModel;
      private JButton btnEngedelyez;
      private JButton btnBezaras;
      private JButton btnAdatFrisst;
      private JTextField textFieldPort;
      private JTextField textFieldMac;
```

```
	private String port;
	private String mac;
	private String nev;
	private String portOID;
	private String IP;
	private JTextField textFieldNev;
	private JTextField textFieldIP;

	private final static Logger LOGGER =
Logger.getLogger(proba.class.getName());




	/* adatok tablazathoz adasa */
	public static void tablahozAd(String macCim, String port, String
nev, String ip) {

		if (macCim.isEmpty() || port.isEmpty()) {
			return;
		}

		DefaultTableModel model =
(DefaultTableModel)tablazatKitilt.getModel();


		/* ket ugyanolyan portot nem ad hozza */
		boolean vanEMar  = false;
		for (int j = 0; j < model.getRowCount(); j++) {
			if (port.equals(model.getValueAt(j, 1))) {
				vanEMar = true;
				break;
			}
		}


		if (!vanEMar) {
			model.addRow(new String[] {macCim, port, nev, ip});
		}
	}



	public kitiltottGepek() {
		setResizable(true);
		setTitle("Kitiltott \u00E1llom\u00E1sok
enged\u00E9lyez\u00E9se");
		setBounds(100, 100, 479, 289);
		getContentPane().setLayout(null);

		btnEngedelyez = new JButton("Enged\u00E9lyez");
		btnEngedelyez.setActionCommand("Engedelyez");
		btnEngedelyez.setBounds(233, 142, 111, 23);
		getContentPane().add(btnEngedelyez);

		JScrollPane scrollPane = new JScrollPane();
		scrollPane.setBounds(23, 11, 421, 120);
		getContentPane().add(scrollPane);
```

```
            tablazatKitilt = new JTable();
            scrollPane.setViewportView(tablazatKitilt);


      tablazatKitilt.setSelectionMode(ListSelectionModel.SINGLE_SELECTION
);
            tablazatKitilt.setRowSorter(new
TableRowSorter<TableModel>(tablaModel));
            tablazatKitilt.setAutoCreateRowSorter(true);



            sor = new String[][] { };
            oszlop = new String[] { "MAC cim", "Port", "Nev", "IP"};

            /* tablamodel, editalas letiltas */
            tablaModel = new DefaultTableModel(sor, oszlop) {
                  private static final long serialVersionUID = 1L;

                  @Override
                  public boolean isCellEditable(int row, int column) {
                        return false;
                  }
            };

            tablazatKitilt.setModel(tablaModel);

            btnBezaras = new JButton("Bez\u00E1r\u00E1s");
            btnBezaras.setActionCommand("Bezaras");
            btnBezaras.setBounds(165, 207, 89, 23);
            getContentPane().add(btnBezaras);

            btnAdatFrisst = new JButton("Friss\u00EDt\u00E9s");
            btnAdatFrisst.setActionCommand("frissit");
            btnAdatFrisst.setBounds(134, 142, 89, 23);
            getContentPane().add(btnAdatFrisst);

            textFieldPort = new JTextField();
            textFieldPort.setEditable(false);
            textFieldPort.setBounds(134, 176, 72, 20);
            getContentPane().add(textFieldPort);
            textFieldPort.setColumns(10);

            textFieldMac = new JTextField();
            textFieldMac.setEditable(false);
            textFieldMac.setBounds(10, 176, 119, 20);
            getContentPane().add(textFieldMac);
            textFieldMac.setColumns(10);

            textFieldNev = new JTextField();
            textFieldNev.setEditable(false);
            textFieldNev.setBounds(216, 176, 79, 20);
            getContentPane().add(textFieldNev);
            textFieldNev.setColumns(10);

            textFieldIP = new JTextField();
            textFieldIP.setEditable(false);
            textFieldIP.setBounds(305, 176, 139, 20);
```

```
		getContentPane().add(textFieldIP);
		textFieldIP.setColumns(10);


		this.btnBezaras.addActionListener(this);
		this.btnEngedelyez.addActionListener(this);
		this.btnAdatFrisst.addActionListener(this);

	tablazatKitilt.getSelectionModel().addListSelectionListener(this);

	}




	@Override
	public void actionPerformed(ActionEvent e) {

		/* bezaras */
		if(e.getActionCommand().equals("Bezaras")) {

	((JInternalFrame)getRootPane().getParent()).doDefaultCloseAction();
		}



		/* adatok firssítese */
		if(e.getActionCommand().equals("frissit")) {
			BufferedReader in = null;

				try {
					String sor;
					in = new BufferedReader(new FileReader(new
File("kitiltott.txt")));

					while((sor = in.readLine()) != null) {
						Scanner sc = new Scanner(sor);

						sc.useDelimiter("\\|");

						String mac = sc.next();
						String port = sc.next();
						String nev = sc.next();
						String ip = sc.next();

						tablahozAd(mac, port, nev, ip);
					}
				} catch(FileNotFoundException ex) {
					JOptionPane.showMessageDialog(this,"Nem
található az információkat tartalmazó fájl!", "Hiba",
JOptionPane.ERROR_MESSAGE);
				} catch(IOException ex) {
					JOptionPane.showMessageDialog(this,"I/O
hiba!", "Hiba", JOptionPane.ERROR_MESSAGE);
				} finally {
					if(in != null)
						try {
							in.close();
						} catch (IOException e1) {
```

```
                                        e1.printStackTrace();
                                  }
                            }


                  }



                  /* port engedelyezese */
                  if(e.getActionCommand().equals("Engedelyez")) {
                        port = this.textFieldPort.getText();
                        mac = this.textFieldMac.getText();
                        nev = this.textFieldNev.getText();
                        portOID = ".1.3.6.1.2.1.2.2.1.7." + port;
                        IP = this.textFieldIP.getText();


                        if(!port.isEmpty()) {
                              try {
                                    cam_tabla_eloallit.portEngedelyez(portOID,
new UdpAddress(IP));

                                    LOGGER.log(Level.INFO, "Mac:" + mac + "
Port:" + port + " Nev:" + nev + " IP:" + IP +  "  állomás engedélyezve");

                              } catch (IOException e1) {
                                    JOptionPane.showMessageDialog(this,"A port
engedélyezése sikertelen!", "Hiba!", JOptionPane.WARNING_MESSAGE);
                              }

                              if(tablazatKitilt.getSelectedRow() != -1) {
                                    felhTorol.sortTorol("kitiltott.txt", mac);
                                    ((DefaultTableModel)
tablaModel).removeRow(tablazatKitilt.getSelectedRow());
                              }
                        }
                  }



            }




            @Override
            public void valueChanged(ListSelectionEvent e) {
                  int kijeloltSorIndexe = tablazatKitilt.getSelectedRow();

                  if(kijeloltSorIndexe != -1) {
                        String Mac =
tablazatKitilt.getValueAt(kijeloltSorIndexe, 0).toString();
                        String Port =
tablazatKitilt.getValueAt(kijeloltSorIndexe, 1).toString();
                        String Nev =
tablazatKitilt.getValueAt(kijeloltSorIndexe, 2).toString();
```

```
                    String IP =
tablazatKitilt.getValueAt(kijeloltSorIndexe, 3).toString();
                    this.textFieldMac.setText(Mac);
                    this.textFieldPort.setText(Port);
                    this.textFieldNev.setText(Nev);
                    this.textFieldIP.setText(IP);
              } else if(kijeloltSorIndexe == -1) {
                    this.textFieldMac.setText("");
                    this.textFieldPort.setText("");
                    this.textFieldNev.setText("");
                    this.textFieldIP.setText("");
              }

        }
}
```

A login osztály forráskódja

```
/* Bejelentkezesi felulet */

package GUI;

import user.*;



import java.awt.*;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.event.*;
import java.io.*;
import java.text.ParseException;
import java.util.*;
import java.util.regex.Pattern;



public class login extends JFrame implements ActionListener {
      private static final long serialVersionUID = 1L;

      private static login frame;
      private JPanel contentPane;
      private JTextField textFieldFelhNev;
      private JPasswordField passwordFieldJelszo;
      private JButton btnBelepes;
      private JFormattedTextField formattedTextFieldIP;

      private static felhasznaloTarol felhasznalok = new
felhasznaloTarol();
      private File be = new File("felh.auth");
      private static String felhNev;
      private static String felhJelszo;
      private static String eszkozIP;
```

```
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					frame = new login();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}



	/* admin-kent van-e bejelentkezve */
	public static boolean adminE() {
		if(felhNev.contains("admin"))
			return true;

		return false;
	}


	/* eszkoz IP cime */
	public static String getEszkozIP() {
		return eszkozIP;
	}



	/* IP cim vizsgalata */
	public boolean IPcimE(String IPcim) throws IllegalArgumentException
{
	  Pattern ipPattern =
Pattern.compile("\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}/\\d{1,3}");

	  if(ipPattern.matcher(IPcim).matches()) {
		return true;
	  }
	 return false;
    }



	/* felhasznalokat tartalmazo fajl beolvasasa */
	public static void beolvas(File bemenet) throws IOException{
		BufferedReader in = null;

		try {
			String sor;
			in = new BufferedReader(new FileReader(bemenet));

			while((sor = in.readLine()) != null) {
				Scanner sc = new Scanner(sor);

				sc.useDelimiter("\\|");

				String nev = sc.next();
				String jelszo = sc.next();
```

```
                        felhasznalok.hozzaad(new felhasznalo(nev,
jelszo));
                  }
            } catch(FileNotFoundException ex) {
                  JOptionPane.showMessageDialog(frame,"Nem található a
felhasználói információkat tartalmazó fájl!", "Hiba",
JOptionPane.ERROR_MESSAGE);
            } catch(IOException ex) {
                  JOptionPane.showMessageDialog(frame,"I/O hiba!",
"Hiba", JOptionPane.ERROR_MESSAGE);
            } finally {
                  if(in != null)
                        in.close();
            }
      }



      /**
       * Create the frame.
       * @throws ParseException
       */
      public login() {
            setTitle("Bejelentkez\u00E9s");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(300, 200, 354, 190);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(null);

            JLabel lblFelhasznaloiNev = new
JLabel("Felhaszn\u00E1l\u00F3i n\u00E9v:");

      lblFelhasznaloiNev.setHorizontalTextPosition(SwingConstants.LEADING
);
            lblFelhasznaloiNev.setFont(new Font("Arial", Font.PLAIN,
12));
            lblFelhasznaloiNev.setBounds(22, 28, 108, 14);
            contentPane.add(lblFelhasznaloiNev);

            JLabel lblJelszo = new JLabel("Jelsz\u00F3:");
            lblJelszo.setFont(new Font("Arial", Font.PLAIN, 12));
            lblJelszo.setBounds(22, 53, 108, 14);
            contentPane.add(lblJelszo);

            textFieldFelhNev = new JTextField();
            textFieldFelhNev.setToolTipText("Itt adhatja meg a
felhaszn\u00E1l\u00F3i nevet.");
            textFieldFelhNev.setBounds(141, 26, 130, 20);
            contentPane.add(textFieldFelhNev);
            textFieldFelhNev.setColumns(10);

            this.passwordFieldJelszo = new JPasswordField();
            passwordFieldJelszo.setToolTipText("Itt adhatja meg a
jelsz\u00F3t.");
            this.passwordFieldJelszo.setBounds(141, 51, 130, 20);
            contentPane.add(passwordFieldJelszo);
            passwordFieldJelszo.setColumns(10);
```

```
            btnBelepes = new JButton("Bel\u00E9p\u00E9s");
            btnBelepes.setActionCommand("Belepes");
            btnBelepes.setFont(new Font("Arial", Font.PLAIN, 12));
            btnBelepes.setBounds(117, 111, 89, 23);
            contentPane.add(btnBelepes);

            JLabel lblEszkozIp = new JLabel("Eszk\u00F6z IP:");
            lblEszkozIp.setBounds(22, 78, 108, 14);
            contentPane.add(lblEszkozIp);


            formattedTextFieldIP = new JFormattedTextField();
            formattedTextFieldIP.setToolTipText("Itt adhatja meg az SNMP-
t t\u00E1mogat\u00F3 eszk\u00F6z IP c\u00EDm\u00E9t IP c\u00EDm/port
alakban. \r\n(Pl.: 192.168.1.2/161)");
            formattedTextFieldIP.setBounds(141, 75, 130, 20);
            contentPane.add(formattedTextFieldIP);



            this.btnBelepes.addActionListener(this);
      }



      @Override
      public void actionPerformed(ActionEvent e) {
            felhNev = this.textFieldFelhNev.getText();
            felhJelszo = new String(passwordFieldJelszo.getPassword());
            eszkozIP = this.formattedTextFieldIP.getText();



            /* ha hibas az IP cim */
            if(!IPcimE(eszkozIP)) {
                  JOptionPane.showMessageDialog(frame, "Hibás a megadott
IP cím formátuma!", "IP cím hiba", JOptionPane.ERROR_MESSAGE);
                  return;
            }


            /* Belepes gomb */
            if(e.getActionCommand().equals("Belepes")) {
                  try {
                        beolvas(be);
                  } catch (IOException e1) {
                        JOptionPane.showMessageDialog(frame, "IO hiba!",
"Hiba", JOptionPane.ERROR_MESSAGE);
                  }


                  /* ha a mezok nem uresek */
                  if(!(felhNev.isEmpty() || felhJelszo.isEmpty() ||
eszkozIP.isEmpty())) {

                        if((felhasznalok.elemKeresVane(felhNev,
felhJelszo)) == true) { /* ha igazat ad vissza == van ilyen elem */
                              foablak ujlap = new foablak();            /*
foablak meghivasa */
```

```
                              ujlap.setVisible(true);                    /*
foablak megjelenitese */
                              frame.setVisible(false);                   /*
login ablak elrejtese */

                              ujlap.setTitle("SNMP-menedzsment szoftver @
Miskolci Egyetem, Tompa Tamás, 2011");
                        } else {
                              JOptionPane.showMessageDialog(this,"Hibás
felhasználói név vagy jelszó!", "Hiba", JOptionPane.WARNING_MESSAGE);
                        }
                  } else if((felhNev.isEmpty() && felhJelszo.isEmpty()))
{
                        JOptionPane.showMessageDialog(this,"Nem adott meg
felhasználói nevet és jelszót sem!", "Hiba",
JOptionPane.WARNING_MESSAGE);
                  } else if(felhNev.isEmpty()) {
                        JOptionPane.showMessageDialog(this,"Nem
adott meg felhszanálói nevet!", "Hiba", JOptionPane.WARNING_MESSAGE);
                  } else if(felhJelszo.isEmpty()) {
                        JOptionPane.showMessageDialog(this,"Nem
adott meg jelszót!", "Hiba", JOptionPane.WARNING_MESSAGE);
                  } else if(eszkozIP.isEmpty()) {
                        JOptionPane.showMessageDialog(this,"Nem adta meg
az eszköz IP címét!", "Hiba", JOptionPane.WARNING_MESSAGE);
                  }
            }
      }

} /* osztaly vege */
```

Az ujEszkoz osztály forráskódja

```
package GUI;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.util.regex.Pattern;
import javax.swing.JInternalFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JFormattedTextField;
import javax.swing.JOptionPane;

import org.snmp4j.smi.*;

import CAM_tabla.cam_tabla_eloallit;


public class ujEszkoz extends JInternalFrame implements ActionListener {
      private static final long serialVersionUID = 1L;
      private JButton btnHozzad;
      private JButton btnBezar;
      private JFormattedTextField formattedTextFieldIP;

      private int i = 1;
      private static String ip = null;
      private static Address[] targetAddressTomb = new Address[100];
```

```
	public ujEszkoz() {
		setTitle("\u00DAj eszk\u00F6z hozz\u00E1ad\u00E1sa");
		setBounds(100, 100, 320, 190);
		getContentPane().setLayout(null);

		btnHozzad = new JButton("Hozz\u00E1ad");
		btnHozzad.setActionCommand("Hozzaad");
		btnHozzad.setBounds(51, 96, 89, 23);
		getContentPane().add(btnHozzad);

		btnBezar = new JButton("Bez\u00E1r");
		btnBezar.setActionCommand("Bezar");
		btnBezar.setBounds(145, 96, 89, 23);
		getContentPane().add(btnBezar);

		JLabel lblAzjEszkz = new JLabel("Az \u00FAj eszk\u00F6z IP
c\u00EDme:");
		lblAzjEszkz.setBounds(10, 28, 139, 14);
		getContentPane().add(lblAzjEszkz);

		formattedTextFieldIP = new JFormattedTextField();
		formattedTextFieldIP.setToolTipText("Itt adhatja meg a
tov\u00E1bbi eszk\u00F6z/eszk\u00F6z\u00F6k IP c\u00EDm\u00E9t.");
		formattedTextFieldIP.setBounds(65, 53, 148, 20);
		getContentPane().add(formattedTextFieldIP);

		this.btnBezar.addActionListener(this);
		this.btnHozzad.addActionListener(this);
	}


	/* IP cim vizsgalata */
	public boolean ipCimE(String IPcim) throws IllegalArgumentException
{
	  Pattern ipPattern =
Pattern.compile("\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}/\\d{1,3}");

	  if(ipPattern.matcher(IPcim).matches()) {
		return true;
	  }
	 return false;
    }


	public static Address[] getEszkIP() {
		return targetAddressTomb;
	}


	@Override
	public void actionPerformed(ActionEvent e) {

		if(e.getActionCommand().equals("Bezar")) {

	((JInternalFrame)getRootPane().getParent()).doDefaultCloseAction();
		}
```

```
            if(e.getActionCommand().equals("Hozzaad")) {
                  ip = this.formattedTextFieldIP.getText();


                  if(!ip.isEmpty()) {
                        if(!ipCimE(ip)) {
                              JOptionPane.showMessageDialog(this, "Hibás
a megadott IP cím formátuma!", "IP cím hiba", JOptionPane.ERROR_MESSAGE);
                              return;
                        } else {
                              try {
                                    targetAddressTomb[0] =
foablak.getTarget();

                                    targetAddressTomb[i] = new
UdpAddress(ip);

      foablak.setTarget(targetAddressTomb[i]);


      cam_tabla_eloallit.snmpKapcs(targetAddressTomb[i]);


                              } catch (IOException ex) {

      JOptionPane.showMessageDialog(this,"Kapcsolódási hiba!", "Hiba",
JOptionPane.WARNING_MESSAGE);
                              }
                              i++;
                        }
                  }
            }



      }
}
```