Technische Universität München

Institut für Informatik

Fortgeschrittenenpraktikum

Implementierung von Funktionen zur Unterstützung des ATM-Managements am LRZ

Themensteller: Univ.-Prof. Dr. Heinz-Gerd Hegering
Betreuer: Anja Schuhknecht, Thomas Schwerdtner
Bearbeiter: Kilian Frühauf
Carsten Hecht
Christian Seiler

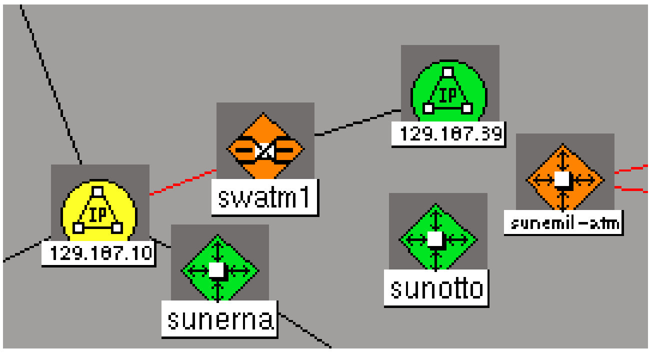
Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

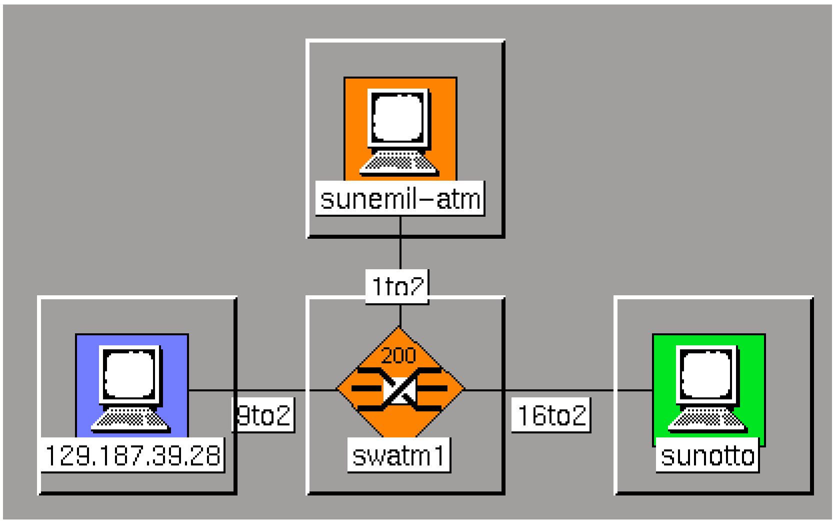
Als eine der zukunftsträchtigsten Netzwerktechnologien gilt der Asynchrone Transfer Modus (ATM). Im Rahmen dieses Fortgeschrittenenpraktikums sollten Tools zur Verwaltung eines ATM-Netzes am LRZ entwickelt werden. Dafür stand das Netzwerkmanagementsystem HP-OpenView (OVW) - ein System für das Internetmanagement - zur Verfügung.

Das im Lieferumfang von OpenView enthaltene IP-Map kann die Konfiguration eines Netzes nur auf IP-Ebene erfassen und darstellen. Zudem erzeugt OpenView äußerst unübersichtliche Strukturen:



Da die ATM-Technik auf den unteren Schichten des OSI-Referenzmodells arbeitet, ist jedoch eine Sicht auf die physikalische Verknüpfung der Netzkomponenten nötig. Es existieren zwar herstellerspezifische, auf OVW aufsetzende Lösungen. Diese erlauben jedoch kein integriertes Netzmanagement.

Das Fortgeschrittenenpraktikum bietet eine übersichtliche Lösung:



Darüberhinaus verlangt die ATM-Technologie erweiterte Funktionalitäten, wie z.B. genaueres Erfassen des Zellflusses.

1.2. Arbeitsumgebung

Das Testnetz, für das das Fortgeschrittenenpraktikum realisiert worden ist, besteht aus einem Switch (Fore ASX-200) und drei Hosts (siehe Abb. oben), die jeweils über einen Fore-Adapter an das Netz angeschlossen sind. Die Managementplattform OpenView läuft auf dem Host „sunotto", der damit die Rolle des Managers im Netz übernimmt.

Als Arbeitsraum stand zeitweise ein Raum in der Leopoldstraße zur Verfügung. Aufgrund der räumlichen Distanz wurde ein Videokonferenzsystem installiert, das die ständige Kommunikation zwischen Betreuern und Bearbeitern sicherstellte.

1.3. Aufgaben und Funktionalitäten

Das Fortgeschrittenenpraktikum gliedert sich in drei Teilbereiche:

- Erfassen der physikalischen Konfiguration des ATM-Netzes
- Ermittlung der Verkehrscharakteristik im ATM-Netz
- Aufbereitung der gesammelten Informationen

Die Fopra-Applikationen arbeiten auf einer eigenen Submap, in der der Administrator die Komponenten des ATM-Netzes manuell einfügt und verbindet. Diese Editieroperationen erzeugen OVW-Events, welche der erste Teil über sogenannte Callback-Funktionen abfängt und in Konfigurationsdateien speichert. Auf diese Weise können doppelte Einträge eines Gerätes bzw. einer Verbindung in der Submap vermieden werden. Außerdem kann so für jedes hinzugefügte Objekt überprüft werden, ob herstellerspezifische Informationen, die für MIB-Abfragen benötigt werden, vorhanden sind.

Der zweite Teil erfaßt die Netzlast im ATM-Netz. Dabei werden nur die aktiven Phasen mitprotokolliert, die der Adminstrator über Thresholds mit dem OVW-Nodemanager definiert.
Bei Überschreiten der Schwellwerte rufen die erzeugten Events die entsprechenden Loggingprozesse auf. Das Ermitteln der Last erfolgt auf zwei Ebenen. Auf Linkebene wird der gesamte Verkehr pro Port betrachtet, wobei zwischen ein- und ausgehendem Link unterschieden wird. Auf Verbindungsebene wird die Last erfaßt, die auf einzelnen Kanälen aufgetreten ist. Dabei kann entweder nur ein Durchschittswert oder ein genaues Profil für eine aktive Phase erstellt werden.

Im dritten Teil werden die gewonnenen Informationen schließlich ausgewertet und dem Administrator aufbereitet zur Verfügung gestellt. Dazu gehören sowohl Statistiken, die sich auf einzelne Switches (Konfiguration und aufgetretene Last auf Linkebene) beziehen, als auch netzweite Verbindungsinformationen, aus denen hervorgeht, wieviele Zellen über welche Ende-zu-Ende-Verbindungen zu welchem Zeitpunkt übertragen worden sind. Weiterhin besteht die Möglichkeit, eine konfigurierte Verbindung durch das Netz zu verfolgen und dabei pro Switch aktuelle MIB-Variablen oder Switch-Statistiken anzeigen zu lassen.

1.4. Namenskonvention für Dateien

Die meisten, in diesem System verwendeten Dateien haben zusammengesetze Dateinamen. Folgendes Beispiel erläutert dieses Prinzip:

Eine Datei, in der die Anzahl der auf einem Link eingegangenen Zellen enthält, hat den Namen „%SWITCH%_%LINK%_in_cells“. Dabei stellt „%SWITCH%“ einen Platzhalter für den Selection Name des Switches dar, „%LINK%“ steht für die entsprechende Linknummer.

Platzhalter existieren außerdem für VP- und VC-Identifikatoren: „%VPI%“ und „%VCI%“.

2. Benutzerschnittstelle

2.1. Menüführung

Netzkonfiguration:

- Hinzufügen von Objekten in der ATM-Submap:
  Menü: Edit→AddObject
  Dialog: Eingabe von Objektname und Selectionname des neuen Objekts

- Hinzufügen von Verbindungen in der ATM-Submap:
  Menü: Edit→AddConnection
  Submap: Anfangs- und Endpunkt der neuen Verbindung anklicken
  Dialog: Name der Verbindung eingeben mit folgender Syntax:
  „xtoy“- wobei x und y für die Nummern des Ports an den Start- und Endobjekten stehen, an denen die Verbindung anliegt.
  Wegen der Zuordnung dieser Nummern ist bei der Eingabe von Start- und Endpunkt der Verbindung die Reihenfolge von Bedeutung

- Löschen von Objekten oder Verbindungen aus der ATM-Submap:
  Submap: Anklicken der zu löschenden Komponente
  Menü: Edit→Delete
  Dialog: Bestätigen

Parameter:

- Festlegen der Logging-Parameter:
  Submap: Anklicken eines Switches
  Menü: Fopra→Edit Logging Parameters
  Fenster: Editieren der in der Eingabedatei befindlichen Parameter (siehe 4.2.2.)

- Festlegen der Parameter für die Statistikerstellung:
  Menü: Fopra→Edit Statistic Parameters
  Fenster: Editieren der in der Eingabedatei befindlichen Parameter

Switch-Statistiken und netzweite Informationen:

- Erstellung einer temporären Switch-Statistik:
  Submap: Anklicken des Switches, zu dem eine Statistik erstellt werden soll
  Menü: Fopra→Calculate current Switch Statistics

- Erstellung aktueller netzweiter Informationen:
  Menü: Fopra→Calculate current Net Statistics

- Anzeigen netzweiter Informationen:
  Menü: Fopra→Show Net Statistics

Trace

- Submap: Doppelklick auf ein Objekt oder
- Menü: Fopra→Trace nach einfachem Anklicken der Startkomponente
  Fenster: Ist die Startkomponente ein Switch:
  Auswahl zwischen Trace oder Ausgabe von Statistiken zum Switch
  Ist die Startkomponente ein Host: nur Trace möglich

  Bei Trace: Auswahl eines konfigurierten Kanals
  Wahl einer Option:

```
Route ......................... (1)
Route + History ............... (2)
Route + aktuelle MIB-Werte .... (3)
Route + History + MIB-Werte ... (4)
```

2.2. Eingabedateien

Sämtliche Eingabedateien befinden sich im Verzeichnis „input".

2.2.1. Parameterdatei zur Festlegung der ATM-Submap

Diese Datei („lrz_atm_id") hat als einzigen Eintrag den Selection Name der ATM-Submap.

2.2.2. Hersteller-Dateien

Die Gesamtheit dieser Dateien ist die generische MIB, die ausführlich in Kapitel 4.5. erläutert wird. Jede Netzkomponente benötigt eine derartige Datei, deren Name sich aus der MIB-II-Variable „sysDescr" ergibt.

2.2.3. Parameterdatei für Erstellung von Switch-Statistiken und netzweiten Informationen

Diese Datei („statistik_info") wird für die Erstellung von Switch-Statistiken und netzweiten Informationen benötigt und sieht folgendermaßen aus:

```
# Switch-Statistiken
# ------------------
LogIntSwitchStat:   15      # Intervalllänge in Minuten
DelFreqSwitchStat:   3      # nach x Intervallen alte/
                              ausgewertete Daten loeschen
# Netzweite Statistik
# -------------------
LogIntNetStat:      15      # Intervalllänge in Minuten
DelFreqNetStat:      3      # nach x Intervallen dekonfigu
                              rierte VCs loeschen
MaximumHopDiff:      1      # max. Verzögerung von aktiven
                              Phasen pro Hop in Sekunden
```

Die einzelnen Parameter haben folgende Bedeutung:

- *LogIntSwitchStat*:
  Dieser Parameter legt fest, in welchem Intervall (in Minuten) Switch-Statistiken erstellt werden sollen. Bei der Wahl dieses Parameters ist zu berücksichtigen, daß die Statistikerstellung längere Zeit in Anspruch nehmen kann und Dateien lockt, auf die auch andere Prozesse zugreifen. Je mehr Switches sich im Netz befinden, desto größer sollte das Intervall gewählt werden.

- *DelFreqSwitchStat*:
  Hiermit kann der Administrator bestimmen, nach jeder wievielten Switch-Statistik-Erstellung bereits ausgewertete Daten (Teile der Logging-Dateien pro Switch) gelöscht werden. Diese werden nicht gelöscht, falls der Administrator eine Statistik für einen bestimmten Switch anfordert.

- *LogIntNetStat*:
  Dieser Parameter legt fest, in welchem Intervall (in Sekunden) netzweite Informationen erstellt werden sollen. Hier ist zu berücksichtigen, daß die Erstellung der Ausgabedatei umso mehr Zeit in Anspruch nimmt, je größer das Netz ist.

- *DelFreqNetStat*:
  Hiermit kann der Administrator bestimmen, nach jeder wievielten Erstellung netzweiter Informationen nicht mehr konfigurierte VCs gelöscht werden. Diese werden auch dann gelöscht, wenn diese „von Hand" (also nicht im Intervall) erstellt werden.

- *MaximumHopDiff*:
  Findet beim Auftreten einer aktiven Phase ein Datenfluß über mehrere nacheinandergeschaltete Kanäle statt, ist es möglich, daß für diese leicht variierende Zeitpunkte für

den Beginn der aktiven Phase protokolliert werden. Um eine aktive Phase nachträglich durch das Netz verfolgen zu können, müssen diese Differenzen berücksichtigt werden. Durch Setzen dieses Parameters kann der Administrator wählen, wie groß Verzögerungen zwischen zwei hintereinandergeschalteten Kanälen höchstens sein dürfen. Die Einheit Sekunden orientiert sich an der Genauigkeit, die der Network Node Manager verwendet.

2.2.4. Parameterdatei für Trace-Ausgabe

In dieser Datei („tracemibs“) befinden sich für jeden Hersteller die MIB-Variablen, die bei einem Trace pro Switch ausgegeben werden sollen.

Beispiel:

```
/* Instanz-Typen                                    */
/* LRZ_ATM_INSTANCE_ADAPTER        1                */
/* LRZ_ATM_INSTANCE_LINK           2                */
/* LRZ_ATM_INSTANCE_PATH           3                */
/* LRZ_ATM_INSTANCE_CHANNEL        4                */
/* LRZ_ATM_INSTANCE_CHANNELROUTE   5                */
/* LRZ_ATM_INSTANCE_LINK_ERROR     6                */
-------------------------------------------------------------

/***********************************/
/* MIB Variablen fuer Fora Switches */
/***********************************/
VendorIndex:            2
ChanCells:              ".1.3.6.1.4.1.326.2.2.2.1.4.1.1.8"
InstanceTyp:            4
ChanRejectedCells:      ".1.3.6.1.4.1.326.2.2.2.1.4.1.1.11"
InstanceTyp:            4
IfError:                ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.10"
InstanceTyp:            6
IfOverflow:             ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.9"
InstanceTyp:            6
```

Die einzelnen Parameter haben folgende Bedeutung:

- *`Vendorindex:`*
  Gibt an, von welchem Hersteller und Typ der Switch ist, da die MIB-Abfragen jeweils unterschiedlich zu generieren sind.

- *`ChanCells, ChanRejectedCells, IfError, IfOverflow:`*
  Beispiel für eine auszugebende MIB-Variable. Der Name kann frei gewählt werden, sollte aber erläutern, welche Variable hinter der folgenden OID steckt. Hier kann jede MIB-Variable, die auf dem entsprechenden Switch existiert angegeben werden.

- *`InstanceTyp:`*
  Für das Abfragen einer MIB-Variable ist außer ihrer OID und dem Vendorindex der Instancetyp anzugeben. Je nach dem, ob die MIB-Variable sich z.B. auf einen ganzen Link, oder nur den verfolgten Channel bezieht, sind bei ihrer Abfrage vom Programm an die OID unterschiedlich spezifische Instanzen anzuhängen.

2.3. Installation und Start

Zur Installation und zum Start des Systems sind folgende Schritte nötig:

1. Entpacken der Archivdatei:
   Das Archivfile „lrz_atm.tar.gz“ in das Zielverzeichnis kopieren.
   Zuerst mit „gunzip lrz_atm.tar.gz“ entpacken.
   Dann mit „tar xf lrz_atm.tar“ die Dateien aus dem Archiv kopieren

2. Setzen der Umgebungsvariable „FOPRA_PATH“:
   Einige der Fopraapplikationen benötigen diese Umgebungsvariable. Am besten setzt man sie im Skript „.login“ oder im Initialisierungsskript der Shell (z.B. „.bashrc“ oder „.tcshrc“).
   Sie muß den Pfad des Fopra-Verzeichnisses enthalten, also z.B. „/home/sunotto/a2824ca/fopra“.
   Es ist auch darauf zu achten, daß die „PATH“-Umgebungsvariable diesen Pfad enthält.
   Hier ein exemplarischer Auszug aus einer „.login“-Datei:

   setenv FOPRA_PATH /home/sunotto/a2824ca/fopra
   setenv PATH $FOPRA_PATH/bin:$PATH

3. Kompilieren:
   In das „src“-Verzeichnis wechseln. Ein einfacher Aufruf von „make“ kompiliert sämtliche Quelltexte. Das Makefile verzweigt in die entsprechenden Unterverzeichnisse und ruft die dortigen Makefiles auf.

4. OVW-Registration-File:
   Die Datei „fopra.reg“, die sich im Hauptverzeichnis befindet, muß in das Registrationfile-Verzeichnis von OVW kopiert werden, damit die Fopraapplikation eingebunden wird.

5. Festlegen der LRZ-ATM-Submap
   Der Selectionname der ATM-Submap muß in der Datei „input/lrz_atm_id“ gespeichert werden.

6. Einrichten der NodeManager-Events

   Das Überwachen der Schwellwerte übernimmt wie bereits erwähnt der Network-NodeManager. Der Netzadministrator muß somit für jeden Switch entsprechende NNM-Events einrichten: Zwei Events für die Linkebene (Incoming und Outgoing) und einen für die Verbindungsebene (in Klammern stehen jeweils die Werte für den ForeRunner ASX-200):

Linkebene:

a) Suchen der MIB-Variablen für Zellfluß in beiden Richtungen (portReceivedCells, portTransmittedCells). Diese OIDs sind auch in den Herstellerdateien zu spezifizieren.
Alle folgenden Punkte sind für beide MIB-Variablen durchzuführen.

b) Eintragen des (Start-) Schwellwerts: Dieser Wert richtet sich nach der Anwendung. Eine Schranke von 100 kbps würde einem Wert von 235 entsprechen.

c) Eintragen des Rearm-Schwellwerts: Dieser Wert darf vor allem bei Links, bei denen ständig ein gewisses Grundniveau (während des Praktikums ca. 10-15 cells/sec) an Netzverkehr herrscht, auf keinen Fall zu klein gewählt werden, da nur ein Unterschreiten dieses Schwellwerts die aktive Phase beendet und das Logging stoppt. Es hat sich gezeigt, daß ein Wert von 20 cells/sec das absolute Minimum darstellt, da ansonsten der Rearm-Wert nie unterschritten und das Logging nie beendet wird.

d) Im Instanzenfeld „List" anwählen und die am Switch benutzten Links angeben (0,1,8,9,15)

e) Auf keinen Fall „Check Thresholds And Store" sondern „Check Thresholds" anwählen, da der NNM ansonsten riesige Logdateien anlegt.

f) Pollingfrequenz: Ein sehr heikler Punkt. Das Pollingintervall stellt die minimale Ungenauigkeit bei allen Messungen dar. Sie ist dem Internetmanagement zu verdanken, das die Schwellwertprüfung dem Manager überträgt. Diese Pollingfrequenz hat nichts mit der eigentlichen Logginggranularität (siehe 4.2.3.) zu tun, sondern bestimmt, wie schnell NNM das Über- bzw. Unterschreiten der Schwellwerte erkennt (1 bis 5 sec).

g) Nun müssen die NNM-Events eingerichtet werden. Dies geschieht mit „Configure Threshold Event" bzw. „Configure Rearm Event". Die Namen und NNM-Portnummern der Events können beliebig gewählt werden. Danach müssen noch die Pfade der Loggingprozesse (siehe 4.2.2) eingetragen werden. Dazu wählt man „Modify" an. Wiederum erscheint ein neues Dialogfenster. Jetzt den Pfad für die Loggingprozesse eingeben, z.B. „/home/sunotto/a2824ca/fopra/bin/ lrz_atm_ac_link $2 $4 $6" bzw. „.../lrz_atm_deac_link" beim Rearm-Event. Wichtig ist die Übergabe der Parameter $2, $4 und $6 (Hostname, OID, Instanz), die von den Loggingprozessen benötigt werden. Das Feld „Popup Notification" sollte gelöscht werden, damit nicht ständig Popup-Dialoge beim Eintreten der Events erscheinen.

h) Einschalten der Schwellwertüberprüfung mit „Resume".

Verbindungsebene:

a) Suchen der MIB-Variable für Zellfluß auf den Kanälen (chanCells)

b) siehe Linkebene

c) siehe Linkebene. Der Rearmschwellwert kann hier oft auch auf Null gesetzt werden, da es im Gegensatz zu Links (über die mehrere Verbindungen laufen) auch Phasen gibt, in denen überhaupt kein Netzverkehr zu beobachten ist.

d) Hier muß mit regulären Ausdrücken gearbeitet werden. Hier sollte der Systemadministrator sowenig Verbindungen auswählen wie nötig
((16.0)|(1.0)|(8.0)|(9.0).*).

e) siehe Linkebene

f) siehe Linkebene

g) siehe Linkebene. Anstatt „.../lrz_atm_ac_link" bzw. „.../lrz_atm_deac_link" müssen hier die Pfade aber „.../lrz_atm_ac_vc" und „.../lrz_atm_deac_vc" lauten

h) siehe Linkebene

Dieses Vorgehen ist für alle Switches im ATM-Netz zu wiederholen. Dies bedeutet relativ viel Arbeit, die vom Administrator sorgfältig zu erledigen ist. Spätere Versionen der Fopra-Applikation könnten diese Eintragungen (zum Teil) automatisieren.

7. Starten des Channel-Daemons:
„lrz_atm_chand hostname interval (in ms) oid instance1 instance2 ..."
Die Parameter sind im einzelnen folgende:

Hostname:	Name des zu überwachenden Switches
interval:	Intervall in ms, in dem die MIB-Variable mit den gespeicherten Werten zu vergleichen ist
oid:	Hier ist die oid der MIB-Variablen anzugeben, die die Zahl der Channels pro Link angibt
instancen:	Liste von Instanzen, die angibt, welche links auf dem Switch zu beobachten sind

Beispiel: lrz_atm_chand swatm1 3000 .1.3.6.1.4.1.326.2.2.2.1.3.1.1.6
0.0 1.0 8.0 8.1 9.0 16.0 16.4 16.6

8. Starten des Daemons, der Switch-Statistiken erstellt:
„lrz_atm_logswitchstat"

9. Starten des Daemons, der netzweite Informationen erstellt:
„lrz_atm_lognetstat"

3. Erfassen der physikalischen Konfiguration des ATM-Netzes

3.1. Überblick

Aufgabe des ersten Teils ist es, Eingaben des Benutzers in der ATM-Submap festzustellen, sie semantisch und syntaktisch zu kontrollieren und entsprechend die für die weiteren Programmteile notwendigen Konfigurationsdateien zu erstellen oder anzupassen. Ein Check dieser Dateien wird außerdem bei jedem Programmstart durchgeführt, um festzustellen, ob sie auf dem aktuellen Stand sind (wird nach Installation der Foprasoftware OVW gestartet, so wird über das OVW-Registrationfile die Routine `lrz_atm_start` aufgerufen, die u.a. diesen Konsistenzcheck durchführt).

Eingabedateien:

- “input/lrz_atm_id“ enthält den Selectionname der ATM-Submap
- „conf/lrz_atm_components“ enthält die bisher konfigurierten Komponenten
- „conf/lrz_atm_connections“ enthält die bisher konfigurierten Verbindungen
- %SWITCH%_VC-Dateien enthalten Informationen zu den an einem Switch konfigurierten Channels

Ausgabedateien:

- „conf/lrz_atm_components“ enthält die konfigurierten Komponenten
- „conf/lrz_atm_connections“ enthält die konfigurierten Verbindungen
- %SWITCH%_VC wird hier gegebenenfalls aktualisiert

3.2. Prinzip einer eigenen ATM-Submap

Wie bereits im ersten Kapitel erläutert ist OVW durch seine Sicht lediglich auf logische Verbindungen und den Mangel an entsprechender Funktionalität nicht für das ATM-Netzmanagement geeignet. Daher verwendet dieses Fopra OVW lediglich als Grundlage, auf der eigene Applikationen aufsetzen. Dabei arbeitet es auf einer eigenen ATM-Submap, deren Name bei Programmstart aus der Datei „input/lrz_atm_id“ eingelesen wird (bei Installation ist hier der Name der ATM-Submap von Hand abzulegen). Die Routine `lrz_atm_GetSubmapId` ermittelt die von OVW zu diesem Namen verwendete SubmapId und speichert sie in die einzige im Programm verwendete globale Variable, damit sie allen Programmteilen zur Verfügung steht.

Durch die Verwendung einer eigenen ATM-Submap wird außerdem der ungeeignete Autodetect umgangen. Wie bereits unter 1.1. auch durch Bilder angedeutet, erzeugt der OVW-Autodetect teilweise erheblich unübersichtliche Strukturen und bietet außerdem keine Sicht auf die für ATM relevante physiaklische Ebene. Der Verzicht auf Autodetect bedeutet allerdings, daß sämtliche Netzkomponenten und die Verbindungen zwischen ihnen vom Netzmanager von Hand eingegeben und gelöscht werden müssen. Dabei ist folgendermaßen vorzugehen:

3.2.1. Zufügen von Objekten (Hosts oder Switches)

1. Aktuelle Submap = ATM-Submap
2. Anwählen des Menüpunktes Edit→AddObject
3. Es öffnet sich ein Fenster, in dem der Name des Objektes und seine Bezeichnung in der ATM-Submap einzugeben sind

3.2.2. Zufügen von Verbindungen (connections)

1. Aktuelle Submap = ATM-Submap
2. Anwählen des Menüpunktes Edit→AddConnection
3. Es sind nun zuerst Anfangs- und Endpunkt der Verbindung durch Mausklick in der ATM-Submap zu bestimmen, wobei es wichtig ist, die richtige Reihenfolge einzuhalten.
4. Es öffnet sich ein Fenster, in dem der Name der Verbindung einzugeben ist, bei dem sich der Anwender an folgende Syntax halten muß: ‘xtoy’- wobei x und y für die Nummern des Ports an den Start- und Endobjekten (Hosts oder Switches) stehen, an denen die Verbindung anliegt. Wegen der Zuordnung dieser Nummern ist bei der Eingabe von Start- und Endpunkt der Verbindung die Reihenfolge von Bedeutung.

3.2.3. Funktionsweise des Programms beim Zufügen von Objekten und Verbindungen

Die Fopra-Applikation wird beim Einfügen in der ATM-Submap erst tätig, nachdem vom Anwender die oben beschriebenen Schritte vollständig durchgeführt wurden. Und zwar werden die entsprechenden Callbackfunktionen (s.u.) (OVwConfirmAddSymbolCB, OVwConfirmConnectSymbolsCB), die beim Programmstart eingerichtet wurden von OVW gestartet. Sie bekommen als Parameter u.a. einen Zeiger auf die Map- und einen auf die Symbol-Infostruktur und stellen als erstes fest, ob es sich um eine Aktion in der ATM-Submap handelt (`lrz_atm_IsSymbolInATMMap`), denn sie werden bei jedem Zufügen aufgerufen, auch wenn dies in anderen Submaps geschieht. Ist dies der Fall werden die Funktionen aufgerufen, die die Fileeinträge in den Fopra-Konfigurationsdateien anpassen (`lrz_atm_AddObject`, `lrz_atm_AddConnection`). Diese kontrollieren zunächst, ob das neue Symbol (steht für Objekt oder Verbindung) schon einmal in der ATM-Submap existiert (`lrz_atm_IsSymbolUniqueInATMMap`) und ändern die Konfigurationsfiles nur, wenn dies nicht der Fall ist, da im Sinne der Funktionalität und Übersicht nur ein Symbol pro Objekt sinnvoll ist. Falls das Objekt(oder die Verbindung) bereits durch ein Symbol in unserer Submap dargestellt wird, ist das Vorgehen für Objekte und Verbindungen unterschiedlich:

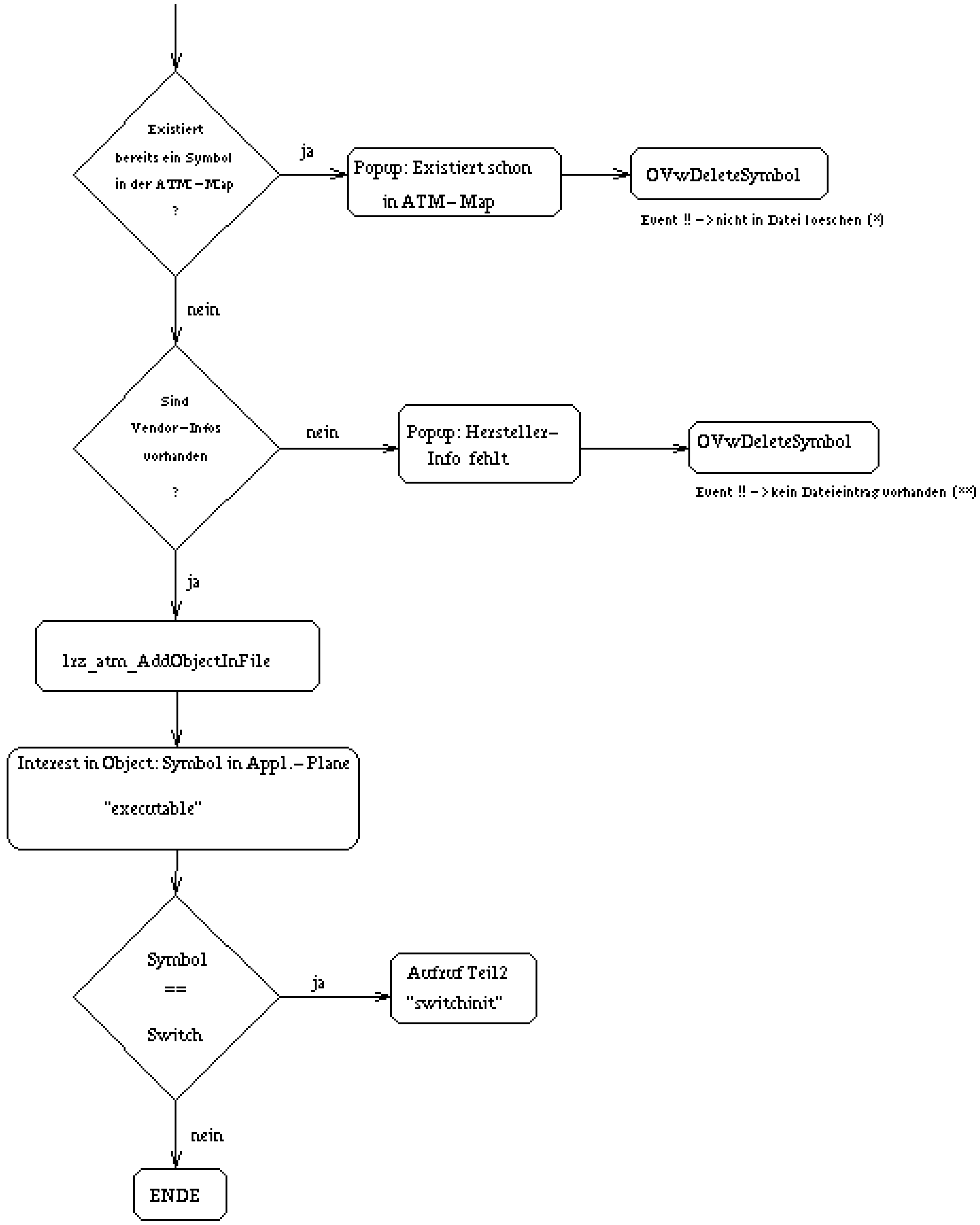
- bei Objekten kann das zusätzliche Symbol ohne Probleme wieder gelöscht werden

- bei Verbindungen entsteht aber ein Problem, nämlich das OVW evtl. sogenannte 'Metaconnections' aufbaut, wenn zwischen zwei Komponenten eine zweite Verbindung eingegeben wird. Diese sind aber in unserem Fall nicht verwendbar und außerdem in OVW falsch implementiert, so daß sie nicht einmal mit API-Funktionen zuverlässig wieder gelöscht werden können. So muß das Programm den Anwender bitten, selbst nach dem rechten zu sehen und evtl. Metaconnections von Hand wieder zu löschen. Hier sollte der Anwender bereits beim Hinzufügen von Verbindungen darauf achten, keine zweite Verbindung zwischen zwei Objekten zu legen.

Bei Objekten, d.h. bei neuen Netzknoten, wird jetzt die MIB-II-Variable „sysDescr“ abgefragt, um den Namen der Herstellerdatei zu ermitteln. Dies geschieht mit einem SNMPGET-Request. Auf dem Knoten muß also der SNMP-Daemon laufen. Nun überprüft das Programm, ob eine entsprechende Herstellerdatei im „input“-Verzeichnis existiert. Falls dies nicht der Fall ist, wird nach entsprechender Meldung das Objekt wieder gelöscht.
Anschließend wird beim Zufügen von Verbindungen noch die Syntax des Labels überprüft. Ob allerdings die Nummern der Ports auch stimmen, wird von Programmseite aus nicht überprüft, hier wäre mit Sicherheit ein weiterer Punkt für mögliche Verbesserungen der Software.

Erst nach diesen Tests wird das Objekt oder die Verbindung in die entsprechende Datei eingetragen. Für Objekte ist dies die Datei „conf/lrz_atm_components“, für Verbindungen die Datei „conf/lrz_atm_connections“.

Das folgende Flußdiagramm stellt den Ablauf beim Zufügen einer Komponente noch einmal übersichtlich dar:



3.2.4. Löschen von Objekten oder Verbindungen

Zum Löschen von Objekten oder Verbindungen muß der Anwender zunächst die zu löschende Komponente anklicken, dann den Menüpunkt Edit→Delete anwählen.

3.2.5. Funktionsweise des Programms beim Löschen von Komponenten und Verbindungen

Auch für das Löschen ist eine beim Programmstart erklärte Callbackroutine zuständig (OVwConfirmDeleteSymbolsCB), die zuerst feststellt, ob das gelöschte Symbol in der ATM-Submap lag, dann für Objekte und Verbindungen zum Löschen aus den jeweiligen Dateien in verschiedene Routinen springt (`lrz_atm_DeleteObject`, `lrz_atm_DeleteConnection`). Wird ein Objekt gelöscht, muß außerdem die Connection-Datei angepaßt werden, damit Verbindungen nicht zu einer bereits gelöschten Komponente führen, des weiteren werden Log- und Statisktikdateien aus den Teilen 2 und 3 (Logging und Statistikerstellung), die zu der gelöschten Komponente gehören, ebenfalls gelöscht.

Wichtig ist, daß Änderungen an der ATM-Submap nur stattfinden, während die Fopra-Applikation läuft (d.h. die AddObject-Events abgefangen und ausgewertet werden), da sonst die Konfigurationsdateien mit den in OVW dargestellten Symbolen nicht mehr übereinstimmen. Dies wird zwar mit dem Konistenzcheck bei Programmstart überprüft, dennoch ist dies natürlich eine mögliche Fehlerquelle.

3.3. Die erzeugten Konfigurationsdateien

Die beiden grundlegendsten Dateien für die Fopra-Applikation sind die im ersten Teil erstellten „atm_components“ und „atm_connections“. Beide liegen im Verzeichnis „conf“ und werden entweder beim Konfigurationscheck (beim Programmstart oder auf Anwahl des Menüpunktes „Consistence-Check“) oder beim Hinzufügen/Löschen von Komponenten/ Verbindungen verändert. Die anderen Programmteile greifen nur lesend auf sie zu.

Beispieldateien zur Erläuterung:

3.3.1. atm_components:

In dieser Datei werden die in der ATM-Submap eingefügten Objekte sequentiell mit folgenden Informationen aufgelistet:

- *`ComponentSelName`*:
  Der beim Einfügen vom Benutzer festgelegte Name. Dabei handelt es sich um den OVW-Selectionname (wie bei allen restlichen Einträgen mit der Kennung „SelName“)

- *`ComponentVendorName`*:
  Der volle Name des Produktes

- *`SymbolId`*:
  Die OVW-interne Identifikationsnummer des Symbols

- *`IsSwitch`*:
  Eine im Fopra eingeführte Booleanvariable, die für Switches auf True steht

```
----------------------------------------------
ComponentSelName:             "swatm1"
ComponentVendorName:          "Fore Systems ASX-200"
SymbolId:                     3428
IsSwitch:                     TRUE
----------------------------------------------
ComponentSelName:             "129.187.39.28"
ComponentVendorName:          "Fore Systems ATM Host (SunOS 4.1.3_U1
sun4m)"
SymbolId:                     3477
IsSwitch:                     FALSE
----------------------------------------------
ComponentSelName:             "sunotto"
ComponentVendorName:          "Fore Systems ATM Host (SunOS 4.1.3 sun4m)"
SymbolId:                     3374
IsSwitch:                     FALSE
----------------------------------------------
ComponentSelName:             "sunemil"
ComponentVendorName:          "Fore Systems ATM Host (SunOS 4.1.4 sun4m)"
SymbolId:                     3653
IsSwitch:                     FALSE
```

3.3.2. atm_connections

Der Aufbau dieser Datei ist bedeutend komplizierter als der von atm_components. Sie besteht aus einer Sektion für jede Komponente, deren Name am Anfang unter *`SourceSelName`* steht. Dann werden nacheinander alle Informationen zu jeder Verbindung (unabhängig davon, ob sie beim Einrichten der Verbindung als source oder destination angegeben wurde) aufgelistet, die zu dieser Komponente bestehen. Das bedeutet, daß jede Verbindung zweimal aufgelistet ist, einmal unter der beim Erstellen als Source, einmal unter der als destination angegebenen Komponente. Diese Datenredundanz führt zwar auf Softwareseite zu relativ hohem Aufwand beim Erstellen und Erweitern, vereinfacht dafür aber den häufigeren Fall des Lesens erheblich. Es ist zu beachten, daß die hier verwendeten Begriffe *`SourceSelName`* und *`DestSelName`* nichts mit „source“ und „destination“ beim Einfügen der Verbindung in die ATM-Submap zu tun haben, sondern nur eine Bedeutung innerhalb eines Abschnittes bis zum nächsten Separator („------------“) haben. Dies ist lediglich programmtechnisch nötig und spielt für den Anwender keine Rolle.

Die Elemente eines Eintrags bedeuten im einzelnen folgendes:

- `SourceSelName`:
  Name der Komponente, von der bis zum nächsten Separator („-----------------“) alle aufgelisteten Verbindungen ausgehen.

- `SymbolId`:
  hier wie im folgenden die OVW-interne Identifikationsnummer der jeweiligen Komponente oder Verbindung

Jetzt kommen nacheinander die Blöcke, in denen jeweils eine Verbindung zu der oben genannten Komponente beschrieben wird - getrennt durch Leerzeilen:

- `ConnectionSelName`:
  Name der Verbindung nach folgender Syntax: „‘xtoy“- wobei x und y für die Nummern des Ports an den Start- und Endobjekten (Hosts oder Switches) stehen, an denen die Verbindung anliegt.

- `DestinationSelName`:
  Name der Komponente, die über diese Verbindung mit der „Sourcekomponente“ verbunden ist

- `SourceLink`:
  Nummer des Ports, an dem die Verbindung bei der „Sourcekomponente“ liegt

- `DestLink`:
  entsprechend Portnummer an der „Destinationkomponente“

```
----------------------------------------------
SourceSelName:    "swatm1"    # von dieser Komponente gehen
                              # bis zum nächsten Separator
SymbolId:          3428       # alle aufgeführten
                              # Verbindungen aus

ConnectionSelName:      "1to2"
SymbolId:               3655
DestinationSelName:     "sunemil" # Angaben zur Verbindung ‘1to2’
SymbolId:               3653
SourceLink:             1
DestLink:               2

ConnectionSelName:      "9to2"
SymbolId:               3478
DestinationSelName:     "129.187.39.28"
SymbolId:               3477        # Angaben zur Verbindung ‘9to2’
SourceLink:             9
DestLink:               2

ConnectionSelName:      "16to2"
SymbolId:               3452
DestinationSelName:     "sunotto"
SymbolId:               3374
```

```
SourceLink:             16
DestLink:               2
---------------------------------------------
SourceSelName:          "sunotto"
SymbolId:               3374

ConnectionSelName:      "16to2"
SymbolId:               3452
DestinationSelName:     "swatm1"
SymbolId:               3428
SourceLink:             2
DestLink:               16
---------------------------------------------
SourceSelName:          "129.187.39.28"
SymbolId:               3477

ConnectionSelName:      "9to2"
SymbolId:               3478
DestinationSelName:     "swatm1"
SymbolId:               3428
SourceLink:             2
DestLink:               9
---------------------------------------------
SourceSelName:          "sunemil"
SymbolId:               3653

ConnectionSelName:      "1to2"
SymbolId:               3655
DestinationSelName:     "swatm1"
SymbolId:               3428
SourceLink:             2
DestLink:               1
```

4. Ermittlung der Verkehrscharakteristik im ATM-Netz

4.1. Überblick

Der zweite Teile ist der Kernpunkt des Fortgeschrittenenpraktikums. Er verwendet die im ersten Teil erzeugten Dateien, die die physikalische Verknüpfung des ATM-Netz beschreiben, und erstellt die Logdateien des Netzverkehrs für den Statistikteil. Der zweite Aufgabenbereich besteht im wesentlichen aus drei Teilen:

- Erfassen der Netzlast
- Protokollieren der geschalteten Verbindungen
- Protokollieren von Fehlern

4.2 Erfassen der Netzlast

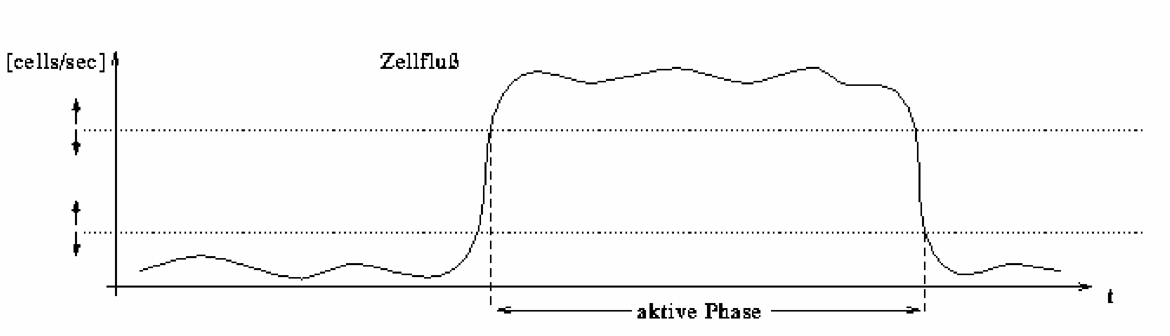
Das Ermitteln des Zellflusses auf den Switches findet auf zwei Ebenen statt. Es gibt die *Linkebene*, auf der die Zellen gezählt werden, die auf den Ports der Switches (also auf den einzelnen Kabeln, die am Switch angeschlossen sind) ein- und ausfließen. Aufgrund dieses bidirektionalen Datentransfers auf den Links existieren also pro Switch bis zu zwei (bzw. drei) Logdateien (siehe auch 4.2.4.).

Genauere Informationen über den Zellfluß erhält man auf der *Verbindungsebene*. Virtuelle Verbindungen (auch Kanäle genannt) sind integraler Bestandteil des ATM-Standards. Diese unidirektionalen Verbindungen sind über mehrere Switches geschaltet und werden dabei innerhalb eines Switches von einem Link auf einen anderen geroutet. Die Verbindungsidentifikatoren VPI (Virtual Path Identifier) und VCI (Virtual Connection Identifier) können sich dabei von Switch zu Switch ändern. Es kann jede virtuelle Verbindung auf einem Switch gelogged werden. Pro Verbindung legt die Fopraapplikation eine eigene Logdatei an (siehe auch 4.2.4).

Das Loggen auf Verbindungsebene gestattet ein Zuordnen von Netzverkehr zu bestimmten Verbindungen, während auf Linkebene nur der gesamte Verkehr auf einem „Stecker“ überwacht wird.

4.2.1. Definieren aktiver Phasen durch Nodemanager-Thresholds

Die folgende Abbildung zeigt einen exemplarischen Zellfluß, wie er sich auf einem Link oder einer Verbindung ergeben könnte. Eine wesentliche Anforderung an das Praktikum war, nicht einfach den gesamten Verlauf mitzuprotokollieren (dies könnten auch die mit OVW mitgelieferten Tools erledigen), sondern nur die *aktiven Phasen* rauszufiltern, die sich vom sonstigen Rauschen bzw. von der Normallast abheben, da ein ständiges Logging viel zu große Logdateien erzeugen würde.



Dieser Filermechanismus wurde mit Hilfe der OVW-NodeManager-Thresholds umgesetzt. Der obere Schwellwert markiert den Beginn der aktiven Phase, der untere beendet diese. Die Pfeile deuten an, daß die Schwellwerte auf Erfahrungswerten basieren und deren sinnvolle Festlegung noch nicht abgeschlossen ist bzw. es überhaupt keine allgemein gültigen Werte geben kann, da der Zellfluß bei verschiedenen Anwendungen sehr unterschiedlich sein kann (z.B. MPEG-2-Video oder eine Telnet-Session). Trotzdem sind bei der Konfigurationsprozedur (siehe unten) einige Vorschläge für Schwellwerte enthalten.

Das Überwachen der Schwellwerte übernimmt wie bereits erwähnt der NetworkNodeManager. Der Netzadministrator muß somit für jeden Switch entsprechende NNM-Events einrichten: Zwei Events für die Linkebene (Incoming und Outgoing) und einen für die Verbindungsebene. Dabei sollte er sich an das in der Installationsanleitung vorgestellte Vorgehen halten, damit keiner der Schritte vergessen wird.

4.2.2. Einstellen der Loggingparameter

Es gibt zwei verschiedene Log-Modi, in denen die aktive Phase protokolliert werden kann: Bei der ersten Betriebsart nimmt der Loggingprozeß nur am Anfang und Ende der aktiven Phase jeweils einen Meßwert und bestimmt damit die durchschnittliche benutzte Bandbreite. Welche Minima und Maxima sich während der Phase ergaben, läßt sich somit nicht feststellen. Dafür ist der zweite Modus gedacht, in dem das Loggingprogramm in definierbaren Abständen (z.B. 100 Millisekunden) den aktuellen Wert vom entsprechenden Switch per SNMP-GET abfrägt. Somit ergibt sich ein (mehr oder weniger) genauer Verlauf der aktiven

Phase vom Anfang bis zum Ende.

Der Betriebsmodus und die zeitliche Granularität des Protokollierens der aktiven Phasen läßt sich für Link- und Verbindungsebene getrennt einstellen. Dies hat denn Vorteil, daß man sich bestimmte Verbindungen aussuchen kann, die man detaillierter protokollieren kann. Alle Einstellungen beziehen sich auf einen einzelnen Switch, welche der Administrator in den Dateien „log/%SWITCH%_log_info“ eintragen muß. Hier ein Beispiel für den Switch „swatm1“:

```
# swatm1_log_info
#
#
# Fuer jeden angeschlossenen Link muss in dieser Datei ein Eintrag vor-
# handen sein. IntervalLink und IntervalVC spezifizieren die Logging-
# Intervalle fuer die Link-Logfiles bzw. VC-Logfiles.
#
# Die Intervalle haben folgende Semantik:
# -1 => kein Profil, sondern nur eine Logzeile fuer gesamte aktive Phase
#  0 => feinstmoegliches Profil (Schleife ohen Sleep)
# >0 => Granularitaet des Profils in Millisekunden


LinkNumber:       0           #Linknumber 0
IntervalLink:     -1
IntervalVC:       -1

LinkNumber:        1          # --> sunemil
IntervalLink:     -1          # no smooth profile
IntervalVC:       -1          #   "   "    "

LinkNumber:        8          # --> sunotto
IntervalLink:     -1
IntervalVC:       1500        # smooth profile, one sample per 1.5 secs

LinkNumber:        9          # --> sunerna
IntervalLink:     -1
IntervalVC:       -1

LinkNumber:        16         # --> sunotto
IntervalLink:     -1
IntervalVC:       1500        # smooth profile, one sample per 1.5 secs
```

Alle Zahlenangaben haben die Einheit Millisekunden. Der Wert „-1“ bewirkt, daß das Loggingprogramm in der Betriebsart ohne genaues Profil arbeitet. Der Eintrag „IntervalLink“ definiert dabei den Modus für die Logdateien auf Linkebene und „IntervalVC“ den Modus auf Verbindungsebene (für alle Kanäle auf einem bestimmten Link). In der obigen Beispieldatei gibt es keine genauen Profile für Links, nur für die Kanäle auf Link 8 und 16 wird mit einer zeitlichen Auflösung von 1,5 Sekunden der genaue Verlauf festgehalten.

Der Administrator kann die Datei von Hand mit einem beliebigen Editor verändern (z.B. „vi %SWITCH%_log_info“) oder im Menü „LRZ-ATM→Edit Logging-Parameter“ anwählen, wobei er zuerst den entsprechenden Switch mit der Maus aktivieren muß.

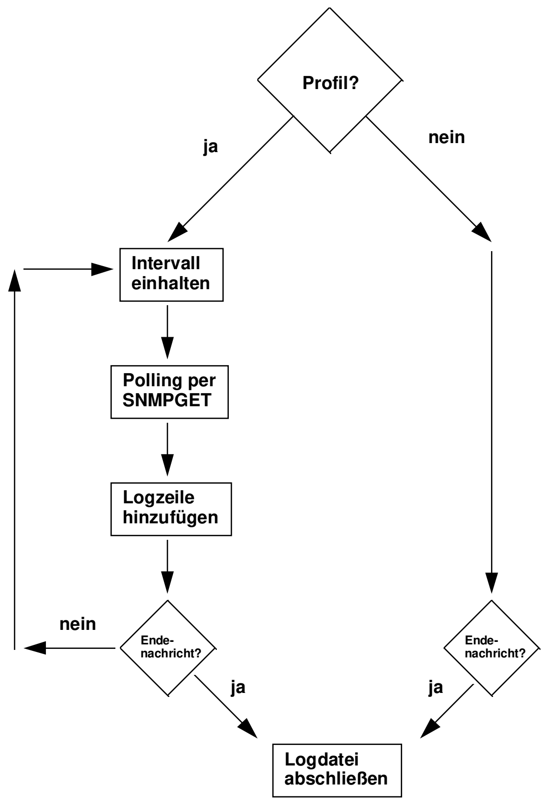
4.2.3. Arbeitsweise der Loggingprozesse

Wie bereits im vorigen Kapitel erwähnt, ruft der NodeManager die Loggingprozesse nach Auftreten der entsprechenden Events auf. Jene stellen ab diesem Zeitpunkt von OVW völlig unabhängige UNIX-Prozesse dar. Die Loggingprogramme stehen im Verzeichnis „bin“ und heißen „lrz_atm_(de)ac_link“ bzw. „lrz_atm_(de)ac_vc“ für Link- bzw. Verbindungsebene. Zu Beginn der aktiven Phase wird ein „Activate“-Prozeß gestartet, der dann auf eine Endenachricht vom „Deactivate“-Prozeß wartet, der am Ende der Phase (beim Unterschreiten des Rearm-Thresholds) aufgerufen wird. Die Interprozeßkommunikation wickelt dabei eine FIFO-Datei ab.

Im Folgenden wird nun die Arbeitsweise exemplarisch für die Linkebene dargestellt, die mit dem Loggen auf Verbindungsebene bis auf wenige Details identisch ist.
Das Programm bekommt drei Parameter von OVW/NNM übergeben: Den Hostname des Knotens im Netz, auf dem die aktive Phase registriert wurde, die MIB-OID und die Nummer der Instanziierung der MIB-Variable.

Nach Öffnen der Logdatei wird die FIFO-Datei erzeugt, über die die Interprozeßkommunikation abläuft. Eine FIFO-Datei ist eine Unix-Datei, die wie eine Warteschlange nach dem FirstIn/FirstOut-Prinzip arbeitet. Der Endeprozeß schreibt in die Datei, während der vom Threshold-Event zuerst aufgerufene Prozeß „am anderen Ende der Datei horcht“. Nun verzweigt das Programm in die Funktion `lrz_atm_DoLogging`, die für das Loggen zuständig ist.

Ein Flußdiagramm soll die Arbeitsweise dieser Funktion verdeutlichen:



Der linke Zweig realisiert den Betriebsmodus mit Profil (siehe 4.2.2.), bei welchem der genaue Verlauf der aktiven Phase durch ständiges Polling per SNMPGET festgehalten wird. Der rechte Zweig steht für den Betriebsmodus ohne Profil (Pollingintervall -1).

Wie bereits erläutert, schickt der „Deactivate“-Prozeß die Endenachricht über die FIFO-Datei, an dessen anderem „Ende“ der „Activate“-Prozeß horcht. Die Endenachricht besteht aus dem Zählerstand am Schluß der aktiven Phase und dessen Zeitpunkt. Mit diesen Informationen kann der „Activate“-Prozeß die aktive Phase in der Logdatei abschließen.

Das „Activate“-Programm für die Verbindungsebene zählt noch zusätzlich in der Datei „log/%SWITCH%_%LINK%_count“ die Anzahl der aktiven Phasen, die es augenblicklich auf dem Link gibt. Diese Information erleichtert die Arbeit des Statistikteils.

Zum Schluß dieses Kapitels muß nun noch ein wichtiger Punkt angesprochen werden. Ganz offensichtlich muß unbedingt dafür gesorgt werden, daß der „Deactivate“-Prozeß aufgerufen wird bzw. der Rearm-Event ausgelöst wird. Ansonsten wartet der „Activate“-Prozeß vergeblich auf die Endenachricht und wird zum „Prozeßzombie“. Aus diesem Grund darf der Rearm-Threshold nicht zu gering gewählt werden, da er aufgrund des ständigen Zellflusses auf den Links nie mehr unterschritten wird (siehe 4.2.1.). Ein Ansatzpunkt zum Überwachen der Events bildet der Menüpunkt „LRZ_ATM→Watch-NNM Events“, der die auftretenden Events auflistet. Ebenso kann der Systemadministrator mit „ps -auxw | grep lrz_atm“ von Zeit zu Zeit feststellen, wieviel Prozesse gerade aktiv sind.

Verschlimmert wird die Situation noch dadurch, daß OVW/NNM von Zeit zu Zeit Events „verschluckt“, was die gleiche katastrophale Auswirkung hat. Insgesamt stellt dies eine unbefriedigende Situation dar. Hier ist zweifelsfrei ein Ansatzpunkt für zukünftige Weiterentwicklungen gegeben. Denkbar wäre zum Beispiel die Entwicklung eines eigenen „NodeManagers“, der die Aufgaben des NNM übernimmt und dazu alle in der Komponentendatei „atm_components“ eingetragenen Netzknoten selbst überwacht. Dies hätte auch den Vorteil, daß das Programm über die gestarteten Prozesse Buch führen könnte und bei Bedarf die Prozesse beenden könnte.

4.2.4. Die erzeugten Logdateien

Es gibt zwei verschiedene Arten von Logdateien der Netzlast. Es gibt einerseits Dateien für die Linkebene und andererseits Dateien für die Verbindungsebene. Auf Linkebene gibt es wiederum drei unterschiedliche Typen, was mit der bidirektionalen Natur der Links zu tun hat. Die Dateien „log/%SWITCH%_%LINK%_in_cells“ repräsentieren den Fall, daß der Switch Zellen empfängt. „log/%SWITCH%_%LINK%_out_cells“ bedeutet den umgekehrten Fall. „%SWITCH%“ und „%LINK%“ stehen für den Hostnamen des Switches bzw. der Linknummer. Schlägt die Richtungsbestimmung fehl (dies sollte eigentlich nie passieren, kann aber bei bestimmten Fehlerabfragen im Programm vorkommen), heißt die Logdatei „log/%SWITCH%_%LINK%_xxx_cells“.

Ein Beispiel: in der Datei „swatm1_9_out.log“ wurde eine aktive Phase auf Link 9 des Switches „swatm1“ protokolliert (mit der Richtung „Senden“). Bei den Verbindungen gibt es nur eine Sorte von Dateien mit folgender Nomenklatur:
„%SWITCH%_%LINK%_%VPI%_%VCI%._cells“.

Der Dateinhalt selbst unterscheidet sich jedoch auf Link- und Verbindungsebene nicht. Hier ein Auszug aus einer Logdatei:

```
Wed, 09.08.1995, 10:12:57:     8.252 kbps passed over 1.747 seconds (34 cells).  @ Exact
time-stamp, cell value:807955977 944488, 3853018
Profil-Start:
Wed, 09.08.1995, 10:12:59:     9.076 kbps passed over 1.495 seconds (32 cells).  @ Exact
time-stamp, cell value:807955979 440221, 3853050
Wed, 09.08.1995, 10:13:00:    16.123 kbps passed over 1.499 seconds (57 cells).  @ Exact
time-stamp, cell value:807955980 940204, 3853107
Wed, 09.08.1995, 10:13:02:     9.045 kbps passed over 1.500 seconds (32 cells).  @ Exact
time-stamp, cell value:807955982 440213, 3853139
Wed, 09.08.1995, 10:13:03:    16.112 kbps passed over 1.500 seconds (57 cells).  @ Exact
time-stamp, cell value:807955983 940241, 3853196
Wed, 09.08.1995, 10:13:05:     9.045 kbps passed over 1.500 seconds (32 cells).  @ Exact
time-stamp, cell value:807955985 440263, 3853228
Wed, 09.08.1995, 10:13:06:     8.486 kbps passed over 1.499 seconds (30 cells).  @ Exact
time-stamp, cell value:807955986 940221, 3853258
Wed, 09.08.1995, 10:13:08:    16.395 kbps passed over 1.500 seconds (58 cells).  @ Exact
time-stamp, cell value:807955988 440224, 3853316
Wed, 09.08.1995, 10:13:09:     8.486 kbps passed over 1.499 seconds (30 cells).  @ Exact
time-stamp, cell value:807955989 940219, 3853346
Profil-End:
```

Zu unterscheiden sind hier die beiden Modi mit und ohne exaktem Profil. Aktive Phasen mit protokolliertem Profil sind von den zwei Zeilen „Profil-Start“ und „Profil-End“ umklammert. Eine aktive Phase mit Profil umfaßt also alle Zeilen zwischen diesen Begrenzungen. Die angebene benutzte Bandbreite bezieht sich dabei immer auf die vorangegangene Zeile.

Anders sieht es bei Zeilen aus, die nicht zu einem Profil gehören. Dann repräsentiert jede Zeile *eine* aktive Phase. Die Bandbreite ist somit ein Durchschnittswert für die gesamt aktive Phase. Die Werte hinter dem Klammeraffen werden programmintern gebraucht.

Schließlich gibt es noch die Dateien %SWITCH%_%LINK%_count“, die Informationen darüber enthalten, wieviele Kanäle wann auf dem Link aktiv waren, z.B.

```
Wed, 16.08.1995, 19:18:04:    Active VCs: 0
Wed, 16.08.1995, 19:23:01:    Active VCs: 1
Wed, 16.08.1995, 19:23:05:    Active VCs: 2
Wed, 16.08.1995, 19:23:07:    Active VCs: 1
```

4.3. Protokollieren der geschalteten Verbindungen

4.3.1. Der Channeldaemon

Es stellte sich heraus, daß der NodeManager beim Überwachen bestimmter MIB-Variablen zu unflexibel ist. Konkret trat ein derartiges Problem beim Monitoring der Anzahl der konfigurierten Kanäle auf. Die „normale“ Lösung zum Erzeugen eines Events beim Ändern der Zahl konfigurierter Channels wäre über sog. MIB-Expressions gewesen, bei denen man in einer eigenen Sprache Bedingungen unter Einbeziehung von MIB-Werten angeben kann, unter denen ein NNM-Event ausgelöst werden soll. Leider ist es aber auch mit den verschiedensten MIB-Expressions nicht möglich, einen Event zu generieren, wenn ein neuer Kanal neu konfiguriert oder ein bestehender gelöscht wird. Diese Situation läßt sich auf folgendes Problem reduzieren: Wie generiert man einen Event, wenn sich eine MIB-Variable vom Typ „Gauge“ (also eine Pegelvariable) *ändert*?

Aus diesem Grund bedurfte es der Notlösung des hier beschriebenen „Channeldeamon“. Die Idee ist, daß man selbst kontollieren muß, ob und welche Channels auf einem Link hinzugefügt oder gelöscht wurden. Dazu ist es nötig, daß ständig für jeden Switch eine Instanz unseres Daemons im Hintergrund läuft und für jeden gewünschten Link (der Anwender gibt beim Start des Daemon die Instanzen der Links an, die beobachtet werden sollen) die aktuelle Zahl von konfigurierten Channels (diese ist bei den uns bekannten Switches über eine MIB-Variable festzustellen) mit der früheren (die der Daemon gespeichert hat) in festzulegenden Intervallen vergleicht. Ist innerhalb dieses Intervalls auf einem Link ein Channel hinzugekommen oder wurde einer gelöscht, so wird die Liste konfigurierter Channels aktualisiert und abgespeichert.

Das Hauptproblem dieser Notlösung ist, daß die dauernden MIB-Abfragen ziemlich zeitaufwendig sind und bei wachsendem Netz (zunehmender Anzahl Switches => zunehmende Anzahl Daemoninstanzen) den managenden Computer immer mehr belasten werden. Aus diesem Problem ergibt sich ein weiteres, denn es ist nötig, einen Kompromiß zwischen einer hohen Genauigkeit und der Belastung der Komponente zu finden. Die mangelnde Genauigkeit ergibt sich dadurch, daß man innerhalb des vorgegebenen Intervalls keine Chance hat, festzustellen, ob evtl. zwar ein Channel hinzugekommen ist, sich aber die Gesamtzahl nicht geändert hat, weil noch während des Intervalls ein anderer oder der gleiche Channel gelöscht wurde. Dies ist ziemlich unbefriedigendend, es wäre auf jedenfall wichtig, herauszufinden, ob evtl. bei neueren Versionen von OVW der NodeManager vielleicht die oben beschriebene Fähigkeit doch hat und dadurch dieser Daemon unnötig würde.

Bedienung des Channeldaemons:
Der Daemon muß „von Hand" gestartet werden; für jeden zu beobachtenden Switch mit einer Liste von Instanzen, die angibt, welche Links auf dem Switch überhaupt überwacht werden sollen.

Der Aufruf sieht folgendermaßen aus:

„lrz_atm_chand hostname interval (in ms) oid instance1 instance2 ..."

Die Parameter sind im einzelnen folgende:

- *Hostname*:
  Name des zu überwachenden Switches
- *interval*:
  Intervall in ms, in dem die MIB-Variable mit den gespeicherten Werten zu vergleichen ist (Probleme s.o.)
- *oid*:
  Hier ist die oid der MIB-Variablen anzugeben, die die Zahl der Channels pro Link angibt. Bei dem uns vorliegenden Fore-Switch war das die im Beispiel angegebene
- *instancen*:
  Liste von Instanzen, die angibt, welche links auf dem Switch zu beobachten sind

Beispiel:

„lrz_atm_chand swatm1 3000.1.3.6.1.4.1.326.2.2.2.1.3.1.1.6
0.0 1.0 8.0 8.1 9.0 16.0 16.4 16.6"

Um festzustellen welche Channels auf einem Switch neu hinzugekommen sind und um diese in die FileChannelRouteList (steht in %SWITCH%_VC) und RealChannelRouteList (die tatsächlich konfigurierten Channels) einzufügen, verwendet der Channeldaemon Funktionen aus dem File „lrz_atm_GetConfChannels.c", die im folgenden beschrieben werden sollen:

4.3.2. Die Funktionen in lrz_atm_GetConfChannels.c

- `lrz_atm_GetConfChannels`:
  Die zentrale Funktion in diesem File erstellt zu einem Switch (Name beim Aufruf übergeben) eine Liste aller auf seinen Links konfigurierten Channels (nur der Kontrollport

wird herausgefiltert, da die auf ihm übertragenen Daten im allgemeinen bei Beobachtung des Netzverkehrs keine Rolle spielen). Der Rückgabewert der Routine ist ein Zeiger auf die erstellte Liste. Der Speicherplatz für die Liste wird dynamisch allokiert, deshalb muß anschliessend lrz_atm_`FreeChannelRouteList` mit dem Zeiger auf diese Liste aufgerufen werden. Informationen über die aktuell konfigurierten Channels holt diese Routine aus der MIB, deswegen ist sie recht zeitaufwendig. Sie ist leider sehr lang geworden, aber ein Zerlegen in Unterfunktionen wäre zu umstaendlich geworden. So wurde versucht sie durch Struktur und Kommentare verständlich zu halten.

- `lrz_atm_ReadFileChannelRouteList`:
  Das Gegenstück zu `lrz_atm_GetConfChannels` liest aus dem File %SWITCH%_VC in eine Liste ein, welche Channels dort zu diesem Switch aufgeführt sind. Gibt einen Zeiger auf diese Liste zurück, so das beide Listen vom ChannelDaemon durch Aufruf der nächsten beiden Funktionen verglichen und aktualisiert werden können.

- `lrz_atm_ConfigureChannelInList`:
  Diese Funktion wird vom Channeldaemon aufgerufen, mit den beiden oben beschriebenen Listen und sucht zu jedem in der preallist enthaltenen ChannelEntry, ob ihm ein Eintrag in der pfilelist entspricht. Ansonsten wird dieser in pfilelist an der entsprechenden Stelle eingefuegt. Anschliessend muss die neue filelist wieder in die Datei zurückgeschrieben werden (mit `lrz_atm_WriteFileChannelRouteList`).

- `lrz_atm_UnConfigureChannelInList`:
  Das Gegenstück zu `lrz_atm_ConfigureChannelInList` vergleicht pfilelist und preallist und setzt evt. nicht mehr konfigurierte Channels in der pfilelist auf invalid und dazu einen Timestamp. Wegen dieser Timestamps reicht es nicht aus, einfach nur die aktuelle Liste der konfigurierten Channels zu speichern, denn früher einmal konfigurierte sollen für nachträgliche Beobachtungen in der Datei %SWITCH%_VC weiterhin auftauchen.

- `lrz_atm_CheckSwitchLinkVCFiles`:
  Diese Routine wird am Ende von `lrz_atm_GetConfChannels` aufgerufen, um die %SWITCH%_%LINK%_VC_Count-Files auf Aktualität zu überprüfen. In diesen Files steht zu jedem Link mit Timestamp die Zahl der dort zu den angegebenen Zeitpunkten konfigurierten Channels.

Die übrigen Funktionen dienen dem Einsortieren, Suchen und Erweitern in den Listen - ihre Funktionsweise ist im Quelltext kommentiert.

4.4. Protokollieren von Fehlern

Neben der Protokollierung des Netzverkehrs gibt es noch die Möglichkeit, bestimmte MIB-Variablen zu überwachen, die nichts mit dem Zellfluß zu tun haben. Besonders interessant hierfür sind MIB-Variablen, die Fehler wie z.B. „hwPortErrors“ anzeigen.
Die auftretenden Fehler werden in einer eigenen Datei pro Switch gespeichert.

4.4.1. Einrichten der Nodemanagerevents

Das Ablaufmuster beim Überwachen der Fehler-MIB-Variablen sieht genauso aus wie beim Überwachen der Netzlast (siehe 4.2.). Das Monitoring übernimmt also wiederum der NodeManager, der bei Auftreten eines Events dann das Programm „bin/lrz_atm_error" aufruft, welches die Eintragungen in der Fehler-Logdatei vornimmt.

Zuerst muß der Administrator die NNM-Events einrichten. Hier kann er sich an folgende Prozedur halten, die er für jeden Switch durchführen muß:

1. Suchen der gewünschten MIB-Variablen. Hierbei muß es sich um eine Variable vom Typ „LRZ_ATM_INSTANCE_LINK_ERROR" handelt. Was das bedeutet, ist in Kapitel 4.5 nachzulesen.

2. Eintragen des (Start-) Schwellwerts: Dieser muß Null betragen, da jeder Fehler eingefangen werden soll.

3. Eintragen des Rearm-Schwellwerts: Ebenfalls Null

4. Instanzen: Das hängt sehr von der Art der MIB-Variable ab. Im Prinzip sollte es sich aber um Variablen auf Linkebene handeln, so daß hier beispielsweise die benutzten Links einzutragen sind. Grundsätzlich gilt: So wenig wie möglich, so viel wie nötig

5. Auf keinen Fall „Check Thresholds And Store", sondern „Check Thresholds" anwählen, da der NNM ansonsten unnötige Logdateien anlegt.

6. Pollingfrequenz: Ein sehr heikler Punkt. Das Pollingintervall stellt die minimale Ungenauigkeit bei allen Messungen dar. Sie ist dem Internetmanagement zu verdanken, das die Schwellwertprüfung dem Manager überträgt.

7. Nun muß noch der Pfad des Loggingprozesses eingetragen werden. Siehe Installationsprozedur in (4.2.1.), nur heißt hier das Programm „lrz_atm_error". Es wird kein Rearm-Prozeß benötigt.

8. Einschalten der Schwellwertüberprüfung mit „Resume".

Jede überwachte MIB-Variable muß der Administrator auch in der „Error Section" der Herstellerdatei spezifizieren, da über diese die Fopraapplikation nach Auftreten des Events den Fehlertyp anhand der vom NodeManager übergebenen MIB-OID identifiziert. Der Fehlertyp ist dann anders als die MIB-OID eine hersteller*un*abhängige Größe.

Der entsprechende Abschnitt könnte folgendermaßen aussehen:

```
------------------------------------------------------------
ErrorSection:
IfError:       ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.10"
IfOverflow:    ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.9"
------------------------------------------------------------
```

4.4.2. Die erzeugten Logdateien

Jeder Link auf jedem Switch besitzt seine eigene Logdatei „log/%SWITCH%_%LINK%_errors“, in der die Fehler gespeichert werden. Jede Zeile enthält den Zeitpunkt des Auftretens, den herstellerunabängigen Fehlertyp und den Inhalt der MIB-Variable (in Klammern):

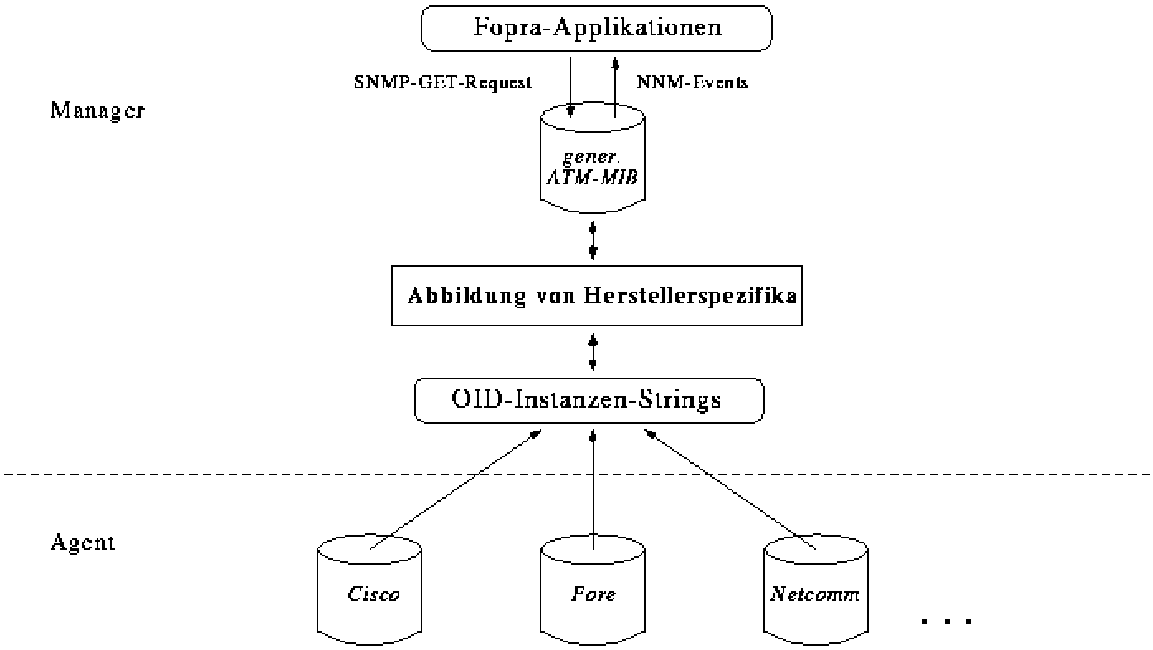
```
Wed, 16.08.1995, 19:23:01: IfError (1)
Sat, 19.08.1995, 12:53:21: IFOverflow (1)
Sun, 20.08.1995, 22:43:37: IfError (1)
```

4.5. Die generische MIB

Zum Zeitpunkt der Bearbeitung des Fortgeschrittenenpraktikums gab es noch keine standardisierte ATM-MIB. Die Praktikanten waren also auf die MIBs der Hersteller angewiesen, um an die für die ATM-Technologie relevanten Informationen heranzukommen.

Damit steht man allerdings vor einem Dilemma: Die ATM-MIBs der Hersteller sind alle verschieden. Es mußte also ein Modell entworfen werden, das all diese unterschiedlichen MIBs aufeinander abbildet, damit die Meß- und Statistikprogramme auf einer einheitlichen Grundlage arbeiten können. Hier kommt die Idee einer *generischen MIB* zum Tragen. Generisch bedeutet in diesem Zusammenhang einerseits, daß es sich nicht um eine MIB im gewöhnlichen Sinne handelt, die in den Agenten implementiert ist, sondern um eine logische MIB, die sämtliche Hersteller-MIBs kapselt. Andererseits heißt generisch, daß die MIB für beliebig viele Hersteller erweitert werden kann.

Die folgende Abbildung stellt das Modell der generischen MIB dar:



Wenn man mit SNMPGET auf eine Variable einer Hersteller-MIB zugreifen will, muß man in der SNMPGET-PDU einen String bestehend aus der MIB-OID und der Instanz der Variable spezifizieren. Auch in umgekehrter Richtung, nach Auftreten eines Events, bekommen die Fopra-Applikationen vom NodeManager diese vom Hersteller abhängigen OID-Instanzenstrings übergeben. Diese müssen also nun unter Berücksichtigung der Herstellerspezifika aufeinander abgebildet werden, wobei man aufgrund der teilweise erheblichen Unterschiede zwischen den einzelnen MIBs, die weit über nur unterschiedliche OIDs der Variablen hinausgehen, nicht um eine Kodierung per Programm herumkommt. Dies geschieht in einem von den übrigen Programmteilen getrennten Modul: „lrz_atm_vendorcode.c“. Ausschließlich dort taucht herstellerabhängiger Programmcode auf, während alle anderen Quelltexte völlig herstellerunabängig sind. Das Ergebnis dieser Abbildung ist die generische ATM-MIB, auf die über die Funktionen aus „lrz_atm_vendorcode.c“ zugegriffen werden kann.

Dieses Modul muß also für jeden zusätzlichen Hersteller entsprechend erweitert werden. Ein Ausschnitt aus dem Programmtext soll den Aufbau dieses Moduls veranschaulichen. Es handelt sich um die Routine `lrz_atm_GetVPIOutOfInstanceString`, die aus dem Instanzenstring die herstellerunabhängige Größe „VPI“ extrahiert (dies entspricht in der vorigen Abbildung der Richtung von unten nach oben):

```
switch (vendorindex)
    {
        /* Fore Switch */
        /* ----------- */
```

```
        case 2:
        /* Falls der VPI in der Instanz vorkommt, steht er bei */
        /* Fore IMMER an der 2. Stelle */
        /* ------------------------------------------------------ */
        text    = strtok (instance_buffer, ".");
        text    = strtok (NULL, ".");
        result  = (long int) atoi (text);
        break;

        /* Fore Host */
        /* ----------- */
        case 3:
        /* Falls der VPI in der Instance vorkommt, steht er bei */
        /* ForeHOSTS IMMER an der 3. Stelle */
        /* ------------------------------------------------------- */
            text    = strtok (instance_buffer, ".");
            text    = strtok (NULL, ".");
            text    = strtok (NULL, ".");
            result  = (long int) atoi (text);
            break;

        default:
            result = -1;
            break;

    }
    return (result);
}
```

Die aufrufende Funktion muß der Routine zwei wichtige Informationen mitliefern:

1. Um welchen Hersteller handelt es sich?
2. Um welchen Typ von MIB-Variable handelt es sich?

Die erste Information wird über den bereits in Kapiel 4.2.3. eingeführten „Vendorindex“ bereitgestellt. Für jeden neuen Hersteller (man beachte auch die zusätzliche Unterscheidung Switch/Host!) muß ein neuer „Case“-Block in der „Switch“ eingefügt werden.

Bei der Erklärung der zweiten Information, des Instanztyps, muß etwas weiter ausgeholt werden. Offensichtlich gibt es verschieden „Klassen“ von MIB-Variablen. Da gibt es z.B. die MIB-Variablen auf Linkebene, deren Instanziierung nur von der Linknummer abhängt, im Gegensatz zu MIB-Variablen auf Verbindungsebene, wo neben Linknummer auch die beiden Verbindungsidentifikatoren „VPI“ und „VCI“ vorkommen. Beispiele bei Fore:

PortReceivedCells.x und ChanCells.x.y.z (x = Port, y = VPI, z = VCI)

Die Information „Linknummer“ steht in diesem Fall immer an erster Stelle. Dies muß bei anderen Herstellern aber nicht unbedingt so sein. Folgende Situation ist denkbar:

LinkIncomingCells.x und ConnectionCells.y.z.x

Die Funktion, die die Linknummer bestimmen soll, kann sich also nicht darauf verlassen, daß die Linknummer bei verschiedenen Typen von MIB-Variablen an der selben Stelle steht. Damit die Funktion den Instanzstring richtig interpretiert, muß der Instanztyp angegeben werden.

Dies gilt natürlich auch für den VPI aus dem obigen Quelltext. Im Fall Fore befindet sich der VPI jedoch immer an zweiter Stelle, was eine Unterscheidung der Instanztypen hier unnötig macht. Bei anderen Herstellern ist aber unter Umständen ein zweiter „Switch"-Block erforderlich:

```
switch (vendorindex)
    {
        /* Robotron Switch */
        /* -------------------- */
        case 2:

           switch(instance_type)
           {
            case 1:  .....
                  break;
            case 2:  .....
                  break;
            .........
           }
        .......
     }
```

Selbstverständlich müssen alle Funktionen aus „lrz_atm_vendorcode.c" bei neuen Herstellern erweitert werden.

Bis jetzt sind folgende Instanztypen in der Includedatei „lrz_atm.h" spezifiziert:

```
/* Instanz-Typen */
#define     LRZ_ATM_INSTANCE_ADAPTER           1
#define     LRZ_ATM_INSTANCE_LINK              2
#define     LRZ_ATM_INSTANCE_PATH              3
#define     LRZ_ATM_INSTANCE_CHANNEL           4
#define     LRZ_ATM_INSTANCE_CHANNELROUTE      5
#define     LRZ_ATM_INSTANCE_LINK_ERROR        6
```

Der erste Typ „LRZ_ATM_INSTANCE_ADAPTER" entspricht der Verbindungsebene bei Hostsadaptern. Instanzen vom Typ „LRZ_ATM_INSTANCE_PATH" enthalten neben Linknummer auch den VPI, ein Beispiel hierfür wären bei Fore die Variable „pathNumChannels". „LRZ_ATM_INSTANCE_CHANNELROUTE"-Variablen enthalten bei Fore sechs Informationen: Source-Link, Source-VPI, Source-VCI und Destination-Link, -VPI, -VCI.

„LRZ_ATM_INSTANCE_LINK_ERROR" bezeichnet MIB-Variablen, die zur Fehlerüberwachung herangezogen werden (siehe 4.4.). Dabei handelt es sich ebenfalls um Variablen auf Linkebene, wobei aber auch andere Typen denkbar sind.

Jetzt fehlt noch eine Spezifikation der MIB-Variablen der generischen, herstellerunabhängigen MIB. Dies erfolgt in den sog. „Vendordateien", wo die Abbildung der MIB-OIDs erfolgt. Der Dateiname entspricht der MIB-II-Variable „sysDescr".

Beim Fore-Switch hat diese Datei folgendes Aussehen:

```
---------------------------------------------------------------
VendorIndex:            2
---------------------------------------------------------------
ChannelMean:            "-1"
ChannelPEAK:            "-1"
ChannelQoS:             "-1"
ChannelStatus:          ".1.3.6.1.4.1.326.2.2.2.1.4.2.1.7"
---------------------------------------------------------------
IsNodeSwitch:           ".1.3.6.1.4.1.326.2.2.2.1.4.2.1.7"
---------------------------------------------------------------
LinkRxCells:            ".1.3.6.1.4.1.326.2.2.2.1.2.2.1.12"
LinkTxCells:            ".1.3.6.1.4.1.326.2.2.2.1.2.2.1.18"
---------------------------------------------------------------
ErrorSection:
IfError:                ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.10"
IfOverflow:             ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.9"
---------------------------------------------------------------
ControlPortNr:           56
```

Zu Beginn ist der Vendorindex des Herstellers definiert. Die Routinen aus „lrz_atm_vendorcode.c" beziehen sich auf diesen Wert, der eindeutig vergeben werden muß. Danach folgen Variablen auf Verbindungsebene. Fore unterstützt einige Funktionalitäten noch nicht, so daß einige Variablen als „-1" markiert zurückgestellt werden mußten. Aufmerksame Leser werden vielleicht die schon öfters erwähnte Fore-MIB-Variable „chanCells" vermissen, mit der die Netzlast auf Verbindungseben gemessen wird. Diese ist jedoch derzeit noch nicht notwendig, da der NodeManager bereits den kompletten OID-Instanzenstring übergibt. Die Loggingprogramme verwenden dann direkt diesen String, ohne den Umweg über die generische MIB gehen zu müssen.

Auf Linkebene dürfen diese Variablen (LinkRx/TxCells) aber nicht fehlen, damit der Loggingprozeß anhand der vom NodeManager übergebenen OID die Übertragungsrichtung identifizieren kann (siehe 4.2.3.).

Die Variable „IsNodeSwitch" wird im ersten Aufgabenbereich gebraucht, wo es darum geht zu überprüfen, ob eine neu hinzugefügte Komponente ein Switch oder ein Host ist. Beim Fore-Switch wird dazu die „ChannelStatus"-Variable benutzt (siehe Datei oben). Gibt es diese Variable, handelt es sich um einen Switch, bei einem Adapter ist diese Variable nicht vorhanden.

Informationen zur „Error-Section" finden sich in Kapitel 4.4. Die Zeile „ControlPortNr" legt die Nummer des Controlports fest, damit die für Statistikzwecke uninteressanten Verbindungen, die zum Controlport hin- und wegführen, rausgefiltert werden können. Ob diese Controlports nur beim ForeSwitch auftauchen und somit eigentlich nichts in der generischen MIB zu suchen haben, oder ob es bei anderen Herstellern ähnliches gibt, bleibt abzuwarten.
Wichtig ist, daß all diese Variablen für jeden Hersteller definiert werden. Die Herstellerdateien haben also immer den genau gleichen Aufbau, nur die OIDs variieren.

Die Gesamtheit der Vendordateien in Vebindung mit dem Modul „lrz_atm_vendorcode.c" bildet also die generische MIB. Es gilt aber noch eine Reihe von Verbesserungsvorschlägen zu implementieren. So könnte die Definition der Instanztypen aus der Includedatei in eine definierte Datei ausgelagert werden. Auch wäre es sinnvoll, gleich bei der Definition der Variablen in den Herstellerdateien den Instanztyp zu definieren, so daß dies nicht an anderer Stelle (siehe 4.4.2 bei „tracemibs“), wo der Instanztyp benötigt wird, erfolgen muß. Das hätte eine konsistentere und kompaktere Beschreibung der generischen MIB zur Folge. Die Bezeichnung von Vendorindex und Instanztyp mit Strings statt mit Zahlen schließlich würde dem Administrator mehr Komfort bieten.

4.6. Problemkreise

4.6.1. OVW/NodeManager-Probleme

Es stellte sich heraus, daß der NodeManager beim Überwachen bestimmter MIB-Variablen zu unflexibel ist. Konkret trat ein derartiges Problem beim Monitoring der Anzahl der konfigurierten Kanäle auf. Auch mit den verschiedensten MIB-Expressions ist es nicht möglich, einen Event zu generieren, wenn ein neuer Kanal neu konfiguriert wurde oder ein bestehender aufhörte zu existieren. Diese Situation läßt sich auf folgendes Problem reduzieren: Wie generiere ich einen Event, wenn sich eine MIB-Variable vom Typ „Gauge“ (also eine Pegelvariable) *ändert*?
Aus diesem Grund bedurfte es der Notlösung „Channeldeamon“ (siehe 4.3.). Dieser Hintergrundprozeß übernimmt an dieser Stelle die Ausgaben des NodeManagers.

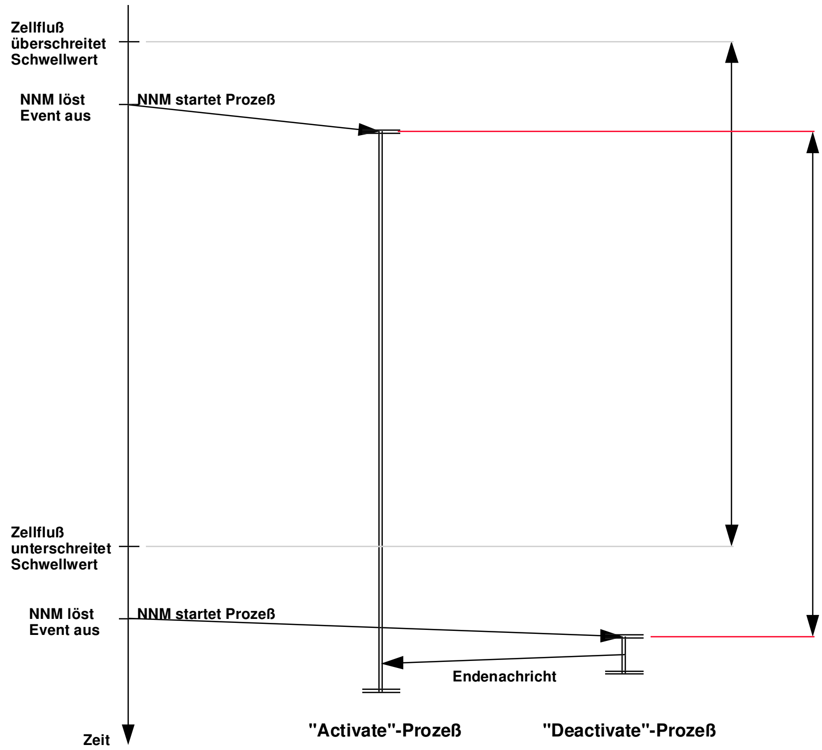
Ein weiteres, bereits im Kapitel 4.2.3. angesprochenes Problem ist das Auslassen von Events durch den NodeManager. Das Problem scheint umso häufiger aufzutreten, je höher die Pollingfrequenz ist. Vor allem beim Überwachen der Netzlast wirkt sich das Auslassen von Events aufgrund der eventuell fehlenden Endenachricht fatal aus.

4.6.2. Eingeschränktes Testszenario

Ein wesentlicher Anspruch des Fortgeschrittenenpraktikums war die Herstellerunabhängigkeit (siehe 4.4.). Das diesem Praktikum zugrundeliegende Testnetz bestand jedoch lediglich aus einem Switch (ForeRunner ASX-200). Es waren zwar noch die MIBs andere Hersteller verfügbar, diese waren aber aufgrund fehlender Switches nicht mit Leben gefüllt und somit relativ wenig hilfreich.

4.6.3. Prinzipielle Probleme des Internetmanagement

Nicht ohne Grund erfreut sich das Internetmanagement so großer Beliebtheit, sind doch die Aufgaben extrem asymmetrisch zu Gunsten des Agents verteilt. Es fällt den Herstellern somit wesentlich einfacher, einen SNMP-Agenten für ihr Produkt zu implementieren als einen OSI-Agenten. Die Schwellwertüberprüfung ist Aufgabe des Managers. Er muß ständig SNMPGET-Requests absetzen, um den aktuellen Status zu erhalten. Dieses kontinuierliche Polling bringt außer zusätzlicher Netzlast auch eine Ungenauigkeit der Messungen mit sich: Tritt kurz nach einer Abfrage durch den Manager ein Überlauf statt, wird dieser erst bei der nächsten SNMP-Abfrage festgestellt. Innerhalb dieses Zeitraums können aber bereits Hunderttausende von Zellen „verlorengegangen“, d.h. nicht erfaßt worden sein. Die folgende Abbildung soll diesen Zusammenhang verdeutlichen:



Der linke doppelte Pfeil markiert die eigentliche aktive Phase, aber der effektiv protokollierte Zeitraum verschiebt sich um eine Zeitspanne, für die neben anderen Einflußgrößen (Aufrufzeit der Prozesse, Abarbeitungszeit der Prozesse) vor allem das Pollingintervall schuld ist.

5. Aufbereitung der gesammelten Informationen

In diesem Kapitel wird die Auswertung und Aufbereitung der bereits gesammelten Daten erläutert. Es wird dargelegt, auf welche Weise Informationen erzeugt und dem Netzadministrator zur Verfügung gestellt werden.

5.1. Überblick

Die Aufbereitung gesammelter Informationen basiert nicht auf dem Auslesen von MIB-Variablen. Sie stützt sich vielmehr auf der Auswertung von Dateien, deren Struktur und Erzeugung bereits in den vorangegangenen Kapiteln vorgestellt worden ist. Als die beiden wichtigsten Dateitypen sind zu nennen:

- Konfigurationsdateien
  (repräsentieren die Konfiguration des ATM-Netzes)
- Netzverkehrsdateien
  (geben Auskunft über die Last auf Links und Channels)

Aus den daraus gewonnen Informationen werden folgende Statistiken bzw. Dateien erzeugt, die dem Netzadministrator schließlich zur Verfügung stehen:

- Switch-Statistiken
- Netzweite Informationen über Verbindungen

Switch-Statistiken geben Auskunft über die innerhalb eines bestimmten Zeitraumes konfigurierten Verbindungen sowie die Verkehrscharakteristik eines Switches.

Netzweite Verbindungsinformationen betrachten nicht einzelne Komponenten, sondern Ende-zu-Ende-Verbindungen. Sie zeigen, zu welchem Zeitpunkt welche Verbindungen zwischen zwei Hosts in welchem Umfang genutzt worden sind. Aus ihnen ist außerdem ersichtlich, ob und auf welchen Kanälen Zellen verlorengegangen sind.

5.2. Switch-Statistiken

5.2.1. Bedienung

Statistiken bzw. Statistikdateien über die Konfiguration und Verkehrscharakteristik aller Switches im ATM-Netz, auf die auf verschiedene Weisen zugegriffen werden kann, werden durch den Aufruf des Programms `lrz_atm_logswitchstat` regelmäßig - in wählbaren Intervallen - erzeugt. Es ist für einen späteren Zeitpunkt vorgesehen, daß dies automatisch mit dem Start von OpenView geschieht. Der Administrator kann aber auch jederzeit eine Statistik für einen bestimmten - in der ATM-Submap per Mausklick ausgewählten - Switch auf zwei Arten erzeugen und anzeigen lassen:

- Menüauswahl: Fopra→Calculate current Switch Statistics
- Shell: Erzeugen: „lrz_atm_logswitchstat %SWITCH%“
  Anzeigen: (z.B. per „more“) von „.../statistik/%SWITCH%.tmp“
  (wobei %SWITCH% der Selection Name des entsprechenden Switches ist)

Dieses Vorgehen hat auf die intervallweise Erstellung von Statistiken keinen Einfluß, d.h. es wird für den entsprechenden Switch eine eigene, temporäre Datei erzeugt.

Zum Anzeigen einer intervallweise erzeugten Switch-Statistik gibt es zwei Alternativen:

- Menüauswahl: Fopra→Show Switch Statistics
  Hier ist darauf zu achten, daß zuvor der entsprechende Switch mit der Maus angeklickt worden ist.
- Shell: Anzeigen (z.B. per „more“) der Datei „.../statistik/%SWITCH%“
  (wobei %SWITCH% der Selection Name des entsprechenden Switches ist)

Definieren der Parameter:

Für die Erstellung von Switch-Statistiken werden Parameter benötigt, die in einer Datei gespeichert sind. Zur Änderung der Parameter kann der Administrator auf diese Datei auf zwei verschiedene Arten zugreifen:

- Menüauswahl: Fopra→Edit Statistic Parameters
- Shell: Editieren (z.B. per „vi“) der Datei „.../input/statistik_info“

5.2.2. Eingabedateien

Die Erzeugung von Switch-Statistiken wird - wie bereits erwähnt - von Parametern beeinflußt, die sich in der Datei „.../input/statistik_info“ befinden:

- `LogIntSwitchStat`:
  Dieser Parameter legt fest, in welchem Intervall (in Minuten) Switch-Statistiken erstellt werden sollen. Bei der Wahl dieses Parameters ist zu berücksichtigen, daß die Statistikerstellung längere Zeit in Anspruch nehmen kann und Dateien lockt, auf die auch andere Prozesse zugreifen. Je mehr Switches sich im Netz befinden, desto größer sollte das Intervall gewählt werden.

- `DelFreqSwitchStat`:
  Hiermit kann der Administrator bestimmen, nach jedem wievielten Intervall bereits ausgewertete Daten (Teile der Logging-Dateien pro Switch) gelöscht werden. Diese werden nicht gelöscht, falls der Administrator eine Statistik für einen bestimmten Switch anfordert.

Hier der entsprechende Ausschnitt aus der Eingabedatei:

```
# Switch-Statistiken
# ------------------
LogIntSwitchStat:   15      # Intervalllänge in Minuten
DelFreqSwitchStat:   3      # nach x Intervallen alte/
                              ausgewertete Daten loeschen
:
:
```

Das Programm zur Erstellung von Switch-Statistiken stützt sich auf folgende Dateien:

- Komponentendatei
  (= Verzeichnis aller im Netz befindlichen Nodes)
- Verbindungsdatei
  (= Verzeichnis aller physikalischen Verbindungen)
- %SWITCH%_VC_Dateien
  (enthalten Routingtabelle und Eigenschaften eines Switches)
- %SWITCH%_%LINK%_VC_Count-Dateien
  (enhalten Informationen über
  - die Anzahl konfigurierter Verbindungen
  - Summen der Mean- und Peak-Raten
  bezogen auf Links eines Switches innerhalb eines Intervalls)
- %SWITCH%_%LINK%_Count-Dateien
  (in diesen Dateien wurde regelmäßig für jeden Link die jeweils aktuelle Anzahl aktiver Phasen mitprotokolliert)
- Fehler-Dateien %SWITCH%_%LINK%_Error
  (enthalten auf Linkebene aufgetretene Fehler)

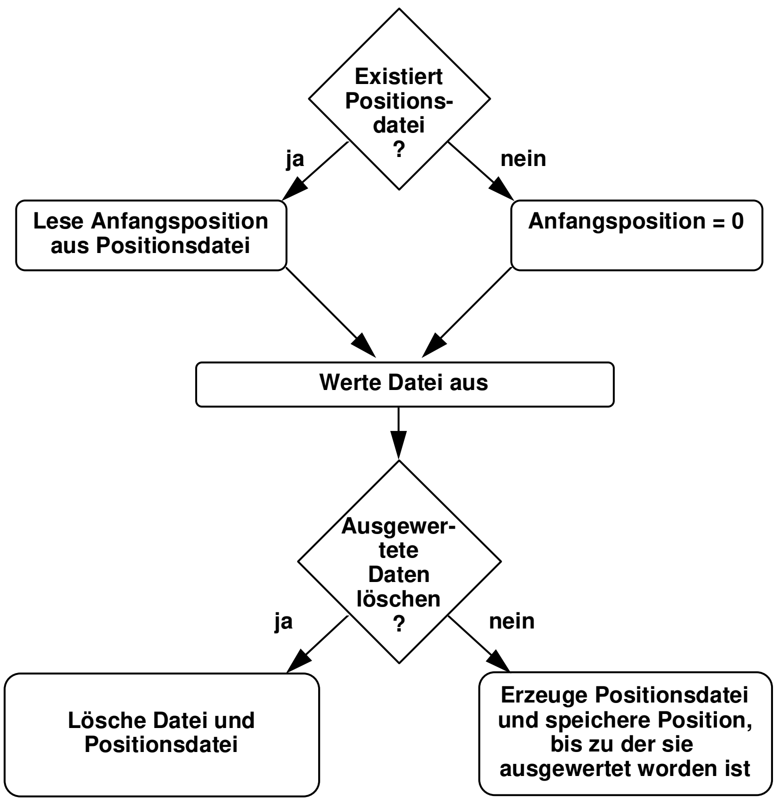
- Positionsdateien %DATEI%.pos
  (enthalten die Positionen von Dateien, bis zu denen diese bereits ausgewertet worden sind)

5.2.3. Positionsdateien

Wie bereits erwähnt, kann der Netzadministrator durch Parameterwahl bestimmen, nach wievielen Intervallen ausgewertete Daten gelöscht werden sollen. Es kann also vorkommen, daß sich in einigen Dateien (%SWITCH%_%LINK%_VC_Count-, %SWITCH%_%LINK%_Count- und %SWITCH%_%LINK%_Error-Dateien) *bereits* ausgewertete und *noch nicht* ausgewertete Daten befinden.

Damit noch nicht gelöschte Daten, die bereits in den Statistiken erfaßt worden sind, nicht wiederholt ausgewertet werden, muß registriert werden, bis zu welcher Position die davon betroffene Datei schon ausgelesen worden sind. Diese Position wird jeweils in einer Datei gespeichert, deren Name aus dem der ausgelesenen Datei und dem Suffix „.pos“ zusammengesetzt wird.

Den Ablauf einer Routine, die eine Datei unter Verwendung einer Positionsdatei auswertet, zeigt folgendes Diagramm:



5.2.4. Ausgabedateien

Die für einen Switch erstellte Statistikdatei „statistik/%SWITCH%“ (bzw. „statistik/%SWITCH%.tmp“ für einen aus der ATM-Submap ausgewählten Switch) enthält für jedes Intervall einen Eintrag, dessen Struktur am folgenden Beispiel erläutert wird.

Zunächst erhält der Administrator Auskunft darüber, auf welchen Zeitraum sich die folgende Statistik bezieht:

```
-------------------------------------------------------
Interval Start:              Fri, 18.08.1995, 06:00:00
Interval End  :              Fri, 18.08.1995, 14:00:00
Duration      :              08:00:00 h/mm/ss
```

Informationen über Kanäle:

Es folgt eine Auflistung aller Kanäle, die während des abgelaufenen Intervalls konfiguriert waren und möglicherweise wieder dekonfiguriert worden sind.

```
Connections:

  LINK  VPI  VCI  STATUS   DIR  CONFIG-TIME               DE-CONFIG-TIME             QoS...
  ----  ---  ---  -------  ---  ------------------------- -------------------------- ---...
     9    0   40  invalid  out  Fri, 18.08.1995, 12:57:35 Fri, 18.08.1995, 13:57:35  VBR...
     9    0   40    valid  out  Fri, 18.08.1995, 13:58:24                            VBR...
     9    0   43  invalid   in  Fri, 18.08.1995, 12:57:35 Fri, 18.08.1995, 13:57:35  CBR...
     9    0   43    valid   in  Fri, 18.08.1995, 13:58:24                            CBR...
     :    :    :      :     :             :             :         :          :        :
     :    :    :      :     :             :             :         :          :        :
```

Für jeden Kanal werden folgende Informationen ausgegeben:

- Linknummer, VPI und VCI
- Status: valid (konfiguriert) oder invalid (nicht mehr konfiguriert)
- Zeitpunkt der Konfiguration und (möglicherweise) Dekonfiguration
- Quality of Service (CBR, VBR...), falls in den %SWITCH%_VC-Dateien eingetragen

An diese Daten schließen sich rechts weitere an:

```
LINK  VPI  VCI  ......    MEAN     PEAK
----  ---  ---  ......    -------  -------
   9    0   40  ......        100     2000
   9    0   40  ......        200     4000
   9    0   43  ......        300     2000
   9    0   43  ......        200     3000
```

Hier werden für jeden Kanal die reservierten Mean- und Peakraten ausgegeben, sofern diese in den %SWITCH%_VC-Dateien verfügbar sind. Wurden - wie in diesem Beispiel - Kanäle mehrmals neu konfiguriert, werden für diese zusätzlich die minimalen, maximalen und durchschnittlichen Raten berechnet.

Informationen pro Link:

Nachdem zuvor alle Kanäle betrachtet worden sind, folgen nun Informationen über die einzelnen Links. Zunächst werden Daten über die an dem entsprechenden Link ein und ausgehenden Kanäle zur Verfügung gestellt:

```
Link: 9
===========================

Configured Connections:
                 2 min
                 2 max
                 2 average

Configuration Duration:
              0:01:36 (h:mm:ss) min
              1:00:00 (h:mm:ss) max
              0:30:48 (h:mm:ss) average

Mean Rates:
                            Sums
            +-----------+-----------+
        min |         0 |       500 |
        max |      1000 |      5000 |
    average |       333 |       700 |
            +-----------+-----------+

Peak Rates:
                            Sums
            +-----------+-----------+
        min |         0 |      1000 |
        max |       800 |     10000 |
    average |       200 |      5000 |
            +-----------+-----------+
```

Im einzelnen sind dies die Extrem- und Durchschnittswerte...

- deren Anzahl
  (sollte die dafür benötigte %SWITCH%_%LINK%_VC_Count-Datei nicht existieren, wird die Anzahl der an diesem Link ein- und ausgehenden Kanäle ausgegeben)
- deren Konfigurationsdauer
- deren reservierter Mean- und Peakraten sowie deren Summen (sofern verfügbar)

Weiterhin erhält der Administrator Auskunft über die aktiven Verbindungen an diesem Link:

```
Active Connections:
                 0 min
                 4 max
                 2 average

  Duration of active phases:
             1.245 secs min
             4.219 secs max
             2.421 secs average

  Cells:
            117392 transmitted
             28221 received
       ===========
            145613 total

  Bandwith:
               103 c/s min
              1282 c/s max
              4431 c/s average
```

Dabei werden folgende Fragen beantwortet:

- Wieviele aktive Phasen sind an diesem Link minimal, maximal und durchschnittlich aufgetreten ?
  (Um diese Informationen zu erhalten, wird die jeweilige %SWITCH%_%LINK%_Count-Datei ausgewertet. Sollte diese nicht vorhanden sein, weil z.B. der entsprechende Logging-Prozess nicht lief, wird ausgegeben, wieviele aktive Phasen insgesamt aufgetreten sind.)
- Wie lange dauerten die aktiven Phasen minimal, maximal und durchschnittlich ?
- Wieviele Zellen wurden gesendet und empfangen ?
  (Sollten Zellen übertragen worden sein, deren Richtung nicht bestimmt werden konnte, wird deren Anzahl ebenfalls ausgegeben.)
- Welche Bandbreite wurde minimal, maximal und durchschnittlich erreicht ?

Abschließend werden, falls an diesem Link Fehler aufgetreten sind, die mitprotokollierten Fehlermeldungen aufgeführt.

5.2.5. Realisierung

Im folgenden wird der Ablauf des Programms `lrz_atm_logswitchstat`, das Statistiken für Switches erzeugt, erläutert. Zunächst eine Übersicht:

```
1 | für alle Switches oder für einen bestimmten Switch im Netz:
  |
2 |      ermittle Informationen über alle an diesem Switch (möglicherweise nicht mehr)
  |      konfigurierten Kanäle
  |
3 |      für alle Links dieses Switches ermittle:
  |
4 |            - minimale, maximale und durchschnittliche reservierte Mean- und Peakraten
  |              sowie Minimum, Maximum und Durchschnitt aus deren Summen
  |            - minimale, maximale und durchschnittliche Anzahl konfigurierter Kanäle
  |
5 |            - minimale, maximale und durchschnittliche Konfigurationsdauer
  |              von Kanälen
  |
6 |            - minimale, maximale und durchschnittliche Anzahl aufgetretener
  |              aktiver Phasen
  |
7 |            - Informationen über Last auf diesem Link
  |
8 |            - aufgetretene Fehler
  |
  |        end
  | end
  |
9 | Intervall einhalten oder Programm beenden
```

1.) Es wird geprüft, ob dem Programm ein Selection Name eines Switches als Parameter übergeben worden ist. In diesem Fall wird nur für diesen durch Aufruf von `lrz_atm_AddSwitchStatInFile` eine temporäre Statistik erstellt. Andernfalls werden die folgenden Operationen in einer Schleife über alle Switches ausgeführt. Dazu werden in der Komponentendatei die Einträge gesucht, in denen dem Schlüssel *`isSwitch`* der Wert TRUE zugeordnet ist.

2.) Die Funktion `lrz_atm_AddSwitchStatInFile` erzeugt bzw. öffnet eine bereits bestehende Statistik-Datei. Der Name dieser Datei entspricht dem Selection Name des Switches. Sie erhält das Suffix „.tmp“, falls beim Programmaufruf ein Selection Name übergeben worden ist.

Es werden nun durch Aufruf von `lrz_atm_PrintConfChannels` Informationen über Kanäle an diesem Switch gesammelt. Dabei werden auch die Kanäle berücksichtigt, die nicht mehr konfiguriert sind. Nicht in die Statistik fließen solche Kanäle ein, die über den Kontrollport gehen. Die Linknummer dieses Ports erhält man durch den Aufruf von `lrz_atm_GetControlPortNr`.

Aus der jeweiligen %SWITCH%_VC-Datei werden für jeden ein- und ausgehenden Kanal folgende Daten gelesen und in einer verketteten, sortierten Liste gespeichert:

- Linknummer, VPI, VCI
- Konfigurationszeitraum (bzw. Zeitpunkt der Konfiguration, falls noch konfiguriert)
- QoS: VBR, CBR, ... (falls verfügbar)
- reservierte Mean- und Peakrate (falls verfügbar)

Anschließend werden die Inhalte aus den Elementen der verketteten Liste in die Statistikdatei geschrieben. Sollte ein Kanal innerhalb des vergangenen Intervalls mehrfach neu konfiguriert worden sein (existieren also in der %SWITCH_VC-Datei mehrere Einträge für diesen Kanal), stellt das Programm zusätzlich Minimum, Maximum und Durchschnittswert der reservierten Mean- und Peakraten in der Ausgabedatei zur Verfügung.

3.) Ab hier werden alle Links dieses Switches betrachtet. Welche dies sind, kann aus der Verbindungsdatei gelesen werden. Hier sind für jeden Quell-Node (in diesem Fall ist dies der jeweilige Switch) die ausgehenden Links verzeichnet. Die folgenden Informationen werden innerhalb einer Schleife über diese Links ermittelt und in einer Struktur gespeichert. Der Inhalt dieser Struktur wird nach Beendigung der Schleife in die Ausgabedatei übernommen. Eine Ausnahme bilden eventuell aufgetretene Fehler. Die entsprechenden Meldungen werden direkt in die Statistik-Datei geschrieben.

4.) Die Funktion `lrz_atm_GetConfigChannels` versucht, folgende Daten für die Statistik zur Verfügung zu stellen, indem sie sie in der bereits erwähnten Struktur speichert:

- minimale, maximale und durchschnittliche für Kanäle reservierte Mean- und Peakraten sowie Minimum, Maximum und Durchschnitt aus deren Summen
- minimale, maximale und durchschnittliche Anzahl konfigurierter Kanäle

Die Durchschnitts- und Extremwerte der reservierten Mean- und Peakraten können, sofern diese in den entsprechenden MIBs zur Verfügung standen, aus den Einträgen der jeweiligen %SWITCH%_VC-Datei ermittelt werden, die den Kanälen eines bestimmten Links zugeordnet sind.
Die Summen dieser Raten sowie die Anzahl der jeweils konfigurierten Verbindungen wurden regelmäßig - mit einem Timestamp versehen - in den %SWITCH%_%LINK%_VC_Count-Dateien gespeichert, aus denen Minimum, Maximum und Durchschitt errechnet werden. Sollten diese Dateien nicht vorhanden sein, wird in der Statistik anstelle der Anzahl konfigurierter Verbindungen die Anzahl der an diesem Link ein- und ausgehenden Kanäle ausgegeben.

Bei den %SWITCH%_%LINK%_VC_Count-Dateien ist darauf zu achten, daß bereits ausgewertete Daten nicht ein weiteres Mal berücksichtigt werden. Aus diesem Grund wird für jede %SWITCH%_%LINK%_VC_Count-Datei eine Positionsdatei angelegt, die die Position enthält, bis zu der die Datei bereits ausgewertet worden ist bzw. ab der beim nächsten Durchgang gelesen werden soll. Sollen bereits in die Statistik übernommene Daten gelöscht werden (wurde also die Anzahl von Intervallen erreicht, die der Adminstrator durch den Parameter *DelFreqSwitchStat* vorgegeben hat), wird die ausgewertete Datei auf die Länge 0 gekürzt (und nicht gelöscht, damit sie vorerst noch gelockt bleibt) und deren Positionsdatei gelöscht.

5.) Für die Ermittlung der minimalen, maximalen und durchschnittlichen Konfigurationsdauer von Kanälen ist die Funktion `lrz_atm_GetConfChannelDuration` zuständig.

Die %SWITCH%_%VC%-Dateien enthalten für jeden Kanal einen Eintrag, der Zeitpunkte der Konfiguration und Dekonfiguration enthält. Die gesuchten Werte (Minimum, Maximum und Durchschnitt) erhält man durch Betrachten der Einträge für die Kanäle, die an dem jeweiligem Port konfiguriert waren bzw. sind. Sie werden ebenfalls in der o.g. Struktur abgelegt. Für noch konfigurierte Kanäle wird die Konfigurationsdauer aus der Differenz der aktuellen Zeit und dem Zeitpunkt der Konfiguration gebildet.

6.) In den %SWITCH%_%LINK%_Count_Dateien wurde die Anzahl gerade aktiver Phasen bei Beginn und Ende aktiver Phasen erfaßt. Die Funktion `lrz_atm_GetActiveChannels` greift auf diese - sofern vorhanden - zurück, um daraus die Extremwerte (Minimum und Maximum) und den Durchschnitt zu berechnen. Dazu müssen lediglich die einzelnen Werte zeilenweise gelesen werden.
Es müssen auch hier entsprechende Positionsdateien verwendet werden.

7.) Nun werden für den jeweiligen Link folgende Daten ermittelt:

- Anzahl gesendeter und empfangener Zellen
- minimale, maximale und durchschnittliche Bandbreite
- minimale, maximale und durchschnittliche Dauer aktiver Verbindungen.
- Anzahl der aktiven Phasen

Diese Information können aus den Netzverkehrsdateien gewonnen werden. In diesen wurde die Last zeilenweise oder als Profil mitprotokolliert. Für jeden Link können bis zu drei solcher Dateien existieren, und zwar für:

- gesendete Zellen
- empfangene Zellen
- Zellen, für die nicht bestimmt werden konnte, ob sie gesendet oder empfangen wurden

Um die gewünschten Informationen berechnen zu können, müssen diese Dateien zeilenweise gelesen werden. Eine Zeile wird dabei durch Aufruf der Funktion `lrz_atm_LogLineToStat` ausgewertet.
Damit dieselben Daten nicht mehrmals in die Statistik übernommen werden, müssen

auch hier Positionsdateien angelegt werden.

8.) Die Prozedur `lrz_atm_ErrorsToStat` prüft, ob an dem jeweiligen Link Fehler aufgetreten sind. In diesem Fall existiert eine Fehler-Datei, die entsprechende Meldungen enthält. Diese Meldungen werden zeilenweise in die Statistik-Datei übernommen. Auch hier muß auf Positionsdateien zurückgegriffen werden.

9.) Wurde das Programm mit einem Selection Name aufgerufen, damit es nur eine Statistik für einen bestimmten Switch erstellt, wird dieses nun beendet. Andernfalls wird durch Aufruf der Funktion `lrz_atm_WaitTime` das vom Administrator gewählte Intervall eingehalten, bevor neue Statistiken erstellt werden.

5.3. Netzweite Informationen

5.3.1. Bedienung

Erzeugung netzweiter Informationen:

Die intervallweise Erzeugung netzweiter Informationen bzw. einer entsprechenden Datei (das Intervall wird durch den Administrator bestimmt) beginnt mit dem Aufruf des Programms `lrz_atm_lognetstat`. Es ist für einen späteren Zeitpunkt vorgesehen, daß dies automatisch mit dem Start von OpenView geschieht.
Um innerhalb eines Intervalls einen aktuellen Stand zu erhalten, gibt es zwei Möglichkeiten:

- Menüauswahl: Fopra→Calculate current Net Statistics
- Shell: „lrz_atm_lognetstat xxx“ (wobei xxx ein beliebiges Argument ist, das eine einmalige Programmausführung veranlaßt; wird kein Parameter übergeben, werden netzweite Informationen in einem definierbaren Intervall erstellt)

Anzeigen netzweiter Informationen:

Zum Anzeigen bereits erzeugter netzweiter Informationen gibt es zwei Alternativen:

- Menüauswahl: Fopra→Show Net Statistics
- Shell: Anzeigen (z.B. per „more“) der Datei „.../statistik/netstat“

Definieren der Parameter:

Der Administrator kann Parameter setzen, die die Erstellung netzweiter Informationen beeinflussen. Diese Parameter, die im einzelnen noch vorgestellt werden, befinden sich in einer Datei, die auf zwei Arten editiert werden kann:

- Menüauswahl: Fopra→Edit Statistic Parameters
- Shell: Editieren (z.B. per „vi“) der Datei „.../input/statistik_info“

5.3.2. Eingabe-Dateien

Als Eingabedateien werden benötigt:

- Konfigurationsdateien
- Netzverkehrsdateien
- Parameterdatei

Das Internet-Management bietet keine komponentenübergreifende Information. Es wäre also nicht sinnvoll, für die Gewinnung netzweiter Informationen MIB-Variablen auszulesen. Aus diesem Grund wurden die folgenden Konfigurationsdateien erzeugt, die hier ausgewertet werden, um einst konfigurierte Ende-zu-Ende-Verbindungen rekonstruieren zu können:

- Komponentendatei
  (= Verzeichnis aller im Netz befindlichen Nodes)
- Verbindungsdatei
  (= Verzeichnis aller physikalischen Verbindungen)
- %SWITCH%_VC-Dateien
  (enthalten Routingtabelle und VC-Konfiguration eines Switches)

Damit festgestellt werden kann, welche Last auf den Kanälen einer Verbindung aufgetreten ist, werden außerdem pro benutztem Kanal die Netzverkehrsdateien %SWITCH%_%LINK%_%VPI%_%VCI%_Cells, in denen die Anzahl übertragener Zellen mitprotokolliert worden ist, benötigt.

Die Erzeugung netzweiter Informationen wird - wie bereits erwähnt - von Parametern beeinflußt, die sich in der Datei „.../input/statistik_info“ befinden:

- `LogIntNetStat`:
  Dieser Parameter legt fest, in welchem Intervall (in Sekunden) netzweite Informationen erstellt werden sollen. Hier ist zu berücksichtigen, daß die Erstellung der Ausgabedatei umso mehr Zeit in Anspruch nimmt, je größer das Netz ist.

- `DelFreqNetStat`:
  Hiermit kann der Administrator bestimmen, nach jedem wievielten Intervall nicht mehr konfigurierte VCs gelöscht werden. Diese werden auch dann gelöscht, wenn netzweite Informationen „von Hand“ (also nicht im Intervall) erstellt werden.

- `MaximumHopDiff`:
  Findet beim Auftreten einer aktiven Phase ein Datenfluß über mehrere nacheinandergeschaltete Kanäle statt, ist es möglich, daß für diese leicht variierende Zeitpunkte für den Beginn der aktiven Phase protokolliert werden. Um eine aktive Phase nachträglich durch das Netz verfolgen zu können, müssen diese Differenzen berücksichtigt werden. Durch Setzen dieses Parameters kann der Administrator wählen, wie groß Verzögerungen zwischen zwei hintereinandergeschalteten Kanälen höchstens sein dürfen. Die Einheit

Sekunden orientiert sich an der Genauigkeit, die der Network Node Manager verwendet.

Hier der entsprechende Ausschnitt aus dieser Eingabedatei:

```
:            :
:            :
# Netzweite Informationen
# -----------------------
LogIntNetStat:          15      #    Intervalllänge in Minuten
DelFreqNetStat:         3       #    nach x Intervallen dekonfigu
                                     rierte VCs loeschen
MaximumHopDiff:         1       #    max. Verzögerung von aktiven
                                     Phasen pro Hop in Sekunden
```

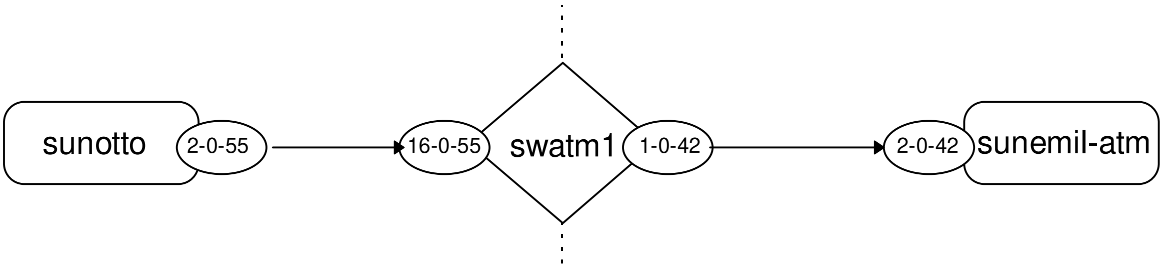
5.3.3. Ausgabedatei

Die Ausgabedatei „.../statistik/netstat“ enthält für jedes Intervall einen Eintrag, dessen Struktur am folgenden Beispiel erläutert wird:

```
==================================================================================
Interval Start:              Fri, 21.07.1995, 14:19:35
Interval End  :              Fri, 21.07.1995, 14:39:35
Duration      :              0:20:00 h/mm/ss
==================================================================================
----------------+----------------+----------------+----------------+----------------+
                |        sunotto |         swatm1 |         swatm1 |    sunemil-atm |
----------------+----------------+----------------+----------------+----------------+
   Link/VPI/VCI |    2    0   55 |   16    0   55 |    1    0   42 |    2    0   42 |
----------------+----------------+----------------+----------------+----------------+
     configured |                |Fri, 21.07.1995 |Fri, 21.07.1995 |                |
                |                |       04:57:25 |       04:57:25 |                |
----------------+----------------+----------------+----------------+----------------+
   deconfigured |                |Fri, 21.07.1995 |Fri, 21.07.1995 |                |
                |                |       21:02:12 |       21:02:13 |                |
----------------+----------------+----------------+----------------+----------------+
           Mean |                |          10000 |          10000 |                |
           Peak |                |          50000 |          50000 |                |
            Qos |                |            VBR |            VBR |                |
----------------+----------------+----------------+----------------+----------------+
     start time |                |Fri, 21.07.1995 |Fri, 21.07.1995 |                |
                |                |       14:25:07 |       14:25:08 |                |
duration (secs) |                |         68.505 |         68.515 |                |
          cells |                |        2175544 |        2175544 |                |
----------------+----------------+----------------+----------------+----------------+
     start time |                |Fri, 21.07.1995 |Fri, 21.07.1995 |                |
                |                |       14:35:03 |       14:35:03 |                |
duration (secs) |                |         66.331 |         66.341 |                |
          cells |                |        2204429 |        2204429 |                |
----------------+----------------+----------------+----------------+----------------+
 DURATION TOTAL |                |        134.836 |        134.846 |                |
    CELLS TOTAL |                |        4379973 |        4379973 |                |
----------------+----------------+----------------+----------------+----------------+
```

Der Kopf eines Eintrags gibt Auskunft darüber, auf welches Intervall sich die folgenden Informationen beziehen. In diesem Fall begann dieses am 21.7.95 und dauerte 20 Minuten. Im Anschluß findet der Administrator je eine Matrix für jede zustandegekommene Verbindung. Hier ist nur eine Verbindung aufgeführt.

Diese verlief vom Host sunotto über den Switch swatm1 zum Host sunemil-atm:



Die aufgeführten Namen entsprechen den in Open View verwendeten Selection Names. Switches schalten virtuelle Verbindungen von einem eingehenden auf einen ausgehenden Kanal. Aus diesem Grund existieren für diese zwei Spalten. Welche Links, Pfade und Kanäle im einzelnen für eine Verbindung gewählt worden sind, ist der Folgezeile zu entnehmen.

Informationen über die Konfiguration dieser Kanäle erhält der Administrator aus dem nächsten Abschnitt. Hier ist aufgeführt, seit wann bzw. in welchem Zeitraum diese gültig waren sowie, ob bzw. welche Mean- und Peakraten für sie reserviert sind. Die für den jeweiligen Kanal zutreffende Dienstgüte (CBR, VBR ...) wird ebenfalls angegeben, sofern diese in den entsprechenden %SWITCH%_VC-Dateien verfügbar waren.

Nachfolgend sind alle aktiven Phasen aufgelistet, die während des Intervalls aufgetreten sind. In diesem Beispiel handelt es sich um zwei aktive Verbindungen. Für diese werden pro Kanal folgende Daten ausgegeben:

- Beginn der Phase (Datum und Uhrzeit)
- Dauer der Phase in Sekunden
- Anzahl der übertragenen Zellen

Diese Angaben müssen für die einzelnen Kanäle einer Verbindung natürlich nicht übereinstimmen. Differenzen bei der Anzahl übertragener Zellen weisen auf verlorengegangene Zellen hin.

Am Ende des Eintrags erfährt der Administrator, wieviele Zellen innerhalb des Intervalls insgesamt gesendet worden sind und wie lange deren Übertragung gedauert hat.

5.3.4. Realisierung

Im folgenden wird der Ablauf des Programms `lrz_atm_lognetstat`, das netzweite Informationen erzeugt, erläutert. Zunächst eine Übersicht:

```
1 | für alle Hosts im Netz:
2 |     für alle Links dieses Hosts:
3 |         für alle Kanäle dieses Links:
  |
4 |             suche eine aktive Phase, die über diesen Kanal während dessen Konfigu-
  |             rationszeitraums geschaltet worden ist
  |
  |             if (aktive Phase gefunden)
  |
5 |                 ermittle anhand der aktiven Phase den Verlauf dieser Verbindung
  |
6 |                 ermittle alle aktiven Phasen, die über die gleiche Ende-zu-Ende-
  |                 Verbindung verlaufen sind und speichere Informationen über diese
  |                 in der Ausgabedatei
  |
  |             end
  |
  |         end
  |     end
  | end
  |
7 | if (ausgewertete Daten loeschen) then nicht mehr konfigurierte VCs löschen
  |
8 | Intervall einhalten oder Programm beenden
```

1.) Das Programm verfügt über drei ineinander verschachtelte Schleifen. In der ersten werden durch Aufruf der Funktion `lrz_atm_GetNextHost` aus der Komponentendatei alle Hosts ermittelt. Dies geschieht durch Suche nach den Einträgen, in denen dem Schlüssel *`isSwitch`* der Wert FALSE zugeordnet ist.

2.) Die zweite Schleife ist dafür verantwortlich, alle von einem gefundenen Host ausgehenden Links und den jeweils darüber erreichbaren Switch zu suchen. Diese sind in der Verbindungsdatei aufgeführt und können mit der Prozedur `lrz_atm_GetNextLink` gesucht werden.

3.) Um alle Kanäle zu finden, die über den jeweiligen Link laufen bzw. gelaufen sind, ist die dritte Schleife notwendig. Die Funktion `lrz_atm_GetNextChannel` ermittelt diese, indem sie die %SWITCH%_VC-Datei (enthält u.a. Routingtabelle eines Switches) auswertet, die für den unter 2. gefundenen Switch angelegt worden ist.

4.) Aufgabe der Prozedur `lrz_atm_BuildVCList` ist es zunächst, zu überprüfen, ob an dem jeweiligen Kanal während dessen Konfigurationszeitraumes eine aktive Phase aufgetreten ist. Aktive Verbindungen werden in den Netzverkehrsdateien für jeden Kanal mitprotokolliert.

5.) Ist dies der Fall, wird versucht, die entsprechende Ende-zu-Ende-Verbindung zu rekonstruieren, die möglicherweise nicht mehr konfiguriert ist. Durch abwechselnde Aufrufe der Funktionen `lrz_atm_GetNextOutgoingVC` und `lrz_atm_GetNext-IngoingVC` wird solange eine verkettete Liste der benutzten Kanäle zusammengestellt bis ein Host erreicht ist. Um an einem Switch den ausgehenden Kanal zu ermitteln, muß wiederum die entsprechende %SWITCH%_VC-Datei ausgewertet werden. Für die Suche nach einem eingehenden Kanal ist zusätzlich die Verbindungsdatei zu untersuchen, um den Selection Name und den Link des Folge-Nodes zu erhalten.

   Jeder Kanal muß daraufhin überprüft werden, ob er zum Zeitpunkt der aktiven Phase überhaupt konfiguriert war (`lrz_atm_WasChannelValidDuringPhase`) und ob er für den Weg der aktiven Verbindung gewählt worden ist (`lrz_atm_FindLogData` sucht nach einem Eintrag oder Profil für die aktive Phase in der Netzverkehrsdatei des Kanals). Die Zeitpunkte, zu denen die Phasen auf den nacheinandergeschalteten Kanälen aktiv wurden, dürfen nicht zu stark differieren. Diese Problematik wird im Kapitel 5.3.5. näher erläutert.

   Konnte die Verbindung bis zu einem Ziel-Host rekonstruiert werden, erfolgt die Speicherung deren Verlaufs in der Ausgabedatei.

6.) Die Prozedur `lrz_atm_NextActiveConnToStat` sorgt bei wiederholtem Aufruf dafür, daß alle aktiven Phasen auf der betrachteten virtuellen Verbindung erfaßt und in der Ausgabedatei aufgeführt werden. Sie erhält als Eingabe die verkettete Liste, in der alle Kanäle verzeichnet sind, über die die Ende-zu-Ende-Verbindung geschaltet worden ist.

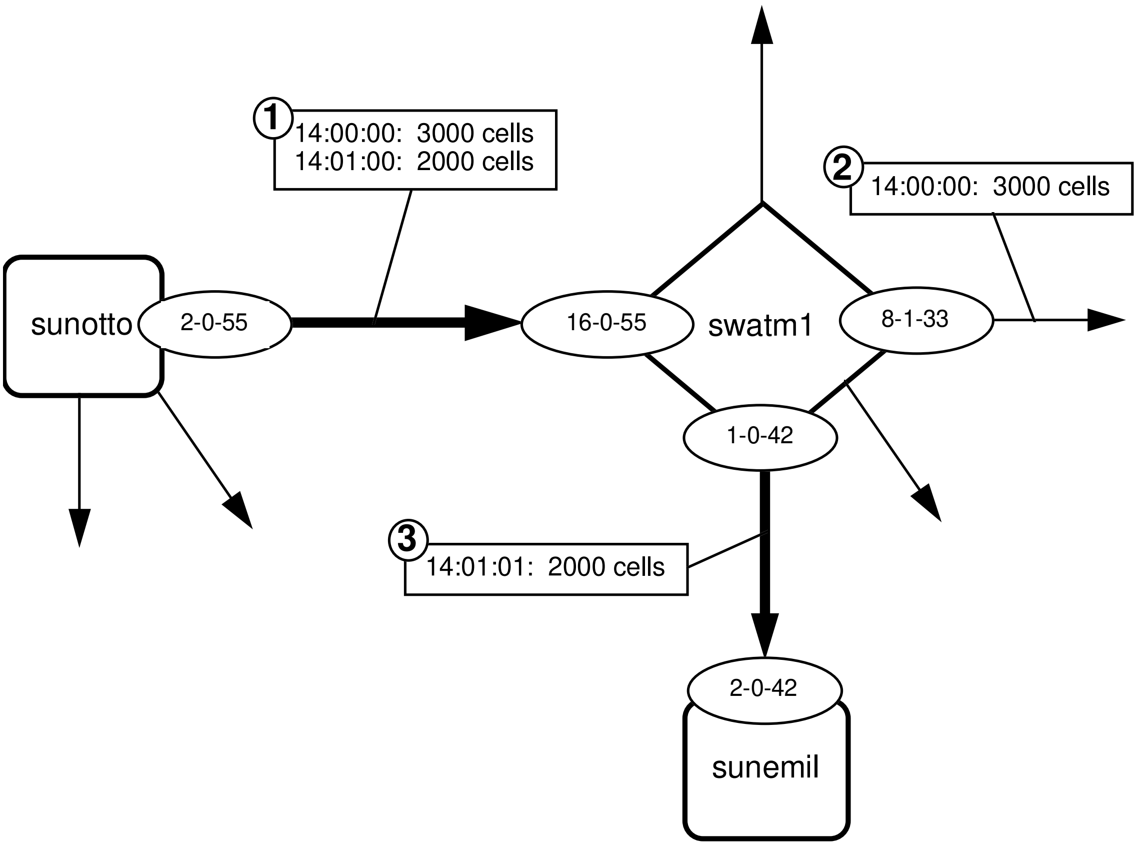
   Sie verfolgt jede aktive Phase durch Suche der entsprechenden Einträge in den Netzverkehrsdateien jedes Kanals. Die Funktion `lrz_atm_FindLogData` lokalisiert diese Einträge, die entweder aus einer Zeile oder einem mehrzeiligen Profil bestehen und wertet diese durch Aufruf von `lrz_atm_LogDataToStat` aus.
   Nach jedem Aufruf von lrz_atm_NextActiveConnToStat werden die ausgewerteten Einträge in den Netzverkehrsdatein gelöscht.

7.) Dekonfigurierte VCs sind zur Rekonstruktion virtueller Verbindungen nötig. Danach können diese jedoch durch das Programm `lrz_atm_lognetstat` aus den %Switch%_VC-Dateien entfernt werden. Der Netzadministrator kann durch Wahl eines Parameters (*`DelFreqNetStat`*) entscheiden, nach wievielen Intervallen dies geschehen soll. Werden netzweite Informationen „von Hand“ erstellt (also durch Aufruf von `lrz_atm_lognetstat` mit einem beliebigen Parameter) werden nicht mehr konfigurierte VCs ebenfalls gelöscht.

8.) Wurde das Programm mit einem beliebigen Parameter aufgerufen, wird dieses beendet. Andernfalls wird durch Aufruf der Funktion `lrz_atm_WaitTime` das vom Administrator gewählte Intervall eingehalten, bevor neue netzweite Informationen erstellt werden.

5.3.5. Problemkreise

Bei der Erstellung netzweiter Informationen sind Probleme bzw. Situationen zu berücksichtigen, die anhand der folgenden Zeichnung erläuert werden. Durch die stärker gezeichneten Pfeile ist die aktuelle Konfiguration zu erkennen (sunotto - swatm1 - sunemil). Die Rechtecke symbolisieren Ausschnitte aus den Netzverkehrsdateien der einzelnen Kanäle. In den Ellipsen befinden sich Linknummer, VPI und VCI der am jeweiligen Node aus- und eingehenden Verbindungen.

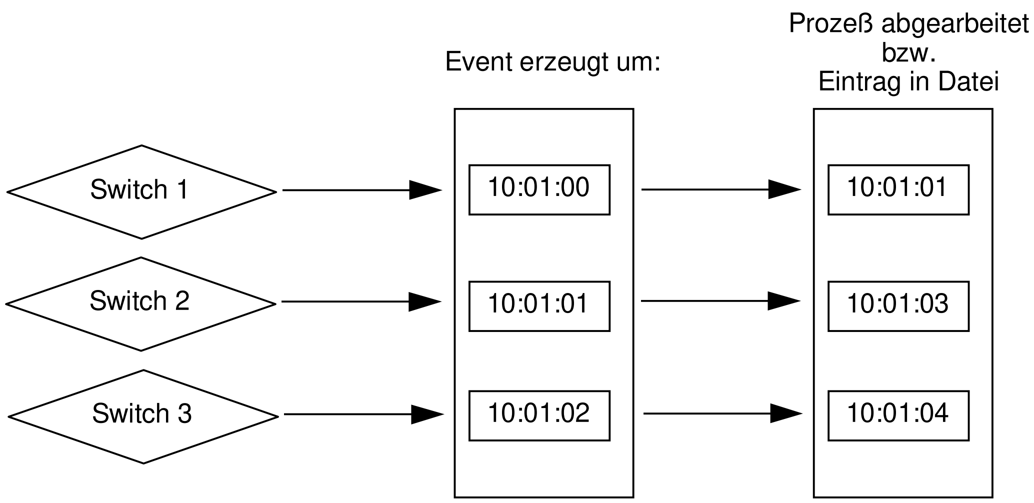


Würde man bei der Erstellung netzweiter Informationen nur die konfigurierten Verbindungen betrachten, könnten einige aktive Phasen nicht oder nur unvollständig rekonstruiert werden. Sie wären damit auch nicht in der Ausgabedatei aufgeführt.
Ein Beispiel dafür ist die aktive Phase, die um 14:00:00 Uhr von sunotto kommend am Switch swatm1 einging (siehe Netzverkehrsdatei 1). Man versucht vergeblich, diese über die aktuell konfigurierte virtuelle Verbindung nach sunemil weiterzuverfolgen. In der Netz-

verkehrsdatei 3 ist kein entsprechender Eintrag zu finden. Untersucht man aber den - mittlerweile nicht mehr konfigurierten - Kanal, der den Switch über 8-1-33 verlassen hat, kann die aktive Phase rekonstruiert werden.

Ein weiteres Problem ergibt sich dadurch, daß aktive Verbindungen an den Switches verzögert werden können bzw. mit unterschiedlichen Timestamps in den Netzverkehrsdateien hintereinandergeschalteter Kanäle protokolliert werden. Als Beispiel wurde in der Grafik in der Verkehrsdatei 1 die Übertragung von 2000 Zellen um 14:01:00 Uhr eingetragen. In der dritten Datei erhielt der entsprechende Eintrag den Timestamp 14:01:01 Uhr. Diese Tatsache ist auf verschiedene Ursachen zurückzuführen. In Versuchen hat sich herausgestellt, daß der Network Node Manager durch Schwellwertübertretung ausgelöste Events teilweise verzögert. Durch jeden Event wird wiederum ein Prozeß gestartet, der die entsprechenden Einträge in den Netzverkehrsdateien vornimmt. Hier kommt die Problematik hinzu, daß die Abarbeitung der einzelnen Prozesse unterschiedlich lange dauern kann.

Ein Beispiel für diese Situation bietet die folgende Abbildung:



Hier wird angenommen, daß eine aktive Verbindung über drei hintereinandergeschaltete Switches verläuft. Die Zeitpunkte, zu denen die Events ausgelöst werden, unterscheiden sich um jeweils eine Sekunde. Bis die entsprechenden Einträge in den Netzverkehrsdateien vorgenommen worden sind, vergehen weitere ein bzw. zwei Sekunden. Um diese aktive Phase nachträglich erfassen zu können, müssen also Differenzen zwischen den Timestamps von bis zu zwei Sekunden berücksichtigt werden. Der Netzadministrator hat die Möglichkeit, die maximal erlaubte Differenz durch Setzen des Parameters *`MaximumHopDiff`* zu wählen.

5.4. Trace

Zum dritten Teil des Fopras gehört neben den oben genannten Funktionalitäten ein Trace-Programm, mit dem es möglich ist, einen Channel ab einem beliebigen Punkt im Netz (Host oder Switch) bis zum Ziel zu verfolgen und dabei Switchstatistiken oder aktuelle MIB-Variablen zu den erreichten Switches auszugeben. Der Trace kann von der Shell, durch Doppelklick auf die Startkomponente oder über Menü gestartet werden.

5.4.1. Bedienung

- Shell:
  lrz_atm_trace hostname, wobei hostname für den Switch oder Host steht, bei dem der Trace begonnen werden soll.

- Doppelklick:
  Innerhalb der ATM-Submap sind alle Symbole von Hosts und Switchs „executable“, d.h. durch Doppelklick wird hier das Traceprogramm gestartet mit der angeklickten Komponente als Startpunkt.

- Menü:
  Nach Anklicken der Startkomponente Auswahl des Menüpunktes ‘LRZ-ATM→Trace’

  Bei Switches wird der Anwender nun gefragt, ob er einen Trace ausführen, oder nur Statistiken zum angewählten Switch anschauen möchte.
  Beim Trace werden zuerst alle auf der gewählten Komponente konfigurierten outgoing Channels mit Link, VPI und VCI aufgelistet und der Anwender wählt denjenigen aus, den er verfolgen möchte. Anschließend erscheint folgendes Auswahlmenu:

```
Route ........................ (1)
Route + History .............. (2)
Route + aktuelle MIB-Werte .... (3)
Route + History + MIB-Werte ... (4)
```

Welche Informationen auf dem Trace angezeigt werden sollen gibt der Anwender mit der entsprechenden Zahl ein. Hierbei bedeutet :

- Route:
  Nur der Name der verfogten Verbindung (link, VPI, VCI) und der Name der erreichten Komponenten wird ausgegeben
- History:
  vorhandene Statistiken werden ausgegeben
- aktuelle MIB-Werte:
  bei jedem erreichten Switch wird in der Datei „input/tracemibs“ unter dem entsprechenden Vendorindex nachgesehen, ob und welche MIB-Variablen zu diesem Switch ausgegeben werden sollen. Die Datei ist nach Wunsch vom Anwender zu editieren.

Die Bedienung des Trace ist noch recht unkomfortabel, wird aber wohl baldmöglichst auf den Interfacearchitekt umgestellt.

5.4.2. Realisierung

Die wichtigsten Funktionen sind die folgenden:

- `lrz_atm_ShowSwitchChannels`:
  gibt für einen Switch die konfigurierten Channels aus, in dem es die entsprechende %SWITCH%_VC-Datei ausliest

- `lrz_atm_ShowHostChannels`:
  gibt für einen Host die konfigurierten Channels aus, in dem es entsprechende MIB-Variablen ausliest

- `lrz_atm_TraceVC`:
  der Kern des Trace. Einen übergebenem Channel (nodename, link, VPI, VCI) traced er so longe, bis ein Host erreicht wurde, oder Fehler beim Verfolgen auftreten. Handelt es sich beim erreichten Hop um einen Switch, versucht diese Funktion zuerst aus der vorhandenen %SWITCH%_VC den zum übergebenen ingoing-Channel gehörenden outgoing zu finden (`lrz_atm_GetOutChannel`). Ist dies nicht möglich, wird diese Datei aktualisiert (`lrz_atm_MakeSwitchVCActual`) und `lrz_atm_GetOutChannel` noch einmal probiert. Wurde der outgoing-Channel gefunden, werden mit `lrz_atm_PrintOnTheFly` die gewünschten Informationen ausgegeben, wenn nicht gibt es auf diesem Switch keine Fortsetzung des verfolgten Channels und es wird eine entsprechende Fehlermeldung ausgegeben.

- `lrz_atm_PrintOnTheFly`:
  gibt aktuelle Routing-Informationen aus und ruft entsprechend dem gewünschten tracemode noch die Funktionen `lrz_atm_PrintLinkStats` und `lrz_atm_Print_TraceMIBValues` auf

- `lrz_atm_Print_TraceMIBValues`:
  liest die Datei ‘input/tracemibs’ aus, in der der Anwender festlegt, welche MIB-Variablen zu einem Typ Switch (über Vendorindex festgelegt) ausgegeben werden sollen. Die Datei hat z.B. folgendes Aussehen:

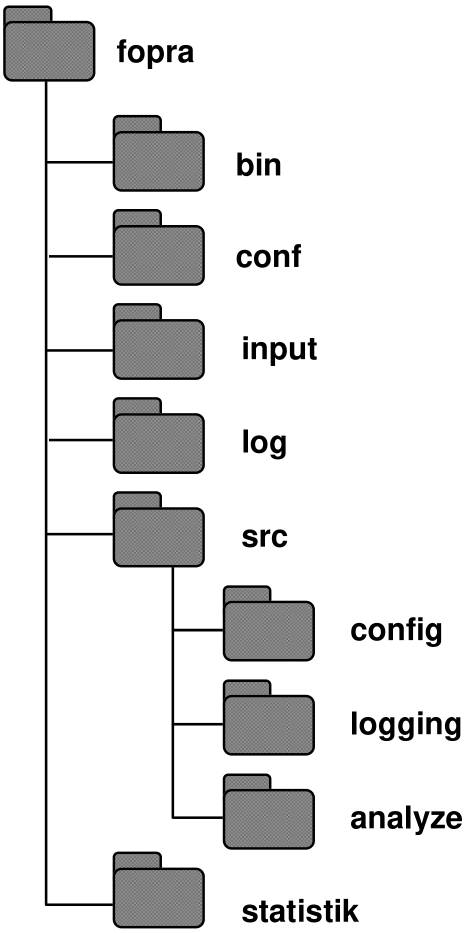
```
/**********************************************/
/* Hier werden die beim Trace auszugebenden     */
/* MIB-Variablen vendorabhaengig definiert      */
/**********************************************/
/* Instanz-Typen                                */
/* LRZ_ATM_INSTANCE_ADAPTER        1            */
/* LRZ_ATM_INSTANCE_LINK           2            */
/* LRZ_ATM_INSTANCE_PATH           3            */
/* LRZ_ATM_INSTANCE_CHANNEL        4            */
/* LRZ_ATM_INSTANCE_CHANNELROUTE   5            */
/* LRZ_ATM_INSTANCE_LINK_ERROR     6            */
-------------------------------------------------------------
```

```
/***********************************/
/* MIB Variablen fuer Fopra Switches */
/***********************************/
VendorIndex:          2     # Gibt an, für welchen Typ von
                            # Komponente die folgenden MIB-
                            # Variablen gelten
ChanCells:            ".1.3.6.1.4.1.326.2.2.2.1.4.1.1.8"
InstanceTyp:          4     # instance-Typ wie oben definiert,
                            #um MIB-OID richtig zu erstellen
ChanRejectedCells:    ".1.3.6.1.4.1.326.2.2.2.1.4.1.1.11"
InstanceTyp:          4
IfError:              ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.10"
InstanceTyp:          6
IfOverflow:           ".1.3.6.1.4.1.326.2.2.1.1.2.3.1.9"
InstanceTyp:          6
```

- `lrz_atm_PrintLinkStats`:
  gibt zu der übergebenen Linknummer die verfügbaren Statistiken aus

6. Verzeichnisstruktur

Das „fopra"-Unterverzeichnis:



Im „bin"-Verzeichnis befinden sich sämtliche ausführbaren Programme.

Die Konfigurationsdateien werden unter „conf" abgelegt. Dazu gehören:

- Komponentendatei
- Verbindungsdatei
- %SWITCH %_VC-Dateien
- %SWITCH %_% LINK %_Count-Dateien
- %SWITCH%_%LINK%_VC-Count-Dateien

Das „input"-Directory beherbergt Dateien, in den Parameter definiert werden, sowie Hersteller- und Trace-MIB-Dateien.

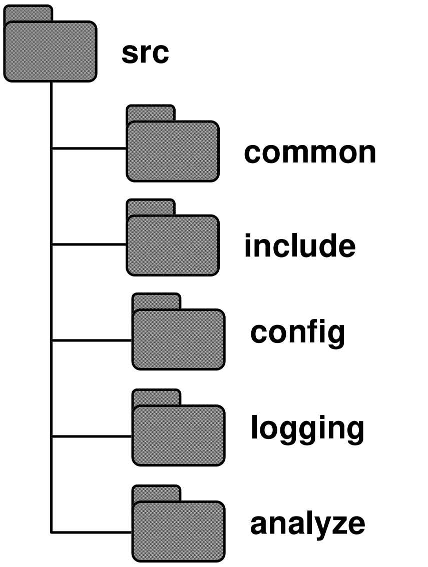
Das „log"-Verzeichnis enthält Dateien, in denen Informationen über Netzlast sowie Fehler mitprotokolliert werden:

- %SWITCH%_%LINK%_%DIR%_Cells-Dateien
  (Netzlast auf Linkebene)
- %SWITCH%_%LINK%_%VPI%_%VCI%_Cells-Dateien
  (Netzlast auf Channelebene)
- Fehlerdateien

Im „src"-Directory befinden sich sämtliche Source-Codes. Der genaue Aufbau dieses Unterverzeichnisses wird später vorgestellt.

Im Verzeichnis „statistik" befinden sich Switch-Statistiken und netzweite Informationen, die den Dateinamen „netstat" erhalten. Werden im Intervall Statistiken für Switches erstellt, erhalten diese als Dateinamen deren Selection Names. Wird für nur einen bestimmten Switch (durch Anklicken in der Submap und Menüwahl) eine Statistikdatei erzeugt, wird der Dateiname zusätzlich mit dem Suffix „.tmp" versehen.

Das „src“-Unterverzeichnis:



Hier befinden sich die Source-Codes aller Teilprogramme des Fopras.

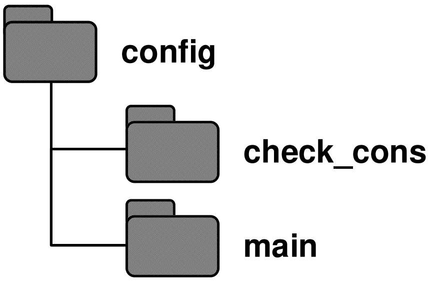
Unter „common“ sind Funktionsbibliotheken (z.B. für Fileroutinen) abgelegt, auf die verschiedene Programme zurückgreifen.

Im „include“-Verzeichnis ist die Include-Datei zu finden, in der u.a. Konstanten und Typen definiert werden.

Die drei folgenden Unterverzeichnisse beherbergen Programm-Code für die drei Fopra-Teilbereiche:

- Erfassen der physikalischen Konfiguration des Netzes
- Ermittlung der Verkehrscharakteristik im Netz
- Aufbereitung der gesammelten Informationen

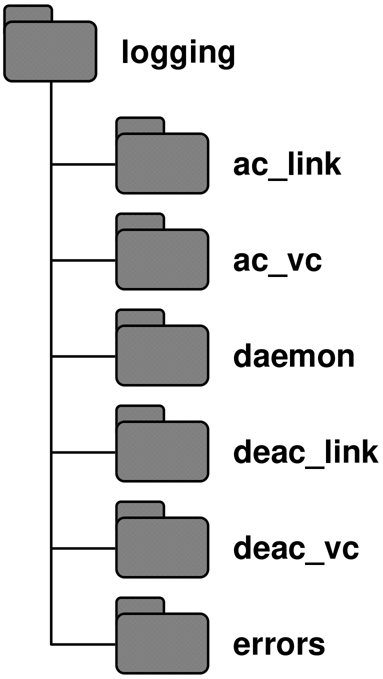
Das „config“-Unterverzeichnis:



Unter „check_cons“ ist der Quelltext für das Programm zu finden, das die Konsistenz von Konfigurationsdateien und OVW-Konfiguration sicherstellt.

Das Verzeichnis „main“ beherbergt das Hauptprogramm des ersten Fopra-Teilbereiches.

Das „logging“-Unterverzeichnis:

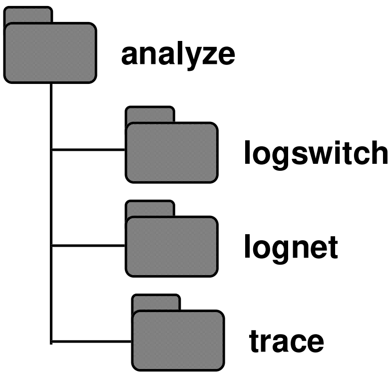


Unter ac_link, ac_vc sowie deac_link und deac_vc sind die Source-Codes der Logging-Programme zu finden, die bei Beginn bzw. Ende von aktiven Phasen auf Link- bzw. VC-Ebene gestartet werden.

Im Verzeichnis „daemon“ ist der Programm-Code für den Channel-Daemon.

In „errors“ ist das Logging-Programm abgelegt, das aufgetrene Fehler protokolliert.

Das „analyze“-Unterverzeichnis:



Die Unterverzeichnisse „logswitch“ und „lognet“ enthalten den Source-Code der Programme, die für die Erstellung von Switch-Statistiken und netzweiten Informationen zuständig sind.

Unter „trace“ ist der Code für das Programm abgelegt, daß dem Netzadministrator erlaubt, eine konfigurierte Verbindung durch das Netz zu verfolgen.