DIPLOMARBEIT

Realisierung eines Network Management & Signalling System für das Prototyp einer Weltraumkommunikationseinrichtung (OBIF)

ausgeführt am Institut für
Technische Informatik
der Technischen Universität Wien

unter Anleitung von

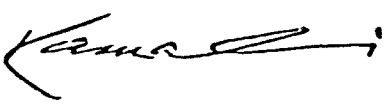
O. Univ. Prof. Dr. Hermann Kopetz

durch

M. Masoud KAMALI
Matr. Nr. 8326940

A-1070 Wien, KAISERStraße 96/8

10.4.1993



Danksagungen

Meinen Eltern und meinem Bruder schulde ich mehr als irgend jemandem sonst. Sie haben mich ermutigt und unterstützt. Ihnen ist diese Diplomarbeit gewidmet.

An dieser Stelle bedanke ich mich bei der Familie Farinpour. Sie haben in meinem Leben eine entscheidende Rolle gespielt, die ich immer in Erinnerung haben werde.

Besonderen Dank schulde ich meinem Freund D.I. Mohsen Emami-Nouri, denn er hat mich ständig angefeuert und gelegentlich kritisiert.

Weiters bedanke ich mich bei meiner Freundin Roja, für ihre Unterstützung und die Geduld, die sie mir während der letzten zwei Jahre entgegenbrachte.

Weitere Personen, die auf unterschiedliche Art und Weise zur Entstehung dieser Arbeit beitrugen:

Dr. D.I. Christian Koza: Für die freundliche Unterstützung und gewissenhafte Betreuung bei der Entstehung dieser Diplomarbeit. Ohne seine Unterstützung und Ideen wäre das Netzwerk-Management-System vom OBIF nicht realisierbar gewesen.

Dr. D.I. Georg Bruckner, D.I. Markus Hadek, Ing. Alfred Kuzmits, Ing. Hans Wimmer und andere Mitarbeiter der Firma Softlab: Für die fruchtvolle und freundliche Zusammenarbeit.

The mind's the standard of the man.
-I. Watts, False Greatness, quoted in [WEB80]

ABSTRAKT

Mit der Erstellung dieser Diplomarbeit war ich im Rahmen eines industriellen Projektes der Firma Softlab Ges.m.b.H. (Österreich) mehr als zwei Jahre beschäftigt. Das OBIF-Projekt wurde für die Europäische Weltraumbehörde (ESA) gemeinsam mit der Firma Spacebel Informatique (Belgien) durchgeführt.

Diese Diplomarbeit ist die akademische Aufarbeitung des Problems durch die Beschäftigung mit der einschlägigen Literatur (Bücher, Papers, Standards) sowie meine Ideen und Erfahrungen aus der Realisierung von einem *Network Management & Signalling System* für OBIF.

Men are not machines, not even ghost-ridden machines.
They are men- a tautology which
is sometimes worth remembering.
-G.Ryle, The concept of Mind [RYL63]

Inhaltsverzeichnis

1. Einleitung

Das OBIF-System wurde unter dem Auftrag von dem in Noordwijk, Holland residierenden *European Space Research and Technical Centre (ESTEC)* gemeinsam von den *Firmen Softlab Ges.m.b.H* (Österreich) und *Spacebel Informatique* (Belgien) entwickelt.

On-board Informatique Test Facility (OBIF) ist ein Prototyp der zukünftigen Weltraumkommunikationseinrichtungen und sollte entwickelt werden um folgendes zu unterstützen:

- Entwurf, Entwicklung und Validierung vom on-board Teil des EETF Prototypes (end to end telematics facility);
- die Implementierung, Bewertung und Validierung der onboard und space/ground Kommunikations-standards und -protokolle (z.B. die CCSDS Protokolle und Standards);
- die Entwicklung und Untersuchung von on-board Computerprototypen und ihrer Laufzeit Unterstützungs-systeme (support-systems);
- die Entwicklung und das Testen von Audio-, Video- und Bildverarbeitungssysteme- und Standards (z.B. das ESA video distribution system),
- die Entwicklung und Untersuchung von Netzwerk-schnittstelleneinheiten (diese umfaßt *Distributive Operatating System (DOS) Boards, Local Operating System (LOS) Boards*, und *Network Interface Boards (NIB)*);
- die Entwicklung und Untersuchung von NM-Aufgaben in den Space- und Groundsegmenten;
- Entwurf und Untersuchung vom Spacelink-subnetwork (siehe Kapitel 3);
- und die Entwicklung und Validierung von neuen (high speed-) LAN und (Lightweight-) Protokol Technologien.

Das Netzwerk Management & Signalling System (NM&S) vom OBIF sollte von Firma Softlab entworfen und realisiert werden. Ein Management-System, das die Steuerung, Kontrolle und Koordinierung von systeminternen und systemexternen Ressourcen (Managed Objects) des OBIF-Systems, mittels Erfaßung und Austausch von Management-Information ermöglicht.

Höher Grad an Software- und Hardwareunabhängigkeit einerseits, und an Automatisierung andererseits zählen zu den wichtigsten Merkmale des OBIF-NM&S-Systems. Sie haben sicher dazubeigetragen, daß das OBIF-System heute nicht nur als

Prototyp einer Weltraumkommunikationseinrichtung für ESTEC angesehen wird sondern schon als unentbehrlicher Bestandteil mehrerer operationellen Projekte vom ESOC (European Space Agency Operating Centre) vorgesehen ist.

Die Entwicklung von OBIF-NM&S-System dauerte fast zwei Jahre. Die Integration und Endabnahme von OBIF-NM&S-System im gesamten OBIF-System wurde am 19.06.92 in Noordwijk abgeschlossen. Das OBIF-System wurde später am 24.09.92 der ESTEC offiziell abgeliefert.

Diese Diplomarbeit mit dem Titel "*Realisierung eines Netzwerk Management & Signalling System für das Prototyp einer Weltraum-kommunikationseinrichtung (OBIF)*" war im Juni 1990 am Institute für technische Informatik (Vors. Prof. Kopetz) der technischen Universität Wien ausgehängt.

Im Rahmen dieser Diplomarbeit habe ich mich zuerst mit den *Consultative Committee for Space Data Systems (CCSDS) Standards*, den OBIF Projektdokumentation, den *European Space Agency (ESA) Standards*, sowie den existierenden Netzwerk-Management-Standards auseinander gesetzt und das erworbene Wissen in zwei Berichten dargelegt. Diese zwei Berichten dienten als Grundlage für die Erstellung von den Kapiteln zwei und drei dieser Arbeit.

Kapitel 2, *"Netzwerk Management in Netzwerken: Ein Überblick über gebräuchlichen Standards und ihre Verwendung"*, behandelt die wichtigsten Netzwerk-Management-Standards und vergleicht sie miteinander. Dieses Kapitel ist eine Einführung in Netzwerk-Management-Standards.

Kapitel 3, *"Advanced Orbiting Systems (AOS): ein Standard für zukünftige Space Data Networks"*, erlautert die Datenverarbeitungsprobleme in Weltraummissionen, verdeutlicht die Merkmale dieser Missionen, und beschreibt die Realisierung und Durchführung der erfundenen Konzepte auf diesem Gebiet.

Kapitel 4, ist dem *On-board Informatics Facility (OBIF)*, das auch *On-board Informatique Test Facility* genannt wird, gewidmet. OBIF ist ein Prototyp des zukünftigen Weltraumkommunikations-einrichtungen. Dieses Kapitel stellt OBIF vor und gibt unter anderem einen ausführlichen Überblick über seine Software- und Hardware-Komponente, Schnittstellen, Applikationen usw.

Kapitel 5, "Network Management & Signalling in OBIF" beschreibt die Realisierung des Netzwerk-Management-Systems für OBIF und ist der Schwerpunkt dieser Diplomarbeit. Es beschreibt das OBIF-NM&S-Konzept und -Model, die Management Domains und

Entitäten, die Structure of Management Information (SMI) und die Management Information Base (MIB) in OBIF-NM&S, usw.

Anhang 1 und 2 sind modifizierte Versionen der *Architectural and Detailed Design Document* der OBIF-NMS-Entitäten, nämlich die *Network Management Station-NMS-* (Anhang 1) und der *Network Management Agent-NMA-* bzw. der *Local Management Entity-LME-* *(Anhang 2)*.

2. Netzwerk Management Standards

2.1. Einleitung

Im Bereich der Rechnerkommunikation lag der Schwerpunkt der Standardisierungsarbeiten lange Zeit auf der Definition von Diensten und Protokollen für die Referenzmodelle. Ein Referenzmodell zusammen mit den Normen für die einzelnen Schichten reicht noch nicht aus, um ein reales Netz zu betreiben. Der Betreiber eines Netzes schafft erst durch das *Netzwerk-Management(NM)* die für einen einwandfreien Betrieb notwendigen Rahmenbedingungen. Nach anfänglicher Zurückhaltung wird das Thema Netzwerk-Management seit einigen Jahren intensiv von wichtigen Standardisierungsorganisationen bearbeitet. Die Vorschläge zu Architektur, Diensten und Protokollen des Netzwerk-Managements stabilisieren sich zunehmend, haben jedoch noch nicht in allen Teilbereichen die Qualität endgültiger, das heißt implementierungsreifer Standards erreicht.

Die Notwendigkeit, im Netzbetrieb auf Managementfunktionen zurückgreifen zu können, hängt sehr stark von der Komplexität des Netzes ab. Historisch gesehen ist daher diese Notwendigkeit bei flächendeckenden Rechnernetzen schon vor längerer Zeit entstanden, und Bezeichnungen wie Netzkontrollzentren oder ähnliche belegen das.

Netzwerk-Management ist seinerseits ein Problemkreis relativ höher Komplexität. Es umfaßt sowohl organisatorische und technologische als auch hardware- und softwareorientierter Maßnahmen.

Bedingt durch die zunehmende Komplexität der Netzwerke, der Heterogenität der angeschlossenen Systeme und auf Grund der Anforderungen der Anwender und Betreiber treten neben der Architektur und den operationallen Diensten, die Architektur und die Funktionen des Netzwerk-Managements immer stärker in den Vordergrund.

Die Haupt-Funktionen des Netzwerk-Managements umfassen:

- Hilfsmittel für das Konfigurieren des Netzes und der zu betreibenden Kommunikations-beziehungen,
- Werkzeuge zur Überwachung des laufenden Netzbetriebes und zur Leistungskontrolle und Leistungssteigerung,
- die Erfassung, das Lokalisieren und das Diagnosieren von Fehlern in der Phase der Inbetriebnahme und im laufenden Netzbetrieb, und

- Hilfsmittel zur Sicherung des Netzes vor unberechtigter Nutzung.

Es wird unterstellt, daß Netzwerk-Management gegenwärtig nicht vollständig automatisch ablaufen kann, sondern die Einbeziehung menschlicher Aufgabenträger (Administratoren) bedarf. Diese Administratoren benötigen unter anderen geeignete Softwareinstrumente, um ihre Aufgaben erfüllen zu können. Diese Softwareinstrumente sind sehr eng mit der übrigen Netzwerksoftware verflochen; so das ein konkretes *Netzwerk-Management-System(NMS)*normalerweise nur im Umfeld einer konkreten Netzsoftware funktioniert. An dieser Stelle wird ein Ansatz für Vereinheitlichungsbestrebungen sichtbar. Diese vollziehen sich in zwei Richtungen:

1. Innerhalb firmenspezifischer Rechnernetz-architekturen verschmelzen die Benutzungsformen für globale und lokale Netze; in diesem Zusammenhang werden auch einheitliche Managementkonzepte entwickelt. Ein Beispiel dafür ist SNA mit NETVIEW.

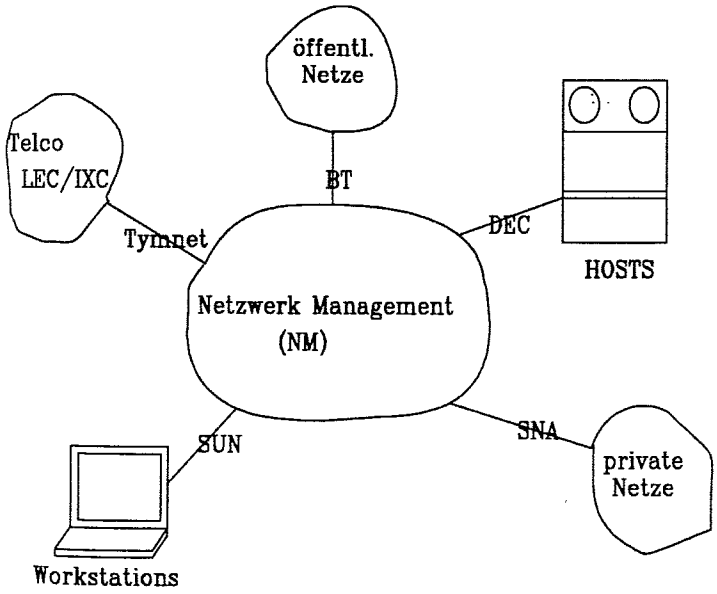


Abb. 2.1) Firmenspezifische NM-Architekturen

2. Internationale Standardisierungsorganisationen arbeiten an Architektur- und Detailkonzepten für das Netzwerk-Management in heterogenen Netzwerken.

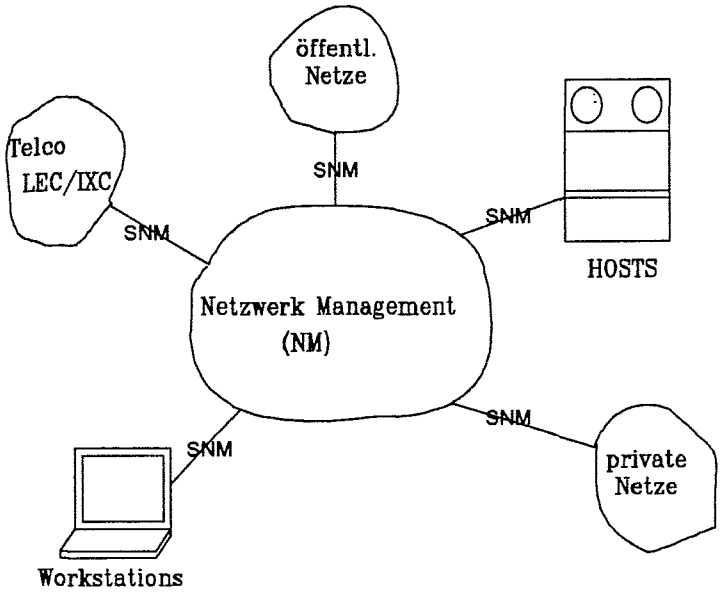


Abb. 2.2) Standardisiertes Netzwerk Management

Dieses Kapitel behandelt hauptsächlich die zwei wichtigsten NM Standards. Zuerst wird das *Open Systems Interconnection (OSI) Model* und seine Protokolle vorgestellt. Die wichtigsten Elementen der OSI NM-Welt, nämlich das *Common Management Information Protocol (CMIP)* und das *Common Management Information Service Element (CMISE)* werden untersucht. Außerdem definiert OSI einige Standards für die fünf Funktionalbereiche der Netzwerk-Management, die auch durchleuchtet werden.

Danach werden die *Internet Protokolle* untersucht. Wir konzentrieren uns dabei auf das zur Zeit am meist verbreiteten Protokoll, nämlich das *Simple Network Management Protocol (SNMP)*und auch auf das *CMOT (CMIP over TCP) Protokoll*.

Neben der zwei erwähnten Standards existiert auch ein drittes, nämlich das von *Institute of Electrical and Electronics Engineers (IEEE)* definiertem *Local Area Networks/Metropolitan Networks (LAN/MAN) Management Standard*. Dieses ist dem formalen Namen *CMIP over LLC(CMOL)* gegeben, obwohl CMIP nicht auf jedem IEEE-LAN fährt.

Alle NM-Standards bieten eine Perspektive für allgemeines herstellerunabhängiges Netzwerk-Management an. An dieser Stelle sollte noch erwähnt werden, daß weltweit bedeutende Initiativen von großen Netzanwendern bzw. -betreibern wie z.B. *MAP* und *TOP* sich nach den besonders von der OSI auf dem NM-Gebiet erarbeitenden Konzepten ausrichten.

Ein wichtiges Ziel von Netzwerk-Management ist es, an einen integrierten Ansatz für das managen von heterogenen Netzen heranzukommen. Man braucht sich nur vorzustellen wie vorteilhaft es wäre, wenn *Ethernet, TCP/IP, Fiber Distributed Data Interface (FDDI)* und *Integrated Services Digital Network (ISDN)* Netze alle dieselben NM-Protokolle verwendeten.

Die folgende Abbildung zeigt eine typische LAN-Konfiguration (Kein NM-Standard fordert, daß Netzerk-Managementkomponenten an bestimmte Orten plaziert werden. In den jetzigen Implementierungen residieren die Agents normalerweise in Komponenten wie Servers, Gateways, Bridges und Routers, und der Managing Process residiert im *Network Control Station (NCS).*

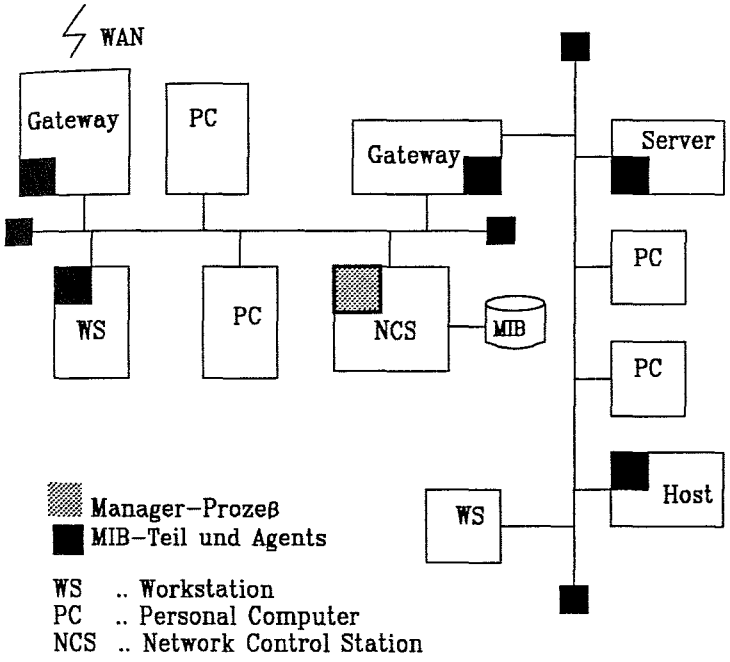


Abb. 2.3) Eine typische LAN-Konfiguration

Es gibt drei NM-Komponente, die in allen genannten Standards vorkommen:

1. *Agent* : Informationen über NM-Elementen werden von den sogenannten Agents gesammelt und an das *Managing Process* geschickt.

2. *Managing Process*: Kontrolliert die Agentaktionen.

3. *MIB*: wird von den *Agent* und *Managing Process* verwendet um die Struktur und den Inhalt von Managementinformation festzustellen.

Ganz abstrakt ausgedrückt, unter einem Netzwerk Management System versteht man nichts anderes als Protokolle, die dazu dienen, um NM-information zwischen mehreren *Agents* und dem *Managing Process* auszutauschen.

Die genauere Definition und die Funktionalität dieser Komponenten ist für jeden einzelnen Standard später in diesem Kapitel gegeben.

2.2. OSI-Management-Konzept

In seinem Originalform, besteht das OSI Referenzmodell nur aus globalen Erklärungen über das Netzwerk-Management. Während der letzten Jahre wurde das OSI Referenzmodell durch das *OSI Management Framework* erweitert.

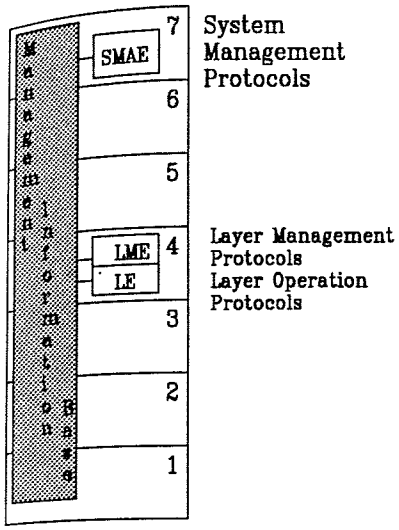
Das OSI Netzwerk-Management ist definiert als "die Möglichkeiten zur Kontrolle, Koordination und Steuerung von Ressourcen, die für Kommunikation in einer OSI-Umgebung verantwortlich sind". Die Ressourcen weisen Datenspeicherung- , Datenverarbeitung- bzw. Datenkommunikationsfähigkeiten auf.

Netzwerk-Management für offene Systeme wird durch die Kooperation folgender, voneinander unabhängiger fünf Netzwerkkomponente ausgeübt:

2.2.1. Die Management Information Base (MIB)

Alle Managementinformationen eines offenen Systems sind in der *MIB* abgespeichert (siehe folgende Abbildung).

Die *MIB* enthält Managementinformation über jedes einzelne Ressource im System.



Der OSI-MIB

Abb. 2.4) Der Management Information Base in OSI

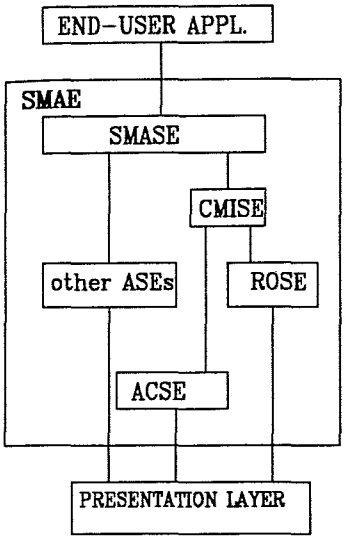
2.2.2. Das System Management (SM)

Das *Systems Management* stellt Mechanismen zur Steuerung, Kontrolle und Koordination von Managed Objects innerhalb von Schichten des lokalen offenen Systems und für den Austausch von Managementinformation mit anderen Systemen zur Verfügung. Diese werden durch den *Systems Management Application Process (SMAP)* ausgeführt. Der *SMAP* erbringt die verschiedenen Arten von Managementfunktionen und ermöglicht dem Systemadministrator über eine Benutzerschnittstelle die Verwendung dieser Managementfunktionen. Die Übermittlung von Management-Information zwischen den *SMAPs* zweier offener Systeme erfolgt durch eine eigene Instanz, die *Systems Management Application Entity (SMAE)*. Sie ist für die Abwicklung der Kommunikationsaufgaben, die für den Austausch von Managementinformation zwischen zwei *SMAPs* notwendig sind, zuständig. Die *SMAE* stellt dem *SMAP* die notwendigen Dienste zur Verfügung, um Meldungen oder Steueraufträge zur Ausführung einer Managementfunktion an den *SMAP* des Partnersystems zu übermitteln. Zwei *SMAEs* kommunizieren miteinander über das *Common Management Information Protocol (CMIP)*.

Das *Systems Management* ist eine *Information Process Application*, die Mechanismen zur Kontrolle, Steuerung und Überwachung von allen Ressourcen eines offenen Systems zur Verfügung stellt. Das *Systems Management* wird von den als *Management Processes* genannten verteilten Aktivitäten durchgeführt.

Die *Management Processes* kommunizieren miteinander durch den Austausch von Directiven. Ein *Management Process* fungiert

entweder als ein *Managing Process* oder als ein *Agent Process*. Als *Managing Process* ist er für die Ausführung einer oder mehrerer Management- Aktivitäten zuständig. Als *Agent Process* ist er für das Managen einer Menge von *Managed Objects* zuständig, wenn dieses von einem *Managing Process* angefordert ist.



Die Struktur von OSI-Management

Abb. 2.5) Die Management-Struktur in OSI

Dieses Mangement-Modell kann verfeinert werden, wenn man die informative, kommunikative und organisatorische Aspekte des *Systems Managements* in Betracht zieht.

2.2.2.1. Die informativen Aspekte des *SM*

Die zu managenden Ressourcen werden im OSI Management- Modell abstrakt als *Managed Objects (MOs)* repräsentiert. Abstrakt in dem Sinne, daß nur die fürs Management relevanten Eigenschaften einer Ressource in seinem *Managed Object* enthalten sind. Ein *Managed Object* kann auch andere *Managed Objects* enthalten. Die *MIB* eines offenen Systems ist definiert als die Menge aller *Managed Objects* des Systems.

Die Eigenschaften eines *Managed Object* werden unter anderem durch seine Attribute beschrieben, wobei nicht jedes *Managed Object* Attribute enthalten sein muß. Ein Attribute ist ein Informationselement, auf das mindestens eine *Management Operation* definiert ist.

Der Standard *"SIM - Structure of Management Information"* beschreibt Formate und Typen von Management Information zu

Managed Objects. Er enthält eine Liste von Informationstypen, die jedes *Managed Object* enthalten kann. Beispiele dafür sind: Zustand, nicht stellbare Zähler, stellbare Zähler, definiertes Ereignis usw.

Für jeden Informationstype spezifiziert der obengenannte Standard das zugehörige Informationselement, die zulässigen Operationen, die vererbbaren Eigenschaften, die implizierten Beziehungen und die Spezifikationseigenschaften.

Managed Objects werden auch durch *Management Operations*, die auf ihnen ausgeführt werden können, durch Notifications (Meldungen,Ereignisse), die sie auslösen können und durch ihre Beziehungen bzw. Relationen zu anderen *Managed Objects* charakterisiert, und sie werden im wesentlichen ebenspezifisch definiert.

2.2.2.2. Die funktionalen Aspekte des *SM*

Die Funktionen des *Systems Management* werden entsprechend der Einteilung, die von der ISO getroffen wurde, systematisch in die folgenden Kategorien bzw. Funktionsbereichen eingeteilt:

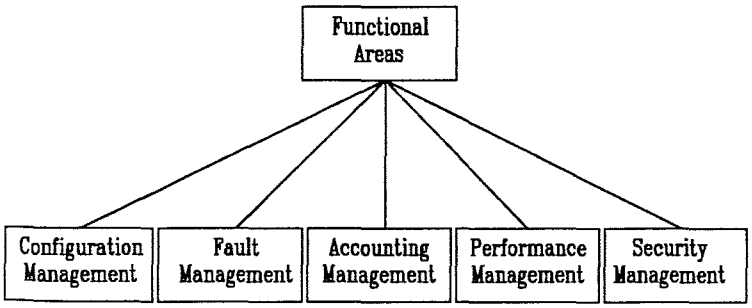


Abb. 2.6) Funktionsbereiche des Systems-Management

a) Das *Configuration Management*

Das *Configuration Management* aktiviert und deaktiviert die OSI Ressourcen und legt ihre Beziehungen untereinander fest. Ein Beispiel für eine Beziehung zwischen zwei OSI Ressourcen ist die Festlegung, welche *(N)-Instanz* (N sei die N-te Schicht) über welchen *(N-1)-SAP* mit der darunterliegenden Schicht verbunden ist. Das Konfiguration-Management sammelt, speichert und stellt Informationen über den aktuellen Netzwerkzustand bereit und steuert die Netzwerkkonfiguration, sowie die Übermittlung von Nachrichten (*Routing*). Dazu ist es in der Lage, Managed-Objekte zu erzeugen und zu löschen; Managementinformation d.h. Eigenschaften, Zustände, Beziehungen zwischen den Managed-Objekten zu setzen und zu lesen; Ereignisberichte nach Veränderungen von Managementinformationen

zu erzeugen usw. Seine Funktionen ermöglichen u.a. die Konfiguration der MOs beim *initial setup,* die Anpassung der MOs an neue Gegebenheiten und die Re-konfiguration von MOs wenn diese erwünscht bzw. notwendig ist.

b) Das *Fault Management*

Das *Fault Management* bietet die zur Übermittlung von Fehlerdaten und zum Anstoß ihrer Ermittlung nötige Funktionalität. Es stellt Fehlerzustände fest und lokalisiert, isoliert und behebt Fehler. Dieses erfolgt sowohl durch Analyse von gemeldeten oder abgefragten Managementdaten als auch durch geeignete Tests. Dem *Fault Management* sind auch Funktionen zugeordnet, die durch entsprechende Eingriffe in die Netzkonfiguration oder in die einzelnen Netzknoten eine Fehlerbehebung ermöglichen. Hier sind, das Durchführen von Back-up-Strategien oder das stufenweise Nutzen vorhandener Redundanz, wie zum Beispiel das Abtrennen von fehlerhaften Knoten oder das Umschalten von Master- auf Slavekonfiguration, zu nennen. Natürlich ist die Ausführung von obenerwähnten Funktionen innerhalb eines Systems nur bei enger Zusammenarbeit zwischen Configuration- und Fault-Management des jeweiligen Systems möglich.

c) Das *Accounting Management*

Das *Accounting Management* soll Daten für die Abrechnung und Zuordnung von Kosten, die durch die Benutzung von Ressourcen oder Diensten der Kommunikation entstanden sind, erfassen. Es unterstützt Kostenanalysen und führt entsprechende Statistiken.

d) Das *Performance Management*

Das *Performance Management* sammelt Informationen über Leistungskennwerte der Ressourcen und steuert diese Informationssammlung. Leistungskennwerte umfassen Daten wie Auslastung des gesamten Netzes oder Auslastung einzelner Ressourcen.

Die gesammelten statistischen Daten können auch verwendet werden, um Trends in der Netzauslastung rechtzeitig zu erkennen. Wenn es sich z.B. auf Grund dieser Daten herausstellt, daß immer häufiger Verbindungen zwischen einem Ort A und einem Ort B aufgebaut werden, dann kann der Systemadministrator einen drohenden Engpaß vermeiden, indem er für eine entsprechende Verbindung eine höhere Übertragungs-kapazität bereitstellt.

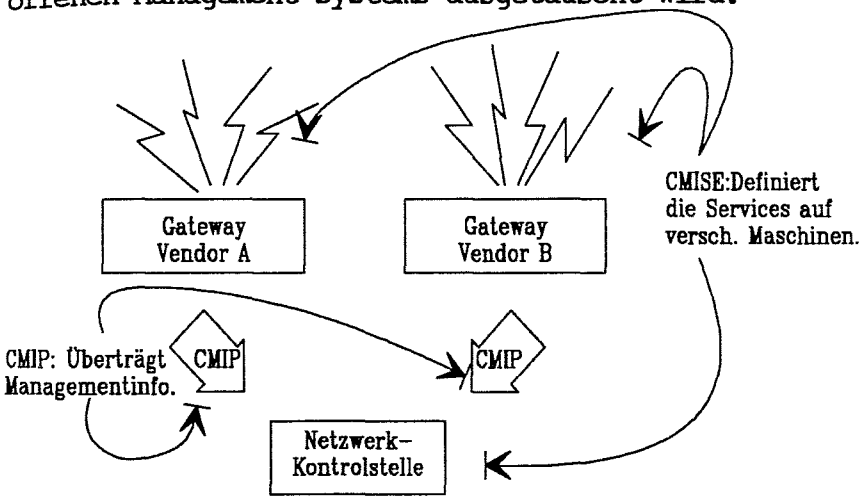
e) Das *Security Management*

Das *Security Management* schützt Ressourcen (*Managed Objects*) vor dem Zugriff durch nicht autorisierte Benutzer. Es umfaßt unter anderen Zugriffskontrolle, Password-Verwaltung und Datenverschlüsselung für die besonders geschützte Diensten.

Unter dem neuen Begriff *"Software Management"* sind jene Management Funktionen zu verstehen, die die Verteilung von Software, das Management von Versionen usw. unterstützen.

2.2.2.3. Die kommunikativen Aspekte des *SM*

Die kommunikativen Aspekte des Systems Management beschreiben, wie die Management Information zwischen den Komponenten eines offenen Management Systems ausgetauscht wird.



Beziehung zwischen CMISE und CMIP

Abb. 2.7) Die Beziehung zwischen CMISE und CMIP

Den Austausch von Management Information wird jetzt in kurze erläutert:

Die *Management Processes* kommunizieren miteinander über den Austausch von Directives und benützen die von der *System Management Application Entity (SMAE)* erbrachten *Application Layer Services*. *SMAE* enthält das *System Management Application Service Element (SMASE)* und das *Common Management Information Service Element (CMISE)*. *SMASE* definiert die abstrakte Syntax und Symmantik der auszutauschenden Managementinformation. *CMISE* erlaubt *SMAEs*, nach den Regeln des *Common Management Information Protocol (CMIP)* zu kommunizieren. *CMISE* und *CMIP* bilden das standardisierte Verfahren für den Austausch von Managementinformation. *CMISE* unterstützt folgende Services:

- *Unconfirmed/Confirmed Event Report* (Ereignisberichte),
- *Unconfirmed/Confirmed Set* zum Setzen von Managementinformation,
- *Confirmed Get* zum Lesen von Managementinformation,
- *Unconfirmed/Confirmed Action* zum Anstoß weiterer Managementfunktionen,
- *Unconfirmrd/Confirmed Create and Delete* zum Erzeugen und Entfernen von *Managed Objects*
- *Linked Reply*

Das *SMAE* kann noch andere *Application Service Elements (ASE)* enthalten; wie zum Beispiel das *Association Control Service Element (ACSE), Filetransfer, Access and Management (FTAM)*, das *Remote Operation Service Element(ROSE)* oder *Transaction Processing (TP)*.

Eine möglich Konfiguration des NM-Modells in der Anwendungsschicht (*application layer*) des OSI zeigt die Abbildung 2.5.

Das *Systems Management Application Service Element (SMASE)* in Abbildung 2.5. generiert und verwendet *Management Application Data Units (MA-PDUs)* um mit anderen Rechnern zu kommunizieren. Es nützt die Services vom *ASEs* oder *CMISE*, wobei das Vorhandensein eines *CMISE*-Elementes das Vorhandensein eines *ROSE*- und eines *ACSE*-Elementes impliziert.

2.2.2.4. Die organisatorischen Aspekte des *SM*

Die organisatorischen Aspekte des Systems Management beschreiben, wie für eine aus offenen Systemen bestehendes globales Netzwerk, das OSI Management administrativ über Management Domains verteilt wird. Ein Management Domain ist eine Sammlung von einem bzw. mehreren Management Prozesse samt mit ihm bzw. mit ihnen assozierter Managed Objects.

2.2.3. Das (N-) Layer Management

Das *(N)-Layer Management* stellt Mechanismen zur Steuerung, Kontrolle und Koordination von Schicht-(N) Ressourcen. Eine *(N)-Layer Management Entity ((N)-LME)* ist für die Abwicklung obengenannter Mechanismen zuständig und kooperiert mit dem *SMAP*.

Die *SMAP* und *LMEs* der sieben Schichten haben Zugriff auf die *Management Information Base (MIB)*, in der alle für das Management notwendigen Informationen, wie z.B. Systemstatus, Statistiken, Zugriffsrechte usw. abgespeichert ist (zu den Aufgaben der *(N)-LME* gehört z.B. die Zuordnung von *(N)-SAPs* zu *(N)-Instanzen)*.

2.2.4. Die (N-) Layer Operation

Die *(N)-Layer Operation* kann die Ausführung von Managementfunktionen für eine Kommunikationsinstanz, die ein Teil der normalen Kommunikation zwischen zwei *(N)-Layer Protocol Entities* ist, unterstützen

2.2.5. Das Local Management

Das *Local Management* ist das Management von lokalen Ressourcen. Dieses berührt nicht die Kommunikation zwischen offenen Systemen, und wird deswegen in den *OSI Management Standards* außer Acht gelassen. Die Designer von Managementsysteme müssen genau definieren, wie auf *Local Management Information* zugegriffen wird, und welche *Local Management Operations* von ihnen unterstützt werden.

2.3. Netzwerk-Management in TCP/IP Netzwerken

Neue Netzwerk-Management Tätigkeiten in der Transmission Control Protocol/Internet Protocol (TCP/IP) Community haben sich auf die Standardisierung zweier NM Protokolle konzentriert, die für den Austausch von Management Informationen sorgen. Diese Protokolle sind das Simple Network Management Protocol (SNMP) und das Common Information Services and Protocol over TCP/IP (CMOT).

2.3.1. Vorgeschichte

Im Frühjahr 1988 hielt das *Internet Activity Board (IAB)* eine spezielle Versammlung zur Festlegung einer Management Strategie für den multilateralen, auf *TCP/IP* basierenden Internets ab. Die Empfehlungen dieser Versammlung wurden offiziell bestätigt und brachten zwei parallele Standardisierungsbemühungen hervor:

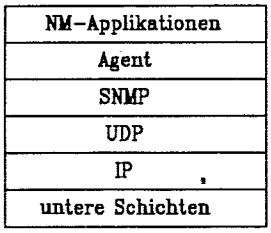
SNMP und *CMOT*. Es empfiehl sich *SNMP* als eine vorübergehende Lösung zu entwickeln, während *CMOT* als eine eventuell langfristige Lösung zu gebrauchen ist.

Eine Gruppe von Experten hat sich auf die Produktionsstandards zur Implementierung von *SNMP* konzentriert. *SNMP* ist die Erweiterung eines früheren NM-Protokolls, genannt *Simple Gateway Monitoring Protocol (SGMP)*, das zur Überwachung von TCP/IP-Gatewaysystemen *(IP-Routern)* entworfen worden war. Die *SNMP* Arbeitsgruppe wurde durch den dringenden Bedarf angetrieben, Netzwerkadministratoren mit einer besseren Lösung zur mittelbaren Handhabung der *TCP/IP-Gateways* und verwandte Netzwerk Interfaces zu versorgen.

Eine zweite Gruppe arbeitete an eine Übereinkunft zur Implementierung von *CMOT*. *CMOT* ist eine Protokol-Stapel-Beschreibung (*protocol stack specification*) für die Entfaltung von *ISO/OSI Common Management Information Protocol (CMIP)* über bereits existierende TCP/IP-Umgebungen.

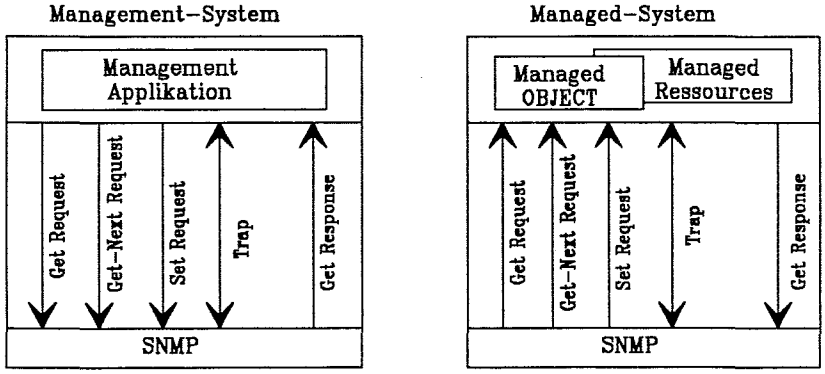
2.3.2. Die Architektur von *SNMP* und *CMOT*

Abbildung 2.3. zeigt die *SNMP* Architektur. *SNMP* ermöglicht Applikationen mit einer einfachen Befehlsreihe, die nur *Get*, *Set* und *Get-Next* enthält. Diese werden nach den Regeln der *Basic Encoding Rules (BER)* und der *ISO Abstract Syntax Notation One (ASN.1)* verpackt und über vorhandene *UDP/IP Services* geschickt. Es gibt auch ein sehr beschränktes *"Trap"* Message, die sechs Standardtypes von unbestätigten Ereignissen *(unconfirmed events)* erlaubt, asynchron gemeldet zu werden.



SNMP Schichten
Abb. 2.8) Die SNMP-Schichten

Neue SNMP-Implementierungen drehen sich um einen Kern aus einem Set von drei Spezifikationen:



Schnittstelle von SNMP mit den oberen Schichten
Abb. 2.9) Schnittstelle von SNMP mit den oberen Schichten

- das *SNMP Protocol* über ein *UDP/IP Protocol Stack*,

- die Regeln für das *Structure of Management Information (SMI)* für den Gebrauch mit *SNMP*,
- und eine Sammlung von ca. 100 standardisierten *Managed Objekts*.

Die oben erwähnte Sammlung von Objekten, genannt *MIB_1*, umfaßt ein *Management Information Base (MIB)*, das für begrenztes Fault- und Konfiguration-Management sorgt. Die *MIB-1* Objekte repräsentieren Parameter, die sich auf *TCP/IP* Protokolle (*TCP,IP,UDP,ICMP* und *EGP*), Systemadress- und Interfacetabellen und Information zur Systemidentifizierung beziehen.

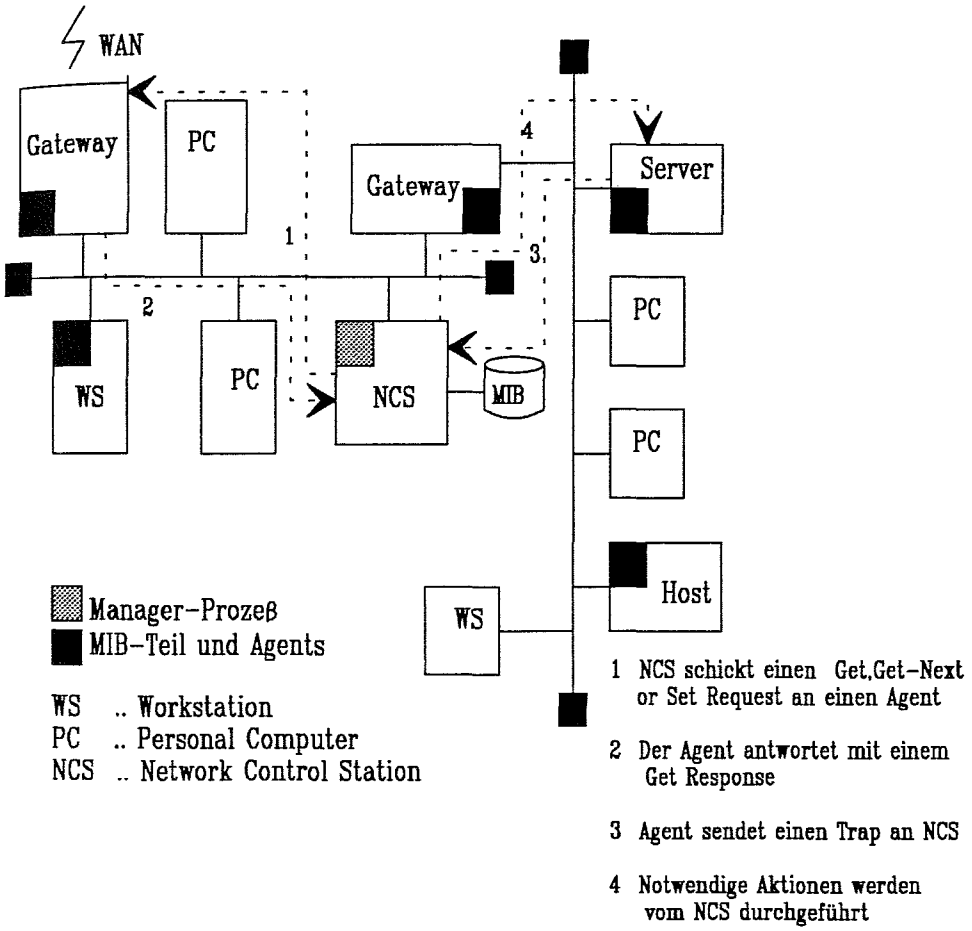
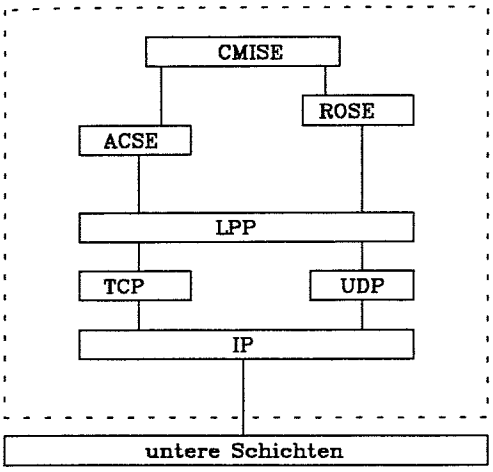


Abb. 2.10) Austausch von NM-Information über eine typische LAN-Konfiguration

Die *CMOT* Architektur ist in Abbildung 2.11. wieder gegeben. Die von *CMOT* zur Verfügung gestellten *Application Services* sind definiert durch *die Common Management Information Services (CMIS)*, die *Service Definition* für das *ISO CMIP Protokoll*. Wie in der Abbildung gezeigt wurde, beruht die *Application Layer* auf OSI und umfaßt das *Common Management Information Service Element (CMISE)*, das *Remote Operation Sevice Element (ROSE)*,

und das *Association Control Service Element (ACSE)*. Die *Transport-* und *Network-layers* sind der Reihe nach *TCP/UDP* und *IP*. Der *Presentation Layer* besteht aus einem *Leightweight Presentation Protocol (LPP)* und ist mit einem Mechanismus zur Unterstützung der OSI *Application Services*, und zwar direkt über *TCP/IP* Umgebungen, ausgestattet.



CMOT Schichten

Abb. 2.11) Die CMOT-Schichten

2.3.2. Technischer Vergleich

In diesem Abschnitt werden *SNMP* und *CMOT* aus drei Perspektiven miteinander verglichen:

- den Applikationen zur Verfügung gestellten *Protocol Services*,
- Management-Paradigmen, und
- Implementierungsvoraussetzungen.

Im allgemeinen stellt *CMOT* (über *CMIS*) *Management Applications* mit einem starken, objektorientierten Interface, während *SNMP* ein relativ primitives und attributorientiertes Interface anbietet.

Die folgenden Unterkapitel vergleichen die Managementkommunikationsmodelle, die angebotenen Funktionen, und Informationsmodelle für *SNMP* und *CMIP*.

2.3.2.1. Managementkommunikation

CMIP ist ein verbindungsorientiertes Protokoll, während *SNMP* verbindungslos ist. Zwei *CMIP Application Entities* können nur dann Nachrichten austauschen wenn sie eine Verbindung

miteinander aufbauen und aufrechterhalten. *SNMP Application Entities* tauschen direkt Nachrichten miteinander; sie brauchen nicht erst die *Application Layer* Verbindungen herzustellen und dann aufrechtzuerhalten. Das verkleinert den *Processing Overhead*, besonders für unregelmäßige *SNMP* Kommunikation. Dennoch kann der Sender einer *SNMP* Nachricht nie sicher sein, ob die Nachricht ihren Empfänger erreicht hat. Für eine garantierte Zustellung brauchen deshalb die Anwendungen, die *SNMP* verwenden, eine Art von Zustellungsversicherung (zum Beispiel eine Erkennungsstrategie, die zwischen den Applications arbeitet)

2.3.2.2. Managementfunktionalität

Multiple Object Selection:

CMIP sorgt für die Durchführung von hochentwickelten Bedingungsbefehlen, basierend auf Objekttyp, Wert und dem relativen Standort im gemanageten Netzwerk. *SNMP* besitzt diese Fähigkeit nicht und verlangt die spezifische Benennung der gerade bearbeiteten Objectinstanze. Die einzige Ausnahme ist die *SNMP Get-Next* Nachricht, die einer Application erlaubt, das relativ zur aktuellen Position nächstes *MIB Object* abzurufen. Eine *Bridge Management Application*, zum Beispiel, auf einem auf *CMIP* basierenden System könnte direkt Portinformationen für alle Bridges des gemanageten Netzwerkes, für die *"Badpacketcount >= 100"* gilt, auffordern. Im Vergleich muß eine *SNMP* benützende Application über alle Bridges im gemanageten Netzwerk Bescheid wissen, die Information aus jedem Bridge individuell apportieren und dann *"Badpacketcount"* überprüfen, um die in Frage kommenden Bridges zu bestimmen.

Synchronisation:

Sowohl *CMIP* als auch *SNMP* können innerhalb einer einzigen Anfrage vielfache Operationen (des gleichen Types) ausführen. Jedoch innerhalb einer einzigen Anfrage können *CMIP* Operationen auf mehreren Objektinstanzen entweder auf *"Best Effort Basis"* oder auf *"atomarer Basis" (atomic basis)* ausgeführt werden. *SNMP* Anfragen sind immer automar, d.h. wenn alle abgefragten, einzelnen Operationen aus irgendeinem Grund nicht vollständig ausgeführt werden können, werden keine der einzelnen Aktionen ausgeführt und eine Errormeldung wird ausgeführt.

Die Tatsache, das die Granularität der Atomarität bei CMIP höher ist als bei SNMP, ist auch ein Grund (siehe 2.5) warum die CMIP-Implementierungen deutlich komplexer sind als die von SNMP.

Linked Reply:

CMIP erlaubt einer Application größere Datenmengen aus verschiedenen gemanageten Systemen über eine einzige Request-Anfrage zu abzufragen. Die Information wird in Form von mehrfach verketteten Antworten *(linked replies)* zurückgeschickt. *SNMP* unterstützt keine *Linked Replies*. Folglich ist man auf die Informationsmenge, die durch eine einzige Request-Anfrage apportiert wird angewiesen. Diese Einschränkung macht *SNMP* für die Erteilung größerer Informationsmengen (z.B. Adresstabellen) ineffizient.

MIB Browsing (Durchblättern der *MIB*):

Sowohl *CMIP* als auch *SNMP* können dazu benützt werden, *MIBs* durchzublättern. *CMIP* stellt diese Eigenschaft durch seine *Multiple Object Selection* Fähigkeit zur Verfügung (man kann auch Wildcards benützen) und *SNMP* mit seinem *Get-Next* (im Fall einer großen *MIB* bestimmt ein weitschweifigster und langsamer Prozeß).

Actions:

CMIP sorgt für die Definition und Ausführung objektspezifischer, imperative Befehle (*imperative commands*, z.B. *"reboot system"*). *SNMP* unterstützt imperative Befehle nicht direkt, nützt aber Nebenwirkungen aus, um spezifische Aktivitäten auszuführen (z.B. könnte das Setzen einer *SNMP Boolean Variable* wie *"reboot MIB"* das System zu Reboot veranlassen.

Event Confirmation:

CMIP Ereignisse können bestätigt oder unbestätigt sein, wobei *SNMP* Ereignisse immer unbestätigt sind. Daraus resultiert, daß ein *Managed Device*, der einen *SNMP Event Report* schickt, nie sicher sein kann, ob der Bericht vom adressierten Managementsystem tatsächlich empfangen worden ist.

Systems Management Functions:

Eine Auswahl von *Systems Management Functions (SMFs)* werden standardisiert, um die von *CMIS* angebotenen Services zu beanspruchen (Beispiele von *SMFs* beinhalten Fehleranzeigen, Vertrauens- und Diagnosetests, und *Log Control*). Im Fall *SNMP* gibt es kein Gegenstück zu den *SMFs*, es sind auch keine geplant. Es wird erwartet, daß zukünftige *SMFs* das *CMIP* ermächtigen können, *Manager-to-Manager*-Kommunikation zu unterstützen. *SNMP* unterstützt spezifische *Manager-to-Manager*-Kommunikation nicht entsprechend bzw. nicht ausreichend. Dieser Mangel kann in den SNMP-Implementierungen, die solche Kommunikationsarten benötigen, durch gewisse Maßnahmen (Proxy

Agents, intelligentere Managing Prozesse) behoben werden, der *Preis* dafür wird allerdings sehr groß sein.

2.3.2.3. Managementinformation

CMIP Objekte (d.h. OSI Management Objekte) sind umfassender als *SNMP* Objekte:

Object Definition:

CMIP Objekte sind nicht gleich *SNMP* Objekte. *CMIP Object* Definitionen beinhalten Listen von Attributen, Events und imperative Actions. Dazu sind Objekttypen in einer "Vererbungshierarchie" struktuiert, und die aktuellen Instanzen von *Managed Objects* werden durch eine *"containment hierarchy"* benannt. Im Vergleich sind *SNMP* Objekte ähnlich den Attributen der *CMIP* Objekte und zusätzlich noch in Gruppen organisiert. Die zusätzlichen Details und die Struktur in *CMIP* Objekten unterstützen direkt die hochentwickelte Funktionalität, die von der *CMIP Service Definition* zur Verfügung gestellt wird.

Object Naming:

Die *CMIP* Nammenshierarchie basiert auf einem hochflexibelen *Containment Modell*, wo *Managed Objects* innerhalb rekursiv definierter Gültigkeitsbereiche einander umfaßt werden. Folglich können Anwendungen Objekte anhand Namen, die bedeutungsvoll und nicht einem bestimmten Netzwerk Adressierungs Machanismus unterworfen sind, bearbeiten. Im Kontrast dazu sind *SNMP Managed Devices* per IP-Adressen benannt und *Managed Objects* innerhalb von Devices sind durch statisch vordefinierte Bennenungsbäume benannt.

2.4. Management-Modelle

Es gibt zwei Modelle (dem *SNMP* und *CMIP/CMOT* grob entsprechend), die für das Netzwerk-Management verbreitet benützt werden:

1. *Polling Based Management*:

   Managed Devices werden über Informationen befragt und senden diese synchron zum Manager zurück (SNMP ist aus mehreren Grunden auf diesem Konzept basiert, von denen zwei hier aufgelistet werden:

   - Polling hält die Komplexität des Systems in Grenzen.
   - Weil Polling von SNMP Manager kontrolliert wird, kann das Verkehrsaufkommen limitiert werden.

2. *Event Based Management*:

Managed Devices senden *pre-configured* Informationen asynchron zum Manager. *Beim System-Startup vom OBIF zum Beispiel, senden alle Managed Object -asynchron- eine Registration-Request-Nachricht (ein SNMP-Trap) an ihre Manager, und das innerhalb eines kurzen Zeitintervalles. Während dieser Phase, kann der SNMP Manager das Verkehrsaufkommen nicht mehr direkt beinflußen. Nachrichten können in diesem Fall dupliziert übertragen werden oder auch verloren gehen. Aus diesem Grund schicken OBIF-MOs ihre asynchrone Event-Meldungen periodisch zum Manager.*

2.5. Implementierungsvoraussetzungen

Die Implementierung und Entwurf von *CMOT* ist viel komplexer als *SNMP*. Eine *SNMP* Implementierung braucht im Gegensatz zu *CMIP* (im *CMOT*) die aktuelle Information über den Zustand von **Managed Objects** bzw. über die Assoziation zwischen diesen, nicht zu speichern, und aus diesem Grund braucht sie weniger Speicherplatz. Nach Angaben von *Netlab (eine Netzwerk Management Firma)* verbraucht ein *SNMP Source-code* um 2/3 weniger Speicherplatz als der von *CMOT* (excl. *TCP/IP code*) . Hinzu kommt noch, daß *SNMP* für jede Host Datenstruktur etwa 300 Bytes benötigt, das sind noch um etwa 500 Bytes weniger als die Größe von einer *CMOT* Host Datenstruktur. Dies bedeutet wiederum, daß *CMOT* viel mehr *Dynamic Memory* bedarf.

Network Management Protocols werden von Applikationen für die Ausführung von Managementoperationen verwendet. Aus der Sicht von Applikationen kann das relative Performance der beiden Protokolle miteinander verglichen werden, indem man die Anzahl der Management Operationen (z.B. *Get requests*), die Innerhalb einer gegebenen Zeitperiode ausgeführt werden können, mißt. Eine einfache Messung ist die *Management Operations Per Second (MOPS)*. *MOPS* ist allerdings von systemspezifischen Parametern, wie *Millions of Instructions Per Second (MIPS)* abhängig. Daraus resultiert ein systemunabhängiger Faktor für die Netzwerk-Managementprotokolle: *MOPS / MIPS*. Die ersten Testversuche bei *Netlab* ergaben für *SNMP* einen *MOPS / MIPS Faktor* von 46 und für die *CMOT* Protokolle einen von 18 (es wurden mehrere Batchfiles mit je 200 einfachen *Get-Request* Befehlen in einem *Send Loop* ohne *User Interaction* ausgeführt).

Es ist an dieser Stelle zu erwähnen, daß ein größerer *MOPS/MIPS Faktor* nicht unbedingt bessere Leistungskennwerte mit sich bringt, weil man mit den *CMOT* Protokollen zu größeren Informationsmengen gelangen kann als mit *SNMP*. Eine Applikation könnte deshalb *SNMP* und *CMOT* Management-Operationen mit unterschiedlichen *Management Performance Faktoren* ausführen.

2.6. Zukünftige Richtungen

Während *SNMP* als gegenwärtiger, tatsächlicher Standard zur Bearbeitung von *TCP/IP* Netzwerken hervorgetreten ist, verbleibt *CMOT* als die offiziell bestätigte Langzeitlösung für *TCP/IP* Netzwerk-Management.

Wie auch immer, ist es wahrscheinlich, daß sich *SNMP* jenseits der traditionellen *TCP/IP* Umgebungen ausbreitet. Zusätzlich zum *TCP/IP Ressource Management*, wird es möglicherweise von Betriebsystemen (z.B. *Berkley Unix*), von *LAN Services* (z.B. *LAN Manager*) und von *OSI Networking Protocols* fürs *Ressource Management* verwendet werden.

SNMPs gegenwärtiger Erfolg beruht auf der Tatsache, daß es relativ einfach zu implementieren ist und keine großen Mengen von Verarbeitungs- und Speicherressourcen benötigt. Es stellt geringe Anforderungen an Atomarität der Ausführung der Aktionen. Dies erlaubt es *SNMP*, in bereits existierenden Produktstrukturen integriert zu werden - ein wichtiger Punkt, weil dies bedeutet, daß kleinere Produkte nicht mehr wieder entworfen werden müssen, um gemanaged zu werden. Außerdem sind die *SNMP* Spezifikationen relativ unkompliziert und erlauben auch einige Alternativen.

Trotz seiner zunehmende Entfaltung, bringt *SNMP* einige Beschränkungen für die NM-Anwender mit sich:

Das *SNMP* Management Paradigma (großteils auf *Polling* basiert) ist nicht für die Überwachung größerer Netzwerke geeignet. Für große Netzwerke führt *Polling* zu regelmäßigen *SNMP Messages* und ergibt nicht akzeptable Antwortzeiten für die Problemermittlung.

SNMP ist ineffizient und für die Erteilung großerer Datenmengen von den *Managed Objects* zu langsam (z.B. das Abrufen der Routingtabelle eines Routers).

SNMP Traps (Event Botschaften) sind unbestätigt. Sie haben bei der Übertragung über *UDP/IP* keine Zustellungsgarantie. Ein *Managed Device*, der eine *Link Down Trap* schickt kann nie sicher sein, ob die Botschaft das Zielmanagementsystem erreicht hat.

Da die gegenwährtigen *SNMP* Ausführungen nur mit einer trivialen Authentizität ausgestattet sind, ist ihr Einsatz in Kontrollaufgaben nützlicher als in Überwachungsaufgaben.

SNMP sorgt nicht für die Definition und Ausführung eines produktspezifischen imperativ Befehls (z.B. *"perform self test"*).

Das *SNMP Management Informations Modell* ist beschränkt. Die meisten gegenwärtigen Implementierungen sind nicht mit Anwendungen ausgestattet, die Netzwerkverwaltern hochentwickelte Fragen basierend auf Objektwerte und Objekttypen erlauben.

SNMP bietet keine Lösung zum *Management-to-Management* Problem. Es gibt keinen Mechanismus, der einem *Management System* erlaubt, näheres über das von einem anderen Managementsystem bearbeitete Netzwerk in Erfahrung zu bringen (siehe 2.3.2.1. Systems Management Functions).

OSI Netzwerk-Management Spezifikationen betreffen die oben erwähnten *SNMP* Beschränkungen. Die Implementierung eines OSI Netzwerk-Managements bietet auch das Potential an, eine breite Palette von integrierten Managementservices zur Verfügung zu stellen, die bereits vorhandenen *OSI Applications*, wie *Delivery Service (X.500)*, *File Transfer (FTAM)* und *Virtual Terminal* umfassen.
Die auf *CMOT* und *SNMP* basierenden Management- Transaktionen können gegenwärtig nur zwischen zwei Systemen stattfinden (ein einziger Manager kommuniziert mit einem einzigen Agenten). Die Implementierung von *CMOT* erlaubt es aber einem, zukünftige *SI Applications*, wie *Commitment, Concurrency und Recovery (CCR)*, *Transaction Processing (TP)* und *Remote Database Access(RDA)* zu nützen, die ein Konzept für das Management von *Multicasting und Distributed* Umgebungen anbieten.

CMOT ist trotz dieser potentielle Vorteile nur von wenigen implementiert worden. Einige Gründe dafür sind:

Die *CMIP* und *CMIS* Konzeptspezifikationen sind während der letzten Jahre Gegenstände vieler, detaillierte Veränderungen gewesen und haben sich kürzlich zum IS-Stadium entwickelt.

Bezügliche Dokumente, jene, die *OSI Management Framework* umfassen, sind noch immer nicht vollständig und sind derzeit Veränderungen unterworfen. Daraus resultierend ist es schwer, OSI bearbeitete Objektdefinitionen in den Produkten zu implementieren.

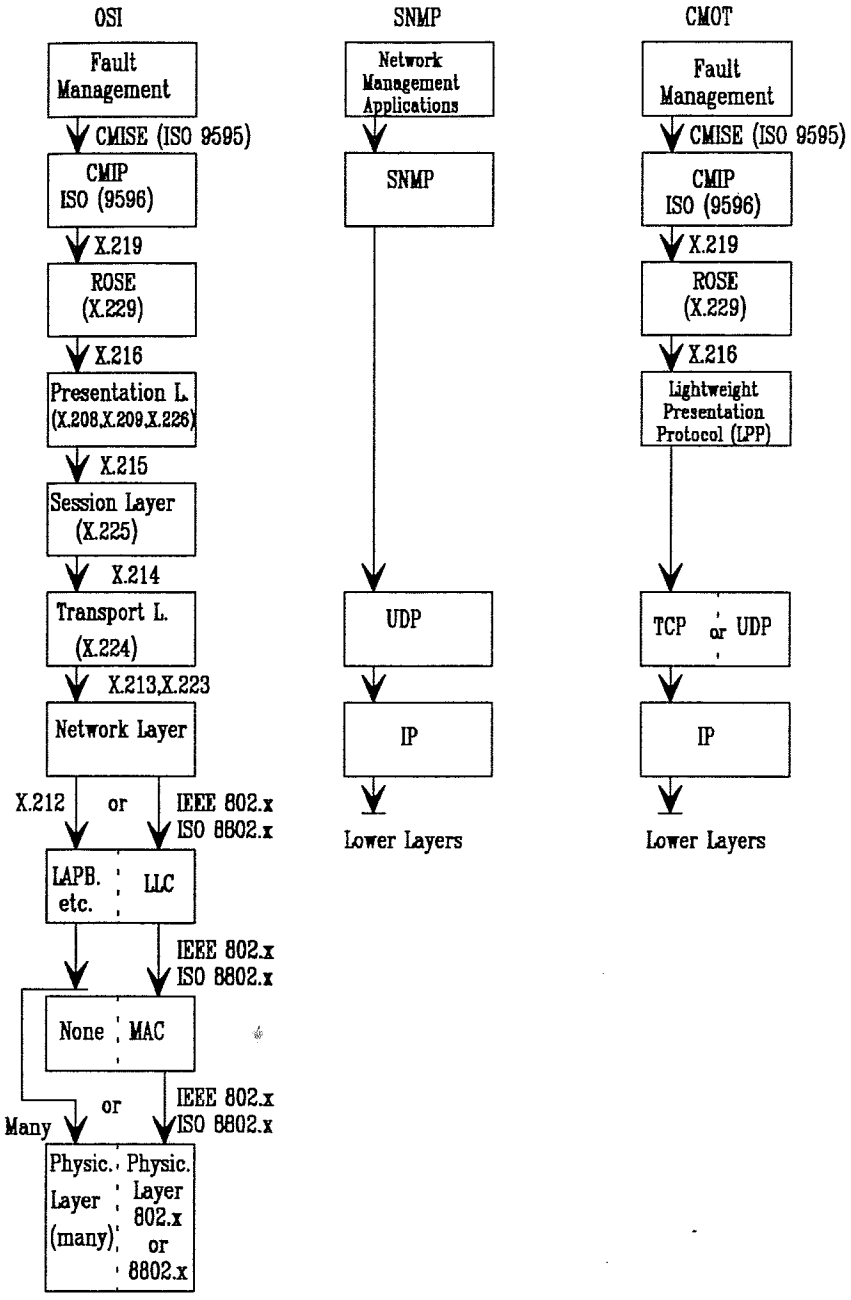
CMOT Implementierungen benötigen signifikante Speicher- und Verarbeitungsressourcen. Das heißt, daß für viele Produkte die *CMOT* Implementierung oft den Neuentwurf von vorhandenen Produkten mit sich bringt.

Das zukünftige *TCP/IP Network Management* ist nicht mehr als das Management von Änderungen, eine Vielfalt von Herstellern, Technologien, Interfaces und Versionen einbeziehend. NM-Protokolle, wie *SNMP* und *CMOT*, stellen nur einen kleinen, aber entscheidenden Teil des für das integrierte *TCP/IP* Netzwerk-

Management benötigte Rohrwerk dar. Es sollen noch viele Mechanismen, wie *Remote Login*, User- und Workinggroupverwaltung usw. standardisiert und implementiert werden. Andere, auf User Interfaces, Expertensystemen und objektorientierten Databases bezogene Technologien sind auch der Schlüssel zur Grundsteinlegung zukünftiger *TCP/IP Network Management Applications*.

Aufbauend auf die vorherige Einführung in das *OSI, SNMP und CMOT* Managementkonzept sind in Abb. 2.12 die Protokolstacks dieser drei Protokolle abgebildet:

Von dieser Abbildung is es einfach zu sehen, wie komplex der OSI-Protokolstack im Vergleich zu SNMP-Protokolstack ist. Die Einfachheit des SNMP-Protokolstacks hat dazu geführt, daß SNMP-Implementierungen weniger Speicherplatz benötigen und außerdem, für die Durchführung von Managementoperationen, weniger Zeit brauchen.



Vergleich von OSI-, SNMP- und CMOT-Schichten

Abb. 2.12) Vergleich von OSI- , SNMP- und CMOT-Schichten

3. Advanced Orbiting Systems (AOS)

3.1. Einleitung

In den späten Siebzigern wurde besondere Aufmerksamkeit auf Datenverarbeitungsprobleme in Weltraummissionen gelegt. Ein Hauptgrund für diese Anstrengungen war das Fehlen von Normen für die Weltraumdatenkommunikation. Jedes Raumschiff verwendete seine eigene Daten- bzw. Protokolformate, und infolgedessen mussten für jedes Projekt neue Systemkomponenten entworfen und bedient werden.

Weiters haben Missionen immer die Kommunikations-verbindungen in Grenzen gehalten und die Fehlerrate bei der Datenübertragung war ziemlich groß. Diese Faktoren machten die Automatisierung unmöglich und bewirkten, daß die Missionskosten durch personal-intensive Operationen gestiegen sind und daß die Zuverlässigkeit darunter gelitten hat.

Als Lösung für diese Probleme wurde das Konzept der *Packet-Telemetry* entwickelt. Entsprechend dem OSI Schichten-Modell sollten folgende Zielen erreicht werden:

Es sollte ein *high-performance Space Data Link* Kodierungssystem entwickelt werden, das eine saubere Behandlung der Daten im Bodennetzwerk erlaubt. Es sollte ein *standard data link framing Protokol* entwickelt werden, daß ein gemeinsames Interface zwischen Raumfahrzeug und Bodensystem darstellt. Die Onboard-instrumenten und -payloads sollten in ihrer eigenen Kommunikationsschicht gestellt werden, und durch die Möglichkeit, daß sie synchrone Pakete bilden können, die einem Standardformat entsprechen, sind sie von der *TDM*-induzierter Anforderung eines synchronen Datentransfers befreit. Die Standardpaketen werden zu Standardrahmen übergeben, die Rahmen werden durch ein Standardkodierungsschema geschützt und ein Standardbodensystem wird dem Prozeß umkehren um die Daten den Benützern zu präsentieren.

Die Realisierung und Durchführung obengenannter Konzepte wurde durch die revolutionären Fortschritten in der Microprocessortechnik der letzten zwanzig Jahren und der Einfluß dieser auf den Entwurf von Raumfahrzeugen vereinfacht.

3.2. Standardisierung von Datenkommunikations-techniken

Zwei Systeme, gleichgültig wie verschieden sie sind, können miteinander wirksam kommunizieren, wenn sie folgende Gemeinsamkeiten erfüllen:

- Sie führen dasselbe Set von Kommunikationsfunktionen aus.

- Diese Funktionen sind in demselben Set von Schichten organisiert.
- Partnerschichten müssen dieselben Services erbringen (es ist nicht notwendig, daß sie diese Leistungen in derselben Art und Weise erbringen) und ein gemeinsames Protokol teilen.

Um das obengenannte sicherzustellen, werden Normen gebraucht. Normen müssen die Funktionen und Leistungen definieren, die von einer Schicht erbracht werden. Normen müssen auch die Protokole zwischen den Partnerschichten definieren. Jedes Protokol muß für die beiden Partnerschichten identisch sein. Um eine Reihe von Protokollen zu standardisieren, muß eine gemeinsame oder eine normierte Architektur verwendet werden.

Das *Consultative Committee for Space Data Systems (CCSDS)* ist eine internationale Organisation, die von Experten von allen wichtigen Weltraumbehörden der Welt besetzt ist. Ihr Ziel ist die Entwicklung von Standarddatenkommunikationstechniken, sodaß die Gelegenheit für *interagency cross support* geschaffen und somit die Durchführung von komplexen internationalen Missionen möglich ist.

Eine Untergruppe von *CCSDS* wurde im Jahre 1985 gebildet, um Datenkommunikationskonzepte für bemannte Missionen und große unbemannte Raumfahrzeuge zu entwickeln.

Zukünftige Missionen, die von den sogenannten *Advanced Orbiting Systems (AOS)* durchgeführt werden, sind durch ihren Bedarf an Austausch von großen Datenvolumina zwischen Raumfahrzeug und Bodenstation, ihre große und wechselnde Benützergemeinschaft, ihre Anforderung, menschliche Stimme und Video mit der Übertragung von Telemetry- und Telecommanddaten (siehe folgenden Absatz) zu vereinigen, ihre diverse Leistungsanforderungen und ihre spezielle Netzwerktopologie, oder besser gesagt, ihre adaptierbare Netzwerktopologie (*Payload*) charakterisiert.

Unter *Telemetry Data* versteht man große Datenvolumina, die vom Raumfahrzeug an Bodenstation gesendet wird (z.B. Satellitenbilder, die für Wettervorhersage verwendet werden). Unter *Telecommand Data* versteht man die Menge der *Commands* (Befehle), die von der Bodenstation an Raumfahrzeug gesendet wird.

Die Defintion der AOS-Protokolle ist in Form einer *Architectural Specification,* die AOS Blue Book [AOS89-4] genannt wird, genauer beschrieben.

Die Produkte von *CCSDS* werden in Form von Empfehlungen herausgegeben, die selbst keine Normen sind. Sie Stellen die Übereinstimmung von technischen Empfehlungen bzw. Vorschlägen

der Panelexperten hinsichtlich wie jede Behörde ein bestimmtes Weltraum-systeminterface normieren sollte.

Normenkommissionen innerhalb jeder Behörde sind für die Entwicklung und das Erzwingen der Lokalnormen verantwortlich, die mit den *CCSDS Recommendations* kompatible sein müssen.

Die ersten Serien von *CCSDS Recommendations* waren in erster Linie auf Normierung von Weltraum/Bodendatenübertragung für konventionelle und unbemannte wissenschaftliche Raumschiffe konzentriert, die mit gemäßigten Datenraten operieren und einer relativ statischen Benützergruppe dienen. Mit dem Blick auf die *Advanced Orbiting Systems* der Zukunft, wie erdbeobachtende Plattformen, bemannte Weltraumstationen und neue Weltraumtransportationssysteme gewiint die *CCSDS* zunehmend an Bedeutung.

Zukünftige Missionen werden weitverbreitet internationalisiert sein und können einfach ohne eine hohe Stufe der Normierung nicht ausgeführt werden. Sie werden eine ungleiche, wechselnde Benützergruppe versorgen und in einem Zeitrahm fliegen, wenn weltweit OSI seinen würdigen akzeptablen Platz innerhalb irdischer Netzwerke eingenommen hat.

3.3. AOS Grundsätze

Die Grundsätze von *AOS* Weltraum/Boden-Datenkommunikations-architektur sind:

- *high-performance symmetric-forward/backward-link-layer* Protokolle;
- Anwendung von *State-of-the-Art* Kodierungstechniken um fehlerfreie Datenübertragung zu ermöglichen;
- gleichzeitige Unterstützung von mehreren Benutzer-Dienstleistungen, unter anderem spezielle Techniken um den *high-data-rate* Übertragungsanforderungen von Erdsensoren und den strengen Isochronitäts-anforderungen von Audio- und Videodaten entsprechen zu können;
- und Bestimmungen für den asynchronen Datenaustausch in Netzwerkschichten.

3.4. Das CCSDS Principal Network Service Model

Die *AOS-Architektur* modelliert die bidirektionalle Interkommunikation von Daten zwischen Raumschiffen und ihren unterstützenden Bodennetzwerken in Sinne eines abstrakten *"CCSDS Principal Network"*. Das *CPN* enthält drei Subnetzwerke: ein *"Onboard Network"* in einem Weltraumvehikel verbunden via ein *CCSDS "Space Link Subnetwork"* entweder zu einem *"Ground*

Network" oder zu einem Onboardnetzwerk in einem anderen Weltraumfahrzeug.

Das *SLS* ist das zentrale Komponent eines *CPN* und ist dort, wo *CCSDS* den vollen Protokolstack festsetzt. *CCSDS* bestimmt nur die Mechanismen, um die Onboard- und Ground-Netzwerke, deren Implementierung von den kommerziellen Applikationen abhängig ist, bei der Network-Schicht oder darüber zu durchqueren.

Innerhalb *SLS* sucht *CCSDS* nach Möglichkeiten für die Realisierung von *"inter-agency cross support"*. Sie ist definiert als die Möglichkeit einer Agency, die Daten einer anderen Agency bidirektional zwischen Boden- und Weltraumsystemen mittels ihrer eigenen Betriebsmittel zu übertragen.

Das CPN-Modell ist in Abb. 3.1 veranschaulicht. Das Modell ist in Übereinstimmung mit dem bekannten Siebenschichtenmodel von OSI und ist in beiden Richtungen der Weltraum/Boden-kommunikationsschnittstellen symmetrisch aufgebaut. Die Datenkommunikationsprotokolle sind so entworfen worden, um bidirektional zu operieren (z.B. vom Weltraum zum Boden, vom Boden zum Weltraum oder vom Weltraum zum Weltraum).

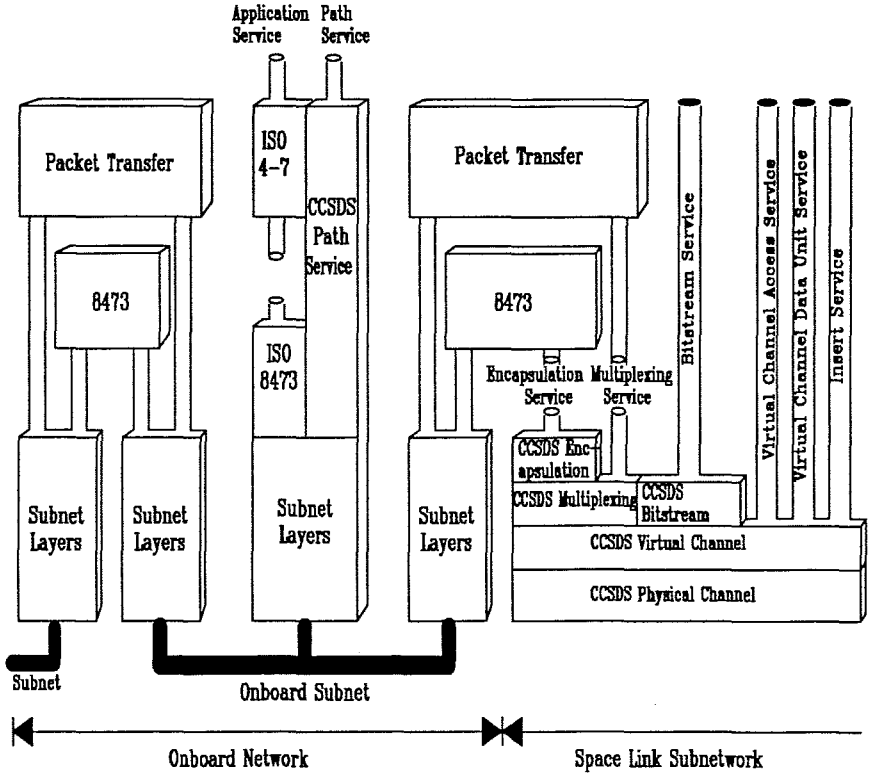


Abb. 3.1) CPN Model und CCSDS Services

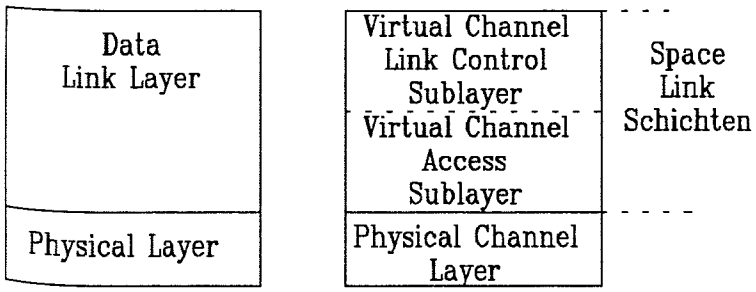
3.4.1. CCSDS Spezifikationen

Die CCSDS spezifiziert jene Protokolle, die für die Datenübertragung über das CPN (End-to-End) verwendet werden müssen, und diejenigen, die für die Übertragung über das SLS zu verwenden sind. Sie spezifiziert aber nicht, welche Protokolle innerhalb der Bodennetzes bzw. des Onboardnetzes zu verwenden sind. Die CPN Protokolle sind alle bidirektional. (obwohl die CPN aus zwei symetrischen Hälften besteht, muß die physische Implementierung der beiden Hälfte nicht identisch sein).

Da die *Physical* und *Data Link Layer* des Onboard- bzw. des Ground-Netzwerkes je nach Bedarf unterschiedliche Implementierungen aufweisen, werden die Protokolle dieser zwei Schichten nicht von CCSDS spezifiziert. Da aber die Benützung von kompatiblen Schicht-2 Protokollen, das Anschließen mehrere Onboard-Netzwerke (, die von verschiedenen *CCSDS Agancies* betrieben werden) stark vereinfachen kann, wird von CCSDS folgendes vorgeschlagen:

- Der Data Link Layer der Onboard-Netzwerke muß gemäß ISO 8802.2 LAN LLC Standard implementiert werden.
- Innerhalb des LLC Sublayer, die Class II ist bevorzugt (Siehe Seite 11 in [AOS89-3]).
- Die durch MAC Layer spezifizierte Adresse muß 48 Bit long sein (Anpassung an ISO 8802.X).

Das CCSDS Main Recommendation definiert jedoch, die Data Link Layer und die Physical Layer Protokolle von SLS. Eine optimale Ausnützung des Space Channel war das wichtigste Ziel bei der Entwicklung obengenannter Protokolle.



Die Schichten des Space Link Subnetwork
Abb. 3.2) Die SLS-Schichten

Die Abb. 3.2 veranschaulicht die SLS-Schichten und ihre äquivalente OSI-Schichten. Die von CCSDS definierte und zu (OSI-) Data Link Layer äquivalente Schicht heißt Space Link Layer und besteht aus zwei Unterschichten; einem CCSDS Virtual Channel Link Control (VCLC) Sublayer und einem CCSDS Virtual Channel Access (VCA) Sublayer. Das VCLC Sublayer generiert zwei unterschiedlichen PDUs, die für die Übertragung über das Physical Channel Verwendung finden, nämlich das Multiplexing Protocol Data Unit genannt M_PDU, und das Bitstream Protocol Data Unit, genannt B_PDU (siehe Abb. 3.3).

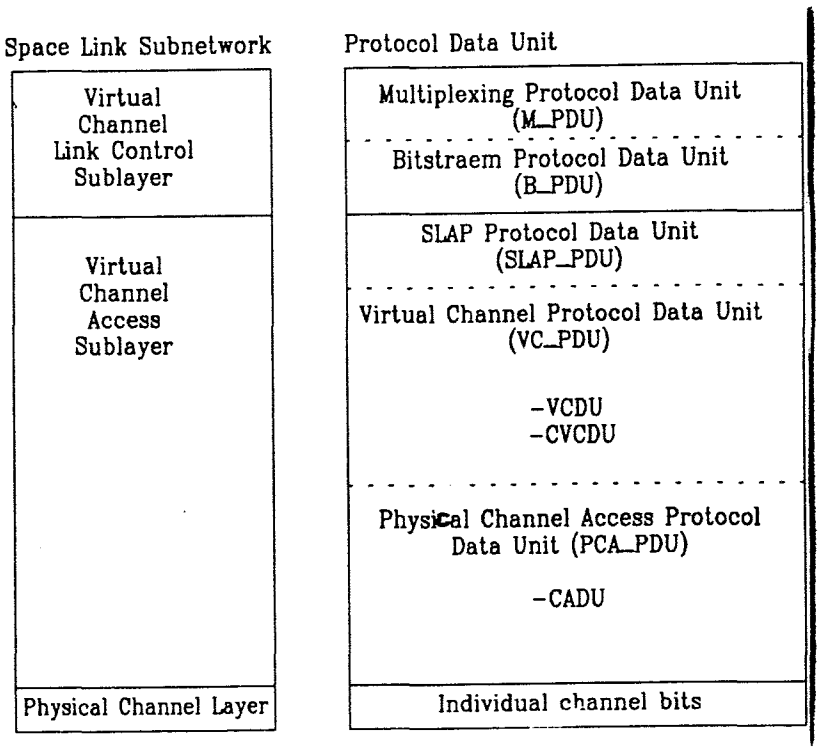


Abb. 3.3) Die SLS-Schichten und Protokolle

Um Daten über das Space Link zu versenden, werden Virtual Channel Protocol Data Units (VC_PDUs) von dem VCA Sublayer generiert. Dafür werden entweder Virtual Channel Data Units (VCDUs) oder Coded Virtual Channel Data Units (CVCDUs sind VCDUs mit Prüf- bzw. CRC-Feldern) verwendet (s. Abb. 3.4.b). Das VCA Sublayer unterstützt auch eine isochrone Datenübertragung (d.h. bei niedriger Bitübertragungsrate werden kleine Datenblöcke in VCDUs bzw. in CVCDUs eingesteckt

Die Abb. 3.2 veranschaulicht die SLS-Schichten und ihre äquivalente OSI-Schichten. Die von CCSDS definierte und zu (OSI-) Data Link Layer äquivalente Schicht heißt Space Link Layer und besteht aus zwei Unterschichten; einem CCSDS Virtual Channel Link Control (VCLC) Sublayer und einem CCSDS Virtual Channel Access (VCA) Sublayer. Das VCLC Sublayer generiert zwei unterschiedlichen PDUs, die für die Übertragung über das Physical Channel Verwendung finden, nämlich das Multiplexing Protocol Data Unit genannt M_PDU, und das Bitstream Protocol Data Unit, genannt B_PDU (siehe Abb. 3.3).

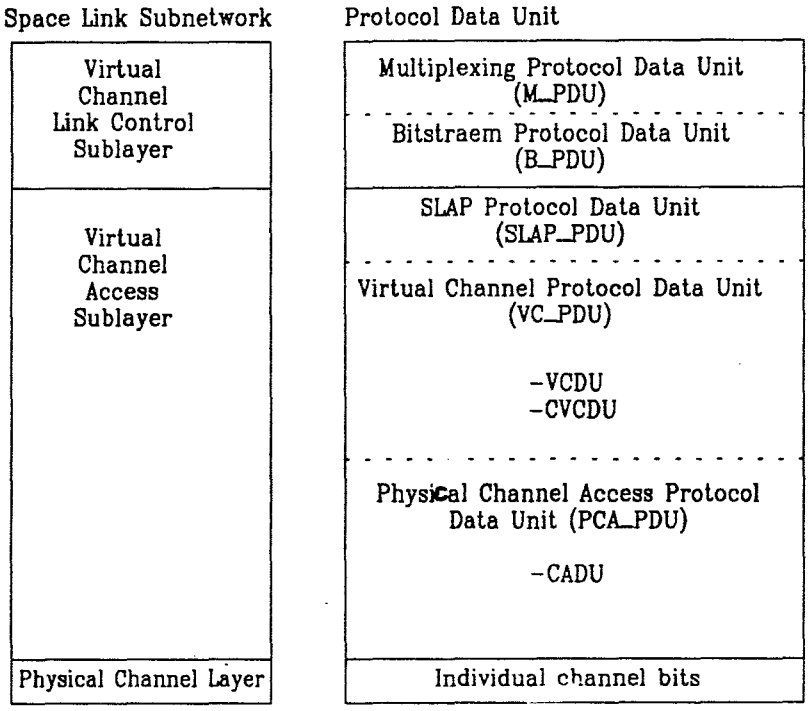
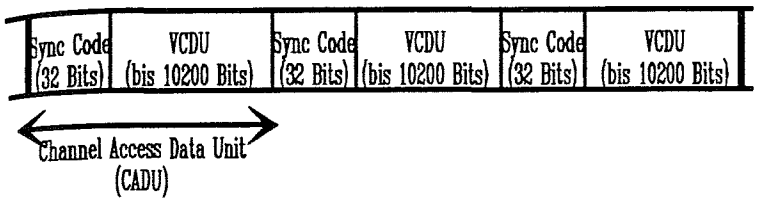


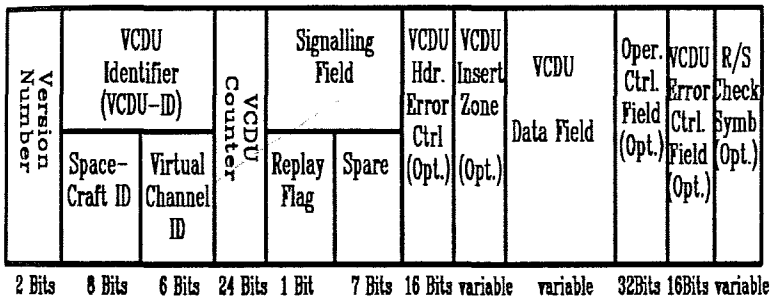
Abb. 3.3) Die SLS-Schichten und Protokolle

Um Daten über das Space Link zu versenden, werden Virtual Channel Protocol Data Units (VC_PDUs) von dem VCA Sublayer generiert. Dafür werden entweder Virtual Channel Data Units (VCDUs) oder Coded Virtual Channel Data Units (CVCDUs sind VCDUs mit Prüf- bzw. CRC-Feldern) verwendet (s. Abb. 3.4.b). Das VCA Sublayer unterstützt auch eine isochrone Datenübertragung (d.h. bei niedriger Bitübertragungsrate werden kleine Datenblöcke in VCDUs bzw. in CVCDUs eingesteckt

und damit ihre isochrone Übertragung sichergestellt) und eine *retransmission control procedure (genannt die Space Link ARQ Procedure)*. Danach werden die VCDUs und CVCDUs um *Synchronization Marker* erweitert und formen damit die Channel Access Data Units (CADUs). Der ununterbrochene Fluß der CADUs, wird dann als Physical Channel Access Protocol Data Unit (PCA_PDU) über das zu OSI-Physical Layer äquivalente Schicht des SLS Subnetzes, nämlich das Physical Channel Layer übertragen.



a) LINK-STRUKTUR



b) VCDU-STRUKTUR

Abb. 3.4) a. PCA-PDU b. VCDU

Die CCSDS Services werden in zwei Kategorien untergeteilt: Space_Link_Services und End_to_End Services. Die Space_Link_Services sind nur von dem Vorhandensein der Space_Link_Subnetwork Schichten abhangig; die End_to_End Services von dem der Internetwork und End_to_End Schichten.

Die CCSDS spezifiziert jene Protokolle, die für die Datenübertragung über das CPN (End-to-End) verwendet werden müssen, und diejenigen, die für die Übertragung über das SLS zu verwenden sind. Sie spezifiziert aber nicht, welche Protokolle innerhalb der Bodennetzes bzw. des Onboardnetzes zu verwenden sind. Die CPN Protokolle sind alle bidirektional. (obwohl die CPN aus zwei symetrischen Hälften besteht, muß die physische Implementierung der beiden Hälfte nicht identisch sein).

Acht verschiedene Benützerservices werden erbracht. Zwei von diesen Leistungen, *Internet* und *Path*, operieren bei oder über der Networkschicht, um die Kommunikation *end-to-end* über das gesamte *CPN* zu ermöglichen. Die restlichen sechs Services (*Encapsulation-,Multiplexing-,Bitstream-,Virtual Channel Access-,Virtual Channel Data Unit- und Insertservice*) sind nur innerhalb von *SLS* für spezielle *point-to-point* Anwendungen, wie *Audio, Video, high-rate Payloads, tape playback* und die zwischenliegende Übertragung von Path- und Internetdaten vorgesehen.

3.4.1.1. End-to-End Services und Protokolle

Die CCSDS definiert zwei End_to_End Services: das Path Service, das Internet und End_to_End Protokolle und Services standardisiert, und das Internet Service, das nur Internetwork Protokolle und Services standardisiert.

Das Path Service basiert auf CCSDS Paket. Es gibt zwei Gründe dafür:

1- Um Kompatibilität mit den existierenden (konventionellen) CCSDS Paket Telemetry und Telecommand Standards zu garantieren.

2- Um dasselbe Paket in mehreren verschiedenen Schichten des Kommunikationspfades verwenden zu konnen (erhöht die Effizienz).

Die Path- und Internetservices ermöglichen beide verschiedenen Anforderungen an end-to-end Userservices gerecht zu werden.

Das Internetservice wird für spontanes Übertragen, bei relativ niedrigen Datenraten , niedrig bis mittelmäßigen Mengen von strukturierten und abgegrenzten Daten zwischen einer Source und einer Destination, wo der Standort der Endpunkte sich oft ändern könnte, verwendet (*maximum flexibilility for intaractive applications at the cost of speed and efficiency*).

Die CPN End-to-End Protokolle, werden in Abb. vorgestellt und ihre Eigenschaften kurz zusammengefßt.

Das Internetservice leitet sich aus der ISO-8473 Paketstruktur. Es ermöglicht damit den Weltraummissionsbenützern den Zugang zu vollen Stapeln von Leistungen der höheren Schichten der OSI-Struktur. Es ist ein Serviece, das für viele dynamische, interaktive Aktivitäten, die sprunghaft operieren nützlich sein könnte.

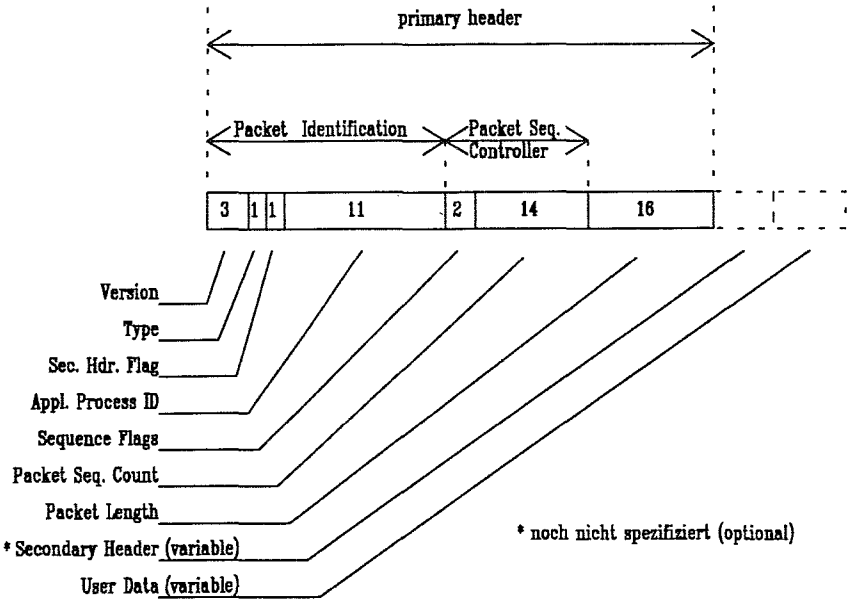
	octet-Nummer
Network Protocol Identifier	1
Length Indicator	2
Version/Protocol ID Extension	3
Lifetime	4
SP | MS | E/R | Type	5
Segment Length	6-7
Checksum	8-9
Dest. Adrs. Length Indicator	10
Destination Address:	11-24
Authority & Format ID	11
International Code Designer	12-13
Organisation Identifier	14-15
Subnet Identifier	16-17
End System Identifier	18-23
NSAP Selector	24
Src. Adrs. Length Indicator	25
Source Address:	26-29
Authority & Format ID	26
International Code Designer	27-28
Organisation Identifier	29-30
Subnet Identifier	31-32
End System Identifier	33-38
NSAP Selector	39
Data Unit Identifier	40-41
Segment Offset	42-43
Totsl Length	44-45
User Data	n-m

Aufbau eines ISO 8473 bzw. Internet Packet

Abb. 3.5) Aufbau eines ISO 8473 bzw. eines Internet Pakets

Im Gegensatz dazu, wird das Pathservice für ununterbrochenes Übertragen großer Volumina von strukturierten, abgegrenzten Dateneinheiten zwischen den statischen Source- und Destinationendpunkten verwendet. Pathpaketen sind in einem sehr einfachen, CCSDS-Format Paket codiert, um höhere Geschwindigkeit, Kommunikationseffizienz und einen besseren Durchsatz zu ermöglichen (very high processing speed and efficiency, at the cost of some fexibility). Das Path Service wird für Anwendungen wie Telemetrytransfer verwendet.

Da dieses Service auch für die Übertragung von NM-Information zwischen OBIF Management Entities (siehe Kapitel 5) zuständig ist, wird es im Vergleich zu anderen Services detailierter besprochen.



Aufbau eines VERSION-1 CCSDS Packet

Abb 3.6) Struktur eines Path Pakets

Das AOS Pathservice und -protokol stehen auch den Benützern der konventionellen Missionssysteme zur Verfügung. Das Hinzufügen von der Internetservice bzw. von den Internetfunktionen vergrößert den früheren Packet Telemetry Spielraum, sodaß die leistungsreiche Infrastruktur von OSI sich in den Weltraum vertreiben kann.

	Path	*Internet*
Benützer	Path Service	Internet Service
Packet Bez.	CCSDS Version-1	ISO 8473
Datenart	telemetry	interactive
Verbindungsart	verbindungsorientiert	verbindungslos
Flexibilität, Durchsatz, Fehlerrate	sehr höhe Durchsatz sehr niedrige Fehlerrate niedrige flexibilität	höchste Flexibilität niedriger Durchsatz höhe Fehlerrate
OSI Schicht	Network Layer	Network Layer
Schnittstellen	oben: User Application unten: Data Link Layer	oben: Transport Layer unten: Data Link Layer

CCSDS End-to-End Protokolle: Path und Internet

Abb. 3.7) CCSDS End-to-End Protokolle im Vergleich zueinander

3.4.1.2. Space Link Services und Protokolle

Ein Grundmerkmal von SLS ist die Einrichtung von *Virtual Channels*, die das Multiplexing mehrere "Virtueller Kanäle" über einem physikalischen Kanal erlauben. Jeder "Virtuelle Kanal" kann unterschiedliche Serviceanforderungen erfüllen.

Virtuelle Kanaldateneinheiten (*virtual channel data units*) mit fixen Längen gestatten dieses Linklayerprotokol. Ein high-performance Reed_Solomon Blockcode kann von den linklayer PDUs angewendet werden, um eine möglichst fehlerfreie Datenübertragung über den Spacelink zu verschaffen. Der ausgewählte Standard Reed-Solomon Code (welcher auch bei den konventionellen CCSDS-systemen verwendet wird) ist der Eckstein der neuen Generation von Weltraummissions-datenhandhabungs-systemen. Seine Leistung ist so hoch, daß es eigentlich virtuelle fehlerfreie Datenübertragungen auch noch von den entferntesten Bereichen des Sonnensystems ermöglicht. Durch die Versorgung unverfälschter Datenübertragung wird Onboard-datenverdichtung und -datenvorverarbeitung möglich und dadurch, daß die Quelle die Datenmengen autonom zusammenstellen und etikieren kann, wird asynchrones Datasampling möglich. Außerdem, da den hereinkommenden Datenströmen vertraut werden können, ist die Automatisierung der Datenhandhabungsnetzwerke erleichtert.

Innerhalb von SLS unterstützen zwei Services (Dienstleistungen) die asynchrone Datenübertragung. Das Multiplexingservice verwendet ein spezifisches Subnetwork Protocol, welches die gleichzeitige und effiziente Übertragung von octet-aligned Pakete variabler Länge mehrerer Benützer durch einen "Space Data Link", auf demselben Virtualkanal erlaubt. Dieses Service erlaubt variabel lange asynchrone CCSDS Pakete in fixer Länge, synchronen VCDUs zu verknüpfen bzw. zu verketten (siene Abb. 3.8).

M-PDU Header		Packet Zone				
Spare (5 Bits)	First Header Pointer (11 Bits)	End of Previous CCSDS Packet	CCSDS Packet		CCSDS Packet	Start of CCSDS Packet

Multiplexing Data Unit

Abb. 3.8) Multiplexing data Unit

Das Encapsulationservice erlaubt, die Übertragung von octet-aligned variabler Länge UDUs (User Data Units) , z.B. das ISO-8473 Paket des Internetservices (Abb. 3.5), in einem spezifischen Standard Local Subnetwork Protocol (ein CCSDS Paket).

Die Umformung der Internetpaketen in CCSDS-Format ermöglicht daß Path- und Internetpakete dasselbe Multiplexingservice verwenden und auf denselben Virtualkanal gesendet werden (siehe Abbildung 3.1).

Die restlichen vier SLS-Services dienen hauptsächlich der synchronen Datenübertragung. Sie sind für das AOS-Environment entwickelt worden. Da es keine Gegenstücke zu diesen Services in den konventionellen Systemen gibt, ist die künftige Erweiterung dieser Systeme um diese Services unkompliziert. Bei diesen Services geht es im Prinzip um *Multimedia-* und *Broadband Networking*.

Das Bitstreamservice erlaubt einen Strom von Bits, dessen innere Struktur und Abgrenzungen nur den beiden Applikationen auf Sende- und Empfangsseite bekannt sind, durch ein Space Data Link (SDL) auf einen dafür bestimmten Virtual Channel zu übertragen (Abb. 3.9). Es kann auch verwendet werden, um playback Daten eines Onboardrecorders oder einen Strom von verschlüsselter Symbole zu übertragen.

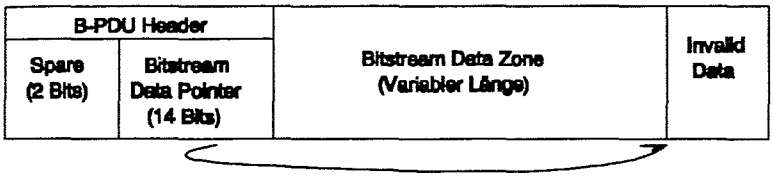


Abb. 3.9) Bitstream Data Unit

Das Virtual Channel Access Service (VCA) erlaubt einem Benützer, einen Virtualkanal zu addressieren, um point-to-point Übertragung von Datenblöcken fixer Länge durch ein SDL zu ermöglichen. Es könnte z.B für den Transfer von Video- und Audiodaten verwendet werden.

Das Virtual Channel Data Unit Service (VCDU) erlaubt es vertrauenswürdigen Anwendern, ihre eigenen VCDUs für die Übertragung durch SDL frei zu gestalten. Es kommt zum Beispiel zur Anwendung, wenn der Datenstrom (VCDUs) eines Raumfahrzeuges in der eines anderen integriert werden soll.

Das Insertservice erlaubt, octet-aligned Dateneinheiten fixer Länge, wie Digitalaudiodaten, isochron durch ein SDL zu übertragen, wobei die gesamte Datenübertragungsrate niedrig sein muß.

3.5. Adressierung

In den *CCSDS Standards* hat man versucht, bei der Spezifikation der Adressierungsart, eine möglichst einfache Adressierungs-schema mit hoher Flexibilität darzubieten.

Es sind zwei Adressierungs-Strukturen für einfache Missionen definiert worden:

- *Virtual Channel (VC-id)*, und
- *Path (AP-id)*.

Bei komplizierten Missionen wird das *Spacecraft Id* (im *VCDU Header* in Bild 3.4.b) verwendet, um zwischen verschiedenen Virtual Channels (zur Zeit ist die maximale Anzahl der Virtual Channels auf 2^6 beschränkt) unterscheiden zu konnen.

In *Space Link Subnet* stehen nur zwei Adressierungsmechanismen zur Verfügung [ESA91-2], nämlich

- der *VC-id* und
- der *Spacecraft-id*.

Für die Ground-Systeme bilden der *Spacecraft-id* und der AP-id den sogenannten *Path-id*, damit Pakete verschiedener Missionen miteinander nicht verwechselt werden.

In AOS Recommendations wird für die Adressierung ein eher unklares, noch dazu ein abstraktes Konzept eingeführt. Hier wird einen *AP-id Qualifier* vorgestellt, was die Identifizierung von bis zu 2048 verschiedenen Datenströmen ermöglicht.

Wie es aus den in diesem Kaptiel vorgekommenen Abbildungen zu sehen ist, werden die *Physical Channels* nie explizit spezifiziert. Für die meisten Services ist dies aber irrelevant, da das *Physical Channel* implizit gegeben ist. Diese Unvollkommenheit führt allerdings bei *Insert Service* (Dieses Service wurde eingeführt, um die Übertragung von isochronen Daten geringer Bandbereite wie Audio zu ermöglichen. Zu diesem Zweck wird ein Datenbereich fixer Länge im Benützerdatenbereich der VCDUs - siehe Abbildung 3.4.b - reserviert. Dies ermöglicht das periodische Übertragen von Bitblöcken fixer Länge.) zu Schwierigkeiten, weil verschiedene Instanzen eines *Insert Services* nicht mehr voneinander unterschieden werden konnen (Das *AOS Blue Book -issue 1-* (siehe [ESA91-2]) spezifiziert den *Spacecraft-id* als der Adressierungsparameter für das *Insert Service*, was klarerweise nicht ausreichend ist: auf ein bestimmtes Link wird das *Insert Zone* jedes *VCDUs* ausgefüllt und zwar unabhängig von dem Spacecraft-id der *VCDUs*).

Die Weltraumforschung ist komplex und teuer. Die Information- und Kommunikationssysteme zukünftiger Raumstationen werden eine diverse und geographisch verteilte Gemeinschaft mit unterschiedlichen Kommunikationsleistungen unterstützen müssen. Das unterscheidet sich wesentlich von den Anforderungen der

früheren und gegenwärtigen Weltraummissionen, welche meistens von kurzer Dauer sind, und für eine kleine Zahl von Anwendern, missionsspezifische Leistungen erbringen.

Unter Standardisierung von Datenkommunikations-techniken durch CCSDS und die Durchführung zukünftiger Missionen durch AOS werden Wiederverwendbarkeit und geringere Reconfigurations-aufwand der Resourcen auf einer Seite und die gleichzeitige Unterstützung von mehreren Missionen die gesamte Raumfahrtindustrie finanziell attraktiver machen.

4. Onboard Informatique Test Facility (OBIF)

4.1. Einleitung

OBIF ist ein Prototyp eines Space Link Subnetwork (SLS) für Validierungszwecke und sollte entwickelt werden um folgendes zu unterstützen:

- Entwurf, Entwicklung und Validierung vom on-board Teil des EETF Prototypes(end to end telematics facility);
- die Implementierung, Bewertung und Validierung der onboard und space/ground Kommunikations-standards und -protokolle (z.B. die CCSDS Protokolle und Standards);
- die Entwicklung und Untersuchung von on-board Computerprototypen und ihrer Laufzeit Unterstützungs-systeme (support-systems);
- die Entwicklung und das Testen von Audio-, Video- und Bildverarbeitungssysteme- und Standards (z.B. das ESA video distribution system),
- die Entwicklung und Untersuchung von Netzwerk-schnittstelleneinheiten (diese umfaßt *Distributive Operatating System (DOS) Boards*, *Local Operating System (LOS) Boards*, und *Network Interface Boards (NIB)*);
- die Entwicklung und Untersuchung von NM-Aufgaben in den Space- und Groundsegmenten;
- Entwurf und Untersuchung vom Spacelink-subnetwork (siehe Kapitel 3);
- und die Entwicklung und Validierung von neuen (high speed-) LAN und (Lightweight-) Protokol Technologien.

In seiner endgültigen Form wird OBIF ein wichtiger Teil jener Infrastructure sein, die ein End_to_End Testbed über den Spacelink-subnetz darstellt. Ein wichtiges Aspekt bei der Realisierung vom OBIF war die Verwendung von existierenden Softwaremodulen und die Anförderung, daß die existierenden Einrichtungen in der OBIF-Entwicklung logisch und physisch eingebettet werden.

4.2. System-Architektur

Das OBIF-Rechnersystem besteht aus drei Boards (genannt der Central Interconnection Facility..CIF) und einer Workstation. Die folgende Abbildung veranschaulicht die System-Architectur von OBIF.

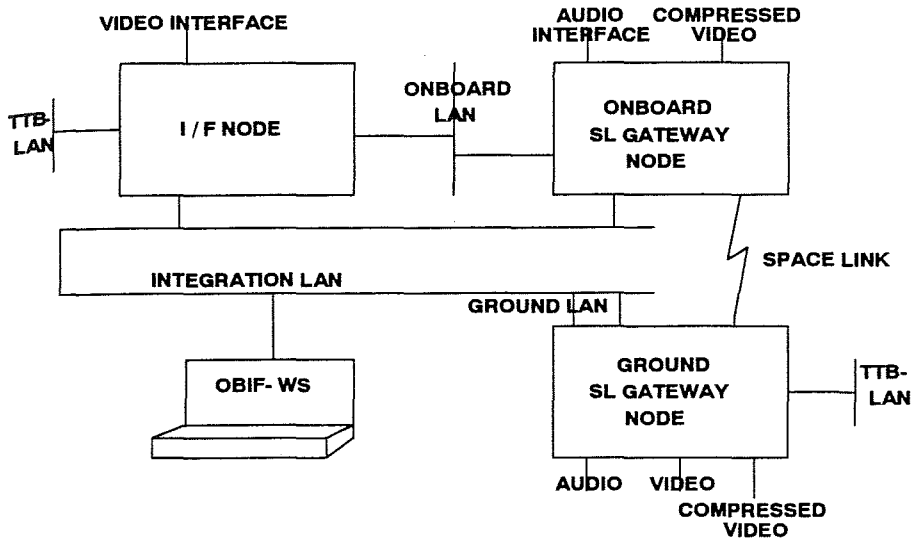


Abb 4.1) System-Architektur von OBIF

Der Onboard-Interface-Node (OIF-Node) und der Onboard-Space-Link-Gateway-Node (OSLGW-Node) representieren die Onboard-Elemente des Systems. Der Ground-Segment wird vom Ground-Space-Link-Gateway-Node (GSLGW) und Workstation-Software representiert. Die Onboard-Nodes sind an einen Ethernet-LAN angeschlossen. Dieser Ethernet-LAN wird OBIF-System-LAN genannt. Die Onboard-Node-Applikationen können die Leistungen der OBIF-Communication-Services in Anspruch nehmen, um miteinander interaktiv zu kommunizieren (siehe Kapitel 4.6.).

Der OIF-Node stellt Schnittstellen zu Telescience-Test-Bed (TTB) und zu einem Videoaufnahmegerät zur Verfügung und benützt die vom OSLGW-Node erbrachten Communication-Services (Leistungen). Der OSLGW-Node verfügt über Schnittstellen zu einem Compressed-Video-Encoder und einem Audio-Subsystem. Der GSLGW-Node besitzt Schnittstellen zu Audio-Subsytem, Compressed-Video-Decoder, TTB-Ethernet-LAN und einem Video-Ausgabegerät. Der GSLGW-Node und die Workstation kommunizieren über den Ground-LAN miteinander. Die OBIF-Workstation (OBIF-WS) stellt die MMI (fürs Testen, Steuern, etc.) zur Verfügung. Die gesamten OBIF-Communication-Services und Applikationen werden von hier aus gemanaged. Das Integration-LAN wird für den Informationsaustausch zwischen dem WS und in den Onboard-Nodes integrierten Test- und Steuermodulen verwendet und simuliert das Space-Link.

4.3. Externe-Schnittstellen

4.3.1. Telescience-Test-Bed-Interface

Diese Schnittstelle (Abb. 4.1) besteht aus zwei Teilen, dem CCSDS-Path-service und dem TTB-Gateway-management. Applikationen, die auf OIF-Node und GSLGW-Node laufen, können den Path-service (über Gateways) end-to-end benutzen.

4.3.2. Video-Interface

Video-Daten werden im OBIF-System von einem an den OIF-Node angeschlossenen Aufnahmegerät zu einem an den GSLGW-Node angeschlossenen Monitor übertragen (Abb. 4.1). Video-Signale werden dafür in/von Packetformaten umgeformt und mit CCSDS-Path-service übertragen.

4.3.3. Compressed-Video-Interface

Die Compressed-Video-Interface (Abb. 4.1) akzeptiert einen isochronen Bitstream vom OSLGW-Node und gibt diesen an GSLGW-Node weiter. Diese Schnittstelle nimmt die Leistungen des CCSDS-Bitstream-services in Anspruch. Da die CCSDS-SLS-protokolle von Natur her asynchron sind, ist Isochronismus garantiert. Die Bitübertragungsrate an diser Schnittstelle beträgt 2,048 Mbps.

4.3.4. Audio-Interface

Diese Schnittstelle (Abb. 4.1) ist voll-duplex. Audio-Daten werden unter Verwendung des CCSDS-VCA-services zwischen GSLGW-Node und OSLGW-Node übertragen. Ein audio-Mutiplexer generiert Datenblöcke fester Länge. Die Länge dieser Blöcke stimmt mit der Länge des Datenfeldes des Space-Link-VCDUs überein (siehe Abb. 3.1). Der OBIF-Operator informiert den Audio-Operator über diese Länge. Ein Demultiplexer bearbeitet die Datenblöcke an der Empfängerseite. Der maximale Jitter zwischen dem Empfangen eines kompletten Datenblockes und dessen Ablieferung beträgt 100ms.

4.4. Hardware-Architektur

4.4.1. Hardware-Konfiguration

Die Hardware-Architektur von OBIF ist in Abb. 4.2. gegeben (vgl. Abb. 4.1). Es besteht aus drei VME-crates und einer SUN-workstation. Alle Komponeten sind an das Integration-LAN angeschlossen.

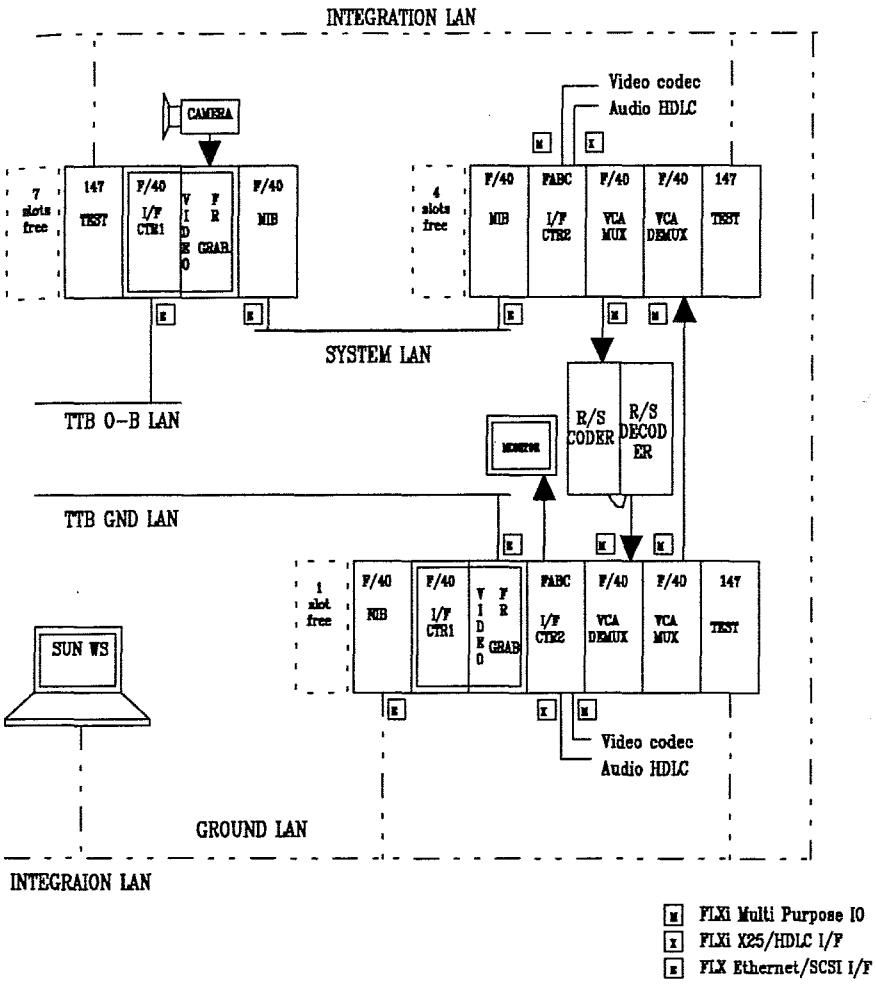


Abb. 4.2) OBIF Hardware-Architektur

4.4.2. Hardware-Komponenten

Für an Hardware-Details interessierte Leser wird im Folgenden ein Überblick über die einzelnen verwendeten VME-Boards und ihre Merkmale gegeben.

Motorola MVME147 Board

Die Hauptmerkmale dieses Boards sind:
- Motorola VME Master/Slave Schnittstelle
- 20 MHz 68030 CPU
- 8 Mbyte DRAM
- 1 Ethernet-port angeschlossen über DMA,
- 1 SCSI Bus-port angeschlossen über DMA

Force Board Family

Als Local-to-VME-Bus-Schnittstelle, benützen alle Force Karten einen FGA-002 ASIC. Dieser ASIC beinhaltet den VME Master/Slave interface, Location Monitors, Lokalbus-arbitration, und einen DMA Kontroller.

Die Merkmale des FLXi Buses sind:

- Relativ einfache Schnittstelle, basiert auf den 68020 - asysnchronen -Bus;
- 32-bit (non multiplexed) Address- und Databus, mit Master/Slave- und Interuptfähigkeit;
- Kompatibilität mit Standardkomponenten der 68000-Familie;
- Externe Uhr bis 50Mhz
- Bandbreite biz zu 30Mbytes pro Sekunde;
- Signal-paths sind klein und kompakt;

Die FORCE CPU-40 und CPU-41 Boards weisen folgende Merkmale auf:

- standard Force VME Interface
- 25 MHz CPU
- 4 MByte DRAM (CPU-40) oder 4 MByte SRAM (CPU-41)
- 1 FLXi Interface

Der FORCE IBC mit den folgenden Merkamalen:

- standard Force VME Interface
- 25 MHz 68020 CPU
- 4 MByte DRAM
- 2 FLXi Interfaces

Dieser Board stellt einen flexiblen front-end-Processor mit konfigurierbaren I/O-Treibern zur Verfügung. Dadurch hält man das VME-Bus-Traffic in Grenzen und benützt weniger VME-Bus-Slots (eine optimale Lösung für die Grouppierung von Audio- und Compressed-Video-(CV-)Schnittstellen).

Das Eagle-1 Module besitzt filgende Merkmale:

- FLXi Interface
- SCSI Port
- Ethernet Port
- 64 KByte Local Interface Memory

Die Platzierung von den 64KByte-SRAM-Puffern an einen entkoppelten Bus reduziert Cycle-stealing. Dadurch kann das Kopieren von SRAM-Puffern von CPU oder DMA unterstützt weden.

Das Eagle-3 Module bietet FLXi-Schnittstellen und besitzt ein MC 68302 IC mit drei Kanälen, die HDLC unterstützen. Dieses Module wird in Audio-Subsystem eingesetzt.

Die Compressed-Video und Read-Solomon-codes werden von dem Multi-Purpose I/O Board (MPIO) erfaßt und vorverarbeitet. Dieser Board besitzt FLXi-Schnittstellen, Serial-parallel-Konvertor, einen FIFO-Kontroller und einen DMA-Kontroller.

Frame Grabber Boards

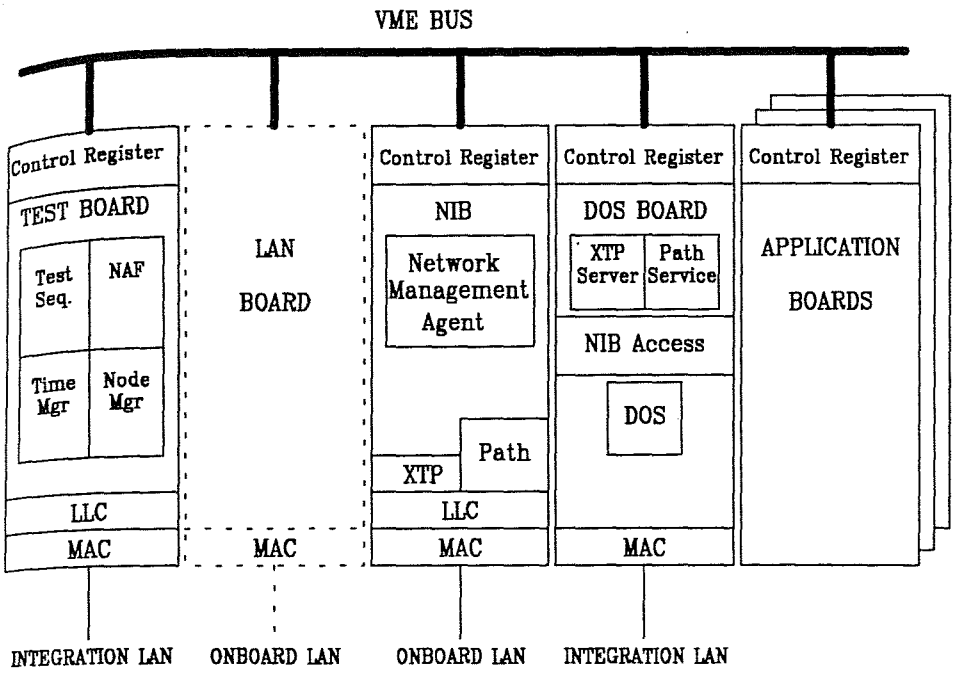
Der DIGIMAX Board (Video-A/D und Video-D/A Umsetzer Board) besitzt einen programmierbaren 8-Bit A/D Umsetzer, der die analogen (RS-170) Videosignale in digitale Signale umsetzt. Der D/A Umsetzer ermöglicht das lokale Erscheinen von Digital-Video.

Der ROI-Store Board (Video Memory Board) erlaubt den simultanen Zugriff auf einen VME-Master, eine Digital-Video-Eingabe und eine Digiatal-Video-Ausgabe. Es representiert ein flexibles programmierbares Module, das dem User die Angabe von verschiedenen (Raum-) Auflösungen (von 64*64 bis zum 512*512 bpi) ermöglicht.

4.5. Software-Architektur

4.5.1. Software-Konfiguration

Die OBIF-Sofware-Elemente sind in den nächsten Abbildungen dargestellt. Es gibt mehrere Boardarten, die alle über Backplane (und Mithilfenahme von Control Register Concept) miteinander kommunizieren.



Allgemeine Architecture eines Onboard-Nodes

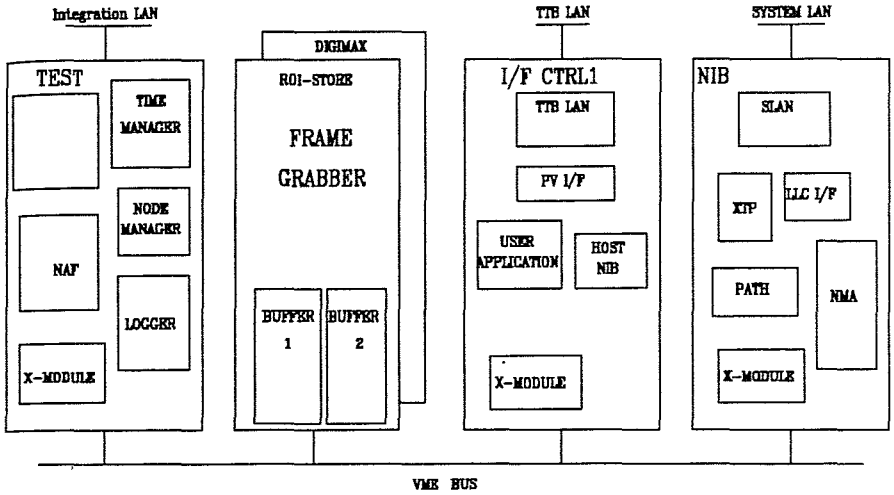
Abb. 4.3) Allgemeine Architektur eines Onboard-Nodes

DOS NIB-Board stellt allgemeine Kommunikations- und LAN-Services zur Verfügung. Er representiert eine LLC-Service-Schnittstelle für die Verwendung von XTP und CCSDS Protokollen. Der Network Management Agent (NMA) tauscht Management-Information mit dem Network Management Station aus (siehe Kapitel 5). Dazu benützt er die Leistungen des CCSDS Path Service.

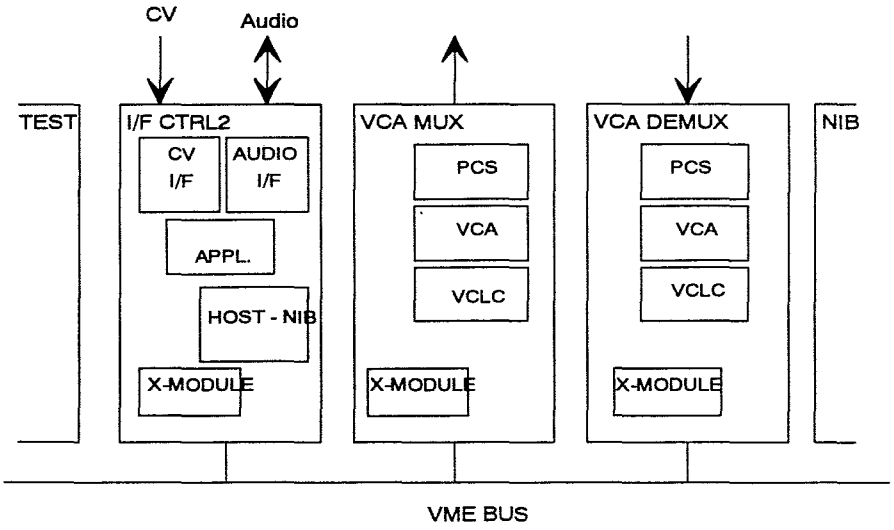
Der Test-board besitzt Ethernet MAC und LLC Software Modules, um auf das Integration-LAN zugreifen zu können. Auf diesem Board sind Applikationen wie die Test Sequence, das Network Assesment Facility (NAF), der Time-Manager und der Node-Manager untergebracht.

Der DOS-Board umfaßt das Distributed Operating System (DOS) samt aller auf dem Knoten notwendigen DOS-Applikationen.

Die Application-Boards sind entweder spezielle anwendungsorientierte Boards oder beinthalten eine VxWORKS-Umgebung für die Ausführung von Applikationen.



a)Test-Board,NIB,Frame Grabber und I/F CTRL1 Boards



b) I/F CTRL2, VCA MUX und VCA DEMUX Boards

Abb 4.4) Architektur einzelener OBIF-Boards

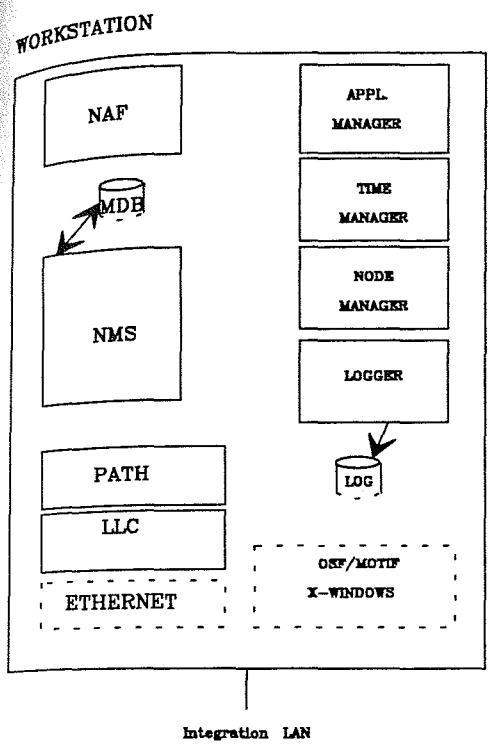


Abb. 4.5) Software-Komponenten in OBIF-WS

4.5.2. Software-Komponenten

Der OBIF-Software besteht aus implementierten generischen Modulen und aus spezifischen Modulen. Die Software-Komponenten von OBIF werden jetzt beschrieben.

4.5.2.1. Die NIU-Architektur

Die OBIF-Nodes sind gekennzeichnet durch folgende Kriterien:

- Hohe Datentransferrate (Video-,Audio- und TTB-Traffic) und kurze Antwortzeit müssen garantiert werden.
- Ressourcenbeschränkungen in Bezug von CPU, Speicher und Bandbreite.
- Das gleichzeitige Vorhandensein von Local-Operating-Systems (LOS) und der damit verbundenen Betriebsmittelteilung und Kommunikation.
- Eine Architekur, die fähig ist, verteilte Betriebssysteme (Distributed Operating Systems..DOS) unterzubringen.

Zugriff auf Communication-Services ist über den Host-NIB (HNIB) Interface möglich. Diese Services umfaßen Zugriff auf den XTP-Transport-Layer, Zugriff auf LLC-type-1-Link-Layer In jedem Host-Node, die Host-NIB-Interface und Zugriff auf das CCSDS-Path_service.

Ein geeignetes Network Interface Board (NIB) stellt Mechanismen zur Verfügung, damit diese Leistungen optimal ausgenützt werden. Der NIB dient dazu, den an Host liegenden Overhead bei der Bearbeitung von Netzwerkprotokollen so weit wie möglich zu reduzieren.

Die Anforderungen an die Host-NIB-Schnittstelle sind groß; in der Folge sind einige aufgelistet:

- Mit einem Minimum an memory-to-memory Kopieren auszukommen,
- Host-Load und backplane-traffic zu minimalisieren,
- Latenz-Zeit zu minimalisieren,
- Durchsatz zu maximalisieren,
- Sofware-Reusability in unterschiedlichen Konfigurationen.

4.6. Interprocesskommunikation

Inter Process Communication (IPC) ist der verwendete Mechanismus, der den Informationsaustausch zwischen CIF-Modulen ermöglicht. Dieser Mechanismus macht sich sowohl lokal als auch inter-board (über den VME-Bus) zunutze.

Die Kommunikation zwischen den Modulen findet durch Nachrichtenaustausch statt. Die ausgetauschten Nachrichten werden im weiteren *Control Blocks (CBs)* genannt.

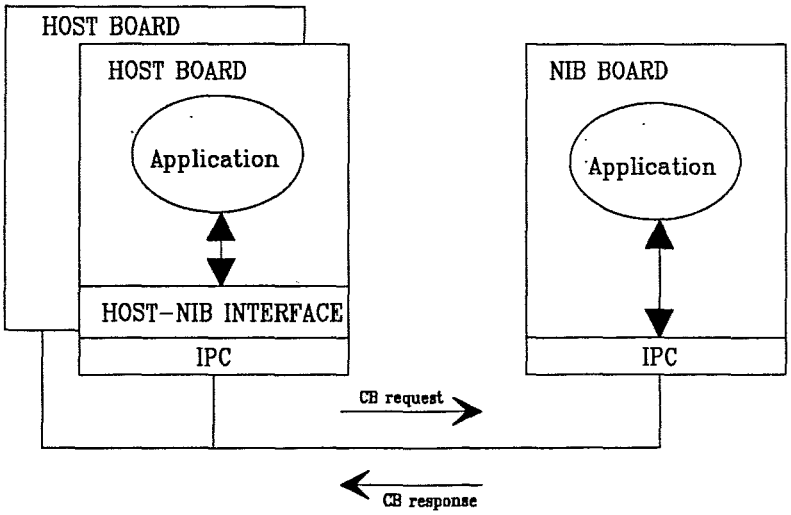


Abb. 4.6) Interprocess-Communication in OBIF Nodes

Wie in Abbildung 4.6 veranschaulicht, ermöglicht IPC den Datenaustausch zwischen dem Host-NIB-Interface-Entitiy auf dem Host-Board und mehreren Transport-Layer-Entities, die auf dem NIB-Board residieren.

Die Funktionen der Transportschicht (für Checksumming, Packetizing und Segmentaion) sind ausschließlich in NIB-Board implementiert. Der Host-Teil ist nur zuständig für die User-Interface und für den Datenaustausch mit dem NIB-Board.

Der Informationsaustausch zwischen einem Host-Board und dem NIB-Board erfolgt nach einem Request/Response-Kommunikationsmodell: wird vom Host ein Communication Service initiiert, so wird ein request Control Block (CB) an NIB Board geschickt. Mit einem response CB bestätigt der NIB die positive oder negative Durchführung des gefordeten Services.

Der User spezifiziert in einem send-CB in welchen Puffer, die zu übertragenen Daten gespeichert sind. Der receive-CB enthält Informationen über Puffer, wo die empfangenen Daten gespeichert werden müssen. Dies impliziert, daß Boards, die Daten über das Netzwerk empfangen oder senden wollen, selber fürs Data Copy zuständig sind. Somit kontrolliert der User das Verkehrsaufkommen von Daten an die Empfängerseite.

Der NIB Board fungiert wie ein "Firewall" und ignoriert jene Daten, für die kein Receive CB zur Verfügung gestellt wurde.

Die zwischen Host und NIB Board ausgetauschten CBs können, je nach Art des gefordeten Services, unterschiedliche Länge aufweisen. Die maximale Länge ist jedoch vom IPC-Mechanismus festgelegt und darf nicht überschritten werden. CBs enthalten "short" user data um die Latenz-zeit zu minimalisieren. Falls größere Datenmengen zu übertragen sind, werden nur die Addressen übertragen, damit die Memory- und Bus-Zugriffszeit möglichst klein gehalten werden. Data Copy Philosophy in OBIF ist in [BAW91-1] zu finden.

4.7. Node Management

Das Node-Management stellt dem OBIF-System Werkzeuge und Möglichkeiten bereit, die das Managen von CIF Nodes ermöglichen. Diese sind unter anderen:

- Mechanismen zur Spezifikation von Sofware und zum Download dieses an einem Board oder an einem Node. Übers Node-Management können Node und Boards resettet oder auch hochgefahren werden.

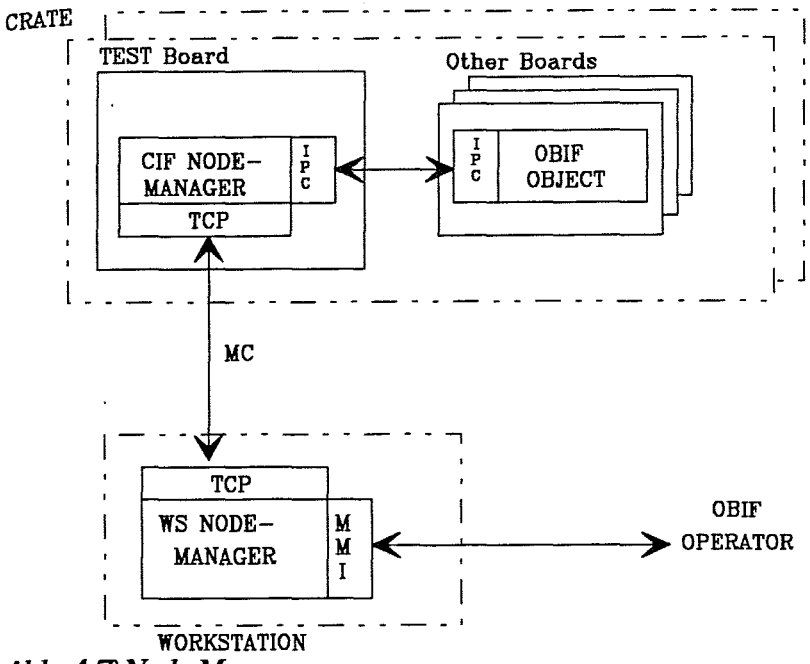


Abb. 4.7) Node Manager

Der Node-Manager ist über CIF-Nodes und OBIF-WS verteilt. Der WS-Teil des Node-Managers wird von dem Operator kontrolliert und dessen Services werden über MMI in Anspruch genommen. Folgende Befehle können vom OBIF-Operator an das OBIF-System erteilt werden: Reset a node, Restart a node, Shutdown a node, Restart a board und Object list of a board. Außerdem schickt der CIF Node Manager, wenn der Zustand eines Nodes sich ändert, ein Report-packet an den WS-Node-Manager.

Für den Austausch von Node-Manager-Primitives zwischen den WS-Node-Managment-Teilen und dem CIF-Node-Management-Teilen wird TCP/IP über das Integration LAN verwendet.

4.8. Time Manager

Der Time-Manager versorgt das OBIF-System mit einer systemweiten Zeitreferenz. Als Zeitreferenz wird die Ground-Workstation-Zeit verwendet. Die Granularität beträgt 10ms. Zur Zeit des Bootens ist die Zeitreferenz an allen OBIF Nodes über Integration LAN zur Verfügung gestellt. Dort wird die Zeit lokal verarbeitet. Es werden periodisch (vom WS Time Manager) Zeitreferenzpackete an jeden OBIF Node geschickt, um die Zeitabweichungen zu korrigieren. Der WS-Operator kann über eine MMI-Schnittstelle die Frequenz des periodischen Absendens der Packete (nur in Sekunden) ändern. Der CIF Time Manager läuft auf das Test-Board jedes Nodes. Er ist zuständig für das Aufrechterhalten der Lokalzeit. Er inkrementiert die Lokalzeit

alle 10s (durchgeführt von einer Interrupt-Serviceroutine die an das System Clock angeschlossen ist).

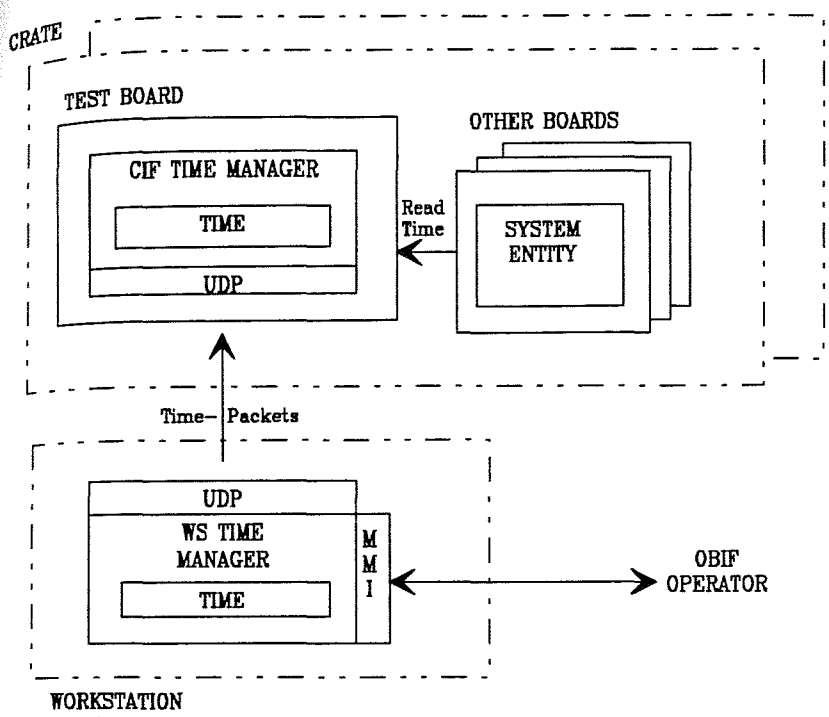


Abb. 4.8) Time Manager

Falls der CIF Time Manager ein Zeitreferenzpacket erhält, dessen Zeitstempel kleiner ist als das eigene, korregiert er die Lokalzeit. In allen anderen Fällen wird das Packet einfach ignoriert bzw. keine Korrektur durchgeführt. Man hat sich für diese einfache Strategie entschieden, damit des hohen Netzwerkverkehrsaufkommens wegen spät angekommene Zeitpackete einfach abgeworfen werden können. Um zu verhindern, daß Lokaluhren schneller laufen als die WS-Uhr, bringt man die WS-Uhr dazu, schneller zu laufen. Diese Strategie hat den Nachteil, daß CIF-Uhren von Zeit zu Zeit nachgestellt werden müssen. Allerdings stellt dies kein Problem dar, weil CIF-Uhren hauptsächlich für die Identifizierung von Reihenfolge der Ereignisse benützt werden.

Die WS- und CIF-Time-Managers kommunizieren miteinander über Integration LAN, und verwenden das UDP Protokol (Das UDP Protokol wurde für seine Fähigkeit, den Broadcasting-Machanismus zu unterstützen, ausgezeichnet).

4.9. Application Manager

Der Application-Manager ermöglicht dem Operator das Laden, Durchführen und Debuggen von OBIF Applikationen. Über eine generische Schnittstelle hat der Operator Zugang zu VxWORKS Services, die das Applicationmanagement unterstützen.

Der Application Manager läuft auf der OBIF-WS. Er benützt das TCP/IP-rlogin-Service des VxWORKS (über das Integration-LAN) um die gefordeten Services auszuführen.

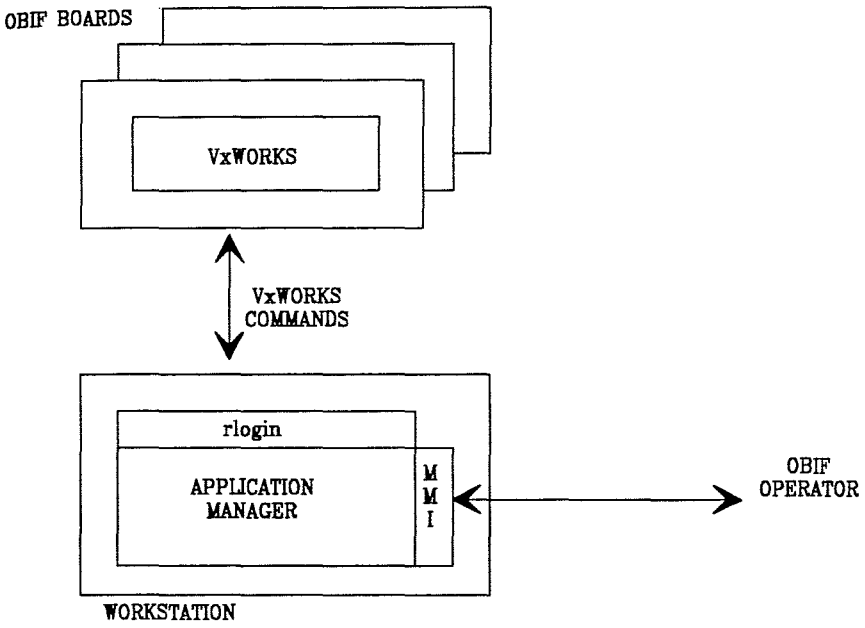


Abb. 4.9) Application Manager

4.10. Network Management & Signalling

Das Network Management & Signalling von OBIF ist für das Managen des OBIF Systems zuständig. Es ist auf allen Knoten (auch WS) verteilt. Der auf die OBIF-WS laufende Teil wird Network Management Station (NMS) genannt und steht unter der Kontrolle vom OBIF-Operator.

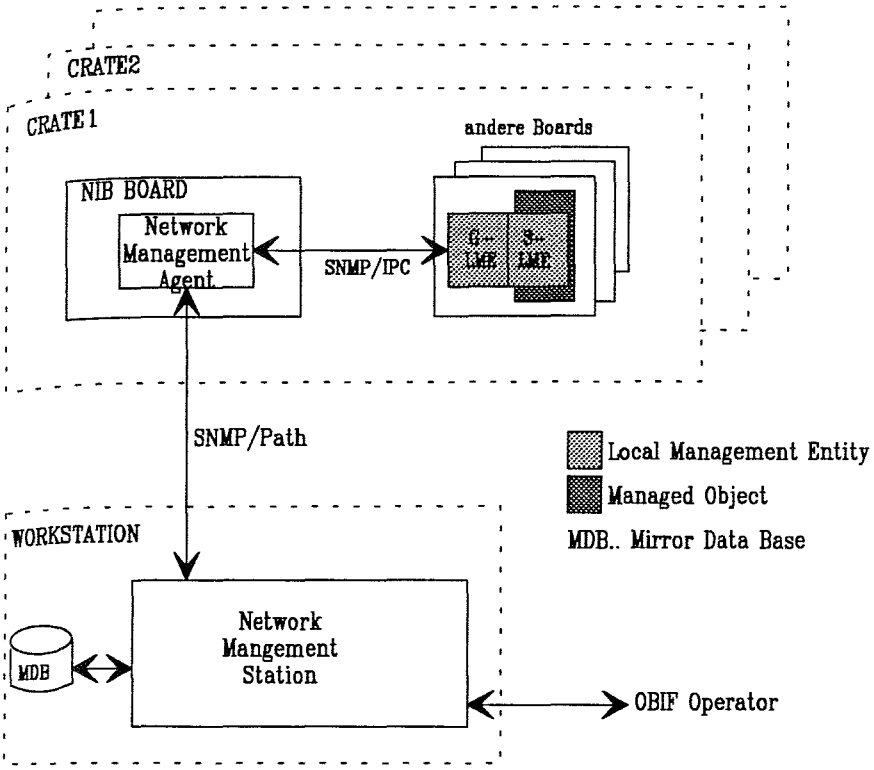


Abb. 4.10) Network Management & Signalling

Es gibt genau einen NMA pro Crate. Dieser residiert auf dem NIB und kommuniziert mit den LMEs von einer, und mit dem NMS von der anderen Seite. Network-Management-Information wird über SNMP ausgetauscht. Die SNMP Packets werden zwischen dem NMS und den NMAs über das Path Service übertragen. Zwischen jedem NMA und seinen untergeordneten LMEs erfolgt dies über den IPC-Mechanismus. Jeder Local Management Entity ist fürs Managen eines bestimmten Managed-Objects (z.B. Path, PV-Interface, VCA, etc) zuständig. Der letzte Stand von Attributen der Managed Objects werden in den Mirror Data Base Files gespeichert. Der Operator kann über eine X-Window/Motif-Schnittstelle jede Zeit die implementierten Management-Oprationen durchführen.

Das Netzwerk-Management im OBIF ist das Thema des nächsten Kapitels und wird dort sehr ausführlich diskutiert. Aus diesem Grund wird hier nicht näher auf dieses Thema eingegangen.

4.11. System Logger

Der System Logger stellt dem OBIF-System einen systemweiten Logging-Machanismus zur Verfügung.

Der System Logger ist über die WS und CIF Nodes verteilt. Die Node Loggers laufen auf die Test Boards. Sie sammeln und speichern alle Logging-Messages (mit Hilfe von TCP/IP-NFS). Es existiert genau ein Log-File pro Node. Der Operator kann jede Zeit ein Logfile schließen und neu öffnen.

Es gibt drei verschiedene Arten von Log-File Einträgen:

1. die Error Messages,
2. die Warning Messages und
3. die Log Messages.

Die ersten zwei werden an den WS Logger geschickt. Der Operator wird dann über die MMI informiert. Der Operator muß explizit jede einzelne Error Message quitieren.

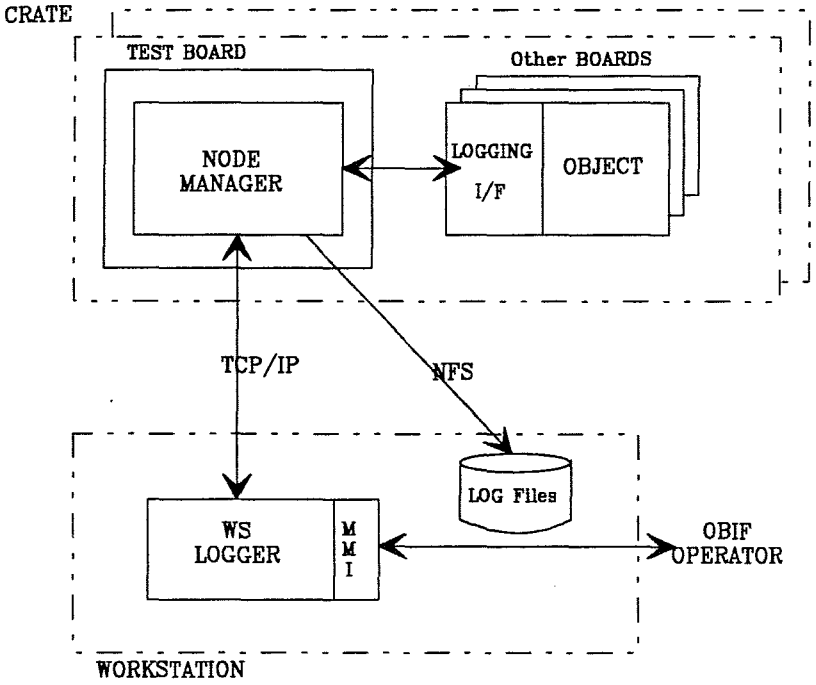


Abb. 4.11) System Logger

Die WS- und Node-Loggers kommunizieren über Integration LAN und verwenden das TCP/IP. Innerhalb eines Nodes wird auf den zuständigen Node Logger über den IPC Mechanismus zugegriffen. Dazu ist eine prozedurale Schnittstelle implementiert, sodaß Details den auf denselben Node laufenden OBIF Entities verborgen bleiben.

4.12. Space Link Subnetwork

Der Space Link Subnetwork (SLS) vom OBIF umfaßt folgende Schichten (vergleich mit Abb. 3.1, 3.2 und 3.3):

1- die CCSDS-Virtual-Channel-Link-Control- (VCLC-) Layer, mit Encapsulation, Multiplexing und Bit-stream Services.

2- die CCSDS Virtual-Channel-Access- (VCA-)Layer, und

3- die Physical-Channel-Simulator- (PCS-)Layer.

Diese Schichten entsprechen den im CCSDS Blue Book [AOS89-4] für VCLC- und VCA-Layer definierten Anforderungen und Definitionen (siehe Kapitel 3).

Auf jedem Gateway-Node (GSLGW und OSLGW) befinden sich ein MUX Board (für Transmission) und ein DEMUX Board (für Reception). Der SLS Sofware läuft ausschließlich auf diese zwei Boards.

4.13. Schnittstellen

4.13.1. Packet Video Interface

Diese Schnittstelle ermöglicht die Kontrolle des OBIF Packet Video (PV) Systems, um PV Packets zwischen die CIF Nodes zu übertragen.

Der PV Interface kann entweder an einen Frame Grabber eines Aufnahmegerätes oder an einen Monitor angeschlossen werden. Das CCSDS Path Packet Service wird für den Austausch dieser Packete verwendet.

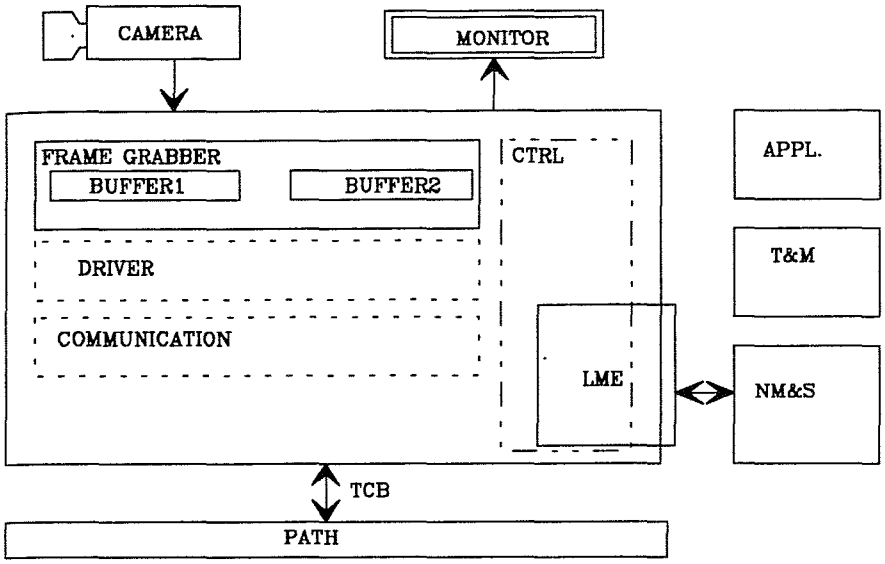


Abb. 4.12) Packet Video Interface

Der PV-Interface kann entweder von einer auf den IF1 Board laufenden Applikation oder vom Netzwerk-Management kontrolliert werden. Über das Netzwerk-Management kann man das Senden bzw. Empfangen von Images stoppen bzw. starten, die Imageauflösung kann auch modifiziert werden.

4.13.2. Compressed Video Interface

Der Compressed Video (CV) Interface ermöglicht die Überwachung des OBIF CV Systems, das für die Übertragung von CVs zwischen die OBIF-Nodes zuständifg ist.

Diese Schnittstelle kann entweder an einen Compressor (genauer bezeichnet an das MPIO-Komponent vom CV System) oder an einen Monitor (über einen Decompressor) angeschlossen werden. Die Video-Images werden in Bitstreams umgewandelt und übertragen. Für diese Übertragung wird das CCSDS VCA Unit Data Service benützt.

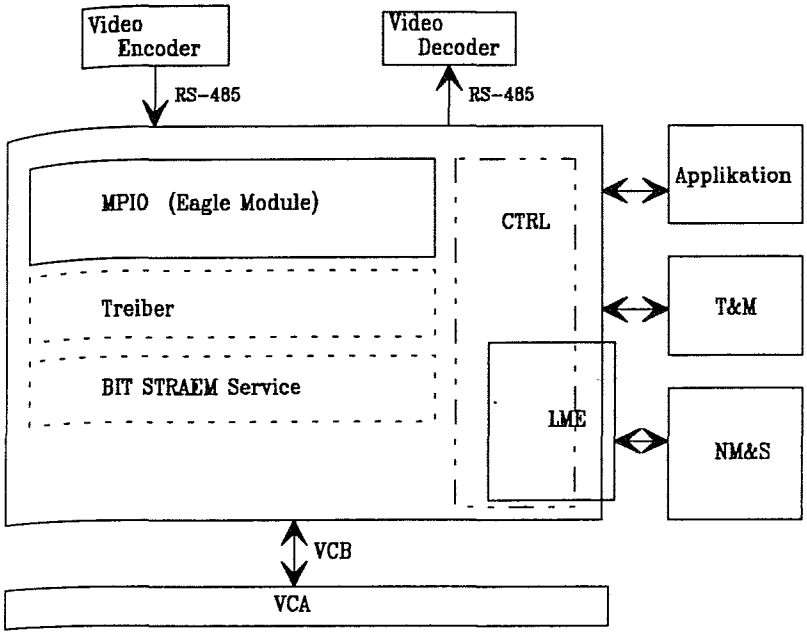


Abb. 4.13) Compressed Video Interface

Das CV System kann entweder von einer auf den IF2-Board laufenden Applikation oder von dem Netzwerk-Mangement (z.B. das Senden/Empfangen von CVs kann gestartet oder gestoppt werden) kontrolliert werden.

If one picture is worth a thousand words,
then one model is worth a thousand
pictures.
-D. Hunt and E.Sullivan,
Between Psychology and Education[HUN74]

5. Netzwerk-Management in OBIF

5.1. Einleitung

Dieses Kapitel stellt das Network Management System in OBIF vor und ist der Schwerpunkt dieser Diplomarbeit. Es soll den Lesern einen Einblick in das Netzwerk Management Konzept von OBIF geben und sie Schritt für Schritt mit allen Aspekten und Besonderheiten dieses Systems vertraut machen.

Zur Entwicklung von NM&S in OBIF

Für OBIF sollte ein Management System entwickelt werden, das Konfigurationsmanagement- und Faultmanagement-Funktionen und Services anbietet.

Im folgenden werden die wichtigsten Anforderungen des SRD (System Requirements Document) zum Thema NM&S in OBIF eingeführt:

"

NM&S services model

1) The services model for the OBIF NM&S system shall be hierarchically layered.

2) The High Level Management layer shall be centralized within the OBIF Control Centre (i.e. OBIF WKS). This level shall be responsible of:

*) the definition of High Level Management Operations;
*) the decomposition into Low Level Management Operations;
*) the sequencing of their execution.

3) The Low Level Management layer shall be decentralized within several Management Agents within OBIF located within each OBIF node. This Level should be responsible for the scheduling and local execution control of Low Level Operations and of the notification and reporting to the High Level Management.

....

4) For the purpose of Network Management, the things that are managed such as protocol machine, connections, addresses and so on, shall be modelled as Managed Objects.

.....

5) Basically, the Managed Objects shall be defined by:
*) attributes, ...
*) operations, ...
*) events,
*) relations to other Managed Objects indicating dependencies.

....

"

Das SRD war die Grundlage füe alle Design-Entscheidungen in der Realisierung des OBIF NM-Systems.

Lösungsansätze

Wie es aus dem SRD zu entnehmen ist, schien ein zentrales Management-System, das eine sternförmige Topologie aufweist, eine vernünftige Grundlage für die Definition der Lösungsansätze zu sein.

In einem Kommunikationssystem muß das Netzwerk Management System sehr stark mit anderen Teilsystemen abgestimmt werden. Deswegen muß ein Netzwerk-Management-System parallel mit den anderen Teilsystemen geplant werden.

Der Teil des SRDs, der die Anförderungen an das NM-System spezifizieren sollte, enthielt:

a) eine Liste von Managed Objects samt ihrer Attribute, setzte aber implizit voraus, daß

b) die zukünftigen Änderungen in derselben Liste die NM&S-Services nicht beinträchtigen bzw. einschränken dürfen (was natürlich und logisch erscheint),

und verlangte zuzätzlich

c) einen mit wenigstem Aufwand durchführbaren Prozess zur Integration von neuen bzw. modifizierten MOs.

Zusammengefaßt: Das NM&S System sollte die MO-Designer entlasten und ihm parallel freie Hand zur Entwicklung seiner MOs anbieten.

Das OBIF-NM&S-System weist einen hohen Grad an Software- und Hardwareunabhängigkeit auf. Es wurde versucht, ein generisches Modell zu konzipieren, das für mehrere Systeme ähnlicher Natur einsetzbar ist und vor allem **kostengünstige Erweiterbarkeit des Manager-Spielraumes** verspricht.

Um ein verteiltes System, das aus mehreren missionskritischen Applikationen bestehen kann, managen zu können, soll das Netzwerk-Management einen hohen Grad an Automatisierung aufweisen. Andernfalls wird das Hinzufügen neuer Applikationen an Netzwerk sowohl für den Netzwerk Manager als auch für den Applikationsbenützer eine mühsame Angelegenheit.

Das, was OBIF-NM&S von den anderen gegenwärtigen Netzwerk-Management-Systemen deutlich unterscheidet, ist die automatische Durchführung von sämtlichen notwendigen Schritten, die die Aufgaben des MO-Designers bei der Integration seines

MOs nur auf gewisse Tätigkeiten, die später detailiert beschrieben werden reduzieren (siehe Kapitel 5.6.).

[illegible] daß das OBIF-NM&S-System derzeit eines [illegible] vollautomatischen (siehe Kapitel 5.6.) NM-Systems ist. Es stellt ein mächtiges Tool zur Verfügung, womit die gesamte Kommunikation zwischen Management-Entities und die Integration dieser im OBIF-NM-System keine Programmierarbeit von den MO-Designers voraussetzt. Sämtliche NM relevanten Source-Codes der oberen drei Schichten, sowohl auf der WS als auch auf der CIF-Nodes, werden von den sogenannten Generatoren erzeugt. Die gesamte MMI wird auch generiert, sodaß die MO-Designers keine Window-Programming Erfahrung erbringen mussen.

5.2. Definitionen

Da im OBIF-SRD der Begriff *Network Management & Signalling (NM&S)* anstatt des üblicheren Terms *Network Management* verwendet wird, folgt gleich die Definition von NM&S:

Das *Network Management* von OBIF wird als die Menge jener Funktionen und Services innerhalb des OBIF Systems, die die Steuerung, Kontrolle und Koordinierung von systeminternen und systemexternen Ressourcen (Managed Objects) ermöglichen, definiert.

Unter *Signalling* versteht man die Menge jener Funktionen und Services innerhalb des OBIF Systems, die die Erfaßung und Austausch von Management Information ermöglichen.

Die von NM&S erbrachten Management-Services kann man unter die zwei Funktionsbereiche des Netzwerk Managements bringen: nämlich das Configuration Management und das Fault Management bringen.

Das OBIF Konfiguration-Management bietet die nötige Funktionalität zur Sammlung, Speicherung und Erstellung vom Netzwerkzustand, und zur Steuerung der Netzwerkkonfiguration und Übermittlung von Management-Nachrichten.

Das Konfigurationsmanagement in OBIF beschränkt sich auf die Konfigurierung von:

- den CCSDS Space Link Subnetwork (SLS) Protocol Entities,
- den CCSDS Path Protocol Entities, und
- den externen Schnittstellen (PV,CV,Audio,..).

Das OBIF Fault-Management umfaßt Maßnahmen, die zur Lokalisierung von Fehlerzuständen und dem Weiterleiten dieser an WS zwecks Fehlerbehebung dienen.

Das Faultmanagement in OBIF beschränkt sich auf:

- die Lokalisierung von Fehlern in Managed Objects und
- die Übermittlung von Fehlern.

Die Fehlerisolierung und Fehlerbehebung sollten vom Operator manuel durchgeführt werden, im Gegensatz zur Fehler-lokalisierung und Fehlerubermittlung, die automatisch vom System unterstutzt werden.

Der OBIF NM&S System Designer sollte die Tatsache in Betracht ziehen, daß externe Subsysteme und ihre Schnittstellen zu OBIF-System nicht unbedingt identisch sein können. Aus diesem Grund wurde in System Requirements Document (SRD) von OBIF ein Object Oriented Design des OBIF NM-Systems bestrebt, woran wir (Dr. Christian Koza und ich) uns, soweit wie möglich, gehalten haben.

Beim Entwurf von OBIF-NM-System haben wir versucht den folgenden relevanten technischen Aspekten besondere Aufmerksamkeit zu schenken:

- Common Mode Failures: Jedes Element im OBIF NM-System sollte mit Cross Checks and Consistency Checks ausgestattet werden, damit diese Art der Failure einfacher zu lokalisieren und zu beheben ist.

- Traffic: Gleichzeitiges Auftreten von Fehlzuständen in Management Entities kann zu Überlastung und in Folge zu Packetverluste führen. Aus diesem Grund wurde versucht, die Menge der auszutauschenden Nachrichten und die Länge dieser Nachrichten minimal zu halten.

- Robustheit: Die Verhaltensweise eines NM-Systems im Falle von duplizierten oder unerlaubten bzw. fehlerhaften Paketen, besonders bei missionskritischen Systemen, ist von großer Bedeutung. Ein zuverlässiges System soll fähig sein, auch in kritischen Situationen richtige Entscheidungen zu treffen.

- Zentrales bzw. dezentrales Management: Ein zentrales Management System impliziert normalerweise einen zentralen Failure-Punkt. Bei dem Entwurf eines operationalen, verteilten Management-Systems sind unter anderen folgende Punkte zu beachten: consistency (z.B. Data-Bases, Netzwerkzustand), Synchronisation und Häufigkeit der Datenbank-Updates.

- Protocol Standards: Die Adoption von Standards erleichtert die Analyse und behindert diese zugleich. (Darüber ist in Kapitel 2 ausführlich diskutiert). An dieser Stelle sei es zu erwähnen, daß beim Entwurf von OBIF NM-System ein Kompromiß bzw. einen Mittelweg gefunden ist. Als Beispiel wäre die Wahl von XDR als Transfersyntax anstatt der üblichen ASN.1-Kodierung (siehe Kapitel 2) anzubringen.

- Testbarkeit: Ein NM-System mit eingebauten Test-Points macht das ganze Testing viel leichter. Test-Points bestehen auf Schnittstellen und aus Snapshot- und Tracing-Möglichkeiten.

- Erweitbarkeit: Dies bedeutet die Fähigkeit eines NM-Systems, sich an zunehmendes Verkehrsaufkommen anpassen zu können, neue Management-Entitäten einfach zu integrieren und sich in andere Netzwerke leicht integrieren zu lassen.

Außerdem sollte das NM-System einen hohen Grad an Modularität aufweisen, damit späteres Einsetzen von neuen Technologien mit akzeptablem Aufwand möglich ist.

Programmability und Inherent-Reliability sind weitere wichtige Punkte, worauf sich jeder Management-System- Designer konzentrieren muß.

Neben technischen Aspekten sind bei dem Entwurf eines verteilten Systems noch eine Reihe von administrativen Aspekten zu betrachten. Darunter versteht man u.a., Software zu verteilen und Versionen zu kontrollieren, Fehler aufspüren und beheben, Systems-Konfiguration zu managen und Sicherheit (Zugriffskontrolle) zu gewährleisten.

Aus der Sicht der NM ist *OBIF* eine verteilte Testumgebung bestehend aus einem *CCSDS Principal Network Prototype(CPN-p)* mit Schnittstellen zu verschiedensten (externen) Subsytem-Prototypen, wobei jedes Subsystem ein spezielles Kommunikations-Subnetz beinhalten kann. Das OBIF-NMS muß die Kontrolle über alle Ressourcen des OBIF Systems, sowohl die OBIF CPN-p Ressourcen als auch externe Ressourcen gewährleisten können. Um dies zu erreichen muß das OBIF-NMS, den Austausch von Management Information zwischen internen und externen Subsytemen unterstützen.

In den nächsten Unterkapiteln wird das OBIF-NM-System in Detail diskutiert.

5.3. OBIF NM&S Model

5.3.1. Management-Domains in OBIF

Das NM-System von OBIF ist hierarchisch aufgebaut. Wie es in Abb. 5.1. zu sehen ist, sind zwei Network Management Domains zu unterscheiden:

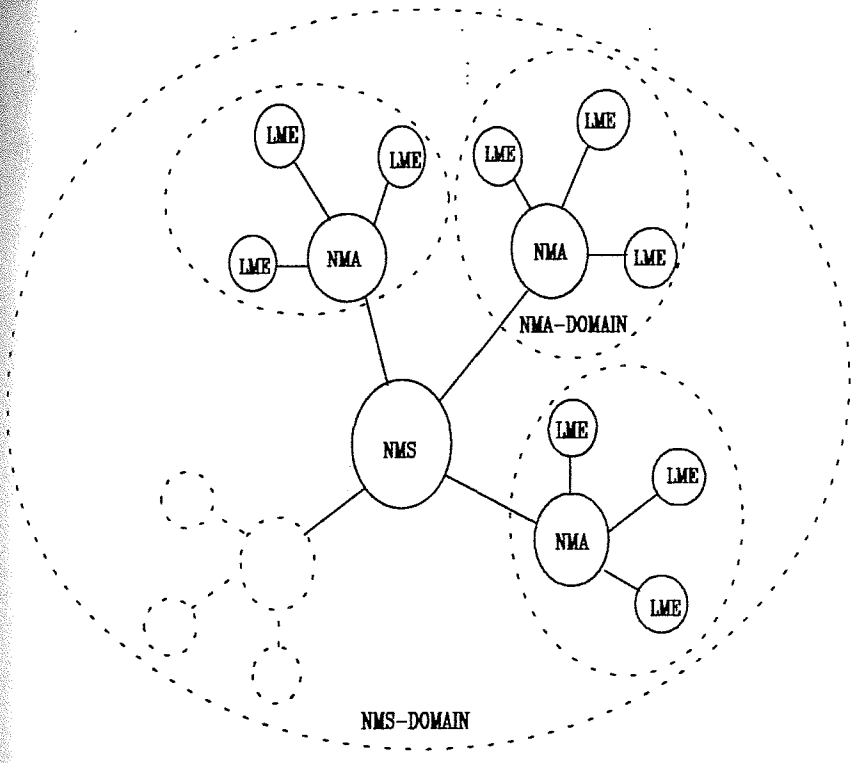
1- Das aus der Network Management Station (NMS) und den Network Management Agents (NMAs) bestehende Network Management Station Domain (NMS-Domain).

2- Das aus jedem Network Management Agent (NMA) und ihm zugeordneten Local Management Entities (LMEs) bestandene Network Management Agent Domain (NMA-Domain).

Der Grund für diese Unterteilung waren folgende Gegebenheiten:

- Funktionen: Gruppierung von Netz- und Managementkomponenten nach ihrer Funktion.
- Organisationsstruktur: Anordnung von Managementkomponenten nach den organisatorischen Einheiten.
- Technologie: Anordnung von Komponenten nach technologischen Kriterien.

Man könnte sich noch ein drittes Domän vorstellen, z.B. ein LME-Domain, dann wären allerdings die aufgelisteten Gegebenheiten nicht einfach voneinander zu trennen. Außerdem tut ein OBIF-NMA fast nichts mehr als Routingaufgaben und Speicherverwaltung.



Hierarchie des Netzwerk-Management von OBIF

Abb. 5.1) Hierarchie des OBIF-NM-Systems

5.3.2. Der OBIF-SMI & OBIF-MIB

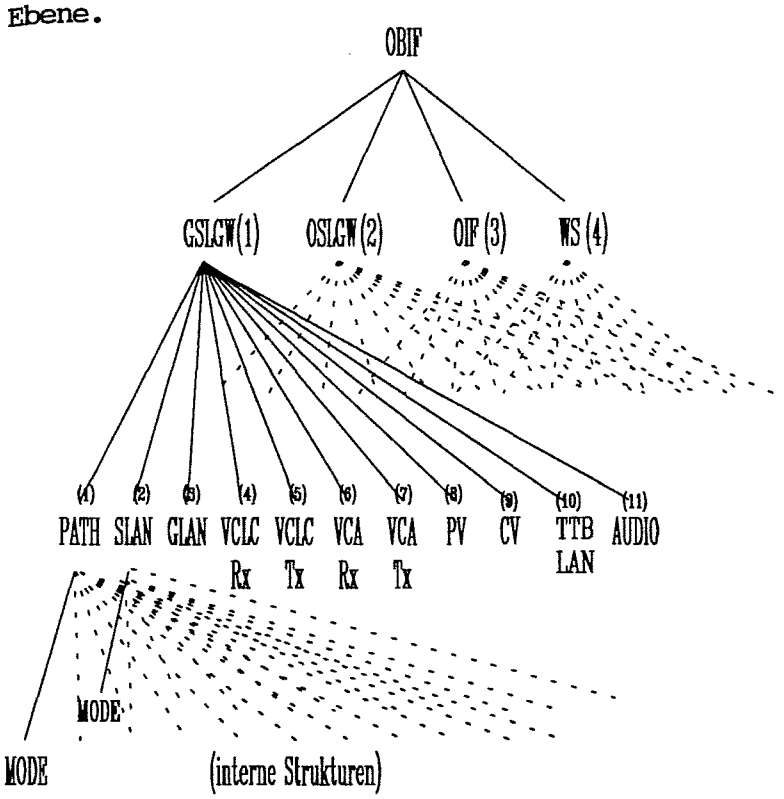
Die für OBIF definierte Management Information Base (MIB) ist in Abbildung 5.2. veranshaulicht.

Die Knoten bzw. die Blätter werden auf alle Ebenen von links nach rechts, mit 1 beginnend, durchnummeriert.

Die OBIF-MIB weist vier (NMA-)Unterbäume auf, von denen je einer einen NMA representiert. Jeder NMA-Unterbaum verzweigt sich wiederum in mehrere (LME-)Unterbäume. Die Anzahl der Unterbäume eines NMA-Unterbaumes ist gleich der Anzahl der von diesem NMA gemangeten LMEs. Die Anzahl der Verzweigungen jedes LME-Unterbaumes ist gleich der Anzahl der LME-Variablen (eine LME_Variable ist eine Sammlung von zusammengehörigen Attributen und kann als eine Datenstruktur verstanden werden) desselben addiert um eins. Diese Addition ergibt sich aus der Tatsache, daß der Zustand eines OBIF Managed-Objects (jedes LME ist zuständig für eine einzige MO) gezwungenermaßen als eine zusätzliche LME_Variable zu verstehen ist. Diese LME_Variable wird Mode genannt und ist Definitionsgemäß mit dem Identifier 1

(um die Addressierung zu vereinfachen) gekennzeichnet. Die LME_Variablen werden wiederum in Attribute untergeteilt.

Aus den obigen ergeben sich vier verschiedene MIB-Ebenen, nämlich die NMA-, die LME-, die LME_Variable- und die Attribute Ebene.



OBIF-MIB

Abb. 5.2) Die Management Information Base

Um die eindeutige Adressierung der einzelnen Attribute gewährleisten zu können, wird folgendes verlangt:

- Die LMEs, die von einem einzigen NMA gemanged werden, müssen eindeutige Namen haben.
- Die LME-Variablen eines einzigen LMEs mussen eindeutige Namen haben.

In NM&S wird jedes NMA-LME Paar durch einen LME-Identifier, genannt Community Name (zum Beispiel pv_lme für Packet_Video), und einen NMA-Identifier (zum Beispiel GSLGW) identifiziert (siehe Abb. 5.3). Im Gegenteil zu Community Name, ist der NMA-Identifier kein Bestandteil des SNMP Pakets (Jeder NMA wird von dem NMS über einen Logical Data Path -LDP-, (ein

Netzwerkverbindung bzw. Session) angesprochen. Die LDPs werden vom Path-Entity dem NMS zur Verfügung gestellt).

Note: In OBIF sind die Community Names nur innerhalb des NMA Domains eindeutig definiert, innerhalb des NMS Domains ist diese Eindeutigkeit aber nicht mehr gewährleistet, da meherere NMAs gleichnamige LMEs managen können.

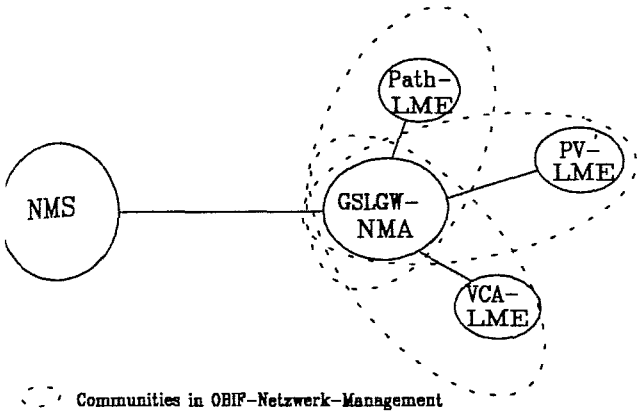


Abb. 5.3) Communities in OBIF-NM&S

Jedes Attribut (vierte Ebene) eines Managed Object besitzt folgende Eigenschaften:

- Type (integer,string,enumeration),
- Gültigkeitsbereich (optional), und
- Zugriffsrechte (READ-ONLY,READ-WRITE).

Jedes Attribut kann auch als Schlüssel (key) definiert werden. Um einzelne Einträge innerhalb LME-Variablen unterscheiden zu können (im allgemeinen ist dies für LME-Variablen, die Tabellen repräsentieren, der Fall), werden zwei Schlüsselebenen definiert, die mit Key-1 und Key-2 gekennzeichnet sind.

Bei der Definition von LME-Variablen kann man auch Traps definieren. Eine Trap-Definition ist sehr eng mit der Attributdefinition verbunden.

5.3.3. Management Entities

5.3.3.1 Network Management Station (NMS)

Der zentrale Baustein des OBIF NM&S ist die Network Management Station (NMS).

Die Network Management Station (NMS) ist in OBIF Workstation (WS) implementiert. Von dieser Stelle aus übt NMS Management Funktionen aus, die über das Netzwerk an Network Management Agents (NMAs) weitergeleitet werden.

Auf der NMS bildet eine OSF/MOTIF Applikation die Benützer Schnittstelle zu NM&S. Über diese Window-oriented Schnittstelle werden Informationen über Management Entities dem Operator zur Verfügung gestellt. Die vom OBIF-Operator verlangten Management Operationen (Set, Get, und Get-Next)werden über das CCSDS Path Service in Form von SNMP Pakets (über NMAs) an LMEs zugeschickt (siehe Abb. 5.4 und Figure A.3).

Die NMS besteht aus fünf Prozessen, die

- Operator Interface Controller,
- NMS Controller (incoming),
- NMS Controller (outgoing),
- Protocol Entity (outgoing) und
- Protocol Entity (incoming)

genannt werden. Mehr Information über die NMS Prozesse, wofür jeder Prozess zuständig ist, oder wie diese Prozesse miteinander kommunizieren, usw. sind in Anhang A, Kapitel A-9 beschrieben.

5.3.3.1.1 Die Mirror Data Base (MDB)

Die NMS verfügt über eine Datenbank, die Mirror Data Base (MDB) genannt wird. Sie enthält die gesamte Konfiguration der OBIF MOs und wird während der automatischen Konfiguration der MOs an die LMEs zugeschickt (die Konfiguration der OBIF MOs wird vom Operator manuell durch das Senden von SNMP-Set-PDUs an die LMEs durchgeführt. Die in den zugehörigen Response-PDUs enthaltene Management Information wird dann in die MDB eingetragen).

Beim Ankommen jedes einzelnen Pakets an NMS lauscht der Incoming-NMS-Controller mit (*snooping*) und teilt dem MDB-Controller mit, falls Änderungen in MDB notwending sind (siehe Anhang A, Kapitel A.7.3. oder [KAM91-1] für mehr Information).

Die Implementierung der MDB war aus drei Gründen notwendig gewesen:

1- Die OBIF Boards verfügen nur über volatile Speicherplatz. Das heisst die OBIF-MOs sind nicht fähig, ihre eigene Konfiguration (das ist die Menge ihrer Attribut-Werte) zu halten. Wenn ein Board

hochkommt bzw. gestartet wird, werden aus diesem Grund alle MDB-Einträge in Form von SNMP-Set-PDUs an die LMEs geschickt und damit die MOs autokonfiguriert.

2- Wie bei allen anderen NM-Systemen auch, ist die Konfiguration der OBIF-MOs eine mühsame und zeitaufwendige Angelegenheit. Die Autokonfiguration entlastet den Operator und erspart ihm viel Zeit.

3- Wie in Kapitel 2 beschrieben, unterstützt SNMP keine polymorphen Funktionen. Aus diesem Grund erfordern bestimmte System-Konfigurationen, die mehrere MOs betreffen (wie zum Beispiel LDP-Konfigurationen), das Setzen von zahlreichen Attributen in mehrere MOs, was klarerweise eine erhöhte Fehlerwahrscheinlichkeit wegen der manuellen Eingabe impliziert.

5.3.3.2. Network Management Agent (NMA)

Die NMAs sind als Router-Knoten des (logischen) Management Netzes zu verstehen. Auf jedem CIF Node gibt es ein NIB Board, worauf die NMAs residieren. Sie geben die von NMS verlangte Information an die LMEs weiter und schicken die von den LMEs an das NMS addressierte Management Information an das NMS.

Unter einem OBIF NMA versteht man einen Task, das auf NIB Board läuft und Management Informationen zwischen die NMS und die LMEs weiterleitet. Er funktioniert als ein SNMP Proxy-Agent. Die Verbindung zwischen dem NMS und jedem einzelnen NMA ist über zwei Logical Data Paths (LDPs) realisiert (Anhang A-2 gibt mehr Information über die OBIF NMAs).

Es gibt drei NMAs in (der Basis-Implementierung von) OBIF:

1- Der Ground Space Link Gateway NMA (GSLGW-NMA), der auf den GSLGW Node laufende LMEs kontrolliert.
2- Der Onboard Space Link Gateway NMA (GSLGW-NMA), der OSLGW-LMEs kontrolliert,
3- Der Onboard Interface NMA (OIF-NMA), der OIF-LMEs unter Kontrolle hat.

5.3.3.3. Local Management Entity (LME)

Die LMEs sind auf LOS Boards bzw. auf NIB Board untergebracht und kontrollieren die Managed Objects.

Die LMEs sind ebenfalls als Tasks zu verstehen. Jeder LME dekodiert die von NMS stammenden Management Pakete und löst über eine Procedural Interface die notwendige Operation auf MOs aus. Die Existenz der Procedural Interface auf jedem LME impliziert eine Unterteilung des LMEs in einen Generic-LME (G-

LME)und in einen Specific-LME (S-LME). Während der G-LME Code von NM&S bereitgestellt wird, muß dies für den S-LME von Managed Object Designer getan werden.

Das Procedural Interface ist definiert als die Menge jener Funktionen (bzw. Prozeduren), die eine 1:1-Abbildung der Operator Requests auf die S-LMEs darstellt. Folgende Abbildung veranschaulicht dieses:

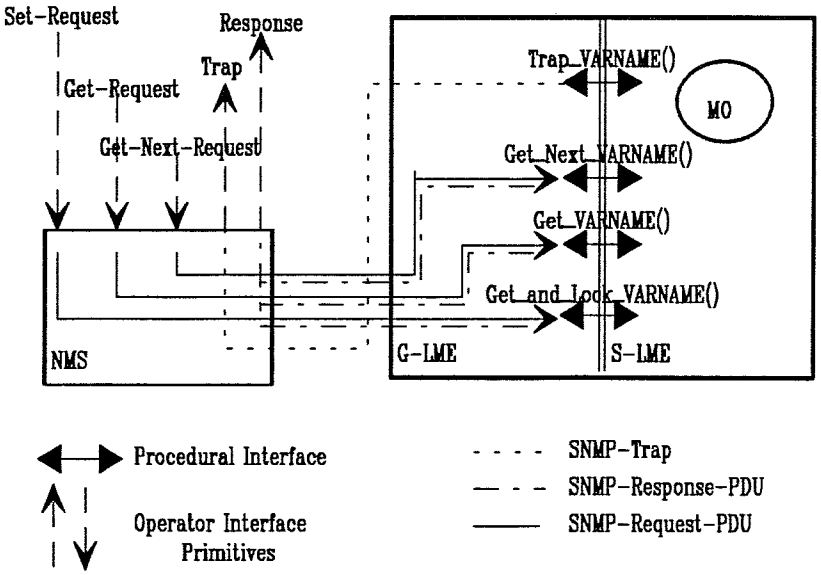


Abb. 5.4) Operator Interface & Procedural Interface

In OBIF wird jedes NMA-LME Paar durch einen LME-Identifier (Name des LMEs, zum Beispiel pv_lme für Packet_Video), genannt Community Name, und der NMA-Identifier (zum Beispiel GSLGW) spezifiziert. Im Unterschied zum Community Name ist der NMA-Identifier kein Bestandteil des SNMP Pakets (Jeder NMA wird von der NMS über einen LDP (bzw. eine Netzwerkverbindung) angesprochen. Dieser wird an NMS von dem Path-Entity zur Verfügung gestellt).

Wie es von der Abb. 5.2 zu sehen ist, sind 11 Managed Objects zu unterscheiden (da in dem ersten System Requirements Document 11 LMEs definiert waren enthält der OBIF MIB Tree 11 unterbäume). Jedes Managed Object representiert genau ein Local Management Entity. Die einzelnen MOs werden in Folge aufgelistet:

1) PATH: representiert jenes LME, das den CCSDS Path Service (siehe Kapitel ..) managed. Der PATH-Unterbaum ist die Sammlung jener Attributen, die über das OBIF NMSystems gesetzt bzw. abgefragt werden konnen.

2) SLAN: representiert das System-LAN-LME.

3) GLAN: representiert das Ground-LAN-LME.

4) VCLC_Tx: representiert das Virtual-Channel-Link-Control LME auf der Senderseite (Tx=Transmission).

5) VCLC_Rx: representiert das Virtual-Channel-Link-Control LME auf der Empfängerseite (Rx=Reception).

6) VCA_Tx: representiert das Virtual-Channel-Access-Sublayer LME auf der Senderseite.

7) VCA_Rx: representiert das Virtual-Channel-Access-Sublayer LME auf der Empfängerseite.

8) PV: representiert das Packet Video LME.

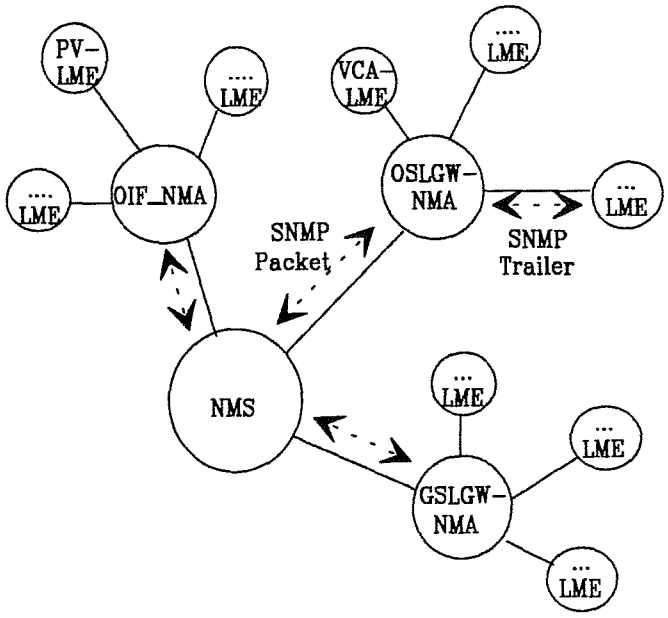
9) CV: representiert das Compressed Video LME.

10) TTB_LAN: representiert das Telescience-Testbed LAN LME.

11) AUDIO: representiert das AUDIO-LME.

5.4. Austausch von Management Information

Für den Austausch von Management Information zwischen OBIF Management Entities hat sich das SNMP Protokol bewährt. Überlegungen, die bei der Auswahl von SNMP für OBIF ausschlaggebend waren sind in Kapitel 2 zusammengefaßt. In XDR (External Data Representation) kodierte Pakete werden über Path-Protokolle zwischen NMS und NMAs übertragen. Die Übertragung von SNMP Paketen zwischen NMAs und LMEs (d.h. innerhalb eines OBIF-Boards) übernimmt der IPC Mechanismus.



Austausch von Managementinformation zwischen Management Entities in OBIF.
Abb. 5.5) Austausch von Management Information

In OBIF sind die vier Standard SNMP Packets implementiert (Get,Set,Get-Next und Trap).

Die Variable-Bindings-List (siehe Abb. 5.6 und [CAS89]) jedes SNMP Pakets enthält die für das Paket in Frage kommenden Attribute:

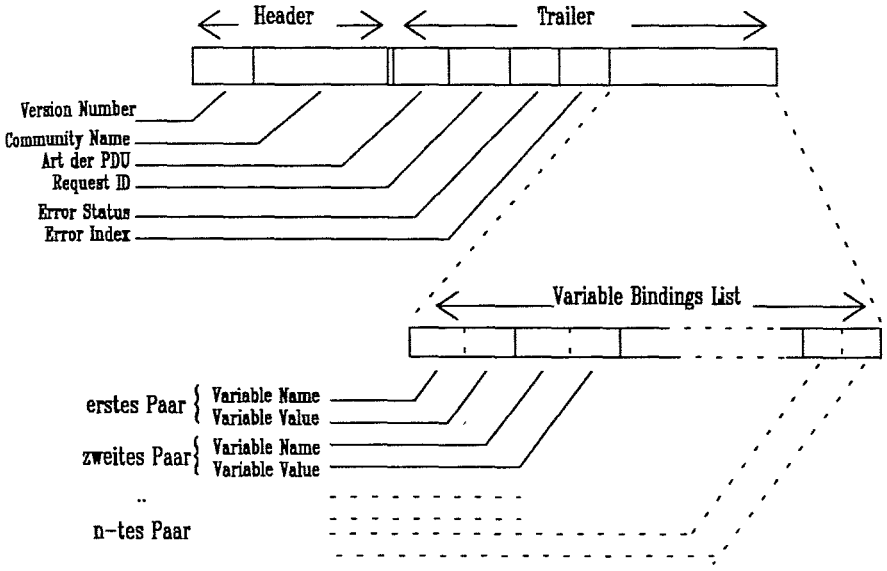
- Für Get und Get-Next sind dies diejenigen Attribute, die READ-WRITE bzw. READ-ONLY Zugriffsrechte besitzen.
- Für Set sind dies diejenigen Attribute, die READ-WRITE Zugriffsrecht besitzen.
- Für Traps sind dies abhängig von den Attributen, die in der Trap-Definition in dem jeweiligen LDF vorkommen.

Auf alle Attribute in der Variable-Bindings-List können vom Operator die NM-Primitiven angewendet werden (siehe Kapitel 2). Der Operator wird vom NM&S-System über die Ausführung der jeweiligen Primitive (über MMI) informiert.

Um die für das NM notwendigen Funktionen Create- und Delete zu realisieren und den Zustand von MOs zu behandlen, wurde die Menge der Standard SNMP Pakete in OBIF-NM&S um drei weitere Elemente erweitert worden, die spezielle Semantik bzw. Ausprägungen der Standard Pakete sind:

Create- & Delete-Packet: Dienen zum Entfernen bzw. Hinzufügen neuer Einträge in MO-Tabellen. Da der Standard-SNMP kein Create- bzw. Delete-Packet definiert hat (es gibt nur Set, Get, Get-Next und Trap), dient in OBIF-NM&S das Set-Request auch zum Kreieren bzw. zum Entfernen. Nach Definition wird jeder Set-Request, dessen Variable-Bindings-List nur aus Schlüssel-Attributen besteht als Delete-Packet interpretiert. Das heißt, beim Empfang eines Delete-Packets soll der S-LME den mit den Schlüssel-Attributen gekennzeichneten Eintrag von der Tabelle entfernen. Dieser Eintrag wird beim Eintreffen des Response-Packets in WS auch aus dem MDB entfernt.

State-Packet: State-Pakets dienen ausschließlich dem Austausch von LME-Variable "Mode" zwischen NMS und LMEs. Sie erfordern ein etwas komplizierteres Protokol Scenario, das eine Mischung von Trap Handling Scenario und Polling Based Scenario ist.



Aufbau von SNMP Packets
Abb. 5.6.) Struktur eines SNMP Pakets

Die SNMP Pakete in OBIF weisen im Gegenteil zu Standard SNMP folgende Unterschiede auf:

a) OBIF-SNMP Pakete sind in XDR kodiert (im Gegenteil zu standard SNMP, das in ASN-1 kodiert wird). Diese Abweichung von Standard ist zweifach begründet:

   1) XDR als Transfer Syntax war notwendig, um die hohe Anforderung an die Performance-Kriterien zu gewährleisten (simplicity of XDR wie z.B. explizite Typebeschreibung führt zu kürzere Pakete).

2) Standard XDR Routinen, die das Kodieren und Dekodieren von SNMP Paketen übernehmen, werden sowohl von SUN/4 (OBIF WS Software lauft derzeit auf ein SUN/4) als auch von VxWORKS (das Betriebssystem auf CIF Boards) unterstützt und zur Verfügung gestellt.

XDR als Transfer Syntax bringt auch Nachteile mit sich, die allerdings während der Entwicklung des OBIF Projektes belanglos waren (OBIF war anfangs nur als ein Prototype gedacht, inzwischen). Die Protocol Entities von OBIF NMS sind allerdings in einer Art und Weise entworfen, daß eine Umstellung auf ASN.1 einfach möglich ist (siehe Anhänge A-1 & A-2).

b) Alle in OBIF ausgetauschte SNMP-PDUs sind von selbem Format,.im Unterschied zu Standard SNMP, wo Trap-PDUs eine andere Struktur aufweisen als Get-, Set- und GetNext-PDUs. An sich liegt der einzige Unterschied zwischen OBIF-SNMP-PDUs in dem Inhalt von drei Feldern:

- Art der PDU: Get, Set, GetNext-, Trap- und State-PDU.
- RequestID: spezifiziert Sonderfälle. Während der Auto-Konfigurationsphase wird RequestID auf AUTO_SET_MODE_C gesetzt, um Set-PDUs von normalen Set-PDUS, die von Operator ausgelöst werden zu unterscheiden (siehe Kapitel Auto-Konfiguration).
- Generic_Trap_ID: ist nur in Trap- bzw. State-PDUs gesetzt und bei allen anderen PDUs ignoriert.

5.4.1. Protokol-Scenarios in OBIF NM&S

Wir errinern uns zuerst an die Adressierungsmethoden der NM-Entitäten, die in Kapitel 5.3. implizit angegeben sind:

NMS-NMA: Jeder NMA ist über eine IP-Adresse von der NMS ansprechbar. Umgekehrt ist die NMS von jedem NMA über ihre IP-Adresse ansprechbar. Dafür sind LDPs definiert (siehe Kapitel 5.3.2. und A.6.2).

NMA-LME: Jeder LME ist über einem Namen, genannt Community Name (siehe Kapitel 5.4, Abb. 5.6 und B.1.2) von dem zuständigen NMA eindeutig ansprechbar (siehe B.1.3).

Wie in Kapitel 2 beschrieben, werden in den SNMP Implementierungen auf einer Seite Polling-Messages periodisch

von Managern an Agents, und auf der anderen Seite Traps von Agents an Manager geschickt.

Das OBIF-NIU-Konzept (kapitel 4.5.2.1 und [BAW91-1]) bestehend aus IPC, Host-NIB-Interface und Data-Copy Kommunikations-services wurde konzipiert, damit die OBIF-Objects (unter Object ist hier ein laufendes Object-Code gemeint, wie z.B. ein Task, ein Prozeß, usw. und nicht ein MO) optimal Nachrichten austauschen.

Das OBIF Netzwerk Management System ist eine Mischung aus Polling-Based und Event-Based Management(siehe Kapitel 2.4.). Auf der Target-Seite benutzen die NMA- und die LME-Objects die obengenannten Kommunikationsservices und tauschen Nachrichten aus (um Management-Pakete auszutauschen, verlassen sich die NMS und die NMAs auf die Leistungen des Path-Entities, was schon erklärt worden ist).

Diese Vorgangsweise ist in Abbildung 5.7. veranschaulicht und wird in folgenden beschrieben.

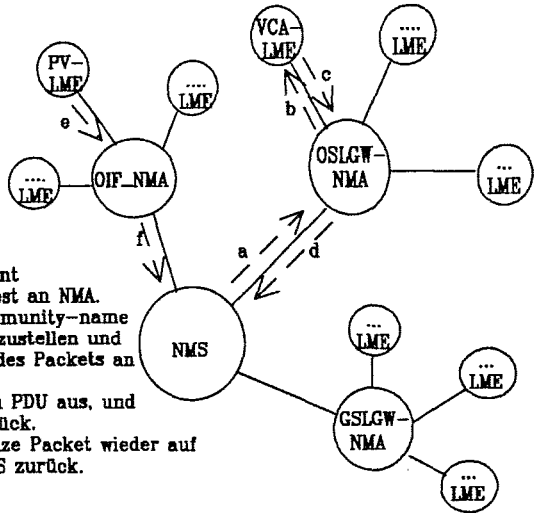
5.4.1.1 Polling-Based Management

a) SNMP-Request-Pakete (Get, Set, GetNext) werden von Netzwerk Management Station (OBIF Workstation) an Netzwerk Management Agents, die auf NIB laufen, geschickt.

b) Der SNMP-Header wird von NMA dekodiert, um aus dem Community-Name das Ziel-LME festzustellen. Jedes NMA verfügt über eine interne Tabelle, wo jedem Community-Name eine physikalische Adresse zugeordnet ist (Jede Applikation, die auf CIF Boards läuft, ist von IPC Mechanismus mit einer Adresse versehen, die Object Identifier genannt wird). Über den IPC Mechanismus wird dann das Ziel LME von dem empfangenen Request-Paket informiert (Das IPC Message, das an LME geschickt wird, enthält die Adresse der PDUs).

c) Nun wird der PDU von LME dekodiert und die verlangte Operation durchgeführt (z.B. bei Get-PDU wird die Variable-Binding List ausgefüllt). Der PDU wird nachher wieder in XDR kodiert. Nachher wird der NMA über IPC Mechanismus informiert.

d) Der Response-PDU wird jetzt an SNMP-Header angehängt und das ganze als ein SNMP-Paket an NMS zurückgeschickt.



Polling-based Management
a) NMS schickt ein request an NMA.
b) NMA benützt den Community-name um den Ziel-LME festzustellen und sendet den PDU-Teil des Packets an LME weiter.
c) Der Ziel-LME füllt den PDU aus, und gibt ihn dem NMA zurück.
d) Der NMA baut das ganze Packet wieder auf und schickt es an NMS zurück.

Traps
e) Der LME informiert sein NMA, daß irgenwelche Daten an NMS zu senden sind, und gibt dem NMA den Trap-PDU.
f) Der NMA baut ein SNMP-Packet und schickt es an NMS.

Polling-Based-Management und Trap-Handling in OBIF

Abb. 5.7.) Mishung von Polling-Based und Event-Based Management Modelle in OBIF NM&S

5.4.1.2 Event-Based-Management

e) Über den IPC-Mechanismus informiert der LME sein NMA, daß es einen Trap-PDU (der schon in XDR kodiert ist) an NMS zu senden hat.

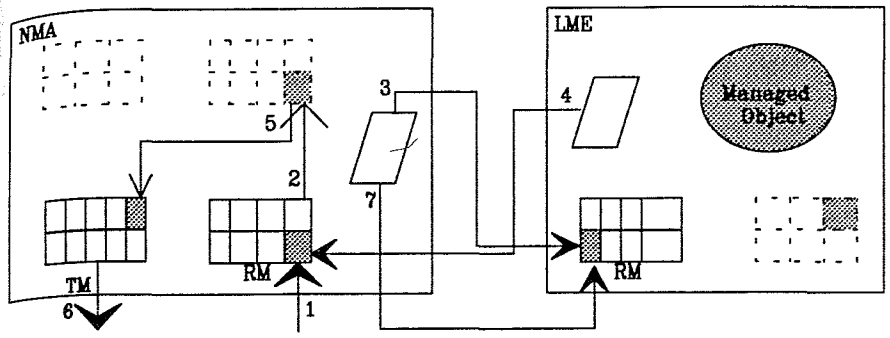
f) Der NMA fügt ein SNMP-Header dazu und schickt das SNMP-Paket an NMS.

Der Austausch von PDUs zwischen einem NMA und seinen LMEs erfolgt in mehreren Schritten, die in nachster Abbildung (Abb. 5.8) veranschaulicht wird.

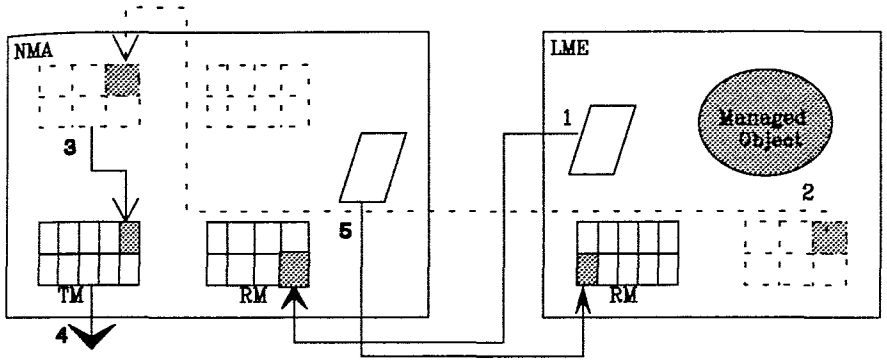
Zu Abbildung 5.8.a

1- Der IPC verlegt eine an NMA adressierte Nachricht in das von NMA definierten Reception-Buffers.

2- Der NMA kopiert den PDU-Teil auf NMA-LME-Buffers. Der Path-Entity wird dann über IPC um die Freigabe des Reception-Buffers (ein Receive-CB) informiert.



a) Data-Copy bei der Bearbeitung von empfangenen Packeten



b) Data-Copy bei der Bearbeitung von Traps

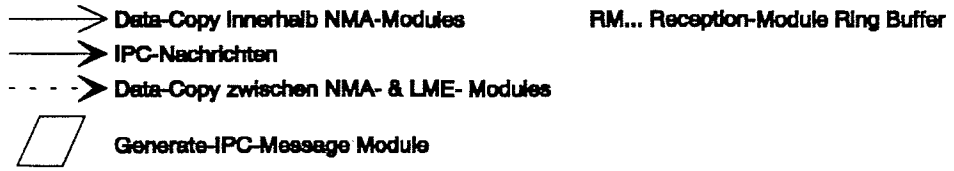


Abb. 5.8.) Data-Copy beim Austausch von Management Information zwischen NMA und LME

3- Der NMA generiert nun eine IPC-Nachricht (genannt Activation-Message) und informiert den LME, daß ein PDU zu verarbeiten ist. Da die Länge eines PDUs die maximale Länge von dafür vorgesehenen Data-Buffer in einer IPC-Nachricht überschreiten kann, wird nur die Adresse des in NMA-LME-Buffers gespeicherten Eintrages dem LME bekanntgegeben.

4- Im normalen Fall (d.h. wenn die PDU-Dekodierung, die Interpretierung des PDU-Inhalts usw. erfolgreich ausgeführt ist) ergänzt der LME die Variable-Bindings-List, kodiert sie und informiert den NMA (wieder über ein IPC-Nachricht).

5- Der von LME kodierte PDU wird jetzt an das dazugehörige SNMP-Header angehängt, in einem SNMP-Packet (Response-Packet) eingekapselt und in Transmission-Buffers gespeichert.

6- Danach informiert der NMA das IPC, daß eine Nachricht an Path-Entity zu schicken ist. Dazu werden die Transmission-Buffers des NMAs, die auch Send-CB genannt werden verwendet.

7- Der NMA informiert den LME, daß die Bearbeitungsprozess eines Requests abgeschlossen sei.

Zu Abbildung 5.8.b

1- Der LME generiert ein Activation-Message und informiert den NMA, daß ein Trap-PDU zu senden ist. Diese Nachricht enthält unter anderen auch die Adresse der im LME-Speicherraum gespeicherten (und in XDR kodierten) Trap-PDU.

2- Der Trap-PDU wird nun mittels einer der Kommunikations-services der NIU, vom LOS-Board auf NIB-Board kopiert.

3- Der von LME kodierte Trap-PDU wird jetzt an ein passendes SNMP-Header angehängt, in einem SNMP-Packet (Trap-Packet) eingekapselt und in Transmission-Buffers gespeichert.

4- Danach informiert der NMA das IPC, daß eine Nachricht an Path-Entity zu schicken ist. Dazu werden die Transmission-Buffers des NMAs, die auch Send-CB genannt werden verwendet.

5- Der NMA informiert den LME, daß die Bearbeitungsprozess des Trap-PDUs abgeschlossen ist, damit der LME die notwendigen Funktionen ausführen kann (wie z.B. Freigabe von Puffer, Übergang in noirmalen Zustand, usw.)

Die Schnittstellen zwischen jedem Management-Entity und dem Path-Entity (Transport Layer) ist für NMS in Anhang A-1 und für NMA und LME in Anhang A-2 zu finden.

5.5. LME-Zustände

Aus der Sicht des OBIF-NM&S kann jeder OBIF-LME vier Zustände haben, wobei der Übergang von Zustand A auf B entweder durch ein Request von NMS (Polling-Requests) oder durch den LME selbst (State-Traps) erfolgt. Dieser Vorgang wird jetzt in Detail beschrieben:

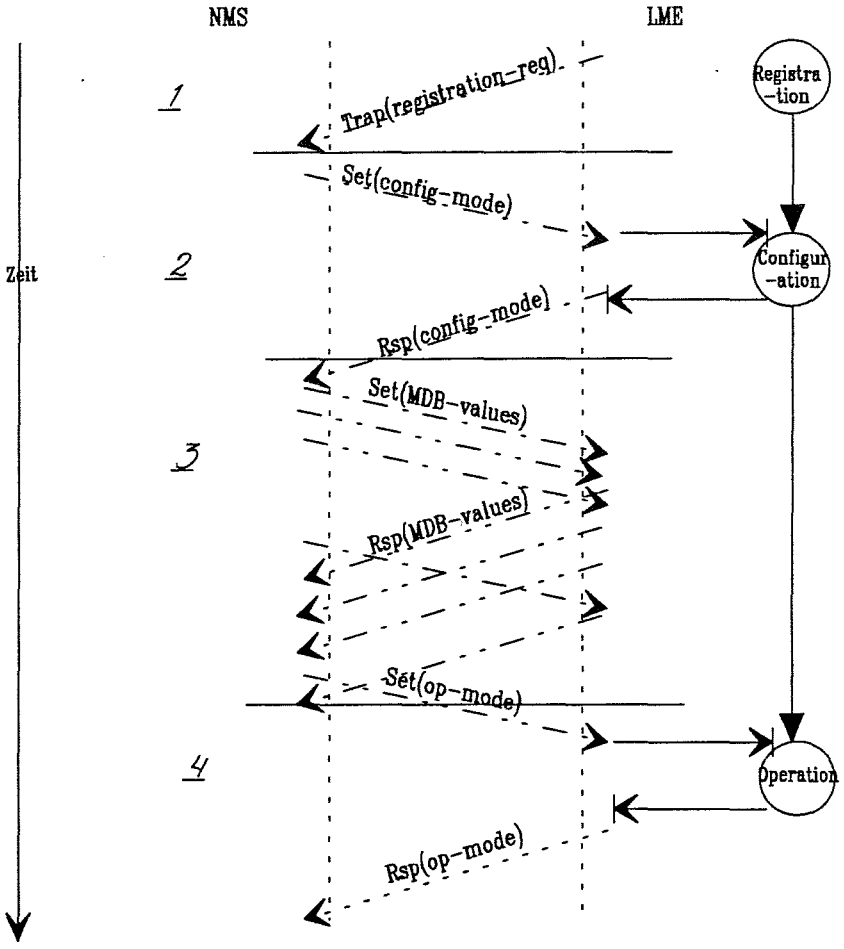
zu Abbildung 5.9

Phase 1:

Wenn ein CIF-Crate hochkommt, werden sich seine LMEs periodisch bei NMS melden. Dies erfolgt mit Auslösen von Request_Registration_Trap PDUs, die von LME asynchron an NMA geschickt werden. Sie werden dann von NMA in SNMP Pakete eingekapselt (diese Pakete werden State-Pakete genannt) und an NMS weitergeleitet (Event-Based Management, s. Kapitel 5.4.1.2). Sobald der LME ein Request_Registration_Trap ausgelöst hat, geht er vom Zustand Deregistration in Registration über.

Phase 2:

Beim Ankommen eines Request_Registration_Trap an NMS, wird ein Set_Request (synchron) an LME zurückgeschickt, der den internen Zustand des LMEs auf Konfiguration zu setzen versucht. Der LME geht in Zustand Konfiguration über, und ist bereit Konfiguriert zu werden. Jetzt wird ein Set_Response von LME an NMS geschickt (Polling Based Management, s. Kapitel 5.4.1.1).



Automatische Konfiguration der LME-Entities in OBIF

Abb. 5.9) Zustandsübergang eines LMEs (während der Autokonfiguration)

Die Phasen 3 und 4 werden auch dem Polling Based Management zugeordnet.

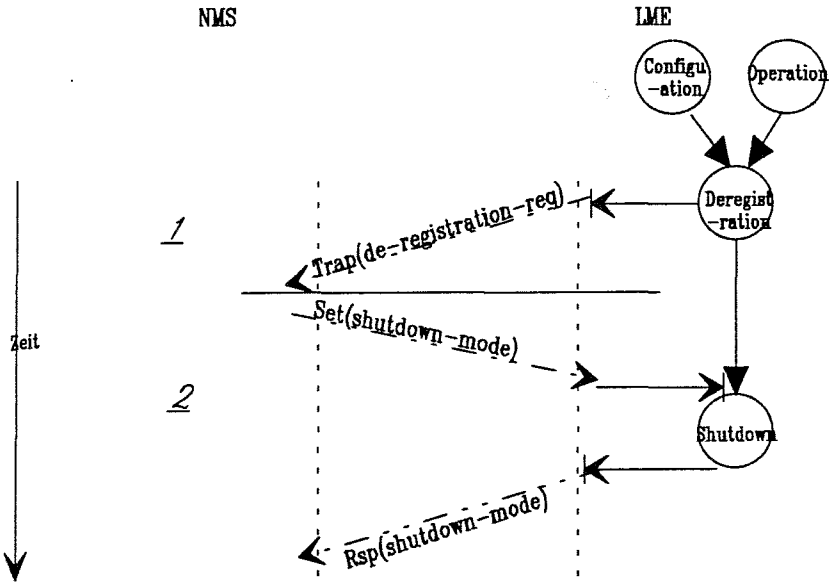
Phase 3:

Die NMS unterscheidet jetzt zwei Falle: entweder existieren schon Einträge in den für den LME existierenden MDB Files oder MDB Files für den jeweiligen LME sind leer. Im ersten Fall werden alle MDB File Entries an LME geschickt und dadurch den LME autokonfiguriert. Im zweiten Fall gibt es kein MDB-Entry, deswegen wird der LME über ein Set-Request gleich auf Operation gesetzt (Phase 4).

Phase 4:

Die NM-Station setzt den internen Zustand des jeweiligen LMEs mittels einen Set-Request auf Operation. Der LME antwortet mit einem Response. Der interne Zustand von LME

darf jetzt vom Operator manuel auf Operation bzw. Configuration gesetzt werden. Management-Information zwischen NM-Station und LME wird mittels Set-, Get-, bzw. Get-Next-Requests und Responses ausgetauscht.



Ablauf eines von einem OBIF-LME aktivierten Deregistration-Trap

Abb. 5.10) Zustandsubergang eines LMEs Anhand von Deregistration-Prozess

zu Abbildung 5.10

Phase 1:
Beim Niederfahren von Crates, setzen sich die LMEs automatisch auf Deregistration, und schicken asynchron Request_Deregistration_Trap PDUs an NMAs. Diese werden wiederum in State-Pakete eingekapselt und an NM-Station weitergeleitet (Event Based Management, Kapitel 5.4.1.2).

Phase 2:
Die NM-Station schickt dann ein Set-Request an LME und setzt ihn auf Shutdown. Diese wird von LME mit einem Response bestätigt (Polling Based Management, Kapitel 5.4.1.1).

Die NMS verfügt über zwei Tabellen, ein State- und ein Description-Table, die während des Informationaustausch mit anderen Management-Entities upgedated werden. Sie spiegeln das gesamte OBIF-System aus der Sicht des OBIF-NMs.

Das State-Table enthält sämtliche Informationen über die Zustände einzelner Netz-Komponenten. Das Description-Table enthält Teile der MIB, Configuration und Operation Flags für

einzelne LME-Variables usw. Die Struktur und genaue Definition dieser Tabellen ist im Anhang 1 Kapitel A-6 zu finden.

5.6. Automatisierungsprozess

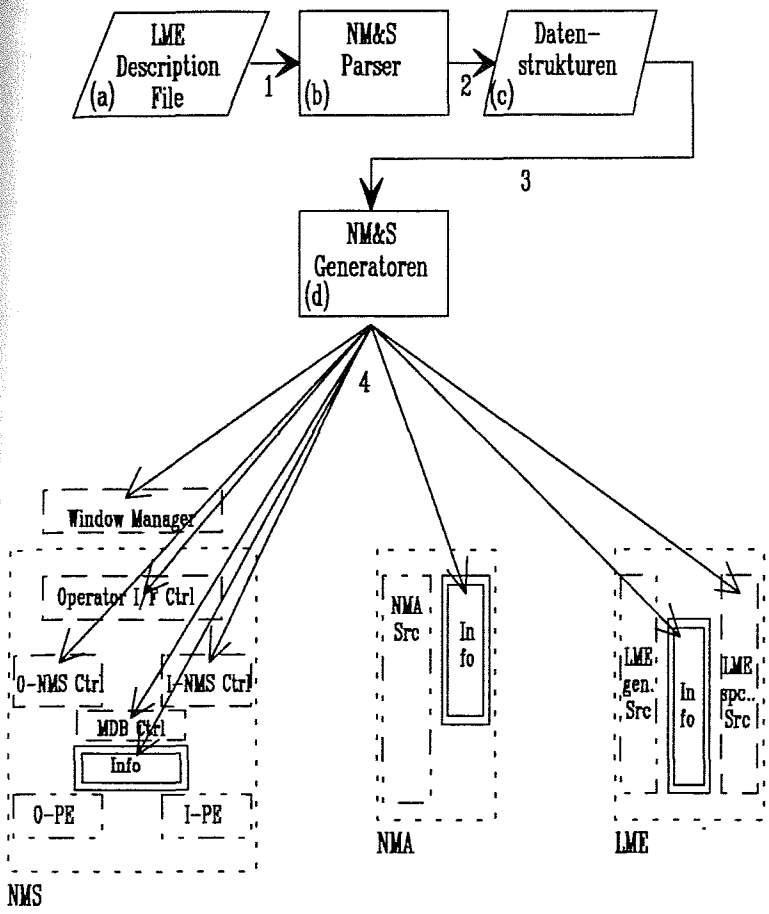
Unter Automatisierungsprozess versteht man sämtliche Leistungen, die von dem NM&S erbracht werden müssen, um mit geringem Aufwand neue MOs (bzw. modifizierte MOs) in dem NM&S System integrieren zu können, damit sie von der OBIF-WS aus kontrollierbar werden.

Der Automatisierungssprozess (für einen einzigen LME) besteht aus mehreren Schritten, die in Folge erklärt werden:

Ausgehend von einem *LME Description File (LDF)* konstruiert der NM&S Parser, C-Strukturen, die das LDF in einem leicht zu verarbeitenden Form enthalten. Die C-Strukturen werden von NM&S Generatoren interpretiert und daraus LME-spezifische Source- und Data-Files, die MMI-Routinen für einzelne LMEs enthalten, generiert. Im Anhang A, Kapitel A.6.1. und im [KAM91-1] sind die generierten Files aufgelistet und beschrieben.

Die generierten Files werden alle in das WS Filesystem abgelegt und falls notwendig (für NMAs und LMEs), in das Target-filesystem z.B. VxWORKS untergebracht.

Der Automatisierungsprozess ist in Abbildung 5.11 veranschaulicht.



Automatische Erzeugung von Source- und Data-files in OBIF-NM&S-Umgebung

Abb. 5.11.) Automatische Generierung für die Integration eines LMEs

a) Um die interne Struktur der einzelnen LME-Variablen (und damit der einzelnen MOs) so einfach wie möglich dem NM&S bereitstellen zu können, wird von dem MO-Designer ein sogenanntes *LME-Description File (LDF)* verlangt. Das LDF eines LMEs enthält sämtliche Informationen, die das NM&S braucht um das LME ordentlich managen zu können. Es enthält Informationen über interne Struktur und Attribute eines LMEs, seine Traps und das vom MO-Designer zum Managen der MOs erwünschter Window-Layout.

Als Beispiel werden unten einige Teile des PATH-LDFs (,die Informationen über LME_Variable 2 und 3 enthalten) wiedergegeben:

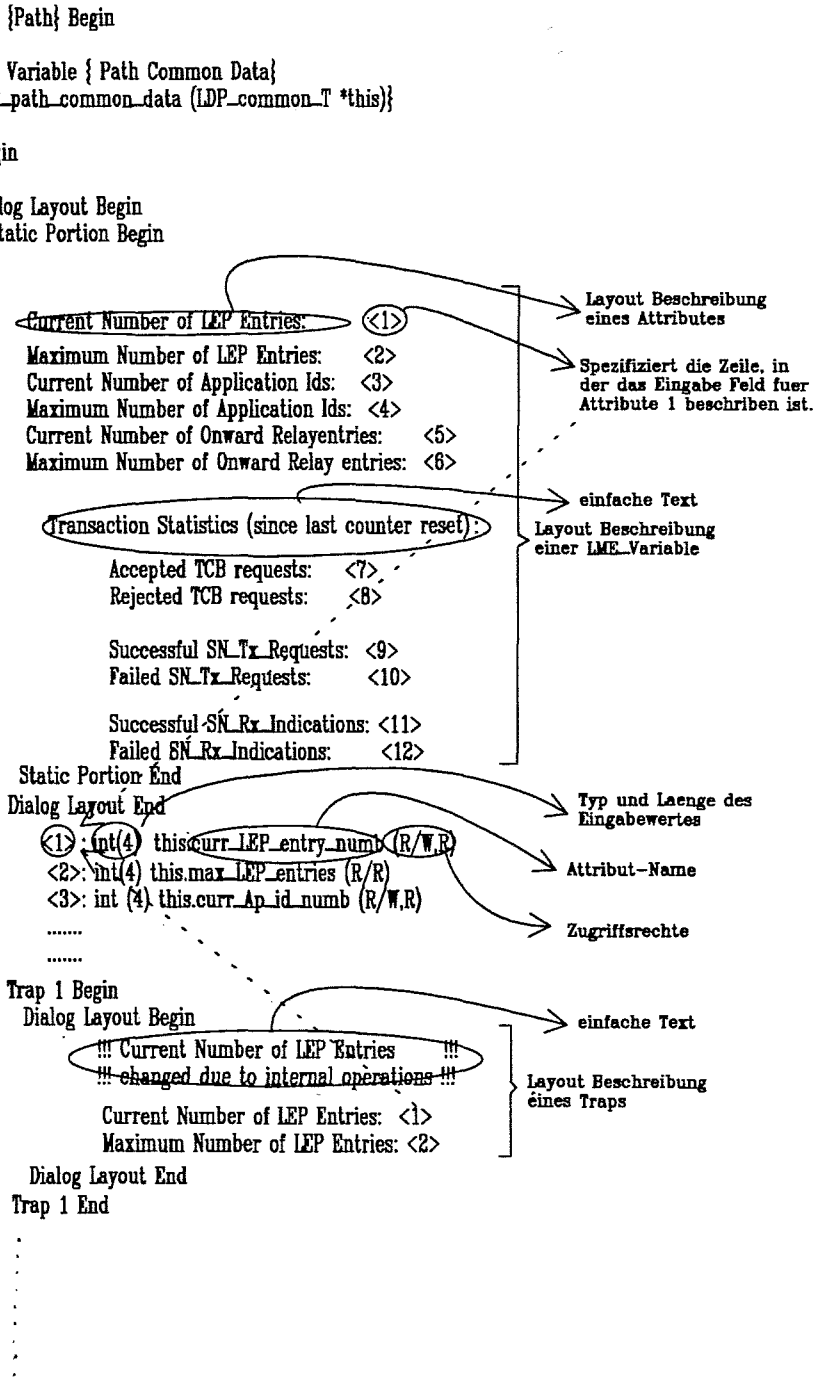


Abb. 5.11.a) Ausschnitt aus dem PATH-LDF

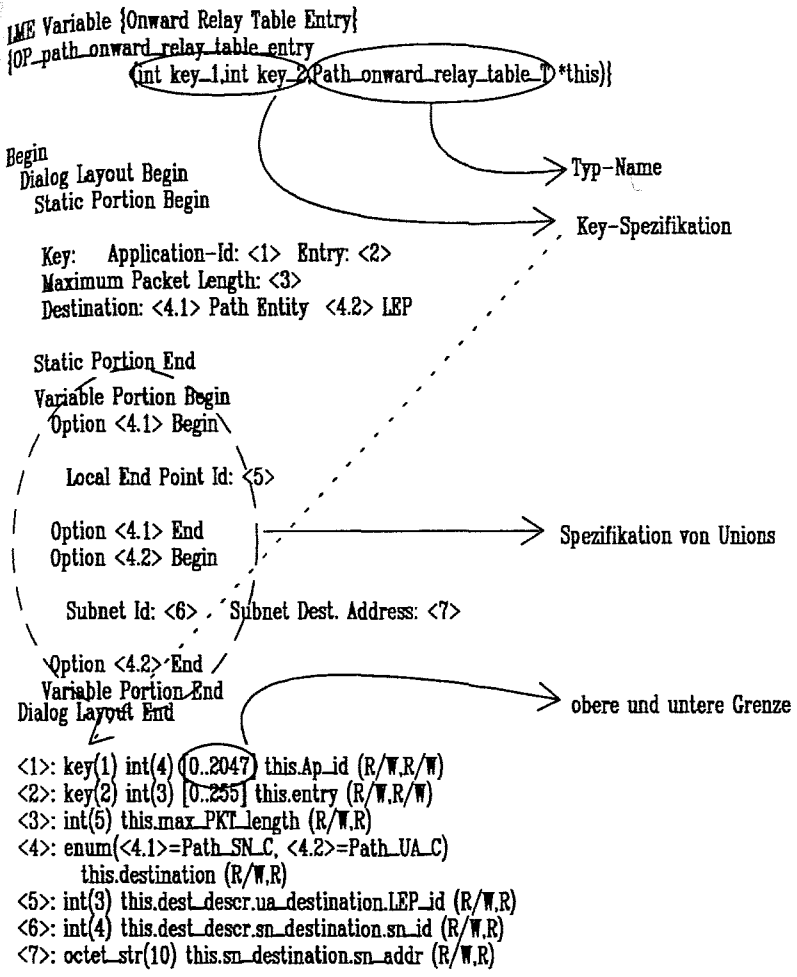


Abb. 5.11.b) Ausschnitt aus dem PATH-LDF

b,c) Der NM&S-Parser (ein LEX-YACC Parser), bekommt ein LDF als Input, und formt die in LDF enthaltene Information in Datenstrukturen um. Diese Umformung ermöglicht eine einfachere Interpretation des LDFs durch die NM&S Generatoren (mehr Information über den OBIF-Parser, z.B. über das LDF-Syntax, findet man in [KOZ91-1] und [KOZ91-2]). Die von dem NM&S Parser erzeugten Datenstrukturen, sind C-Strukturen die einfach, zweifach bzw. mehrfach verkettet sind. Diese Verkettung dient der schnellen Bearbeitung der gespeicherten Information.

d) Bei der automatischen Integration eines LMEs, generieren die NM&S Generatoren jene Teile des gesamten NM&S Software, die notwendig sind um den LME von der NMS aus managen zu konnen.

Das SMI von NM&S-System (eigentlich die LDF-Syntax) erlaubt folgende Typen für die LME-Variablen:

einfache Typen: Die LME_Variable enthält ein einziges Attribut, das vom Typ INTEGER, ENUMERATION, OCTET-STRING, DATE oder TIME ist (z.B. die LME_Variable MODE enthält nur ein einziges Attribute, das vom Typ enumeration ist und den Zustand seines LMEs beinhaltet).

komplexe Typen: Die LME_Variable ist vom Typ STRUCTURE. In diesem Fall faßt sie mehrere Attribute zusammen, wobei alle als einfaches Typ deklariert sind (siehe Abb. 5.11.a). Eine STRUCTURE darf außerdem eine einzige Union enthalten (siehe Abb. 5.11.b).

Liste von komplexen Typen: Die LME_Variable ist vom Typ STRUCTURE_LIST. Eine LME_Variablen, die als dises Typ deklariert ist, representieren normalerweise eine Tabelle, in der jeder Eintrag vom Typ STRUCTURE ist. Der MO-Designer kann maximal zwei Attibute (beide müssen vom Typ INTEGER sein) als Schlüssel definieren um bestimmte Einträge auszeichnen zu konnen (diese Attribute werden im LDF durch den Term *Key* spezifiziert).

Die Generatoren erzeugen Files für alle OBIF-NM&S-Entitäten. Einige Generierungsteilprozesse werden unten angegeben.

Auf der NM-Station:

- Die automatische Generierung jener Datenbank, die die gesamte Netzwerk Konfiguration enthält. Diese Datenbank enthält die Relation zwischen NMS, NMAs, LMEs und LME_Variablen, mit anderen Wörten die gesamte MIB (siehe folgende Abbildung). Diese Datenbank wird beim Hochfahren des NM&S Systems vom OIC eingelesen um die internen Tabelen des OIC zu initialisieren. Dies bedeutet, daß das NM-System ein dynamisches MIB besitzt. Änderungen bzw. Modifikationen in MO-Strukturen eines LMEs brauchen nur in dem jeweiligen LDF durchgeführt werden. Den Rest übernimmt das NM-System. Es interpretiert das LDF und modifiziert die obenerwähnte Datenbank.

  Die folgende Abbildung zeigt die interne Struktur der PATH-MIBs. Alle LME-Variables, außer MODE, sind der OBIF-MIB erst nach dem Generierungsprozess hinzugefügt worden.

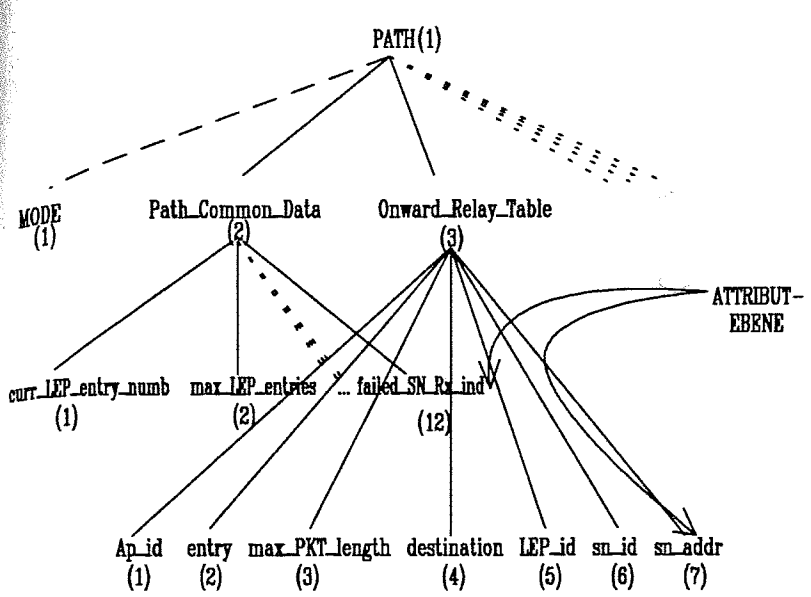


Abb. 5.13) Auschnitte von PATH-MIB in OBIF MIB

Die LME_Varibale 3 (Onward Relay Table Entry) des PATH-LMEs enthält Attribute, die die Charakteristik eines Pfades (Path) reflektieren. Somit spezifizieren die Attribute-Namen 2.3.1, 2.3.2 und 2.3.3 die Application Identifier, Entry und maximale Paketgrosse des jeweiligen Pfades.

- Die automatische Generierung der User Interface Language (UIL) Files. UIL Files beinhalten die gesamte Spezifikation bzw. Layout-Beschreibung (das sind Object-Definitionen und dazugehörige Callback-Routinen) jener MMI-Masken, die zur interaktiven Kommunikation zwischen dem MOs des NM&S Systems und dem Operator notwendig sind. Dies bedeutet, daß der MO-Designer gar keine Window-Programming Erfahrung haben muß. Er schreibt sein LDF, alles andere übernimmt das NM-System. Die in den Abbildungen 5.12.a bis 5.12.d veranschaulichter Layouts sind Beispiele von automatisch generierten MMI-Masken.

- Die automatische Generierung von sämtlichen source code files, die das Verhalten der MMI-Masken kontrollieren. Man kann sich als Beispiel Routinenen vorstellen, die editierbare Eingabefelder auf nicht-editierbar setzen, wenn ein LME vom Zustand Configuration auf Operation übergeht (wie aus dem Path-LDF zu sehen ist, erlauben viele MO-Attribute während der Coniguration Phase Write/Read aber während der Operation Phase nur Read-Only Zugriffsrecht).

- Die automatische Generierung von sämtlichen source code Files, deren Funktionen, die von dem Operator eingegebene

Werte auffassen, überprüfen, und in entsprechenden C-Strukturen umwandeln. Zum Beispiel, beim Überschreiten des maximal erlaubten Wert eines Attributes, beim Nichtsetzen eines Schlüsselattributes usw. erscheinen Error-Windows.

- Die automatische Generierung von sämtlichen source code Files, die aus C-Strukturen SNMP Pakete konstruieren, und eventuell diese C-Strukturen in dem MDB-File System eintragen.

Auf der Workstation erzeugen die Generatoren auch Source-Files für den Operator Interface Controller Prozess und für die NMS Controller Prozesse.

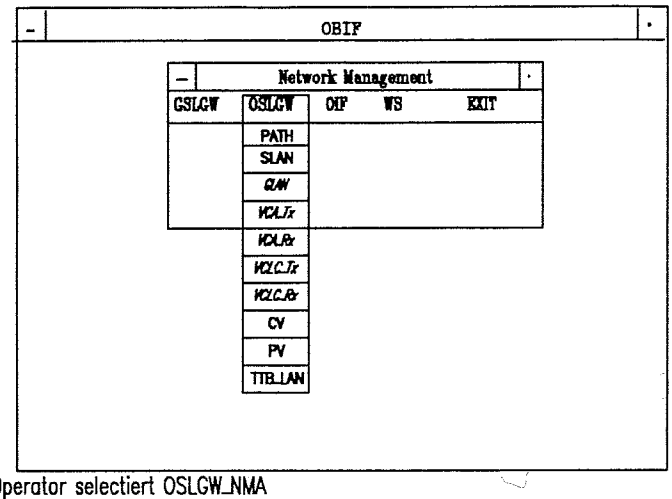
Die Operator-Aktivitäten werden von dem Operator Interface Controller kontrolliert. Jede Aktion, die vom Operator auf ein Dialog-Box ausgeführt wird muss auf Plausibilität geprüft werden. Dies wird von mehreren Routinen (teilweise Callbackroutinen) übernommen, deren Code von den Generatoren erzeugt worden sind. Einige Callbackroutinen garantieren den sauberen Zugriff auf einzelne Attribute, in dem sie die READ_ONLY Attribute in Kursivform darstellen lassen, und die Eingabefelder für solche READ_ONLY Attribute auf not-editable setzen. Andere überprüfen die Eingabefelder und erzeugen Messageboxes falls sie unerlaubte Werte enthalten, oder falls Schlüsselattribute nicht gesetzt sind, usw.

Auch bei der Bearbeitung ankommender Pakete sind generierte source Files voll im Einsatz. Sie übernehmen u.a. De-Encapsulation von Paketen, das Abbilden der SNMP-PDUs auf entsprechende Variable-Bindings-List, Erzeugen von Error-, Warning- und Message-Windows, Eintragen der Response-PDUs in MDB usw.

Das NM&S-System stellt neben dem WS-Software auch den gesamten LME-Software zur Verfügung. Der LME-Software besteht aus einem generierten und aus einem kopierten Teil. Der kopierte Teil ist für alle LMEs fast identisch und wird für neue LMEs einfach von einem Muster kopiert. Nur an gewisse Stellen sind Änderungen notwendig. Die Änderungen sind aber ausschließlich Text-Ersetzungen, die von SED (Stream Editor..UNIX) übernommen werden. Der spezifische Teil wird von den Generatoren erzeugt. Dieser Teil besteht aus Funktionen, die die vom Specific-LME ausgelösten Traps behandeln, für die Procedural-Interface zuständig sind usw.

Das Code aller NMAs ist identisch. Kleine Änderungen, die wiederum nur Text-Ersetzungen betreffen (z.B. die Ersetzung von "####" durch den NMA-identifier) werden von Stream-Editors ausgeführt.

Anhand folgender Abbildungen wird das Verhalten des NM&S-Benützerschnittstelle (die von Operator Interface Controller kontrolliert wird) geklärt.

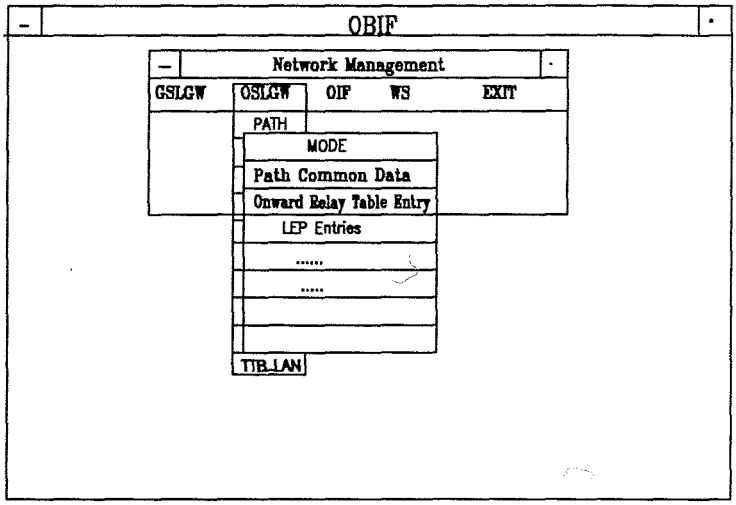


a) Operator selectiert OSLGW_NMA
Die in Kursiv_Schrift gezeigten LMEs sind nicht registriert.

Abb. 5.12.a)

Im Unterschied zu den Abbildungen 5.12.b, 5.12.c und 5.12.d ist das in Abbildung 5.12.a gegebene Layout ein Teil eines UIL Files, das klarereweise nicht generiert worden ist (sie ist unabhängig von den LDFs). Erst ab dem Zeitpunkt des Selektierens eines LMEs wird die Kontrolle an die generierten UIL FIles weitergegeben.

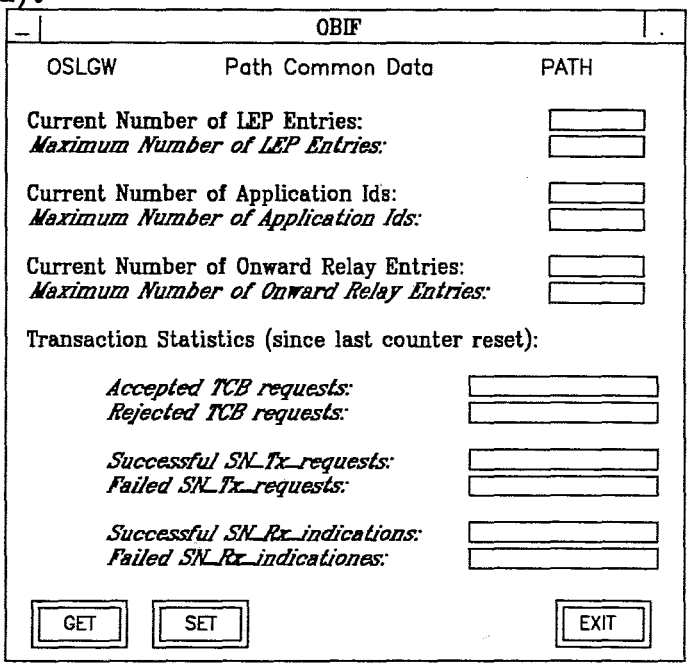
Note: Der Operator Interface Controller erlaubt keinen Zugriff auf nicht-registrierte LMEs.



b) Operator selektiert PATH_LME (nur registrierte LMEs sind selektierbar)
PullDown Menu zeigt die LME_Variablen

Abb. 5.12.b)

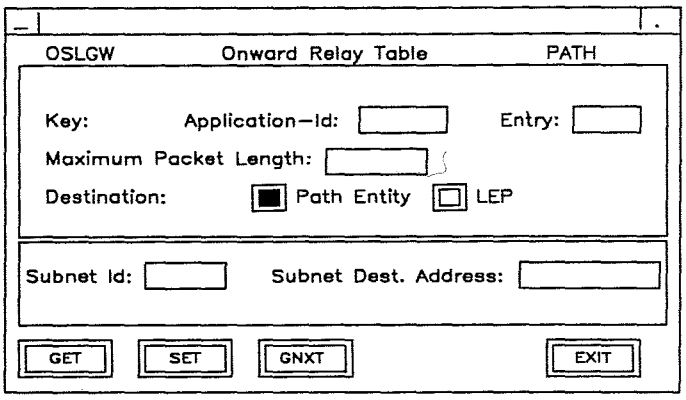
Selektiert der Operator eine bestimmte LME_Variable aus dem Pullright-Menu, so erscheint das in dem LDF für diese LME-Variable gegebene Layout in Form eines Dialog-Boxes (Abb. c und d).



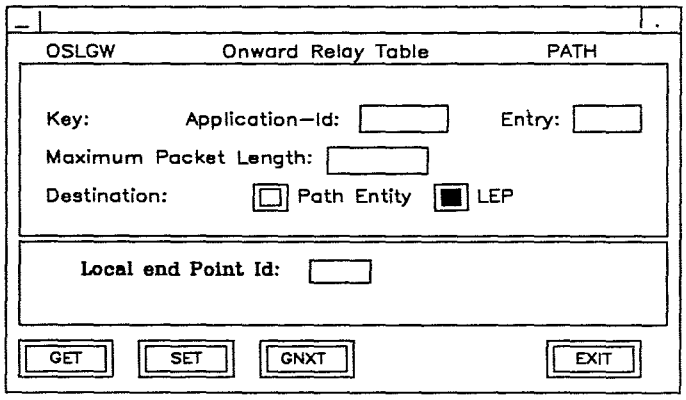
c) Dialog-Fenster fuer LME_Variable 2 (COMMON DATA) in PATH_LME
kursiv: Read-Only Attribut

Abb. 5.12.c)

Figure d gibt das Layout der LME_Variable 3, deren LDF, neben einfache Typen Schlüsselattribute und UNIONs spezifiziert.



d.1) Dialog-Fenster fuer LME Variable 3 (Onward Relay Table) in PATH-LME nachdem Path Entity als Destination angegeben wurde.



d.2) Dialog-Fenster fuer LME Variable 3 (Onward Relay Table) in PATH-LME nachdem LEP (Logical End Point) als Destination angegeben wurde.

Abb. 5.12.d)

Die Generatoren erzeugen Source- und Datenfiles (.c und .h) für alle Management Entities. Das LME Software kann in einem generischen Teil und in einem spezifischen Teil geteilt werden. Der generische Teil ist für alle LMEs identisch und wird für neue LMEs einfach kopiert. Er besteht aus Routinen, die in jedem LME vorkommen, wie z.B. der Schnittstelle zu IPC, dem XDR-Decoder-Encoder, dem generischen Teil des Exception-Handlers usw. Der spezifische Teil wird von den Generatoren erzeugt. Dieser enthält unter anderen die Trap Handling Routines (bilden die über die Procedural Interface erhaltene Trap Information in Variable-Bindings-List um), Mapping Routines (bilden angekommene Variable-Bindings-List in C-

Strukturen um und rufen die richtige Prozedur über die Procedural Interface und umgekehrt).

Aus den obigen Erläuterungen sieht man, dass die Integration eines neuen LMEs bzw. das Erweitern des NM&S um neue LMEs eine ganz einfache Angelegenheit ist:

Der MO-Designer muss auf der WS-Seite dem NM&S einen LDF, und auf der Target-Seite die für die Procedural-Interface definierte Prozeduren dem NM&S zur Verfugung zu stellen. Die gesamte Software, die für die Ausführung der Management Operationen notwendig ist, von MMI auf der Workstation bis zur Procedural Interface auf dem LMEs wird automatisch generiert und verwaltet.

5.7. Conclusion

Obwohl die Basisimplementierung von NM&S eine feste Anzahl von LMEs und NMAs festlegt (namlich 11 LMEs und 4 NMAs) ist das wichtigste was NM&S leistet (und dadurch NM&S von allen anderen gegenwärtigen NM-Implemetierungen unterscheidet), die voll-automatische Integration neue LMEs und NMAs an das NM-System. Mit Integration ist nicht nur das Anschließen neuer Management Entities an NMS gemeint, sondern vielmehr die Tatsache, daß NM&S fahig ist:

Das gesamte NMS, NMA und LME Software, das für den Austausch von Management-Information zwischen LMEs und NMS notwendig ist automatisch zu generieren, und dies für die Application Oriented Layers (siehe Anhang A: Kapitel A.6.1, [KOZ91-1], [KAM91-1] und [KAM91-2]).

Höher Grad an Software- und Hardwareunabhängigkeit einerseits, und an Automatisierung andererseits zählen zu den wichtigsten Merkmale des OBIF-NM&S-Systems. Sie haben sicher dazubeigetragen, daß das OBIF-System heute nicht nur als Prototyp einer Weltraumkommunikationseinrichtung für ESTEC angesehen wird sondern schon jetzt als unentbehrlicher Bestandteil mehrerer operationellen Projekte vom ESOC (European Space Agency Operating Centre) vorgesehen ist.

Strukturen um und rufen die richtige Prozedur über die Procedural Interface und umgekehrt).

Aus den obigen Erläuterungen sieht man, dass die Integration eines neuen LMEs bzw. das Erweitern des NM&S um neue LMEs eine ganz einfache Angelegenheit ist:

Der MO-Designer muss auf der WS-Seite dem NM&S einen LDF, und auf der Target-Seite die für die Procedural-Interface definierte Prozeduren dem NM&S zur Verfugung zu stellen. Die gesamte Software, die für die Ausführung der Management Operationen notwendig ist, von MMI auf der Workstation bis zur Procedural Interface auf dem LMEs wird automatisch generiert und verwaltet.

5.7. Conclusion

Obwohl die Basisimplementierung von NM&S eine feste Anzahl von LMEs und NMAs festlegt (namlich 11 LMEs und 4 NMAs) ist das wichtigste was NM&S leistet (und dadurch NM&S von allen anderen gegenwärtigen NM-Implemetierungen unterscheidet), die voll-automatische Integration neue LMEs und NMAs an das NM-System. Mit Integration ist nicht nur das Anschließen neuer Management Entities an NMS gemeint, sondern vielmehr die Tatsache, daß NM&S fahig ist:

Das gesamte NMS, NMA und LME Software, das für den Austausch von Management-Information zwischen LMEs und NMS notwendig ist automatisch zu generieren, und dies für die Application Oriented Layers (siehe Anhang A: Kapitel A.6.1, [KOZ91-1], [KAM91-1] und [KAM91-2]).

Höher Grad an Software- und Hardwareunabhängigkeit einerseits, und an Automatisierung andererseits zählen zu den wichtigsten Merkmale des OBIF-NM&S-Systems. Sie haben sicher dazubeigetragen, daß das OBIF-System heute nicht nur als Prototyp einer Weltraumkommunikationseinrichtung für ESTEC angesehen wird sondern schon jetzt als unentbehrlicher Bestandteil mehrerer operationellen Projekte vom ESOC (European Space Agency Operating Centre) vorgesehen ist.

A. The Architectural Model of the Network Management Station

The components involved in the management of the Network Management Station (NMS), are shown in the following figure (Fig. A-1).

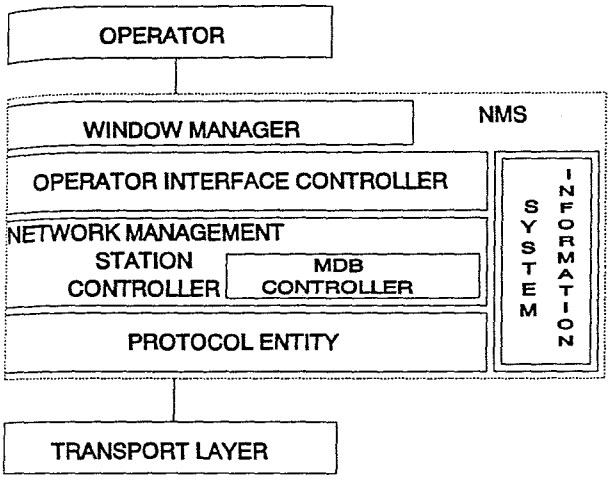


Figure A-1. The Architectural Model of the OBIF NMS

A.1. Window Manager

Interaction between the Network Management System and the Operator, takes place through the Window Manager. Upon NMS requests (e.g. in case of an incoming trap_packet or response_packet sent by one of the LMEs), the Window Manager creates windows, updates window contents, manages or unmanages widgets. Actions taken by the Operator are communicated by this module to the Operator Interface Controller (e.g. activating buttons, selecting an entry of a pulldown menu). It also provides functions for moving and resizing windows, reducing windows to icons, restoring windows from icons and arranging windows on the workspace.

A.2. Operator Interface Controller

The Operator Interface Controller, is responsible to control the behavior of the NMS-Operator Interface.

The functions of this module: supply the Window Manager with the essential data to create windows and update window contents; determine the appropriate action to be taken in

response to requests received from the operator; control the execution of the management primitives or perform local or distributed management services; show the network status to the operator; and perform logging.

A.3. Network Management Station Controller

The Network Management Station Controller, interacts with the Operator Interface Controller and the Protocol Entity. It is the responsibility of this module to control and monitor the flow of management information exchanged between the NMS and the NMAs, as well as providing services to the NMS to operate as expected.

The placement of this component implies that it has to be traversed by each management request and management response operation. This component performs access control on all responses received from other Network Elements, as well as export control on all requests received from the Operator Interface Controller. Thus, management operations performed on the MDB should also pass this component.

A.4. Protocol Entity

The Protocol Entity is responsible for translating the incoming packets sent by LMEs, and the outgoing packets addressed to LMEs, between the SNMP specific form, namely XDR, and the NMS specific representation. It implements the SNMP, and supports the SNMP application entities.

The Protocol Entities residing at the Network Management Station and the Network Management Agents communicate with each other using SNMP.

The Protocol Entity decodes incoming packets sent by NMAs and forwards them to the NMS Controller Module. It also encodes management packets addressed to the NMAs, constructs the appropriate SNMP packets, and forwards them to the underlaying Transport Entity (i.e. the PATH_ENTITY).

Another function of the NMS Protocol Entity is to provide means to its presentation entities, to establish connections, in order to receive and transfer SNMP messages over them.

A.5. Mirror-Data-Base Controller

The Mirror-Data-Base (MDB) Controller, may be regarded as the access point to the MDB. The file structure related calls, that open, and close MDB files, read from and write to MDB files,

access, retrieve and remove entries in the MDB files, are executed merely in the MDB Controller functions.
During system configuration phase, the MDB Controller updates data-structures in the MDB and delivers managed objects data-structures to the NMS Controller, each time a response packet is received by the NMS. The MDB itself, is spread over several files, each containing data-structures of a particular LME_Variable.

A.6. System Information

The System Information Component, is the collection of all information, necessary for the correct operation of the Network Management Station. It contains a description of the network being managed, descriptions of the managed objects that are accessible to and manageable within the system. It also contains information pertaining to the user interface of the system, window layouts, the Mirror Data Base (MDB), and mapping tables for the internal use.

A.6.1.The OBIF NMS Generator

The OBIF NMS Generator, uses the LME_description_structures generated by the OBIF Parser as input, and generates the following files (for each xxx_LME):

Files pertaining to the Operator Interface:

OBIF_xxx_handle_req_proc.c: This file contains that part of the source code of the Operator Interface Controller, which is executed each time a set, get, or get_next button is pushed by the Operator.

OBIF_xxx_rsp_proc.c: This file contains that part of the source code of the Operator Interface Controller, which informs the Operator about the incoming packets.

For each LME_Variable of the xxx_LME there is a procedure named xxx_LME_rsp_lmevar_yyy_proc in this file which is responsible for the creation of the appropriate window of the yyyth LME_Variable with attribute values set to those received from the xxx_LME.

This file also contains xxx_LME trap handling procedures. Trap handling procedures of the xxx_LME are named xxx_show_trap_zzz_proc, where zzz means trap number zzz in the xxx_LME description file. The procedure named xxx_LME_show_response_proc selects and calls one of the above procedures depending on data received from the xxx_LME (e.g. PATH_LME_show_response_proc calls the PATH_LME_rsp_lmevar_3_

proc when GetResponse data is received from the PATH_LME containing attributes of the LME_Variable named LEP_ENTRY);

OBIF_xxx_source.h: This file contains the xxx_LME specific part of the Operator Interface Controller source code (e.g. functions creating xxx_LME pulldown menu, functions managing and un- managing windows of the LME_Variables, etc).

OBIF_xxx_LME.uil and OBIF_xxx_LME_strings.uil: Contain the Object Definition (e.g. the UIL file) for a specific LME.

Files which should be updated (regenerated) only when a new LME has to be added to the MMI or an existing LME should be removed from the MMI:

OBIF_main.uil: the uil file which contains the layout of the main window (and other windows which do not have any parent windows); and its direct children.

OBIF_main.c: the control part for the OBIF_main.uil.

Files involved in the operation of the NMS Controller which handle responses sent by the LMEs:

OBIF_nms_xxx_make_list.h: This file contains procedures named xxx_LME_list_lmevar_yyy_proc. Each of these procedures converts the C_Structure sent by the Operator Interface Controller to the proper variable_bindings_list (i.e. the var_bind_list_arr in nms_pe_if_buf). Another procedure in this file, named xxx_LME_prepare_data_proc selects and calls the appropriate xxx_LME_list_lmevar_yyy_proc.

OBIF_nms_mdb_xxx_cntrl.h: This file contains procedures named: open_xxx_yyy_mdb_for_read (used to open or create the MDB File of LME_Variable yyy of the xxx-LME); fetch_record_yyy_mdb (used to fetch records from the MDB File of the LME_Variable yyy; and close_xxx_yyy_mdb.

Files which have to be updated (regenerated) only when a new LME has to be added to the MMI, or an existing LME is to be removed from the MMI:

OBIF_NMS_LME_cons.h: Contains LME_specific constants like community names, LME_ids, LME_Variable_ids.

Files involved in the operation of the NMS Controller which handels requests invoked by the operator :

OBIF_in_nms_xxx_make_struct.h : Contains procedures named xxx_LME_struct_lmevar_yyy_proc. Each of these procedures converts the incoming variable_bindings_list sent by the

xxx_LME to the proper C_Structure which contains attribute values of the LME_Variable number yyy (i.e. the C_Structure used by Operator Interface Controller). There are some more procedures in this file, named xxx_LME_struct_lmevar_yyy_trap_zzz (i.e. zzz is equal to the trap_id), which construct C_Structures containing trap_specific data). Another procedure, the xxx_LME_prepare_for_Motif_proc, selects and calls one of the above procedures depending on the kind of packet received to the NMS.

OBIF_in_nms_mdb_xxx_cntrl.h: This file contains procedures named: open_xxx_yyy_mdb_for_write.h (used to open or create the MDB File of the yyyth LME_Variable of the xxx-LME; update_record_yyy_mdb (used to update records in the MDB File of the yyyth LME_Variable); and close_xxx_yyy_mdb;.

Files which have to be updated (regenerated) only when a new LME has to be added to the MMI, or an existing LME is to be removed from the MMI:

OBIF_in_nms_lme_cons.h: see OBIF_nms_lme_cons.h.

A.6.2. Description of Management Entities

The internal structure of the NMS_Network_Management_Entity_Description_Table which contains a description of the NMAs, and the LMEs managed by the NMS is described in the following:

```
typedef struct{
 int              NMA_entry_number;
 NMA_data_T       NMA_data_array[max_nma_numb_C];
} NMS_Network_Management_Description_Table_T;
```

NMA_entry_number : gives the number of NMAs managed by the station,
NMA_data_array : each entry of this array contains information on one particular NMA by the NMS,

```
typedef struct{
            int              NMA_id;
            NMA_name_T       NMA_name;
            int              NMA_LME_entry_number;
            LME_data_T       NMA_LME_data_array
                             [NMS_max_lme_numb_C];
} NMS_NMA_data_T;
```

NMA_id : NMA identifier,
NMA_name : name of the NMA,
NMA_LME_entry_number : number of LMEs managed by the NMA,
NMA_LME_data_array : each entry of this array contains information on a particular LME managed by the NMA,

```
typedef struct{
 int                          LME_id;
 LME_name_T                   LME_name;
 int                          LME_Variable_entry_numb;
 LME_Variable_data_T          LME_Variable_data_array
                              [NMS_max_lme_variable_numb_C];
} NMS_LME_data_T;
```

LME_id : LME identifier;
LME_name : name of the LME;
LME_Variable_entry_numb : number of LME_Variables in the LME;

```
typedef struct{
 int                      LME_Variable_id;
 LME_Variable_name_T      LME_Variable_name;
} NMS_LME_Variable_data_T;
```

LME_Variable_id : LME_Variable identifier;
LME_Variable_name : name of the LME_Variable;

The internal structure of the NMS_Network_Management_Entity_State_Table which contains the actual state of all entities managed by the NMS, and the LEP_id of the virtual connections to the NMAs is described in the following:

```
typedef struct{
 int               NMA_entry_number;
 NMA_op_data_T     NMA_state_array[NMS_max_nma_numb_C];
} NMS_Network_Management_Entity_State_Table;

typedef struct{
 NMA_state_T       NMA_state;
 int               NMA_LEP_id;
 int               LME_entry_numb;
 LME_op_data_T  LME_state_array[NMS_max_lme_numb_C];
} NMS_NMA_op_data_T;

typedef struct{
 LME_state_T     LME_state;
 NMS_Time_T      LME_last_change_mode_request;
 NMS_Time_T      NMS_last_change_mode_response;
} NMS_LME_op_data_T;
```

For example, the part of the MIB represented in section 2.2 when stored in the above tables, will look like this:

The NMS_Network_Management_Description_Table_T:

```
NMA_entry_number <-- 4 (four NMAs managed by NMS);
NMA_data_array[3].NMA_id <-- 3 (EQ to LEP_id)
NMA_data_array[3].NMA_name <-- OSLGW;
NMA_data_array[3].NMA_LME_entry_numb <-- 2(two LMEs); managed
by OSLGW: PATH and VCLC);

NMA_data_array[3].NMA_LME_data_array[1].LME_id <-- 1;
NMA_data_array[3].NMA_LME_data_array[1].LME_name<--
                                              PATH;
NMA_data_array[3].NMA_LME_data_array[1].LME_Variable_entry_numb
<-- 3;
and
 ....LME_Variable_data_array[1].LME_Variable_id <-- 1;
          .....[1]LME_Variable_name <-- MODE;
 ....LME_Variable_data_array[2].LME_Variable_id <-- 2;
          .....[2]LME_Variable_name <-- COMMON DATA;
```

The OBIF_Network_Management_Entity_State_Table:

```
NMA_entry_number <-- 4;
```

Entries in NMA_state_array are set and updated during the operational phase.

A.6.3. Description of Managed Objects

NMS_Network_Management_LME_Variable_Access_Table(s):

The NMS_Network_Management_LME_Variable_Access_Table(s), contain(s) access information on properties of the Managed Objects; i.e., the attributes. The number of access tables is equal to the total number of LME_variables declared in the LME_description files. For each LME, the OBIF NMS Generator reads the LME_description file of that particular LME, and generates lines of code, which is used to declare, define, and initialize the Access Tables of that LME.

The naming strategy used by the Generator to declare access tables, is given in the following:

The names of the access tables are identifiers of the form xxx_LME_lme_var_yyy_access_table, where xxx is the name of the LME defined in the LME_description File and yyy is the number of the LME_variable. For example, access tables which provide access information on the PATH_LME shown in Figure 2-2 are the

PATH_LME_lme_var_2_access_table, which contains access information on the attributes of the LME_variable named COMMON DATA; and

```
NMA_entry_number <-- 4 (four NMAs managed by NMS);
NMA_data_array[3].NMA_id <-- 3 (EQ to LEP_id)
NMA_data_array[3].NMA_name <-- OSLGW;
NMA_data_array[3].NMA_LME_entry_numb <-- 2(two LMEs); managed
by OSLGW: PATH and VCLC);

NMA_data_array[3].NMA_LME_data_array[1].LME_id <-- 1;
NMA_data_array[3].NMA_LME_data_array[1].LME_name<--
                                              PATH;
NMA_data_array[3].NMA_LME_data_array[1].LME_Variable_entry_numb
<-- 3;
and
 ....LME_Variable_data_array[1].LME_Variable_id <-- 1;
         .....[1]LME_Variable_name <-- MODE;
 ....LME_Variable_data_array[2].LME_Variable_id <-- 2;
         .....[2]LME_Variable_name <-- COMMON DATA;
```

The OBIF_Network_Management_Entity_State_Table:

```
NMA_entry_number <-- 4;
```

Entries in NMA_state_array are set and updated during the operational phase.

A.6.3. Description of Managed Objects

NMS_Network_Management_LME_Variable_Access_Table(s):

The NMS_Network_Management_LME_Variable_Access_Table(s), contain(s) access information on properties of the Managed Objects; i.e., the attributes. The number of access tables is equal to the total number of LME_variables declared in the LME_description files. For each LME, the OBIF NMS Generator reads the LME_description file of that particular LME, and generates lines of code, which is used to declare, define, and initialize the Access Tables of that LME.

The naming strategy used by the Generator to declare access tables, is given in the following:

The names of the access tables are identifiers of the form xxx_LME_lme_var_yyy_access_table, where xxx is the name of the LME defined in the LME_description File and yyy is the number of the LME_variable. For example, access tables which provide access information on the PATH_LME shown in Figure 2-2 are the

PATH_LME_lme_var_2_access_table, which contains access information on the attributes of the LME_variable named COMMON DATA; and

PATH_LME_lme_var_3_access_table, which contains access information on the attributes of the LME_variable named LEP_ENTRY.

NOTE: xxx_LME_lme_var_1_access_table is reserved for the MODE variable of xxx_LME.

As described in [KOZ91-1],

each attribute of a Managed Object has an Access Mode,which limits the number of management functions (e.g read, write) permitted on that attribute; and

the Access Mode of a particular attribute depends on the actual state of its LME (e.g. operational, configurational).

Considering the above information leads to the definition of the access tables:

```
access_T  xxx_LME_lme_var_yyy_access_table
          [LME_mode_kinds_C][xxx_LME_lme_var_yyy_attr_numb_C]

#define  LME_mode_kinds_C     2;
(Configuration and Operation Modes)

#define xxx_LME_lme_var_yyy_attr_numb_C     ....
(number of the attributes of the LME_Variable number yyy of the
xxx_LME)

typedef enum {
 READ_ONLY_C,
 READ_WRITE_C
} access_T
```

For example, the PATH_LME_lme_var_3_access_table which contains Access Description of attributes of the LME_Variable named LEP_ENTRY in Figure 2-2 will be defined as:

```
access_T  PATH_LME_lme_var_3_access_table
       [LME_mode_kinds_C][PATH_LME_lme_var_3_attr_numb_C];

#define      PATH_LME_lme_var_3_attr_numb_C     8
```

A.6.4. Information pertaining to the Operator Interface

The Operator Interface created for the NMS, uses the User Interface Language (UIL).
The UIL specification files are used to specify the Operator Interface for various LME_description files. For each LME, the OBIF NMS File Generator (see section 3.6) creates two UIL specification files with names that end with the characters

.uil. The name of the first UIL file generated is of the form xxx_strings.uil (e.g. PATH_strings.uil), where the name of the second UIL file is of the form xxx_LME.uil (PATH_LME.uil). The xxx_strings.uil file contains strings used in the xxx_LME.uil file and is included by the related xxx_LME.uil file. The xxx_LME.uil specification file contains a module block that consists of a series of value, identifier, procedure, list and object sections.

It should be noted that, all UIL files are not generated by the generator. For example, the contents of the obifbuttons.uil, obifstrings.uil, most of the obif.uil and most of the obifproc.uil files are not dependent on LME_Description Files and therefore, they do not change when new attributes are added, removed or updated in a LME_Description File.

A.6.5.MDB Files

The MDB Files of the NMS, contain the actual state of attribute values of all managed objects managed by the OBIF Management Entities (i.e. NMS, NMAs and LMEs). Each MDB File contains data structures of one particular LME_Variable. The MDB file structure provides a tree-structured file partitioning, so that files in different directories can have identical names.

The MDB file structure is shown in the following Figure:

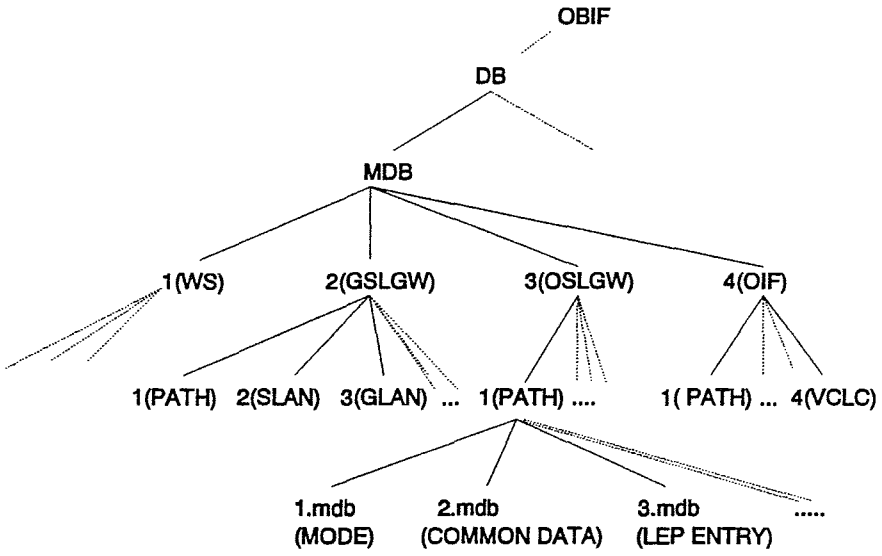


Figure A-2: The MDB File Structure

The root of the MDB file structure is the directory file named MDB. The MDB directory is stored under the DB directory, where other WS configuration parameters are stored. Figure A-2 shows that the MDB directory contains four directory files (i.e. for the four OBIF nodes), and that in each of these files are more directory files (i.e. one per LME) each containing several ordinary files (i.e. one per LME_Variable). Ordinary file names have the form yyy.mdb, where yyy is the identifier of the related LME_Variable in the OBIF MIB Tree (the tree-structured file partitioning, facilitates the different directories, to have files with identical names).

The MDB file structure and naming conventions reflect the OBIF MIB Tree, and its naming conventions discussed in section 2.2.

Note: The WS directory with the path .../DB/MDB/WS, is reserved for the future and will contain the MDB files of the MOs managed by the Work Station LMEs.

A.7. Implementation Architecture of the Network Management Station

This section describes the implementation architecture of the Network Management Station and data_flow between different modules of this entity.

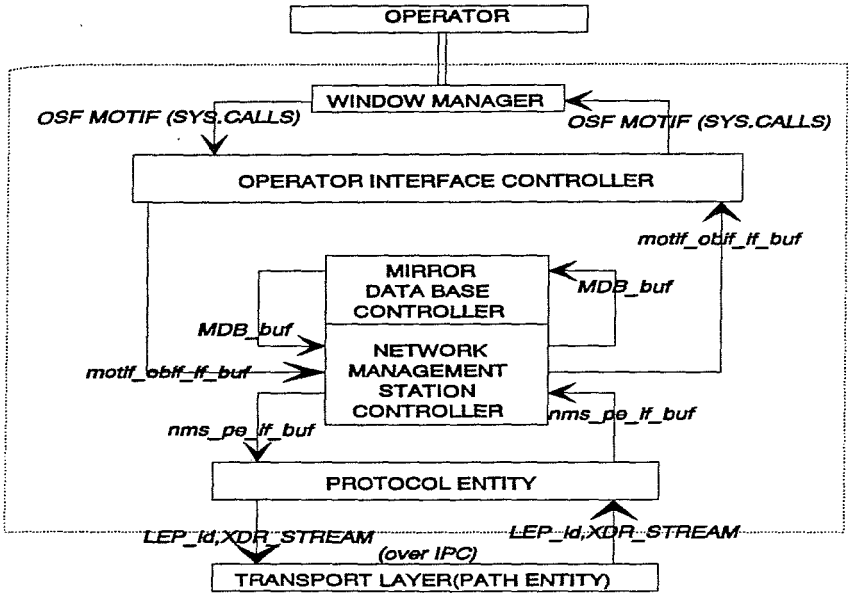


Figure A-3. Data Flow among NMS Modules

A.7.1. Data-Flow in Network Management Station

A.7.1.1. Data Structures exchanged between Operator Interface Controller and NMS Controller

The motif_obif_interface structure (MOTIF_OBIF_IF_buf), contains all the information needed by the NMS Controller to handle the management requests invoked by the Operator (e.g. construct the appropriate request_pdu, and prepare it for encoding), as well as all the information needed by the Operator Interface Controller to inform the Operator, the reception of a response packet received to the NMS.

```
typedef struct {
      NMS_err_T              internal;
      mgt_op_T               mgt_op_id;
      pkt_id_T               pkt_numb;
      obj_id_T               object_id;
      lme_id_T               lme_id;
      lme_var_id_T           lme_var_id;
      lme_var_option_id_T    lme_var_option_id;
```

```
        entity_id_T             nma_id;
        entity_id_T             nms_id;
        snmp_err_T              snmp_req_err;
        snmp_name_T             snmp_req_err_indx;
        snmp_gen_trap_T         snmp_gen_trap;
        snmp_spc_trap_T         snmp_spc_trap;
        time_stamp_T            time_stamp;
        int                     trailer_len;
        two_byte_dummy_T        dummy_T;
        attr_flag_T             attr_flag;
} NMS_motif_obif_if_T;
```

internal : used to specify exceptions;
mgt_op_id (management operation identifier):

From Operator I/F to NMS_Controller :
The mgt_op_id in this case specifies the management activity requested by the operator and its value is equal to

mgt_op_set: operator has invoked a set_request primitive,
mgt_op_get: operator has invoked a get_request primitive,
mgt_op_gnxt: operator has invoked a get_next_primitive,
mgt_op_state: a response to set_request primitive with MODE parameters.

From NMS_Controller to Operator I/F :
In this case the mgt_op_id specifies management activities which should be performed by the Operator Interface Controller in response to packets sent by the LMEs. The value of mgt_op_id can be one of the following:

mgt_op_rsp: the Network Management Station has received a response packet from LME (response to a set, get or get-next),
mgt_op_state: the Network Management Station has received a change_mode_request packet from LME (registration or deregistration request),
mgt_op_trap: the Network Management Station has received a trap packet from LME (i.e., snmp-trap pdus or other traps indicating exeption occurances in one of the LMEs),
mgt_op_op2: is used to handle special cases and is reserved for future use.

pkt_numb (Packet Number): An integer used to map Operator_NMS requests to NMS_Operator responses (for future use).

object_id (Object Identifier): Holds either the object identifier of the destination LME (in case of requests) or the object_id of the source LME (in case of responses). Its value is equal to the object_id defined in lme_description file.

lme_id (Local Management Entity Identifier): Specifies the group identifier of the source/destination LME (see MIB Object Groups).

lme_var_id (LME_Variable Identifier): Specifies the LME_variable subtree identifier,

lme_var_option_id (LME_Variable Option Identifier):

From Operator I/F to NMS Controller:
In this case the lme_var_option_id contains the identifier of the LME Variable Portion (see LME_description File) selected by the operator. Operator Interface Controller fetches widget identifiers as soon as set, get, or get_next buttons are pushed by the operator and sets the lme_var_option_id to its appropriate value.

From NMS_Controller to Operator I/F:
Will be set by the NMS_Controller and is used by the Operator Interface Controller to manage and display the appropriate Window which reflects a Variable-Portion of the responding Managed Object..

nma_id (Network Management Agent identifier): The nma_id field specifies the group identifier of the source/destination NMA (see MIB Object Groups).

nms_id (Network Management Station Identifier): Currently all OBIF-MOs are managed by a single station. The integration of the OBIF System in other Testbeds (just an assumption) may require a communication mechanism among various network management stations. The nms_id field is reserved for the identification of different NMSs in the future.

snmp_req_err (SNMP Error Status): The snmp_req_err field indicates the occurance of exceptions while processing management request packets in LME.

From Operator I/F to NMS Controller:
Always 0.

From NMS Controller to Operator I/F:

Specifies the failure occurred in LME while processing the snmp_set_request, snmp_get_request or snmp_get_next_request packet and its value is equal to:

snmp_err_none : no failure,
snmp_err_too_big : size of the response packet exceeds LME's local limitation,
snmp_err_no_such : the name of an attribute named in the variable_bindings list, does not exactly match the name of any attribute known to the LME,
snmp_err_bad_value: the type of the attribute named in the variable_bindings list, does not match the type of that attribute in the LME,
snmp_err_read_only : the value of an attribute named in the variable_bindings list, could not be altered because its access mode was READ_ONLY (see ACCESS EXPRESSION in LME_description File),
snmp_err_generic : the value of an attribute named in the variable_bindings list could not be retrieved for some internal reasons,

snmp_req_err_indx (SNMP Error Index): Provides additional information by indicating which attribute in the variable_bindings list has caused the exception specified in snmp_req_err.

From Operator I/F to NMS Controller:
Always 0.

From NMS Controller to Operator I/F
Contains the object identifier of the attribute which caused the exception.

snmp_gen_trap (Generic Trap Type): The trap identifier of the trap raised by the LME.

from Operator I/F to NMS Controller:
Always 0.

from NMS Controller to Operator I/F:
Contains the trap_id of the trap raised by the LME.

snmp_spc_trap (Specific Code): Currently always 0. (OBIF LDFs specify generic traps and not specific traps).

time_stamp: Always 0 (In the future, OBIF NM packets may have a time stamp).

trailer_len (Trailer Length): Size of the LME_Variable's C-Structure exchanged between Operator I/F and the NMS Controller.

attr_flag (Attribute Flag): A long integer mask used to specify attributes of the variable_bindings list.

From Operator I/F Controller to NMS Controller:

Indicates which attributes of the LME_Variable are to be sent to the LME. The NMS Controller uses this mask to retrieve those LME_Variable's attributes selected by the Operator and to build the variable_bindings list.

From NMS Controller to Operator I/F Controller:

Indicates which attributes of the LME_variable have been received from the LME. The Operator I/F Controller uses this mask to select the suitable widgets and to set their argument list properly.

The trlr_buff structure: Contains the data structure of the LME_variable and is used to build the variable bindings list of the management packet.

A.7.1.2. Data Structures exchanged between NMS Controller and the MDB Controller

The NMS_MDB_buf structure holds all necessary information the MDB Controller needs to perform file structure related calls on the MDB files.

```
typedef struct{
 NMS_err_T          internal;
 NMS_MDB_op_T        op_id;
 int                nma_id;
 int                lme_id;
 int                lme_var_id;
 int                key_numb;
 int                key_array[nms_max_key_numb_C];
 int                data_struct_len;
 char               data_struct_ptr;
 long               attr_flag;
 } nms_mdb_buf;
```

internal: The only field set by the MDB Controller; gives information on MDB Controller actions:

```
typedef enum{
      .....
      MDB_no_error_C,
      MDB_file_not_found_C,
      MDB_entry_not_found_C,
      MDB_bad_reference_C,
      MDB_continue_C,
      .....
} NMS_err_T;
```

op_id: operation identifier; is either equal to NMS_MDB_config_C or NMS_MDB_op_C.
nma_id: the object identifier of the NMA;
lme_id: the object identifier of the LME;
lme_var_id: the object identifier of the LME_Variable;
key_numb: number of entries in the key_array;
key_array: holds key attributes of the incoming responses;
data_struct_len: length of the data-structure passed between the NMS Controller and the MDB Controller;
data_structure_ptr: reference to the data-structure passed between the NMS Controller and the MDB Controller;
attr_flag: specifies those attributes of the data_structure referenced by the data_structure_ptr whose attribute values must be altered by the MDB Controller;

A.7.1.3. Data Structures exchanged between NMS Controller and the Protocol Entity

The data structure exchanged between the NMS Controller and the Protocol Entity is called the nms_pe_if_buf and is of type nms_pe_if_T discussed in the following.

from NMS_Controller to Protocol Entity:
The nms_pe_if_buf holds the contents of the snmp_request_packet which is to be encoded by the Protocol Entity, an identifier which specifies the destination NMA, and administrative data.

from Protocol Entity to NMS_Controller:
The nms_pe_if_buf holds the contents of a snmp_response_packet or a snmp_trap_packet already decoded to the local syntax of the NMS, an identifier which specifies the source NMA, and administrative data.

```
typedef struct{
      entity_id_T             dest_src_entity;
      NMS_err_T               internal;
      vrs_no_T                version_numb;
      comm_name_T             community_name;
      mgt_op_T                mgt_op_id;
      req_id_T                req_id;
      snmp_err_T              snmp_err_indx;
      snmp_gen_trap_T         snmp_gen_trap;
      snmp_spc_trap_T         snmp_spc_trap;
      time_stamp_T            time_stamp;
      int                     var_bind_list_arr_len;
      var_bind_list_array_T   var_bind_list_arr;
 } nms_pe_if_T;
```

dest_src_entity (Destination Source Management Entity):
An identifier which specifies the source/destination NMA and is used by the Protocol Entity to map between the NMA internal representation in NMS and the LEP_id. (i.e. to send a packet to an NMA the Protocol Entity uses this identifier to retrieve the right LEP_id entry from the NMA_state_array, see section A.3.6.2);

internal: Specifies internal errors occurred in the passing module, its value can be one of the following when set by the Protocol Entity:

```
typedef enum{
      ....
      PE_header_err_C,
      PE_trailer_err_C,
      PE_gen_err_C,
      PE_LEP_entry_not_found_C,
```

```
            PE_no_error_C,
            ....
        } NMS_err_T;
```

version_numb (Version Number):
Not used.

Community_name (Community Name):
Name of the source/destination LME (e.g. path_lme,VCLC_lme,etc),

mgt_op_id, req_id, snmp_err, snmp_err_indx, snmp_gen_trap, snmp_spc_trap, time_stamp :
See the struture exchanged between Operator Interface Controller and the NMS Controller.

var_bind_list_arr_len (Variable Bindings List Array Length):
Contains the number of entries in the Variable Bindings List Array,

var_bind_list_arr (Variable Bindings List Array):
Contains the variable bindings list of the incoming/outgoing snmp packets. Each entry of this array is of type name_value_pair_T

```
typedef struct {
 snmp_name_T      attr_name;
 value_T          attr_type;
 union {
       long   attr_value;
       char  attr_value_str[octet_string_len_C];
  }val;
} NMS_name_value_pair_T;
```

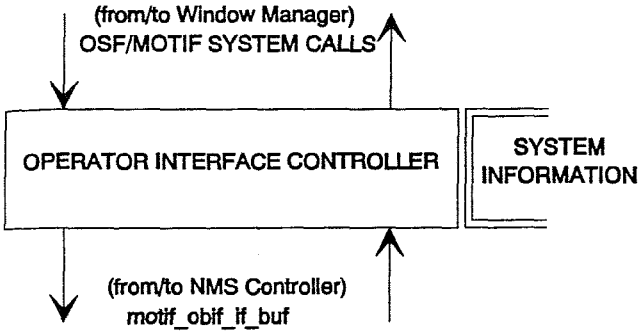
attr_name: name of the attribute (i.e. path in the MIB Tree),
attr_type: type of the attribute (e.g. int, octet_str, date, time),
attr_value and attr_value_str: value of the attribute (val.attr_value_str if attr_type is equal to octet_str or octet_dot_str, and val.attr_value otherwise).

A.7.2. Operator Interface Controller

The functions provided by the Operator Interface Controller are divided in two groups:

1) The Request Handling Functions: functions involved in the processing of the requests invoked by the Operator.

2) The Response Handling Functions: functions involved in the processing of the notifications to the Operator.



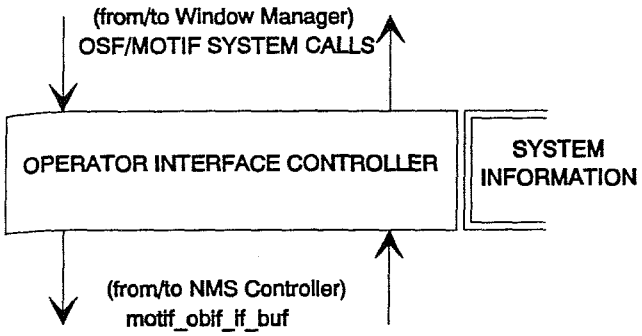
The Request Handling Functions register the actual NMA, LME, and LME_Variable visible to the Operator, handle Get_, Set_, GetNext_request primitives invoked by the Operator and keep track of other Operator activities. This means everytime a Set_, a Get_ or a GetNext_button for a particular LME_Variable is activated, memory for the size of the C_Structure representing the internal structure of that LME_Variable is allocated, and the field named struct_size is set to the size of the allocated C_structure (see section 3.6 OBIF_xxx_LME_set_proc.h and OBIF_xxx_LME_get_proc.h). The NMS Controller needs the size of the allocated C_structure to separate the header of the motif_obif_if_buf from its trailer.

Response Handling Functions:

Upon receipt of a motif_obif_if_buf from the NMS Controller, the Response handling Functions of the Operator Interface Controller retrieve the nma_id, the lme_id, the lme_var_id, the lme_var_option_id, the mgt_op_id and the attr_flag fields from the nms_pe_if_buffer; set the curr_nma, curr_lme, curr_lme_var, and curr_lme_var_option_array and perform the following:

IF mgt_op_id is equal to mgt_op_rsp, which specifies that the incoming packet is a response sent by one of the LMEs then

> the lme_id is fetched to identify the sending LME, and the lme_var_id to identify the LME_Variable of the sending LME,
>
> if the LME_var_id is equal to 1 (i.e. reserved for Mode) and the value of the first attribute of the variable_bindings_list is LME_reg_C or LME_dereg_C then
>
> > the LME_state field of the sending LME in the LME_state_array[lme_id] is updated,



The Request Handling Functions register the actual NMA, LME, and LME_Variable visible to the Operator, handle Get_, Set_, GetNext_request primitives invoked by the Operator and keep track of other Operator activities. This means everytime a Set_, a Get_ or a GetNext_button for a particular LME_Variable is activated, memory for the size of the C_Structure representing the internal structure of that LME_Variable is allocated, and the field named struct_size is set to the size of the allocated C_structure (see section 3.6 OBIF_xxx_LME_set_proc.h and OBIF_xxx_LME_get_proc.h). The NMS Controller needs the size of the allocated C_structure to separate the header of the motif_obif_if_buf from its trailer.

Response Handling Functions:

Upon receipt of a motif_obif_if_buf from the NMS Controller, the Response handling Functions of the Operator Interface Controller retrieve the nma_id, the lme_id, the lme_var_id, the lme_var_option_id, the mgt_op_id and the attr_flag fields from the nms_pe_if_buffer; set the curr_nma, curr_lme, curr_lme_var, and curr_lme_var_option_array and perform the following:

IF mgt_op_id is equal to mgt_op_rsp, which specifies that the incoming packet is a response sent by one of the LMEs then

> the lme_id is fetched to identify the sending LME, and the lme_var_id to identify the LME_Variable of the sending LME,
>
> if the LME_var_id is equal to 1 (i.e. reserved for Mode) and the value of the first attribute of the variable_bindings_list is LME_reg_C or LME_dereg_C then
>
> > the LME_state field of the sending LME in the LME_state_array[lme_id] is updated,

in case of LME_reg_C (meaning that the LME has changed its state to registration in response to a set-packet) the Operator Interface Controller sets the motif_obif_if_buf.mgt_op_id to mgt_op_set, and the request_id NMS_Auto_config_C and sends this structure to NMS Controller and performs no further action (i.e., the NMS Controller will try to send all entries found in MDB -for that particular LME- to the LME when the request_id is set to NMS_Auto_config_C meaning Auto Configuration).

otherwise attr_flag is used to specify which attributes of the LME_Variable of the sending LME were in the packet,

this among some other information (e.g. a copy of the C-Structure constructed by the NMS_Controller that represents the LME_Variable's internal structure) is used to select and activate the appropriate window manager system calls (see section 3.6,"OBIF_xxx_LME_rsp_proc.h").

If mgt_op_id is equal to mgt_op_state, which specifies that the incoming packet is a mode-change request (i.e. request registration trap and request deregistration trap) sent by one of the LMEs then

it uses the lme_id to identify the LME sending that mode-change request,

retrieves the snmp_gen_trap field of the incoming packet which is either equal to the integer 254 (in case of a registration request) or to the integer 255 (in case of a deregistration request), selects and activates the appropriate window manager system call to notify the incoming trap (i.e. to show the appropriate window containing trap_data),

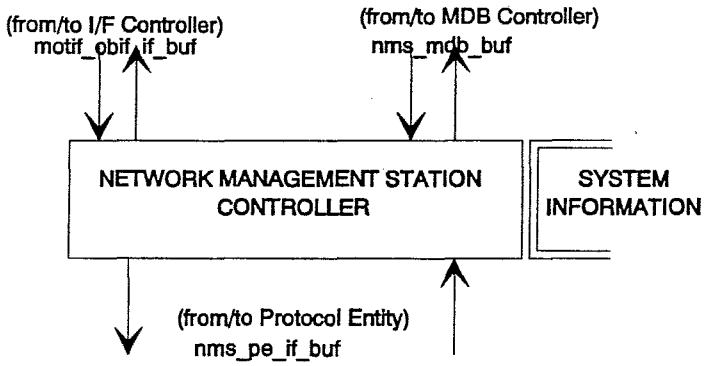
if the ACCEPT_button is pushed by the Operator then the Operator Interface Controller constructs a motif_obif_if_buf structure containing Set_request data, (i.e. pushing ACCEPT button has the same affect as selecting the LME_Variable_1, setting the attribute Mode to LME_reg_C or LME_dereg_C and then pushing the SET button),

IF mgt_op_id is equal to mgt_op_trap, which specifies that the incoming packet contains trap specific data sent by one of the LMEs then

It retrieves the snmp_gen_trap field of the incoming packet to specify the trap invoked by the LME (see section 3.6.1 OBIF_xxx_LME_rsp_proc.h),

The snmp_gen_trap and the C-Structure constructed by the NMS Controller Module are necessary and sufficient to manage and show the proper window. (i.e. in case of mgt_op_trap the lme_var_id and the attr_flag are not relevant: trap_ids are uniquely defined in lme_description files, so a packet containing trap information always holds the same set of attributes).

A.7.3. Network Station Controller Module



Upon receipt of a motif_obif_if_buf from the Operator Interface Controller, functions of this module:

Check whether the requested management operation is snmp_request invoked by the Operator, if so (in this case the mgt_op_id field of the motif_obif_if_buf structure is equal to mgt_op_get, mgt_op_set, or mgt_op_gnxt), the nma_id, lme_id, lme_var_id, lme_var_option_id and attr_flag fields of the motif_obif_if_buf are fetched, the Operator Interface Controller is informed to send the trlr_buffer (trlr_buffer contains the data-structure of the lme-variable selected by the operator). Upon receipt of the trlr_buffer, the nma_id and the lme_id are used to fetch the community name entry from the community_name table. When the entry is found, the community name entry is copied into the community_name field of the nms_pe_if_buf (otherwise the operator is informed), the byte-string containing the lme-var-structure is the copied into the trlr_buf, and size_of_trlr_struct will be set to the length of trlr_buf. The last thing to do is then to construct the variable-bindings-list of the outgoing snmp-packet, (see section 3.6, NMS_make_var_bind_list Sub- Module), to set mgt_op_id, version_numb, req_id, snmp_err, snmp_err_indx,

snmp_gen_trap, snmp_spc_trap, time_stamp and var_bind_list_arr_len fields of the nms_pe_if_buf structure to their appropriate values and to convey this structure to the Protocol Entity.

A.8. The Protocol Specification of the OBIF Management Packets

All OBIF Management Protocols use the same frame structure shown in Figure A-6.a. The Protocol Entities residing in Management Entities of OBIF convert this structure to its XDR standard representation and convey the resulted XDR Stream to their underlaying Transport Layer. A detailed description of the elements of this structure is given in section 4.1.3.

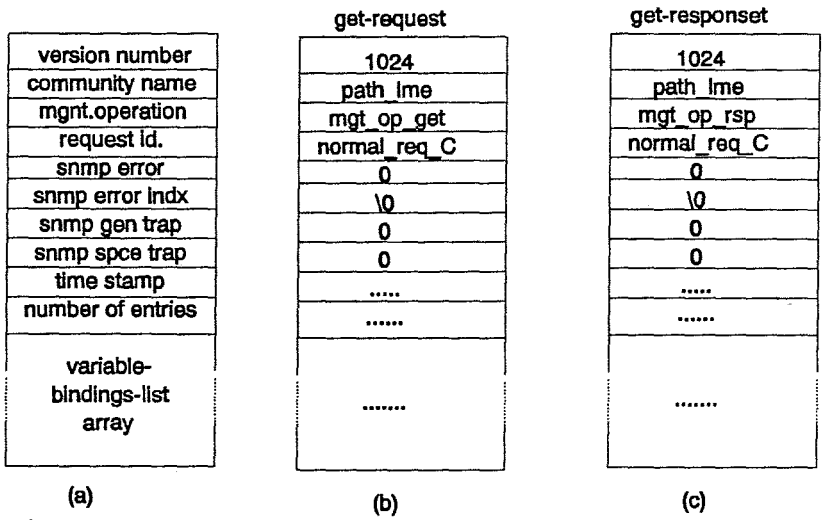


Figure A-6. Frame-structure of request and response packets

Figures A-6.b and A-6.c show a get-request packet and its related get-response packet (i.e. snmp error is set to 0 in get-response meaning that no exception has been occurred by processing the packet in LME). The set-request packet and get-next-request packet are analogous, except that management operation is mgt_op_set or mgt_op_gnxt.

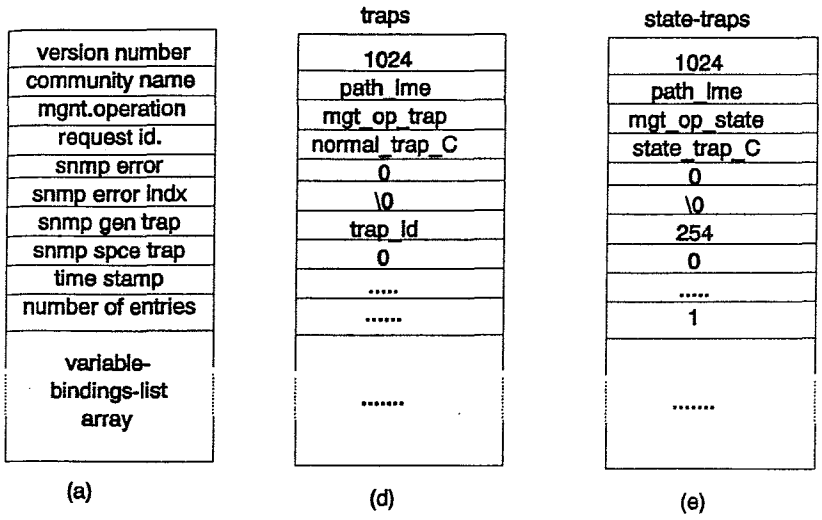
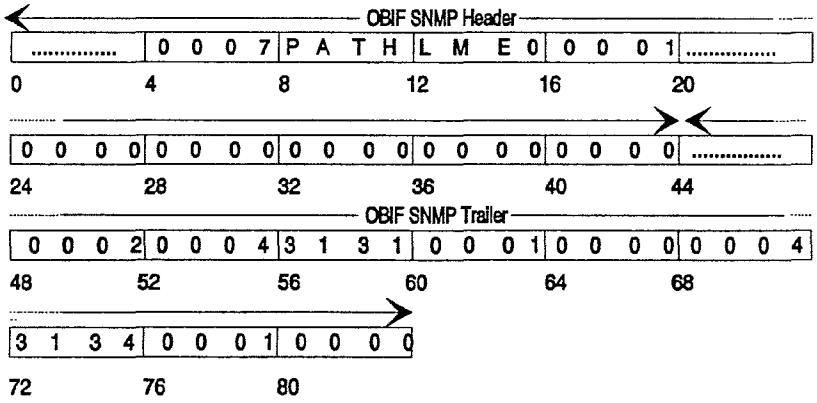


Figure A-7: Frame-structure of traps and state-trap packets.

The trap_request packet and the registration_trap_request packet are shown in Figure A-7.d and A-7.e (i.e. the deregistration_trap_request contains exactly the same data as the registration_trap_request except that snmp generic trap field is 255). The value of the field named number of entries in Change_mode_trap packets (i.e. deregistration and registration trap packets) is always 1, because there exists only one name_value pair in the packet sent by the LME (i.e. the name_value pair of the LME_Variable Mode).

The following example gives the encoded get_request packet (i.e. the XDR Stream) sent by the NMS to the PATH_LME managed by the NMA residing in OSLGW. For the sake of simplicity, we suppose that the get_request packet contains only the two attributes named LEP_id and AP_id both items of the LME_Variable named LEP_ENTRY (see Figure 2-2).



offset 0	version_number
offset 4	length of community name
offset 8	community name
offset 16	management operation
offset 20	request identifier
offset 24	SNMP error
offset 28	length of snmp error index
offset 32	snmp error index
offset 36	snmp generic trap
offset 40	snmp specific trap
offset 44	time stamp
offset 48	number of attributes
offset 52	length of attribute name (attribute 1)
offset 56	attribute name (attribute 1)
offset 60	attribute type (attribute 1)
offset 64	attribute value (attribute 1)
offset 68	length of attribute name (attribute 2)
offset 72	attribute name (attribute 2)
offset 76	attribute type (attribute 2)
offset 80	attribute value (attribute 2)

The get-response packet send by the PATH-LME contains the same data shown above except that bytes 16 to 20 hold the integer 2 (see section 4.1 for a detailed description of management operations), and bytes 64 to 68 and 80 to 84 hold the value of the appropriate attributes.

A.9. The Processes of the Network Management Station

There are five processes involved in the operation of the Network Management Station, which provide services of the session, presentation and application layers. These are as follows:

- The Operator Interface Controller Process, performs those actions described in section A.3.2 (also refer to request & response handling functions in section A.4.2);

- The NMS Controller which is divided in two processes, one handles incoming packets (these packets are already decoded by the Protocol Entity), sent by the NMAs, and the other, handles operator's requests (see section A.3 and A.7.3. for a detailed description on actions performed by these two processes).

- The Protocol Entity is also realized by two processes, one handles incoming data, namely the XDR Stream and LEP_id of the virtual connection, from the Transport Layer and the other handles outgoing data (see section A.4).

Process communication and synchronization is realized by using pipes. All read and write system calls performed on pipes are blocking_read/write.

```
process A:                          process B:

forever                             forever
   write(pipeAtoB[1],...);             read(pipeAtoB[0],...);
   read(pipeBtoA[0],...);              write(pipeBtoA[1],...);
end                                 end
```

The following points have been taken into account to avoid deadlocks:

a) length of exchanged data is always known to both A and B.
b) after process A has sent data over pipe pipeAtoB to B, it waits until acknowledge data is received over pipeBtoA (which is sent by B).

c) the first action performed by process B after receiving data over pipe pipeAtoB sent by process A is to send acknowledge data over pipe pipeBto to process A.

All processes call UNIX write() and read() system calls directly (see Figure A-8) to read to and write from pipes. The only exception is the receiving side of the Operator Interface Controller which receives incoming packets from the NMS

Controller over a pipe using the XtAddInput call (i.e., a X-Window call):

```
Operator I/F:                    NMS Controller(receiving):

forever                          forever
   XtAddInput(pipe1[0],...);        write(pipe1[1],...);
   write(pipe2[1],...);             read(pipe2[0],...);
end                              end
```

The two processes providing services of the NMS Controller have both access to the MDB Files over their NMS_MDB_cntrl procedures. There is no need for a mutual exclusion mechanism to avoid simultaneous access to the MDB Files by these processes, because only one of them updates the contents of the MDB Files (i.e. the NMS Controller handling responses) and the other one just reads them (i.e. the NMS Controller handling requests).

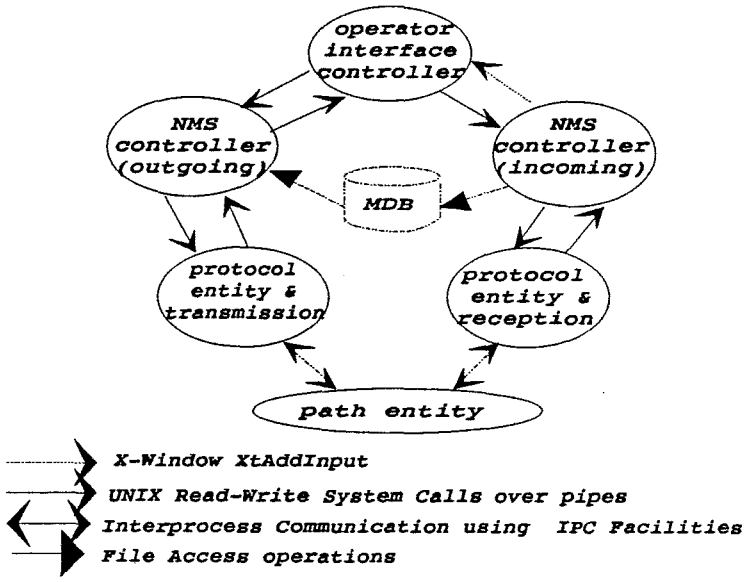


Figure A.8) Interprocess communication among NMS processes

B. NMA & LME

B.1. Network Management Agent

This section describes the breakdown of a system that supports the agent role of the OBIF Management Model. This model will be referred to as the Network Management Agent (NMA).

The following figure shows the breakdown of an NMA.

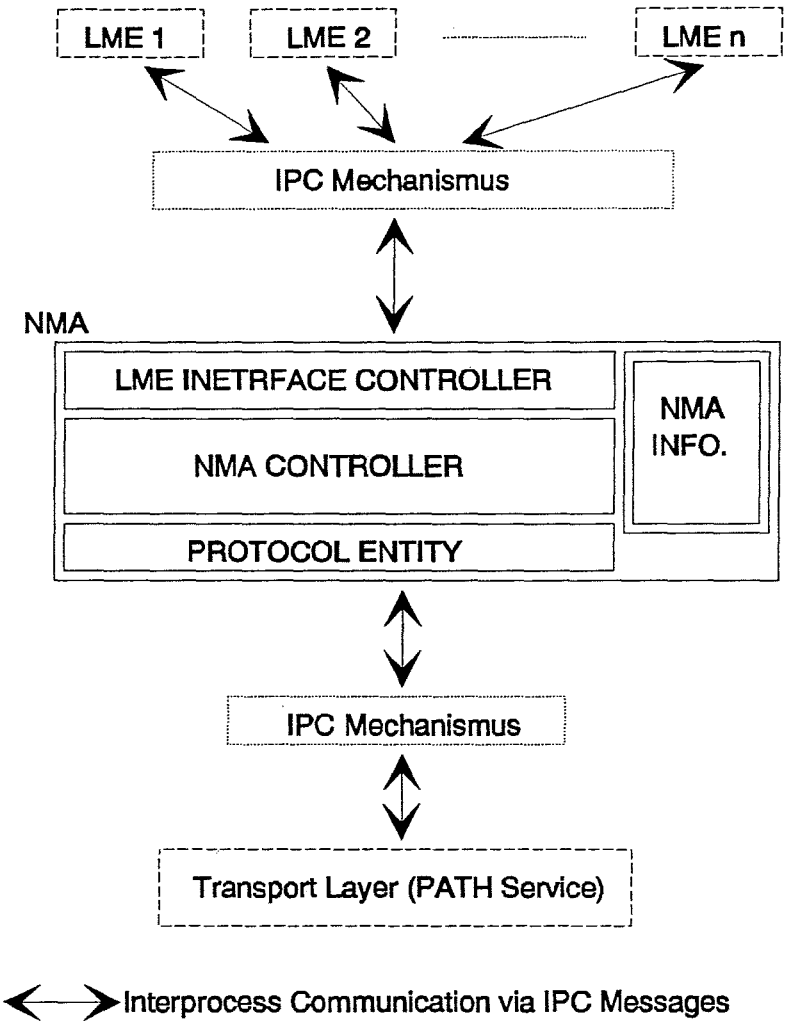


Figure B.1. Breakdown of OBIF NMAs

The following figure shows the data-flow between different modules of an NMA.

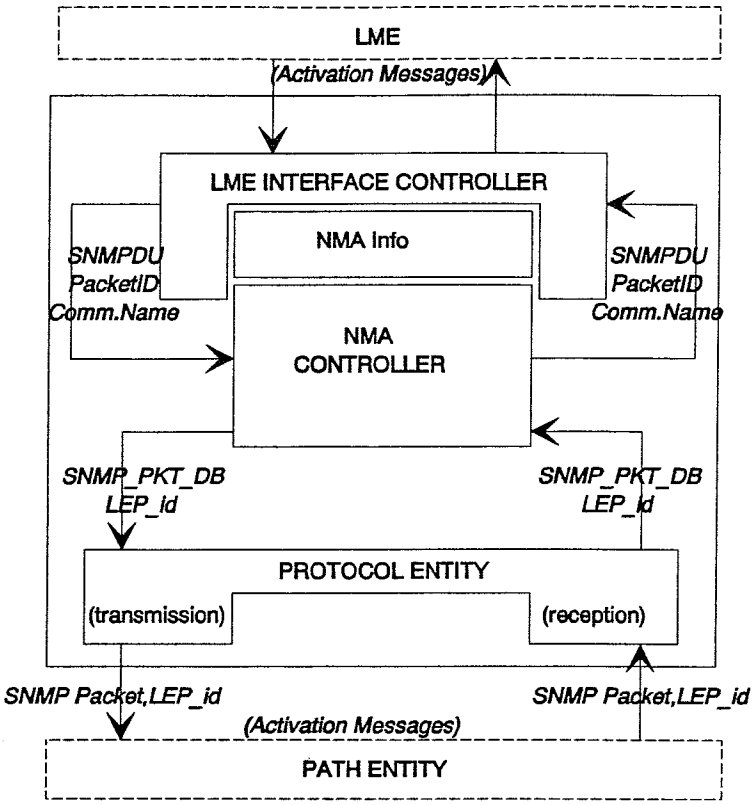


Figure B.2. Data-Flow in NMA

B.1.1. Protocol Entity

The Protocol Entity is responsible for translating incoming requests (i.e. request packets sent by NMS) and incoming responses (i.e. responses sent by LMEs) between the Transfer Syntax of the Protocol Entities (i.e. XDR) and the Local Syntax of the NMA (e.g. C-Data-Structures).

The Protocol Entities residing at Network Management Station and Network Management Agents communicate with each other using the the services of the Transport Layer's Octet String Service [KOZ91-2]. Management Information is exchanged between NMS and NMAs using the Simple Network Management Protocol (SNMP).

Each SNMP Packet consists of a version identifier, an SNMP community name and a protocol data unit (PDU). The header of a SNMP Packet contains the version identifier and the community name. The trailer contains the PDU.

Upon Receipt of a SNMP Packat, the Protocol Entity decodes the Message, stores the results in a buffer specified by SNMP_PKT_DB (i.e. SNMP Packet Description Block), and sends the modified incoming packet to the NMA Controller Module along

with the LEP_id and a parameter named errin which indicates exceptions (for example: failure by parsing the incoming message).

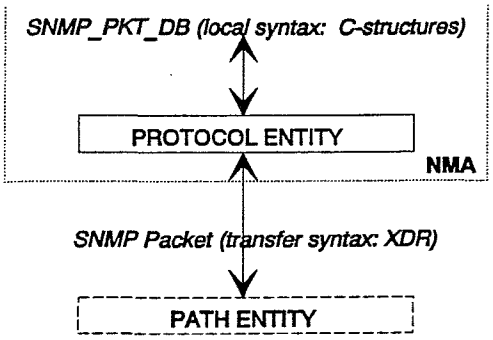


Figure B.3. Protocol Entity

The parameter SNMP_PKT_DB holds the address of the buffer which contains the SNMP Packet sent by the NMS. It is composed of a header part which contains the decoded SNMP Header and a trailer part which contains the SNMPDU.

The local representation of the SNMP_PKT_DB is as follows:

The header is a C-Data-Structure of type snmp_hdr_type representing the SNMP Header:

```
struct snmp_hdr_type
 {snmp_vers_type        version;
  snmp_comm_type        community;
 }header;
```

and the trailer is a fixed length array of integer representing the SNMPDU:

```
int trailer[MaxPduLen];
```

Note: The Protocol Entity must decode the header of the incoming SNMP Packet because the field named community is used to mapping the incoming Packets to the appropriate LMEs. The contents of the trailer are trasparent to the NMA.

B.1.2. NMA Controller Module

The Network Management Agent Controller Module interacts with the LME Interface Controller and the Protocol Entity both residing in the NMA. It is the responsibility of this module to control and monitor the flow of management information exchanged between the NMS and the LME, and also to provide services to the NMA to operate as expected.

The placement of this component implies that it has to be traversed by each management request. This component performs access control on all requests received from other Network Elements.

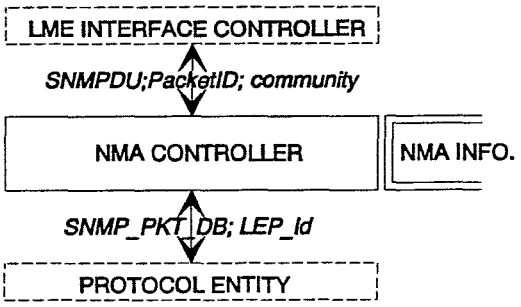


Figure B.4. Interfaces of the NMA Controller Module

This component receives data from the Protocol Entity over the following three parameters:

SNMP_PKT_DB: This parameter contains the SNMP Packet sent by the NMS (already decoded to the Local Syntax of the NMA by the Protocol Entity);
LEP_id: This parameter is used by the NMA Controller Module to specify the originator Protocol Entity (at least in the first phase of OBIF implementation -this parameter is optional; because the only originator Protocol Entity that communicates with the receiving Protocol Entities residing in NMAs is that of the Network Management Station. On the other hand implementing this parameter will simplify realizing facilities which have to support more than one sending Protocol Entity in the future; e.g. Cross Support [6]).
errin: This parameter is used to indicate that an exception occurred while processing the incoming SNMP Packet in the Protocol Entity. In case of such an exception, the NMA Controller Module performs no further actions on the incoming SNMP Packet stored in SNMP_PKT_DB, it just conveys it to the sending Protocol Entity with errout = errin.

The NMA Controller Module sends data to the Protocol Entity over the following three parameters:

SNMP_PKT_DB: This parameter contains the SNMP Packet which is to be sent to the NMS (the Protocol Entity will encode this message).
LEP_id: This parameter is used by the Protocol Entity to specify the destination Protocol Entity (at least in the first phase of OBIF implementation -this parameter is optional; because the only destination Protocol Entity that

communicates with the sending Protocol Entities residing in NMAs is that of the Network Management Station).

errout: this parameter is used to indicate that an exception occurred while processing the incoming SNMP Packet in the receiving Protocol Entity.

The NMA Controller Module sends data to the LME Interface Controller Module over the following three parameters:

SNMPDU: This parameter contains the PDU of the incoming SNMP Packet (i.e. the C-Data-Structure named trailer).

PacketID: This parameter is generated by the NMA Controller Module and is used to distinguish among SNMP Packets which traverse the NMA. Each SNMP Packet arriving at NMA Controller Module is assigned to a unique identifier, the PacketID. The PacketID is used to enable the NMA Modules to handle the SNMP Packets properly.

community: This parameter contains the community field of the header of the arriving SNMP Packet (the community field contains an identifier which is used by the LME Interface Controller to specify the destination LME).

The NMA Controller Module receives data from the LME Interface Controller Module over the following four parameters:

SNMPDU: This parameter contains the PDU of the outgoing SNMP Packet (i.e. the C-Data-Structure named trailer).

PacketID: Packet identifier.

community: This parameter contains the community field of the header of the SNMP Packet which is to be sent to NMS.

error: This parameter is used to indicate that an exception occurred in the LME Interface Controller Module.

The NMA Controller Module retrieves the information that is required for its correct operation from the Network Management Agent Info (NMA INFO) component.

PacketIDs and their associated community names are stored in the NMA INFO. This information enables the outgoing Packet Handler Module (will be discussed later) to map each outgoing SNMP Packet to its corresponding incoming SNMP Packets (note: from now on we will use the term SNMP-Node to refer to the information stored in NMA Info about each incoming Packet).

The NMA Controller Module is broken down into sub-modules, as shown in the following figure:

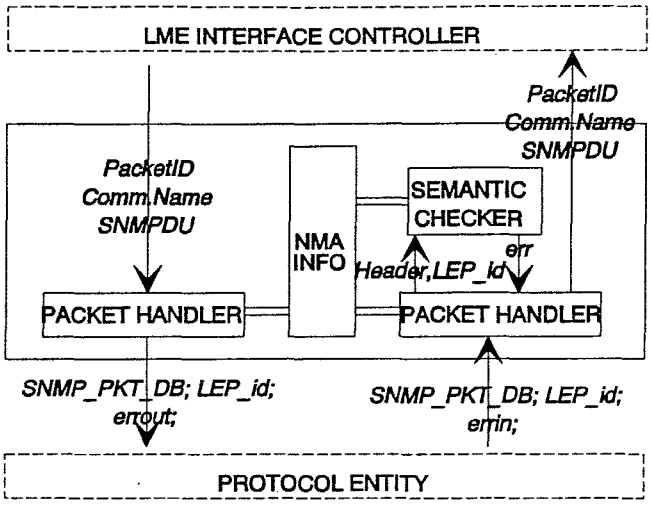


Figure B.5. DATA-FLOW in NMA Controller Module

In the following we will use the term 'receiving' to refer to those sub-modules which handle incoming management data and prepare it for further processing; and the term 'sending' to refer to those sub-modules which participate in generating outgoing management data (e.g. the Packet Handler Sub-Module residing at the left hand side of the figure 3.5. is called the receiving Packet Handler Sub-Module).

B.1.2.1. Packet Handler Sub-Module (receiving)

The following figure shows the breakdown of the receiving Packet Handler Sub-Module:

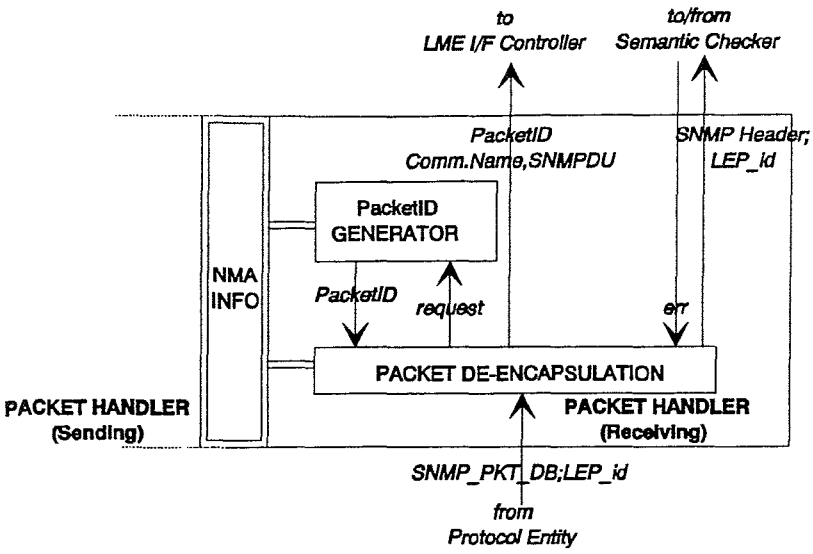


Figure B.6. DATA-FLOW in (receiving) Packet Handler Sub-Module

Upon receipt of a SNMP Packet the receiving Packet Handler Sub-Module performs the following actions

it opens the incoming SNMP Packet and then passes the header to the Semantic Checker Sub-Module along with the LEP_id;

if the parameter err returned by the Semantic Checker Sub-Module indicates

- no failure, then the Packet Handler Sub-Module generates an identifier named PacketID and passes this along with the SNMPDU (i.e. the trailer part of the SNMP Packet) and the community name to the LME Interface Controller Module for further processing;
- some kind of exception (e.g. unauthorized requester of Management Information), then it stores the header and the trailer along with the LEP_id in the NMA Info. The sending Packet Handler Sub-Module retrieves this information and performs the appropriate operations; for example: it reconstructs the SNMP Packet, passes it to the sending Protocol Entity along with the LEP_id in order to send it back to the NMS.

Splitting the SNMP Packet in a header and a trailer part is done by the De-encapsulation Sub-Module.

Upon receipt of a PacketID_request invoked by the Packet De-encapsulation Sub-Module (the PacketID_request is not invoked if the Semantic Checker Sub-Module returns an error), the PacketID Generator Sub-Module generates a PacketID identifier.

B.1.2.2. Semantic Checker Sub-Module

The functionality of this Sub-Module was implicitly described, as we covered services provided by other Modules.The Semantic Checker Sub-Module

performs access control functions; that means it checks the LEP_id to determine whether the requester (the SNMP Application Entity of the requester) is registered in the NMA Info and
checks the version number, if necessary.
It returns the parameter err to the Packet Handler Sub-Module to indicate exceptions (like unauthorized access, version number mismatch, internal error). In this case err is not equal to the predefined value NoError.

B.1.2.3. The Packet Handler Sub-Module (sending)

The following figure shows the breakdown of the sending Packet Handler Sub-Module:

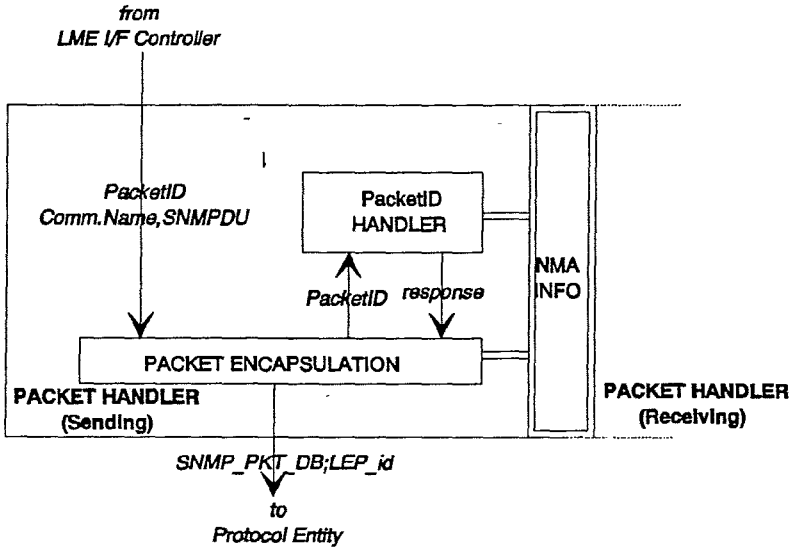


Figure B.7. : DATA-FLOW in (sending) Packet Handler Sub-Module

Upon receipt of a SNMPDU the sending Packet Handler Sub-Module performs the following actions if the parameter error received from LME Interface Controller is equal to the predefined value NoError:

the Packet Encapsulation Sub-Module passes the PacketID identifier received from the LME Interface Controller to the PacketID Handler Sub-Module;

B.1.2.2. Semantic Checker Sub-Module

The functionality of this Sub-Module was implicitly described, as we covered services provided by other Modules.The Semantic Checker Sub-Module

performs access control functions; that means it checks the LEP_id to determine whether the requester (the SNMP Application Entity of the requester) is registered in the NMA Info and
checks the version number, if necessary.
It returns the parameter err to the Packet Handler Sub-Module to indicate exceptions (like unauthorized access, version number mismatch, internal error). In this case err is not equal to the predefined value NoError.

B.1.2.3. The Packet Handler Sub-Module (sending)

The following figure shows the breakdown of the sending Packet Handler Sub-Module:

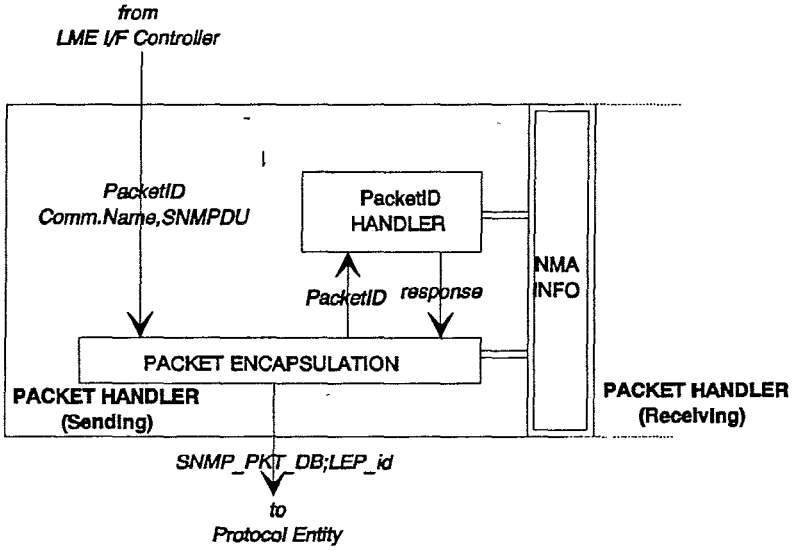


Figure B.7. : DATA-FLOW in (sending) Packet Handler Sub-Module

Upon receipt of a SNMPDU the sending Packet Handler Sub-Module performs the following actions if the parameter error received from LME Interface Controller is equal to the predefined value NoError:

the Packet Encapsulation Sub-Module passes the PacketID identifier received from the LME Interface Controller to the PacketID Handler Sub-Module;

the PacketID Handler Sub-Module retrieves the LEP_id from the NMA Info component using the PacketID identifier and returns one of the following by invoking a PacketID_response:

the LEP_id of the requester and if necessary the version number;

a parameter indicating some exceptional case like:
no SNMP-Node under this PacketID;
no more SNMP-Node under this PacketID because of requester_go_down.

If the PacketID_response returns a LEP_id then the Packet Encapsulation Sub-Module constructs the header using the version number returned by PacketID Handler Sub-Module and the community name sent by LME Interface Controller; then a SNMP Packet is built by concatenating this header and the SNMPDU received from the Interface Controller. The SNMP Packet and LEP_id are then passed to the sending Protocol Entity.

If for some reason the PacketID_response returns a parameter indicating an exception then actions performed by Packet Encapsulation Sub-Module are case-dependent.

B.1.3. NMA Info

The NMA Info (and the NMA-LME Info) are logical components (we will use the term 'passive' to refer to them and the term 'active' to refer to components, which symbolize processes or objects that can be activated). Info components represent an abstract view of the whole bundle of data exchanged between NMA Modules, or required by these modules to function as expected.

We will classify the information stored in the Info components as follows:

configuration information: contains static information on the components of the NMA. This information is distributed among Info components of different NMAs (also among Info components of LMEs) without using services provided by the Network Management System. Consider the table shown in the following:

community name	destination LME
PATH	*PATH_IPC_ObjectID*
VCLC	*VCLC_IPC_ObjectID*
Video	*Video_IPC_ObjectID*

This table is used by the LME Controller Module to specify the destination LME by using the community name of an incoming SNMP Packet. The address of this table and its contents can be stored in NMA Info during the system set-up phase. Configuration Information can also be hard-coded (e.g. the address of the above table can be hard-coded).

conceptional information: refers to the information exchanged between active components of a network element. For example, the pointer to the C-Data-Structure SNMPDU, which is sent to the LME Interface Controller by the NMA Controller Module.

management information: refers to the information exchanged between network elements using management protocols; e.g. the contents of a SNMP or the contents of the C-Data-Structure SNMPDU.

The NMA-LME Info Module contains the information that is necessary for the correct operation of the LME Interface Controller Module (i.e. a description of LMEs managed by the NMA, information about Activation Messages and how to use services provided by the IPC Mechanism and etc.).

The essential information about the NMA itself is stored in the NMA Info Module (i.e. a description of the NMS, access and export control information, structure of SNMP-Node, Time-Out, version number of Management Information Bases (MIBs) supported by the NMA and etc.).

Active components perform read, write or readwrite operations on passive components.

B.1.4. LME-Interface Controller Module

The LME-Interface Controller Module provides services to the NMA so it can communicate with Local Management Entities residing in the same crate using the Activate Operation service provided by the IPC Mechanism. The LME-Interface Controller functions forward request primitives sent by the NMA to the LMEs and return response primitives sent by the LMEs to the NMA. To achieve this, the LME Interface Controller must have access to the reliable information about LMEs managed by NMA, which is stored in NMA-LME Info (e.g. a mapping of community names to ObjectIDs of LMEs in order to route management requests to the appropriate Local Management Entity; how to interact with IPC Mechanism or how to construct an Activation Message; etc.).

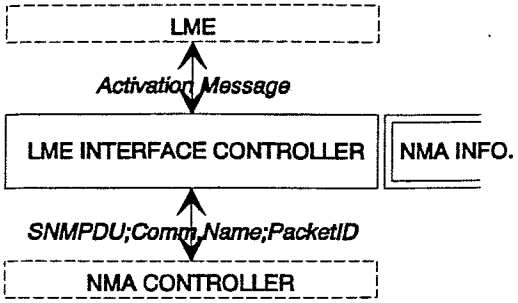


Figure B.8. Interfaces of the LME Interface Controller Module

We have already discussed the interface between the NMA Controller Module and the LME Interface Controller Module (please see NMA Controller Module).

Sending Activation Messages to the LMEs:

In order to send a SNMPDU to the LME specified by the parameter named 'community' the Construct Activation Message Sub-Module of the LME Interface Controller Module constructs an Operation Activation Message which contains:

the Object ID of the sending object (i.e. the Object ID of the LME Interface Controller);
the Object ID of the receiving Object (i.e. the Object ID of the LME);
the Operation ID of the operation that should be activated;
the Error Code and the Error Operation ID;
two Parameters (i.e. the PacketID and a pointer to the SNMPDU which will be sent);
and the Message Length.

Receiving Activation Messages from the LMEs:

Upon receipt of an Operation Activation Message, the Handle Activation Message Sub-Modue of the LME Interface controller Module retrieves

the Source Object ID field to determine the sending LME;
the PacketID parameter;
and the pointer to the SNMPDU.

The Source Object ID is converted to community name to determine the sending LME and is sent along with error, the incoming SNMPDU and the PacketID parameter to the NMA Controller Module.

The following figure shows the breakdown of the LME Interface Controller Module:

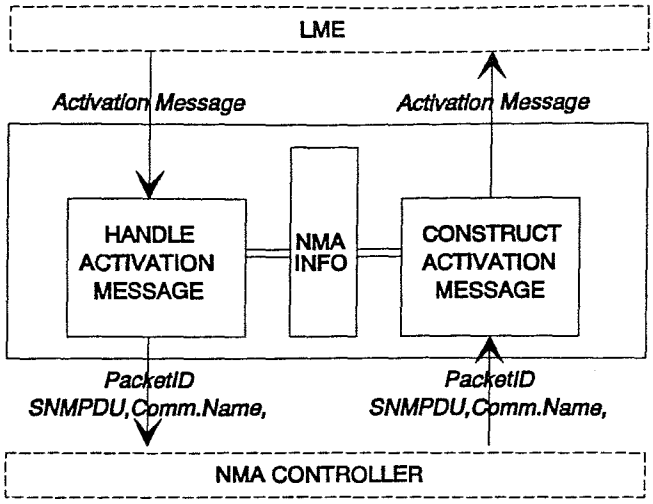


Figure B.9. Breakdown of the LME Interface Controller Module

B.2. Local Management Entity

The following figure shows the breakdown of a LME:

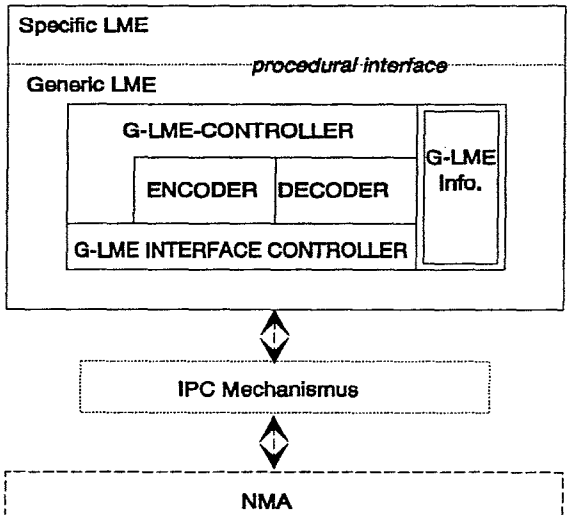


Figure B.10. : Breakdown of the G-LME

The Encoder/Decoder component of LME decodes the contents of the SNMPDUs sent by NMA to the Local Syntax of the G-LME and encodes SNMPDUs sent by S-LME to a SNMP specific representation (i.e. Transfer Syntax: XDR).

Implementation of this component is optional if the Local Syntax of NMA and the Local Syntax of LME are identical. If they are not identical

the incoming SNMPDUs are decoded, then sent to the G-LME Controller Module and after that mapped to a C-Data-Structure.

the C-Data-Structure sent by S-LME will be transformed to the Local Syntax by the G-LME Controller and then passed to the Encoder. The encoded SNMPDU is then sent back to the Interface Controller.

The following figure shows the Data-Flow between the different LME Modules:

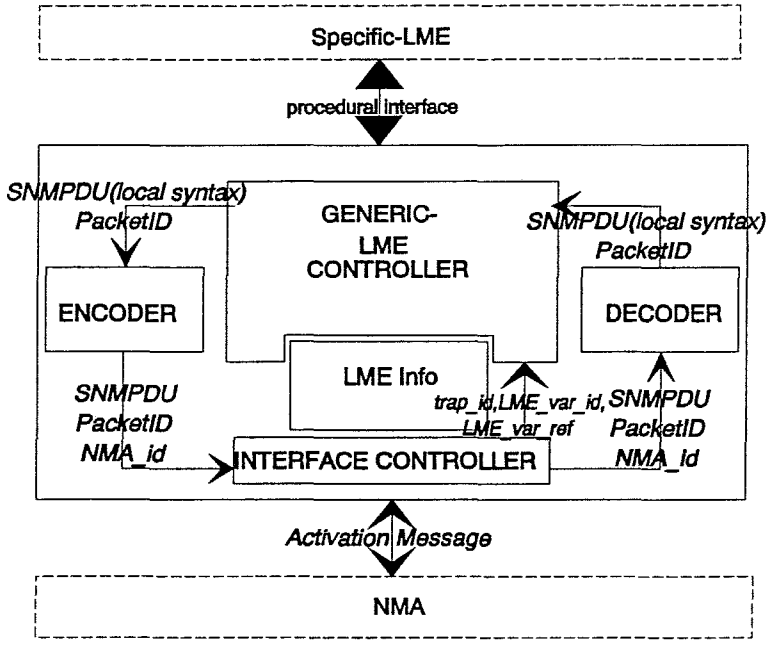


Figure B.11. DATA-FLOW in G-LME

B.2.1. Interface Controller Module

The Interface Controller Module provides services to the LME so it can communicate with the NMA using the Activate Operation service provided by IPC.

The following figure shows the interfaces of the Interface Controller Module:

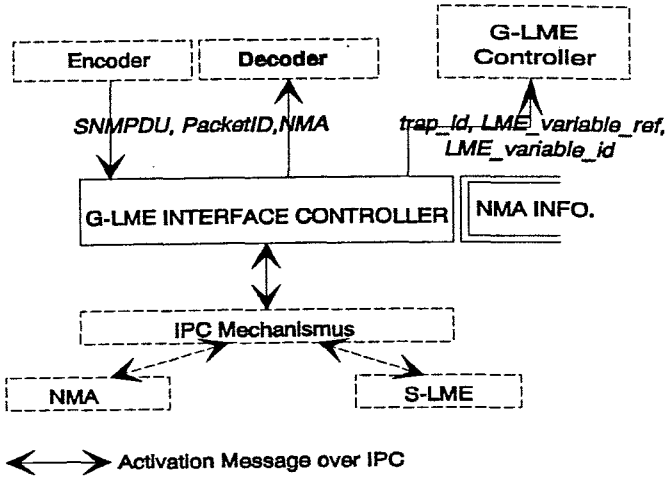


Figure B.12. Interfaces of the Interface Controller Module

Upon receipt of an Operation Activation Message the Interface Controller Module retrieves:

the Source Object ID field to determine the sending NMA (in the first Implementation Phase of OBIF retrieving this field would be optional because it always shows the Source Object ID of the NMA, and in case of Trap-PDUs, the Source Object ID of the LME itself).

If the value of the Operation ID field of the Activation Message is equal to the predefined value PROCESS_G_LME_REQ_C (meaning that the Activation Message is sent by the NMA) then the Interface Controller Module retrieves the first parameter which contains the PacketID; and the second parameter which is a pointer to the SNMPDU.

The incoming SNMPDU and the PacketID parameter are sent to the Decoder Module for further processing. Exceptions are notified by the parameter error (e.g. internal error by interpreting the Activation Message).

If the value of the Operation ID field of the Activation Message is equal to the predefined value PROCESS_TRAP_REQ_C (meaning that the Activation Message is sent by the LME itself), i.e. a Activation Trap Message (see Error Handling), then the Interface Controller Module retrieves:

the first parameter which in this case is the trap_id; trap_id specifies the event occurred in Managed Object;

and the second parameter which contains the LME_variable_ref; LME_variable_ref is a reference to a C-Data-Structure that

represents an instance of a LME-variable containing the attribute which caused the event;

and the third parameter named LME_variable_id which specifies the type of the LME-variable.

In order to send a SNMPDU to the NMA the Interface Controller Module constructs an Operation Activation Message which contains:

the Object ID of the sending object (i.e. the Object ID of the Interface Controller);

the Object ID of the receiving Object (i.e. the Object ID of the NMA)

the Operation ID of the operation that should be activated; Operation ID is predefined and known to the LME;

the Error Code and the Error Operation ID;

the two Parameters: PacketID, and a pointer to the SNMPDU which is encoded by the Encoder Module and should be sent to NMS (in case of a Trap-SNMPDU the contents of PacketID is not relevant and can be used for other purposes);

and the Message Length.

Note: The implementation of the parameter named NMA which is exchanged between the Interface Controller Module and the G-LME Controller Module (this parameter is transparent for Decoder and Encoder Modules) is optional, if each LME communicates with one and only one entity - the corresponding NMA - and not with other LMEs).

The following figure shows the breakdown of the Interface Controller Module:

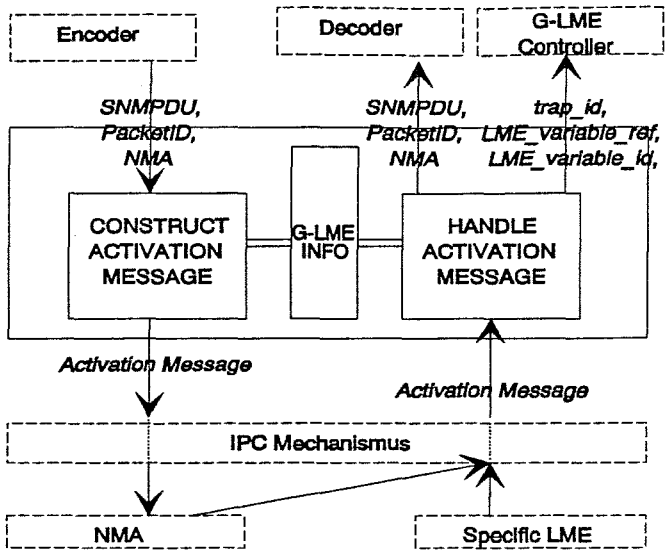


Figure B.13. Breakdown of the Interface Controller Module

B.2.2. G-LME Controller Module

The following figure shows the interfaces of the G-LME Controller Module:

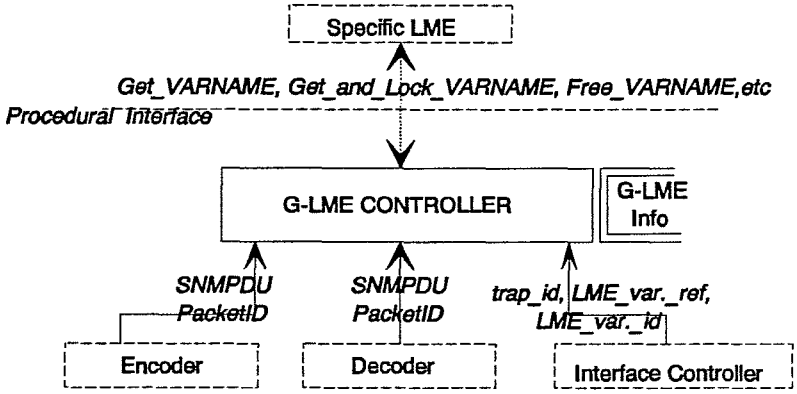


Figure B.14. Interfaces of the G-LME Controller Module

This component receives the following three parameters from the Interface Controller Module:

trap_id: specifies the event occurred in Managed Object
LME_variable_ref: reference to the C-Data-Structure that represents the instance of the LME-variable that contains the attribute which has caused the event;

LME_variable_id: specifies the type of the LME-variable.

This component receives the following three parameters from the Decoder Module:

SNMPDU: the pdu part of the incoming SNMP Packet decoded to the Local Syntax of the LME by the Decoder Module;

PacketID: the packet identifier associated with the SNMPDU (this identifier is transparent to the decoder);

errin: this parameter is used to indicate that an exception occurred while

interpreting the incoming Activation Message in the Interface Controller Module (in this case the value of errin is equal to the value of the parameter error sent to Decoder by Interface Controller);

decoding the incoming SNMPDU in Decoder.

This component sends the following three parameters to the Encoder Module:

SNMPDU: the pdu part of the outgoing SNMP Packet (this pdu will be encoded by the Encoder Module from the Local Syntax of the LME to the SNMP specific representation, i.e. the Transfer Syntax: XDR).

PacketID: the packet identifier associated with the SNMPDU (this identifier is transparent to the Encoder);

errout: this parameter is used to indicate that an exception occurred while

interpreting the incoming Activation Message in the Interface Controller Module or decoding the incoming SNMPDU in the Decoder Module (in this case the value of errout is equal to the value of the parameter errin) or

processing the incoming SNMPDU in the G-LME Controller Module itself (e.g. no suitable C-Data-Structure found).

The G-LME Controller Module is broken down into sub-modules, as shown in the following figure:

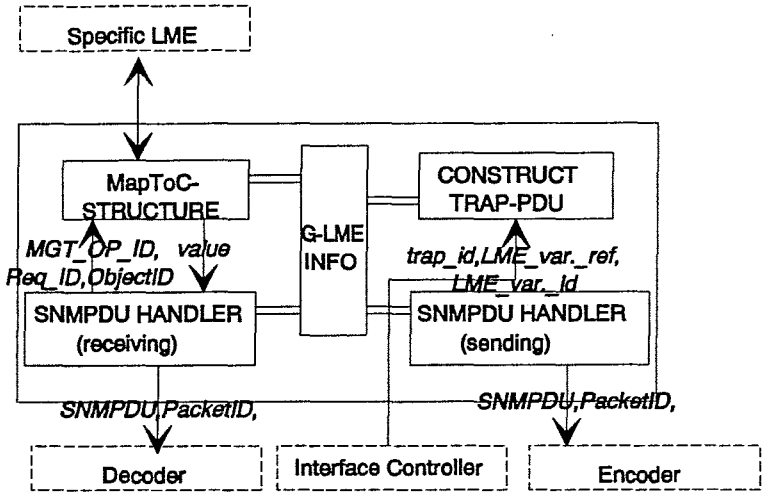


Figure B.15. Data-Flow in the G-LME Controller Module

B.2.2.1. SNMPDU Handler Sub-Module (receiving)

Passing the incoming SNMPDU in form of C-Data-Structures to the S-LME Controller Module:

Each incoming SNMPDU consists of the following fields, all of them already decoded into the Local Syntax of the G-LME by the Decoder Module:

the request-id: is used to distinguish among outstanding requests, duplicated requests and so on;

the error-status: is used to indicate exceptions;

the error-index: provides additional information on the exception identified in error-status (e.g. which attribute of the Managed Object has caused the exception);

variable-bindings list: is a list of variable bindings. Each variable binding refers to the pairing of the name of a variable to the variable's value.

Upon receipt of a SNMPDU, the SNMPDU Handler Sub-Module performs the following top-level actions:

it retrieves the request-id field of the SNMPDU, so it can distinguish among outstanding requests and then performs the following actions:

1. Retrieve the OBJECT IDENTIFIER of the first variable.

Note: This is the OBJECT IDENTIFIER which specifies an attribute of a Managed Object and not the OBJECTID used to specify objects in IPC Mechanism.

2. Send the ObjectID (contains the OBJECT IDENTIFIER) and the ReqID (contains the request-id) and the MGT_OP_ID (a parameter that specifies the type of the SNMPDU - i.e. get, getnext or set) to the MapToC-Structure Sub-Module; in case of a set-request, store the set-value in parameter dummy and pass it to the MapToC-Structure Sub-Module.

3. Get the two parameters error and value returned by MapToC-Structure Sub-Module.

4. If the parameter named error is not noError then, copy this parameter to the error-status field and the OBJECT IDENTIFIER to the error-index field of the SNMPDU and go to 7.

5. Copy the parameter named value to the appropriate field in SNMPDU.

6. If the variable handled above is not the last variable in the variable-bindings field then retrieve the OBJECT IDENTIFIER of the next variable and go to 2.

7. Inform the sending SNMPDU Handler Sub-Module to send the SNMPDU and the associated PacketID to the entity specified by the parameter named NMA.

8. Inform the MapToC-Structure Sub-Module to invoke the Free_VARNAME or the Unlock_VARNAME [KOZ91-1].

B.2.2.2. MapToC-Structure Sub-Module (receiving)

The MapToC-Structure Sub-Module performs the following actions after being called by the SNMPDU Handler Sub-Module:

It uses the MGT_OP_ID and the OBJECT IDENTIFIER received from the SNMPDU Handler Sub-Module to invoke either

the Get_VARNAME(params), if the MGT_OP_ID indicates a GetRequest-primitive,

the Get_Next_VARNAME(params), if the MGT_OP_ID indicates a GetNextRequest- primitive or

the Get_and_Lock_VARNAME(params), if the MGT_OP_ID indicates a SetRequest-primitive.

The prefix VARNAME identifies the LME-variable addressed in the primitive. The MapToC-Structure Sub-Module retrieves the appropriate VARNAME each time it receives an OBJECT IDENTIFIER from the underlying SNMPDU Handler Sub-Module.

After the required data is returned by the S-LME (in params) the MapToC-Structure Sub-Module selects the attribute specified by OBJECT IDENTIFIER and

returns the value of that attribute to the SNMPDU Handler Sub-Module if the management operation requested was permitted. In this case the parameter named error contains the predefined value noError.

if the requested management operation could not be finished successfully because of

an internal error then the parameter named error contains the predefined value Internal; the SNMPDU Handler Sub-Module should then take care of the next processing steps (e.g. informing the NMA or just discarding the Datagram).

an error associated with the requested data then the value of the parameter error returned is NoSuchName, tooBig, badValue, readOnly or genErr depending on the exception.

The MapToC-Structure Sub-Module then invokes

the Free_VARNAME(params) to notify the S-LME that the Get or Get-Next primitive has been finished; or

the Unlock_VARNAME(params) to notify the S-LME that the specified attribute or attributes of the Managed Object (i.e. different LME-variables) have been changed.

B.2.3. Event Handling

Events handled by the OBIF-LMEs can be devided in two groups:

1- Events observed and invoked by the S-LME and
2- Events observed and invoked by the G-LME.

Group 1

The notification of events on the MOs is a matter of the S-LME. Events are notified to the NMS by rising a SNMP-Trap. This is

done by the S-LME by calling a Trap_VARNAME() routine provided by the G-LME.

The parameters provided to Trap_VARNAME() are coded to Activation Trap Messages. They are IPC-messages sent to the generic LME with Operation ID set to PROCESS_TRAP_REQ_C. In addition Trap_VARNAME() invokes the logging of this trap using the LOG_msg service of the OBIF-Logger. Since the NMS itself displays a Trap-Notification on the WS, traps are logged with log_type = Log.

The Interface Controller Module of G-LME can easily distinguish between Activation Trap Messages (sent by S-LME) and normal Activation Messages (sent by NMA) since the IPC invokes different operations of this module.

The arrival of an Activation Trap Message is notified to the Construct-Trap-PDU Sub-Module of the G-LME Controller (note: Activation Messages sent by NMA are notified to the receiving SNMPDU Handler Sub-Module of the G-LME Controller).

In case of an Activation Trap Message, the Interface Controller Module retrieves the three parameters Trap_id, LME_variable_ref and LME_variable_id from the message and passes these three to the Construct-Trap-PDU Sub-Module.

Information associated with these three parameters is retrieved from G-LME Info component. Then, the Construct-Trap-PDU Sub-Module constructs an SNMPDU and conveys it to the receiving SNMPDU Handler. From now on, the actions performed by the SNMPDU Handler Sub-Module are exactly those discussed in B.2.2.1. The parameters passed over the interface between the receiving SNMPDU Handler and the MapToC-Structure are also those discussed in B.2.2.1. and shown in figure B.15. The only distinction is, that in case of a (Trap-)SNMPDU, the ReqID parameter is equal to the predefined value Trap.

Group 2

This group consists of the events (i.e. errors) notified by the G-LME. This includes for example:

decoding or encoding failures,
mapping failures,
unclear termination of procedures called by the G-LME and
other semantic failures.

As mentioned earlier the OBIF-NM&S-System provides an end-to-end error handling mechanism. All pertinent events, exceptions, failures and errors observed in the CIF-Nodes are conveyed to the NMS via SNMP and shown to the Operator in Real-Time. This

services is realized in the NM&S-System by using functions provided by the Event Handling Modules of the NMA, LME and the NMS.

Literaturverzeichnis

- [ALM89] ALMASI G., GOTTLIEB A., Highly Parallel Computing, Benjamin/Cummings Publishing Company, 1989.
- [AOS89-1] CCSDS 701.00-R-3, Recommendation for Space Data System Standards, Advanced Orbiting Systems, Networks and Data Links: Architectural Specification, CCSDS Recommendation, Red Book, Issue-3, June 1989 or later Issue.
- [AOS89-2] CCSDS 703.00-W-1, Recommendation for Space Data System Standards, Advanced Orbiting Systems, Networks and Data Links: Cross Support of Management and Signalling, CCSDS Recommendation, White Book, Issue-1, May 1989 or later Issue.
- [AOS89-3] CCSDS 700.00-G-1, Report Concerning Space Data System Standards, Advanced Orbiting Systems, Networks and Data Links: Summary of Concept, Rationale and Performance, Green Book, Issue-1, June 1989.
- [AOS89-4] CCSDS 701.0-B-1, Recommendation for Space Data System Standards,Advanced Orbiting Systems, Networks and Data Links: Architectural Specification, Blue Book, October 1989.
- [BAW91-1] BAWIN C., On-Board Informatics Test Facility, Network Interface Unit (NIU), DDD, SBI-OBIF-DDD-NIU, Issue 01, 15.03.91.
- [BLA92] BLACK U., Network Management Standards, The OSI, SNMP and CMOL Protocols, McGraw-Hill Inc., 1992.
- [BYT91] BYTE, March 91, Network Management.
- [CAS89] CASE J., FEDOR M., SCHOFFSTALL M., DAVIN J., The Simple Network Management Protocol (SNMP), RFC 1098, University of Tenessee, NYSERNet, Rensselaer Polytechnic Institut, MIT Laboratory for Computer Science, April 89.
- [CCITT-1] CCITT Recommendation Relationship Management Function for CCITT Applications, Colour Book, Vol. ROMAN, Rec X.732,ITU, Geneva, 19xx.
- [CCITT-2] CCITT Recommendation State Management Function for CCITT Applications, Colour Book, Vol. ROMAN, Rec X.731,ITU, Geneva, 19xx.
- [CCITT-3] CCITT Recommendation Reference Model of Open Systems Interconnection for CCITT Applications, Colour Book, Vol. VIII, Rec X.200,ITU, Geneva, 19xx.
- [DAT92] DATACOM, March 92, Sicherheit im Netzwerk.
- [DUP91] DUPUY A., SENGUPTA S., WOLFSON O. & YEMINI Y., NETMATE: A Network Management Environment, IEEE Network Magazine, March 1991.
- [ESA90] ESA PSS-04-107, Packet telecommand standard, June 1990.
- [ESA91-1] ESA PSS-05-0, ESA software engineering standards, Issue 2.
- [ESA91-2] ESA TM-14, AOS handbook, a user's guide to the CCSDS standards for advanced orbiting systems, July 1991.

- [ESA-ESTEC] ESA-ESTEC: Host to NIU prelimenary Interface Specifications, GOMEZ, Noordwijk, February 1990.
- [ESA91-3] ESA PAD-07-612, COES, Human Computer Interaction Standard, April 1991.
- [HUN74] HUNT, SULLIVAN D.E. & E.V., Between Psychology and Education, Hindsdale, I11.: Dryden Press, 1974.
- [IEEE-1] IEEE Software, Special Issue on User Interfaces, January 1989.
- [ISO74-1] ISO 7498-1: Information Processing Systems, Open Systems Interconnection- Basic Reference Model, 1984.
- [ISO74-4] ISO 7498-4: Information Processing Systems, Open Systems Interconnection, Basic Reference Model, Part 4: Management Framework, 1988.
- [ISO85] ISO/TC 97/SC 21 N 24, Information Retrieval, Transfer and Management for OSI, Information Processing - Open Systems Interconnection- Specification of Abstract Syntax Notation One (ASN.1), April 1985.
- [ISO90] ISO/IEC JTC/SC 21, Information Retrieval, Transfer and Management for OSI Secretariat: U.S.A. (ANSI), Final Text of DIS 9595, Information Technology - Open Systems Interconnection- Commonn Management Information Service Definition, January 1990.
- [ISO95-2] ISO 9594-2: Information Processing Systems, Open Systems Interconnection, The Directory, Part 2: Models, 1989.
- [ISO95-3] ISO 9594-3: Information Processing Systems, Open Systems Interconnection, The Directory, Part 3: Abstract Service Definition, 1989.
- [ISOIEC-1] ISO 9595: Information Technology, Open Systems Interconnection, Management Information service Definition.
- [ISOIEC-2] ISO 9596: Information Technology, Open Systems Interconnection, Management Information Protocol Specification.
- [ISOIEC-3] ISO 10040: Information Technology, Open Systems Interconnection, Systems Management Overview.
- [ISOIEC-4] ISO 10165-1: Information Technology, Open Systems Interconnection, Management Information Services, Structure of Management Information, Part 1: Management Information Model.
- [ISOIEC-5] ISO 10165-4: Information Technology, Open Systems Interconnection, Management Information Services, Structure of Management Information, Part 4: Guidelines for the Definition of Managed Objects.
- [JOS90] JOSEPH C. A. & MURALIDHAR K. H., Integrated Network Management in an Enterprise Environment, IEEE Network Magazine,July 1990.
- [KAM91-1] KAMALI M., On-Board Informatics test Facility, Network Management Station, DDD, SLA-OBIF-DDD-NMS, Issue 1.0, October 91.

- [KAM91-2] KAMALI M., On-Board Informatics test Facility, Network Management Agent & Local Management Entity, DDD, SLA-OBIF-DDD-NMA/LME Issue 1.0, October 91.
- [KAU92] KAUFFELS F. J., Netzwerk-Management Probleme-Standards-Strategien, DATACOM, 1992.
- [KOZ91-1] KOZA C.,KAMALI M., On-Board Informatics test Facility, Network Management & Signalling, DDD, SLA-OBIF-DDD-NM&S, Issue 1.0, October 91.
- [KOZ91-2] KOZA C., On-Board Informatics test Facility, Path Entity, DDD, SLA-OBIF-DDD-PATH, Issue 1.0, October 91.
- [MCC88] McCLOGHRIE K., ROSE M. , Management Information Base for Network Management of TCP/IP-based internets, RFC 1066, TWG, August 88.
- [OBI90-1] OBIF Technical Proposal, Version 1.1, EST-SBI-OBIF-VI, 1990.
- [OBI90-2] OBIF Architectural Design Documnet, Version 1.0, EST-SBI-OBIF-ADD, 1990.
- [ONL90] ONLINE'90, 13th European Congress Fair for Technical Communications, Kolloquium C, Network Management: State and Prospects, 1990.
- [ROO90] ROOS J., MARKGRAAFF M., JORDAAN D. & ROOYEN F., Model and architecture of a generalized network management system, computer communications, vol 13 no 9, November 1990.
- [ROS88] ROSE M., McCLOGHRIE K., Structure and Identification of Management Information of TCP/IP-based internets, RFC 1065, TWG, August 88.
- [RYL63] RYLE G., The Concept of Mind, Harmondsworth-England, Penguin, 1963.
- [SCO90-1] SCOTT, Taking Care of Business with SNMP. In: DATA COMMUNICATIONS Spec. LAN Strategies, 3/90.
- [SCO90-2] SCOTT, SNMP Brings Order to Chaos. In: DATA COMMUNICATIONS Spec. LAN Strategies, 3/90.
- [SNE91] SNELL N., Broadening the Management Umbrella, DATAMATION, August 91.
- [STE90] STEENIS H., How to Plan, Develop & Use Information Systems, Dorset House Publishing, 1990.
- [TAN89] TANENBAUM A. S., Computer Networks, Prentice-Hall, 1991.
- [TJA91] TJADEN G., WALL M., GOLDMAN J. & JEROMNIMON C., Integrated Network Management for Real-Time Operations, IEEE Network Magazine, March 1991.
- [WAR89] WARRIER U., Unisys Corporation, BESAW L., Hewlett-Packard, The Common Management Information Services and Protocol over TCP/IP (CMOT), RFC 1095, April 89.
- [WEB80] Webster Encyclopedic Dictionary of the English Language, New York, Avenal Books, 1980.
- [WEI90] WEIHMAYER R. & BRANDAU R., Cooperative Distributed problem solving for communication network management, computer communications, vol 13 no 9, November 1990.

Abkürzungsverzeichnis

ACSE	Association Control Service Element
ADD	Architectrual Design Document
AE	Application Entity
AOS	Advanced Orbiting Systems
ASN.1	Abstract Syntax Notation One
B-PDU	Bitstream PDU
BER	Basic Encoding Rules
CADU	Channel Access Data Unit
CCR	Commitment,Concurrency and Recovery
CCSDS	Consultative Commitee for Space Data Systems
CIF	Central Interconnection Facility
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
CMOL	CMIP over LLC
CMOT	CMIP over TCP/IP
CPN	CCSDS Principal Network
CV	Compressed Video
CVCDU	Coded Virtual Channel Data Units
DDD	Detailed Design Document
ESA	European Space Agency
ESOC	ESA Operation Centre
ESTEC	ESA Tecnical and Research Centre
FFDI	Fiber Distributed Data Interface
FTAM	File Transfer Access and Management
FTP	File Transfer Program
G-LME	Generic LME
GSLGW	Ground Space Link Gateway
IAB	Internet Activity Board
IEEE	Institute of Electrical and Electronical Engineers
IP	Internet Protocol
IPC	Interprocess Communication
ISDN	Integrated Services Digital Network
ISO	International Standardization Organization
LAN	Local Area Network
LDF	LME Description File
LDP	Logical Data Path
LE	Local Etity
LEP	Local End Point
LME	Local Management Entity
LOS	Local Operating System
LPP	Leightweight Presentation Protocol
M-PDU	Multiplexing PDU
MDB	Mirror Data Base
MIB	Management Information Base
MMI	Man Machine Interface
MO	Managed Object
NAF	Network Assesment Facility

NCS	Network Control Station
NFS	Network File System
NIB	Network Interface Board
NM	Network Management
NM&S	Network Management & Signalling
NMA	Network Nanagement Entity
NMS	Network Management Station
NMSys	Network Management System
OBIF	On-Board Informatic Facility
OIC	Operator Interface Controller
OIF	On-board Interface
OSF	Open Systems Foundation
OSF Motif	UNIX-Oberfläche der OSF
OSLGW	On-Board Space Link Gateway
PCA-PDU	Physical Channel Access PDU
PCS	Physical Channel Simulator
PDU	Protocol Data Unit
PE	Protocol Entity
PV	Packet Video
ROSE	Remote Operation Service Element
S-LME	Specific LME
SAP	Service Address Point
SDL	Space Data Link
SDU	Service Data Unit
SGMP	Simple Gateway Monitoring Protocol
SI	System Information
SLS	Space Link Subnetwork
SM	Systems Management
SMAE	Systems Management Application Element
SMAP	Systems Management Application Process
SMASE	Systems Management Application Service Element
SMF	Systems Management Functions
SMI	Structure of Management Information
SNA	System Network Architecture
SNMP	Simple Network Mangement Protocol
SRD	System Requirements Document
TC	TeleControl/TeleCommand
TCP	Transport Control Protocol
TM	Telemetry
TP	Transaction Processing
TTB	Telescience Test-Bed
UDP	User Datagram Protocol
VC	Virtual Channel
VCA	Virtaul Channel Access
VCDU	Virtual Channel Data Unit
VCLC	Virtual Channel Link Control
WAN	Wide Area Network
WS	Work-Station
XDR	Extended Data Representation