FTTX-ANALYSVERKTYG ANPASSAT FÖR TELIAS NÄT

FTTX-ANALYSIS TOOL DESIGNED FOR TELIA’S NETWORK

Andreas Brännback
Sammanfattning

Ett verktyg har utvecklats i programmeringsspråket Python, som analyserar status för uppkopplingar hos Fibre to the X (FTTX)-kunder i Telias nät. Systemet består av en moduluppdeld struktur, där alla analysfunktioner av samhörande typer är uppbyggda i egna moduler. Alla moduler lagras som individuella kodfiler. Systemet är designat för att enkelt kunna vidareutvecklas genom att tillägga fler analysmoduler i framtida projekt.

För att utföra en analys på en specifik kund, hämtar systemet tekniska dataparametrar via den switch som kunden sitter uppkopplad mot. Dessa parametrar jämförs därefter med förbestämda värden för att hitta avvikelser. Simple network management protocol (SNMP) och Telnet är de primära protokollen som används för att hämta relevant data.


Utförligheten i svaret på en utförd analys varierar en aning beroende på switchtypen kunden sitter uppkopplad mot. Switchar av äldre hårdvarutyper presenterar generellt sett mindre kundportsdata jämfört med modernare varianter. Mindre kundportsdata leder till sämre utförlighet i analyssvaret. Därför lämpar sig detta analysverktyg bättre mot de modernare switcharna som finns i Telias nät.
Abstract

A tool for analyzing the status of Fiber to the X (FTTX) customers in Telia’s network has been programmed in the Python programming language. The system consists of a module divided structure where analysis functions of similar types are bundled into module files. The system is designed to be easily further developed by adding more analysis modules in future projects.

To perform an analysis on a specific customer, the system retrieves technical data parameters from the switch which the customer is connected to, and compares these parameters against predetermined values to find deviations. Simple Network Management Protocol (SNMP) and Telnet are the primary protocols used to retrieve data.

Hypertext Transfer Protocol (HTTP) is used to transfer data as system input and output. The result of an analysis is sent as Extensible Markup Language (XML) back to the server that originally requested the start of an analysis. The XML reply contains technical data parameters describing the customer’s connection status and an analytical response based on these technical parameters.

The amount of data presented in the XML response varies slightly depending on the type of switch the customer is connected to. Switches of older hardware types generally present less customer port data compared to more modern switches. Less customer port data leads to poor detail in the analytical response, and therefore, this analysis tool is better suited to the modern switches found in Telia’s network.
Inledning

Bakgrund
Fibre to the X (FTTX) är en term som innefattar flera olika former av nätverksuppkopplingar. Dessa uppkopplingar styr mot en slutdestination (vanligtvis en kund) från en närliggande nätverksnod.1 Telias kunder är främst uppkopplade genom Fibre to the Home (FTTH) och Fibre to the Building (FTTB).2 FTTH innebär att en fiberoptisk kabel terminerar direkt i kundens fastighet (Hus eller lägenhet). FTTB innebär att en fiberoptisk kabel terminerar i en nätverksnod placerad i samma byggnad som kundens fastighet. Därefter går en nätverkskopparkabel vilken terminerar direkt mot kundens fastighet. Samlingsnamnet FTTX används för att innefatta båda dessa termer.3


Syfte
Det nya verktyget avser att ersätta ett tidigare analysverktyg som varit i drift i flertal år. När det tidigare verktyget byggdes, ställdes ett antal analyskrav som ansågs vara lämpliga för dåtidens vanligaste problem. Nu flera år senare har ny hårdvara med fler funktioner satts i drift, nya arbetsmetoder och problem har även framkommit. Dessa faktorer saknar en tydlig representation i det gamla analysverktyget. Därför finns behov av ett nytt uppdaterat verktyg med nya förbättrade funktioner och en bättre anpassning till dagens problemlösningsmetoder.4

Krav från beställare
Ett hållbart analysverktyg behövs framtas i ett modern och lätthanterligt programmeringsspråk. Verktyget ska bestå av en tydligt avgränsad och uppdelad modulstruktur. Modulstrukturen innebär, att varje enskild analysaspekt delas upp i separata filer. Dessa filer ska vara smidiga att modifiera för att enkelt kunna justera analysvar och värden med teknisk data.

Med modulstrukturen ska en systemansvarig lätt kunna bygga vidare på verktyget genom att addera fler moduler. Modulstrukturen ska även kunna garantera en viss säkerhet i att programmet blir körbart inom tidsramen för detta projekt, med eller utan alla moduler.

---
1 Skanova. Ordlista. FTTx. https://www.skanova.se/skanova/Om-Skanova/Ordlista#F-character (hämtad 2018-02-21)
Därutöver behövs endast en "back-end" lösning utvecklas i detta arbete.

Enkelt modifierbara analyssektioner ska finnas med i modulerna. Här ska beslutet tas, huruvida en uppkoppling fungerar stabilt utan störningar, baserat på de hämtade tekniska parametrarna.

Exekveringstiden på dessa moduler ska hållas så kort som möjligt. Kort exekveringstid tillåter anställda att arbeta mer effektivt, vilket i sin tur leder till kortare samtal och minskade telefonköer för kunder som ringer in.

Framtagna resultat från analysverktyget ska, så gott det går efterlikna de manuella analyserna som idag utförs av Telias nätverksdriftingenjörs. Se Tabell 1 för beskrivning över dessa manuella analyser.

Tabell 1. Kort beskrivning av de analyser som utförs av Telia-anställda på Umeå avdelningen idag för att hitta de kända problemen som kan drabba en kund.

<table>
<thead>
<tr>
<th>Typ av analys</th>
<th>Beskrivning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Länk</td>
<td>Kontroll om kunds port har länk.</td>
</tr>
<tr>
<td>IP och MAC</td>
<td>Kontroll att MAC-adresser syns och även att kund har plockat IP adresser till de tjänster som är aktiva på kunds port.</td>
</tr>
<tr>
<td>Felkoppling</td>
<td>Kontroll så kunds MAC-adress på switchporten stämmer med kunds verkliga MAC-adress. (Den verkliga MAC-adressen kan enbart tas fram genom dialog med kund, vilket utförs av kundtjänsten).</td>
</tr>
<tr>
<td>Option82</td>
<td>Kontroll av kunds port ifall den av någon anledning är spärrad.</td>
</tr>
<tr>
<td>Interface</td>
<td>Kontroll av inkommande och utgående errors och discards. Duplexstatus och länkhastighet kontrolleras även här.</td>
</tr>
<tr>
<td>Data</td>
<td>Kontroll av utgående och inkommande fibernivåer på kunds switchport.</td>
</tr>
<tr>
<td>Fibernivåer</td>
<td>Kontroll av kunds switchport i fall kund är uppkopplad mot en fiberswitch.</td>
</tr>
<tr>
<td>TP-kabel mätning</td>
<td>Kontroll av avbrott i nätverkskabeln på kunds switchport i fall kund är uppkopplad mot en koppar switch.</td>
</tr>
<tr>
<td>Switchlogg</td>
<td>Kontroll av loggen som visar upp och ner tider på kunds switchport. Här syns det om en länk till exempel flappar.</td>
</tr>
<tr>
<td>Inloggning på kunds CPE</td>
<td>Kontroll av kunds fiberkonverter/tjänstefördelare. Här syns errors, discards, fibernivåer och länkhastigheter på portar.</td>
</tr>
<tr>
<td>Inloggning på kunds Telia router</td>
<td>Kontroll av kunds Telia router. Här syns errors och länkhastigheter portar.</td>
</tr>
</tbody>
</table>
Mål

- Leverans av ett tydligt uppdelat modulstrukturerat system, som genomgår felsökningsmetoder, liknande de manuella analyser som utförs av Telias anställda idag.

- Åtminstone fem analysmoduler förväntas vara färdigställda innan projektets slut.

- Exekveringstiden för en komplette analys (alla moduler förutom kundstörande kabelmätningsmodul) ska hållas under tio sekunder.

- Analysresultaten ska vara enkelt justerbara.

- Modulerna ska vara felsäkra. Oförväntade inkommande svar ska fångas upp och behandlas.

Beställare

Projektet har beställts av Telia Company AB:s nätverksdriftavdelning i Umeå (GSO SA&O NWOps Access NW B2C Cust IP Swe).
**Teori**

För att ge en bättre överblick på hur systemet fungerar, så beskrivs kortfattat de nätverksprotokoll som används av systemet. Även en kort presentation av Telias olika typer av switchar redovisas här.

**HTTP**

"Hypertext transfer protocol". HTTP är ett applikationslagerprotokoll som skickar förfrågan från en klient(exempelvis en dator), innehållande bland annat förfrågningsmetod, URL, protokollversion och klientinformation. Denna förfrågan går mot en HTTP-server som bland annat svarar med statuskod och data anpassat till förfrågan. Detta protokoll används frekvent inom internet, för att hämta och leverera förfrågningar kring hemsidor.

HTTP anslutningar sker vanligtvis på port 80 via TCP/IP kommunikationsprotokollet. Däremot kan andra portar och andra internetprotokoll användas vid behov. HTTP förmodar endast att ett pålitligt transportprotokoll används.

**Telnet**


Telnet använder sig av TCP/IP kommunikationsprotokollet via port 23. För att översätta den terminalbaserade ASCII-koden till maskinspecifik kod för den lokala servern så används en nätverksvirtuell terminal, även kallad "NVT". Detta gör interaktionen mellan terminalerna smidig och lättbegriplig.

**SNMP**

"Simple network management protocol". I detta protokoll kommunicerar nätverksagenter med en nätverkshanteringsstation kallad "Manager". Protokollet tillåter överföring av driftinformation mellan uppkopplade nätverksnoder. Det finns tre versioner av SNMP. Alla versioner delar samma uppbyggnadsgrundstruktur men skiljer sig i prestanda och på säkerhetsmässiga fronter.

Detta protokoll är användbart i nätverksdriftsituationer. Data från agenter kan enkelt efterfrågas av managern och i fall något fel uppstår så kan agenter skicka ut larm över felet till den dedikerade managern.

---


6 Ibid


8 Ibid


**Telias switchar**


---

Genomförande


![Flödesdiagram](image-url)

**Figur 1. Flödesdiagram som beskriver systemets struktur. Termen ”Technical data” är i detta fall ett samlingsnamn som representerar en samling av flertal variabler.**

10
Allt programmerande skedde utifrån det framtagna lösningsdiagrammet på Figur 1. För att skriva koden användes textredigeraren GNU Nano direkt i en av Telias servrar med Red Hat Enterprice Linux 6.4. Systemet skrevs i ”cgi-bin”-katalogen för att tillåta åtkomst av externa servrar. Rättigheter på filer ställdes därefter in, för att göra programmet exekverbart av alla användare.


**Material**
Nedanstående material har använts för skapandet av programmet.

- **Linux Distribution:** Red Hat Enterprice Linux 6.4
- **Språk:** Python 2.6.6
- **Python-Bibliotek:** sys, time, re, subprocess, telnetlib, urllib2
- **Webbserver:** Apache/2.2.15 (Unix)
Resultat

Ett fullständigt analyssystem som hämtar data via SNMP, HTTP och Telnet har programmerats i Python och installerats på en av Telias servrar. Systemet består av 12 sammankopplade skriptfiler samt en txt-fil. Se Figur 2.


En analys med det framtagna lösningsförslaget kan utföras utan att störa den tillhörande kundens tjänster, så länge modul ”getCableFibreData.py” inte exekveras. Modulen ”getCableFibreData.py” stoppar i vissa fall tillfälligt alla tjänster för den kund som analyseras. Därför ska denna modul hanteras med större försiktighet.

Den servern som detta system installerats på har tillgång till alla Telias switchar, genom bland annat kommunikation via SNMP och Telnet. Brandväggar är även öppnade mot andra externa servrar som detta system förlitar sig på.

Det framtagna lösningsförslaget använde sig av Apache som webbserver. Hela projektets innehåll placeras i cgi-bin katalogen för att tillåta åtkomst via HTTP av externa servrar.

En genomgång av systemets alla delar i den framtagna lösningsmodellen, (Figur 1) beskrivs steg för steg i nedanstående rubriker fram till ”Utskrift”-rubriken. Här presenteras input och output från varje modul. Klammerpareneter i punktlistorna beskriver möjliga värden till den variabel som står vänsterliggande om klammerparenetserna. Värdena string och int presenteras då inga specifika förväntade värden finns.

Figur 2: Översikt av systemets filer

```
root@aa294a1b01 ftx2analysis # ls
getcableFibreData.py getinterfaceData.py getVendor.py snmpcollect.py
getcableFibreData.pyc getinterfaceData.pyc getVendor.pyc snmpcollect.pyc
getCPEdata.py getMacOption82.pyc instancetester.py summaryOutput.pyc
getCustomerMacVendor.py getswitchLog.py masterController.py telnetCollect.pyc
getCustomerMacVendor.pyc getswitchLog.pyc
```

En genomgång av systemets alla delar i den framtagna lösningsmodellen, (Figur 1) beskrivs steg för steg i nedanstående rubriker fram till ”Utskrift”-rubriken. Här presenteras input och output från varje modul. Klammerpareneter i punktlistorna beskriver möjliga värden till den variabel som står vänsterliggande om klammerparenetserna. Värdena string och int presenteras då inga specifika förväntade värden finns.
**Instance Starter**


**Tabell 2. Beskrivning av funktionaliteten kring mastercontroller.py-modulens tredje input-argument**

<table>
<thead>
<tr>
<th>Växel</th>
<th>Moduler som kommer exekveras</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alla moduler förutom <code>getCableFibreData.py</code></td>
</tr>
<tr>
<td>1</td>
<td><code>getVendor.py</code> och <code>summaryOutput.py</code></td>
</tr>
<tr>
<td>2</td>
<td><code>getVendor.py</code>, <code>getIpMacOption82.py</code> och <code>summaryOutput.py</code></td>
</tr>
<tr>
<td>3</td>
<td><code>getVendor.py</code>, <code>getIpMacOption82</code>, <code>getCustomerMacVendor.py</code> och <code>summaryOutput.py</code></td>
</tr>
<tr>
<td>4</td>
<td><code>getVendor.py</code>, <code>getIpMacOption82</code>, <code>getCustomerMacVendor.py</code> och <code>summaryOutput.py</code></td>
</tr>
<tr>
<td>5</td>
<td><code>getVendor.py</code>, <code>getCableFibreData</code> och <code>summaryOutput.py</code></td>
</tr>
<tr>
<td>6</td>
<td><code>getVendor.py</code>, <code>getSwitchLog.py</code> och <code>summaryOutput.py</code></td>
</tr>
<tr>
<td>7</td>
<td><code>getVendor.py</code>, <code>getIpMacOption82</code>, <code>getCPEdata.py</code> och <code>summaryOutput.py</code></td>
</tr>
</tbody>
</table>


Inputen för att starta en analys skickas genom en HTTP-förrågan till filen `instancestarter` med switchinloggnings-IP som första argument. Port som kund i switch sitter på som andra argument och slutligen en växel över moduler som ska köras som tredje argument. De olika alternativen kring växeln i det tredje argumentet redovisas i Tabell 2.

**Master Controller**

Master Controller "masterController.py" har som roll att tolka `option` inputen och utifrån detta starta motsvarande analysmoduler. Här finns alla `option`-val definierade med tillhörande underliggande moduler som eventuellt behövs köras innan den valda modulen kan exekveras. När masterController.py har anropat alla valda analysmoduler skickas därefter resultatet från modulerna till den avslutande modulen "summaryOutput.py".
Get Vendor
Från masterController.py kan modulen ”getVendor.py” anropas. Denna modul anropar därefter en funktion i ”snmpCollect.py”-modulen som utför förfrågningar för att hämta teknisk data till tre olika variabler: vendor, medium och switchUpTime. Data som kommer från snmpCollect.py-modulen innehåller den fullständiga hämtade strängen från SNMP-förfrågan. Rollen som getVendor.py har är, att tolka denna text för att separera ut de sökta värdena som därefter sparas in i motsvarande variabler. Denna modul har enbart switchIp som input. Outputen ser ut enligt följande:
  o vendor: {'Huawei', 'Ericsson', 'Cisco'}
  o medium: {'Fibre', 'Copper'}
  o switchUpTime: {string}

SNMP Collect
Alla SNMP-förfrågningar utförs enbart av modulen ”snmpCollect.py”. SNMP-förfrågningar uträttas med Python biblioteket ”subprocess” som kör SNMP shell-kommandon. SNMP GETBULK används vid tillfällen då många objekt ska hämtas i samma SNMP-session. SNMP GET används vid tillfället då enbart ett eller några få specifika värde eftersöks. Denna modul har enbart en anropning från annan modul som input. Output är variabler innehållande kompletta textsträngar från de SNMP-förfrågningar som utförts.

Get Interface Data
Från masterController.py kan modulen ”getInterfaceData.py” anropas. Denna modul fungerar likt getVendor.py-modulen då den enbart använder sig av data hämtat via SNMP. Från denna modul anropas en funktion i snmpCollect.py som utför en SNMP GETBULK, där en bestämd lista med objekt efterfrågas som därefter skickas tillbaks till getInterfaceData.py-modulen.

Olika objektlista efterfrågas beroende på switchleverantören som beskrivs i input variabeln vendor. De övriga inputen är switchIp och port, dessa används för att efterfråga rätt objekt relaterat till kunds port i switchen. Inkommande data från snmpCollect.py tolkas och sparas i separata variabler. En jämförelse med förbestämda värdet utförs mot vissa av dessa variabler, för att upptäcka potentiella problem i kundens uppkoppling. Resultatet från denna jämförelse sparas i variabeln getInterfaceData_analysisResult. Output ur denna modul ser ut enligt följande:
  o ifOperStatus: {'up', 'down'}
  o ifAlias: {string}
  o dot3StatsDuplexStatus: {'Full', 'Half'}
  o ifInOctets: {int}
  o ifOutOctets: {int}
  o ifInDiscards: {int}
  o ifOutDiscards: {int}
  o ifInErrors: {int}
  o ifOutErrors: {int}
  o ifSpeed: {'10', '100', '1000'}
  o getInterfaceData_analysisResult: {string}
Get Ip/Mac/Option82
Från masterController.py kan modulen ”getIpMacOption82.py” anropas. Data till denna modul hämtas via Telnet genom modulen ”telnetCollect.py”. De kompletta texträngarna som returnerats via modulen telnetCollect.py tolkas sedan av getIpMacOption82.py-modulen. Från texträngarna plockas följande ut: ip-adresser, mac-adresser och option82-strängar. Alla Ip adresser med tillhörande vlan på den valda porten sparar i variabeln allIpAndVlan. Mac adresser med vlan sparar i variabeln allMacAndVlan. Option82-strängvariblerna jämförs med ordet ”abuse”. Om detta ord finns med i strängen sätts variabeln abuseEnabled till värdet ”True” vilket är en indikation på att kunden har obetalda räkningar. Värdet ”False” i abuseEnabled sätts om ordet ”abuse” på inga vilkor hittas. Ip-adresser för specifika vlan plockas även ut här in i separerade variabler som därefter kan användas av andra moduler.

Även i denna modul görs en jämförelse med förbestämda värden, för vissa av modulens variabler i syfte att upptäcka potentiella problem i kundens uppkoppling, resultatet från denna jämförelse sparar i variabeln getIpMacOption82_analysisResult. Inputen till denna modul är vendor, switchIp och port. Output ur denna modul ser ut enligt följande:
- allIpAndVlan: {string}
- allMacAndVlan: {string}
- ipVlan294: {string}
- ipVlan296: {string}
- abuseEnabled: {’True’, ’False’}
- getIpMacOption82_analysisResult: {string}

Telnet Collect
Alla Telnetsessioner utförs enbart av modulen ”telnetCollect.py”. Telnet anslutningar uträttas med Python biblioteket ”telnetlib”. En session med telnetlib fungerar genom att systemet väntar på en specificerad texträng, innan nästa bestämda del av koden kan exekveras. Till exempel när texten ”password:” finns tillgänglig så exekveras nästa rad av koden där en textinput ges. Hela eller delar av textoutputen från Telnetsessionen sparar därefter i en strängvariabel. Likt snmpCollect.py har denna modul enbart en anropning från annan modul som input. Output är variabler innehållande kompletta texträngar från de Telnetsessioner som utförts.

Get CPE data
- CPEdata {string}
Get Switch log
- switchLog {string}
- switchLog_analysisResult {string}

Get Cable/Fibre data
Från masterController.py kan modulen ”getCableFibreData.py” anropas. Denna modul använder sig av telnetCollect.py för att få ut en textsträng från en Telnetsession. Den hämtade textsträngen varierar beroende på det uppkopplingsmedium kunden är ansluten via. Data i textsträngen kommer finnas tillgängligt i fall switchen stödjer utförandet av mätningarna, i alla andra fall så förblir textsträngen tom. Om kunden är uppkopplat via fiber och stöd för fibermätning finns, så innehåller textsträngen fiberoptiska dämpningsvärden i form av utskick och mottagande. Om kunden är uppkopplat via kopparnätverkskabel och stöd för mätning finns i switchen, så kommer textsträngen att innehålla en uppskattad längd på ledningen samt anslutningsstatusen på de 4 kopparparen i ledningen. Varningar på kortslutningar eller andra form av skador på ledningen redovisas här. Även en jämförelse med data från modulen mot förbestämda värden utförs här och sparas i variabeln getCableFibreData_analysisResult. Inputen till denna modul är vendor, medium, switchIp och port. Output ur denna modul ser ut enligt följande:
- fibreValues: {string}
- copperValues: {string}
- getCableFibreData_analysisResult: {string}

Get Customer Mac Vendor
Från masterController.py kan modulen ”getCustomerMacVendor.py” anropas. Denna modul har enbart variabeln allMacAndVlan som input och har som roll att ta fram leverantörer till de mac-adresserna som finns kopplade till den porten som analyseras. Modulen går en i taget igenom alla mac-adresser på porten och jämför dessa med en lista innehållande mac-adresser för 23000 olika leverantörer. Om en match hittas på mac-adresserna så redovisas detta i variabeln allCustomerMacVendor. Output ur denna modul ser ut enligt följande:
- allCustomerMacVendor: {string}

Get summary Output
Utskrift

Den slutgiltiga utskriften av systemet, från en utförd analys är i normala fall enbart i XML-kod. Detta gör det dock aningen besvärligt att redovisa utskriften på ett tydligt sätt i detta dokument på grund av att de flesta webbläsare har ingen möjlighet att visualiserar radbrytningar i XML-kod. För tydligheten skull utförs därför en stdout-print på alla variabler från alla olika moduler direkt via Command-Line Interface (CLI), detta ger en mer organiserad vy på outputen.

Original XML-utskriften redovisas i Figur 10. Alla IP-adresser och delar av MAC-adresser blockeras av en vit ruta i figurerna av säkerhetsskäl.

Variabelutskrift från en kund som sitter kopplad på en Huaweiswitch redovisas i Figur 3. Här finns även motsvarande variabelnamn markerade i vit text. Samma variabelordning på utskriften används även i Figur 4 och Figur 5. Strängen 'n/a' skrivs ut i fallen då data för variabeln saknas.

Variabelutskrift från en kund som sitter kopplad på en Ciscoswitch redovisas i Figur 4.

Variabelutskrift från en kund som sitter kopplad på en Ericssonswitch redovisas i Figur 5.

Utskrift från modulen getCableFibreData.py visas i Figur 6, Figur 7 och Figur 8.

Utskrift från modulen getCPEData.py redovisas i Figur 9.

Figur 3. Utskrift av variabler från en kund uppkopplad på en Huaweifiberswitch. Här redovisas utskriften från alla moduler förutom getCPEData.py och getCableFibreData.py. Variabelnamn redovisas här i vit text.
Figur 4. Utskrift av variabler från en kund uppkopplad på en Ciscokopparswitch. Här redovisas utskriften från alla moduler förutom getCPEdata.py och getCableFibreData.py

Figur 5. Utskrift av variabler från en kund uppkopplad på en Ericssonswitch. Här redovisas utskriften från alla moduler förutom getCPEdata.py och getCableFibreData.py

Figur 6. Resultat av kopparkabelmätning på en Ciscoswitch från modulen getCableFibreData.py
Figur 7. Resultat av kopparkabelmätning på en Huaweiswitch från modulen getCableFibreData.py

```bash
cd /mnt/cluster/cluster
sudo python /root/python/code/CableFibreData.py
```

Huawei Copper

Pair A length: 24 meter(s)
Pair B length: 24 meter(s)
Pair C length: 24 meter(s)
Pair D length: 17 meter(s)
Pair A state: Ok
Pair B state: Ok
Pair C state: Ok
Pair D state: Ok

Figur 8. Resultat av fibermätning på en Huaweiswitch från modulen getCableFibreData.py

```bash
cd /mnt/cluster/cluster
sudo python /root/python/code/CableFibreData.py
```

RX Power(dBm) : -7.94
RX Power High Threshold(dBm) : -1.00
RX Power Low Threshold(dBm) : -24.94
TX Power(dBm) : -5.53
TX Power High Threshold(dBm) : -1.00
TX Power Low Threshold(dBm) : -11.00


```bash
cd /mnt/cluster/cluster
sudo python /root/python/code/CableFibreData.py
```

Port 1
LinkState_1=1000 Mbps, Full-duplex
unicastsReceived_1=87508244
unicastsSent_1=224281844
FCSerrReceived_1=0
FCSerrSent_1=0
Port 2
LinkState_2=Link is down
unicastsReceived_2=0
unicastsSent_2=0
FCSerrReceived_2=0
FCSerrSent_2=0
Port 3
LinkState_3=Link is down
unicastsReceived_3=0
unicastsSent_3=0
FCSerrReceived_3=0
FCSerrSent_3=0
Port 4
LinkState_4=Link is down
unicastsReceived_4=0
unicastsSent_4=0
FCSerrReceived_4=0
FCSerrSent_4=0
Port 5
LinkState_5=Link is down
unicastsReceived_5=0
unicastsSent_5=0
FCSerrReceived_5=0
FCSerrSent_5=0
Port wan
LinkState_6=1000 Mbps, Full-duplex
unicastsReceived_6=224284013
unicastsSent_6=87511121
FCSerrReceived_6=0
FCSerrSent_6=0
Figur 10. XML output utifrån analys med alla moduler förutom getCableFibreData.py. Radbrytningar "\n" visualiseras inte här. Elementen ”Begränsad_Vy” och ”Mer_begränsad_Vy” visar avskalade versioner av innehållet i elementet ”Komplett_Vy.”
Redigerbara analysresultat
Ett av målen till detta projekt var att det skulle vara enkelt att redigera
analysresultaten från modulerna. Figur 11 visar ett exempel på hur
analyssektionen ser ut i modulen getInterfaceData.py. Liknande struktur används
även i andra moduler. Denna sektion kan hittas längst ner i kodfilen och här kan
gränsvärden enkelt modifieras.

![Figur 11. Kod som beskriver villkoren för analysresultatet ur modulen getInterfaceData.py](image)

Totalt finns 15 olika möjliga analyssvar. Alla dessa finns listade i Tabell 3.

<table>
<thead>
<tr>
<th>Nummer</th>
<th>Modul</th>
<th>Analyssvar</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>getInterfaceData.py</td>
<td>No link on port</td>
</tr>
<tr>
<td>2</td>
<td>getInterfaceData.py</td>
<td>Only half duplex communication, potential damage to customer equipment</td>
</tr>
<tr>
<td>3</td>
<td>getInterfaceData.py</td>
<td>Traffic-discard errors on port, do analysis again to check if number of errors regularly increase. No increase means no problem.</td>
</tr>
<tr>
<td>4</td>
<td>getInterfaceData.py</td>
<td>Errors on port, do analysis again to check if number of errors regularly increase. No increase means no problem.</td>
</tr>
<tr>
<td>5</td>
<td>getInterfaceData.py</td>
<td>Linkspeed between Telia switchport and customer equipment is lower than promised speed. Potential port or cable/fibre damage.</td>
</tr>
<tr>
<td>6</td>
<td>getInterfaceData.py</td>
<td>Port linkspeed only 100M. Should be 1000M under normal circumstances for Huawei switches.</td>
</tr>
<tr>
<td>7</td>
<td>getIpMacOption82.py</td>
<td>No IP on port</td>
</tr>
<tr>
<td>8</td>
<td>getIpMacOption82.py</td>
<td>Abuse active on port, have customer paid all bills?</td>
</tr>
<tr>
<td>9</td>
<td>getCableFireData.py</td>
<td>Switchport is receiving very low/no levels of light from customer fibre/media converter.</td>
</tr>
<tr>
<td>10</td>
<td>getCableFireData.py</td>
<td>Switchport is transmitting very low/no levels of light from customer fibre/media converter.</td>
</tr>
<tr>
<td>11</td>
<td>getCableFireData.py</td>
<td>Open connection on atleast 1 pair, TP-cable from switchport have no connection on pair.</td>
</tr>
<tr>
<td>12</td>
<td>getCableFireData.py</td>
<td>Short circuit on atleast 1 pair, possible TP-cable damage.</td>
</tr>
<tr>
<td>13</td>
<td>getCableFireData.py</td>
<td>TP-cable pairs are interfering with eachother. Possible TP-cable damage.</td>
</tr>
<tr>
<td>14</td>
<td>getCableFireData.py</td>
<td>TP-cable pairs have mismatched impedance. Possible TP-cable damage.</td>
</tr>
<tr>
<td>15</td>
<td>getSwitchLog.py</td>
<td>Warning, many entries in switchlog, potential flapping link</td>
</tr>
</tbody>
</table>
**Diskussion**

Överlag har detta projekt gått relativt problemfritt. Jag är själv anställd på den Telia-avdelning som utfört beställningen och har därför redan innan projektstart haft kunskap om hur analyser bör gå till och hur ett FTTX-analyssystem realistiskt skulle kunna byggas inom Telias nät. Detta har underlättat förstudien en aning och har även varit till stor fördel under övriga delar av projektets gång.

Genom att studera Figur 3 till Figur 10, så framstår det tydligt att utskriften och mängden tillgängliga variabler varierar en del beroende på switchleverantör. Detta innebär att analysverktyget kommer att ge aningen mindre utförligt resultat för switchar som saknar redovisning av värden till alla variabler.

Modulen getCableFibreData.py kan enbart redovisa fiberoptiska mätvärden om analysen utförs på en Huawei fiberswitch. Koparkabelmätningar kan enbart utföras av Huawei switchar och moderna Ciscoswitchar. Övriga switchar saknar hårdvara som stödjer utförandet av dessa mätningar. Kabelmätning är en viktig aspekt i felsökningssyfte och detta blir därför ytterligare en negativ aspekt för kunder som är kopplade mot ericssonswitchar.


**Måluppfyllnad**

- *Leverans av ett tydligt uppdelat modulstrukturerat system, som genomgår felsökningsmetoder liknande hur manuella analyser utförs av Telias anställda idag.*
  - Delvis Uppfyllt.

**Motivering:**
Genom att utgå från de framtagna manuella analysmetoderna i Tabell 1 så utförs nästan samma analyser av det programmerade verktyget. Det som saknas är en felsökning av felkopplingar och analys av potentiella fel i kunds Telia Router. Tidsbrist och högre prioritering av andra analysdelar, är orsaken till att lösning för dessa problem saknas i systemet.
• Åtminstone fem analysmoduler förväntas vara färdigställda innan projektets slut.
  – Uppfyllt.

Motivering:
Totalt färdigställdes 8 moduler (inklusive ”Get summary Output”). Av dessa 8 moduler innehåller 5 moduler analysspecifika delar.

• Exekveringstiden för en komplett analys(alla moduler förutom kundstörande kabelmätningsmodul) ska undvikas att överstiga tio sekunder.
  – Ej uppfyllt.

Motivering:

• Analysresultaten ska vara enkelt justerbara.
  – Uppfyllt.

Motivering:
Analysresultaten i moduler med analysspecifika delar är enkelt justerbara via den analysspecifika sektionen som kan hittas längst ner i koden i modulerna. Detta illustreras i Figur 11.

• Modulerna ska vara felsäkra. Oförväntade inkommande svar ska fångas upp och behandlas.
  – Uppfyllt.

Motivering:
Alla moduler hålls felsäkra genom utförlig användning av Pythons ”try” och ”except”. Ifall felaktig förfrågan mot en modul utförs eller att inget svar från modulerna finns tillgänglig så redovisas slutsvaret ”n/a” i de berörda XML elementen.
Källkod
Den slutgiltiga källkoden för systemet är nåbart av Telia-anställda via den server som driftats på, men har avhållits att presenteras till allmänheten på grund av säkerhetsskäl.

I källkodens alla filer finns det noggrant beskrivna kommentarer på engelska som motiverar varför alla val kring programmeringsbesluten tagits. Kommentarerna beskriver även hur mer komplexa koddela fungerar.

Problem kring resultatet


Under tidsramen för projektet, färdigställdes allan moduler vilket översteg förväntningarna från programmerare och beställare. Däremot har ännu inte systemet testats utförligt nog för att vara redo för skarp drift. Alla problem som hittats under projektets gång har blivit åtgärdade men ytterligare problem kan möjligtvis upptäckas genom mer omfattande testning och därför bör detta genomföras innan skarp driftsättning.

Utvecklingsmöjligheter
För att förbättra systemet i framtiden, kan moduler läggas till som utför analyser på fler kunder i switchen och därefter gör jämförelser mellan kunderna för att hitta avvikelse. Detta skulle dock medföra att exekveringstiden kommer bli relativt lång så det blir i så fall extra viktigt att skicka tidseffektiva kommandon för hämtning av data.

En annan potentiell förbättring till detta system är en sammanslagning av alla Telnetförfrågningar. Som systemet nu är designat så kommer två separata Telnetsessioner att öppnas vid utförande av analys med växeln ”o”. Anledningen till denna design är att detta leder till bättre separering och uppdelning av modulerna. Proceduren att öppna en Telnetsession går på ungefär en fjärde sekund till en halv sekund, och så länge enbart ett fält Telnetsessioner används så bör detta vara en acceptabel lösning. Dock om flerta nya moduler läggs till, så lär detta märkas tydligt på exekveringstiden och därför bör detta vara en del av systemet som i så fall byggs om.

Nästa steg mot driftsättning för detta system blir ytterligare testning för att hitta potentiella problem. Därefter kommer en hemsida, som skickar input och redovisar output, behöva programmeras innan systemet är redo för användning av Telias anställda.
Referenslista


