TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Analyse von Methoden und deren Realisierungen zur Erfassung von Netzverkehr für das Abrechnungsmanagement in verteilten, heterogenen Umgebungen

Diplomarbeit

Helmut Schiffelholz

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

Analyse von Methoden und deren Realisierungen zur Erfassung von Netzverkehr für das Abrechnungsmanagement in verteilten, heterogenen Umgebungen

Diplomarbeit

Helmut Schiffelholz

Aufgabensteller: Prof. Dr. H.-G. Hegering

Betreuer: Stefan Schwerdtner

Abgabetermin: 15. Februar 1995

Ehrenwörtliche Erklärung

Ich versichere, daß ich diese Diplomarbeit selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Februar 1995

....

(Unterschrift des Kandidaten)

Inhaltsverzeichnis

1 Einleitung

Das Abrechnungsmanagement, **Accounting Management**, ist ein Funktionsbereich des Netzmanagements, **Network Management**. Die Funktionen, die das Abrechnungsmanagement realisieren muß, sind die Steuerung einer benutzerbezogenen Nutzungserfassung, **Usage Metering**, das dienstbezogene Zusammenführen der erfaßten Daten, die Zuordnung von Kosteninformation und letztendlich anhand dieser Abrechnungsinformation die Rechnungstellung. Demnach sollte jeder Nutzer eine nach von ihm in Anspruch genommenen Diensten aufgeschlüsselte Rechnung erhalten.
Die vorliegende Arbeit befaßt sich hauptsächlich mit der Nutzungserfassung. Dazu werden nach den einleitenden Kapiteln Methoden zur Erfassung von Netzverkehr diskutiert. Betrachtet werden dabei grundsätzlich verteilte, heterogene Umgebungen, im konkreten Fall dieser Diplomarbeit sind dies TCP/IP-basierte Datennetze. Nach der Analyse der Methoden werden die Realisierungen dieser Methoden, die Werkzeuge zur Erfassung von Netzverkehr, näher betrachtet. Dazu werden Kriterien zur Bewertung dieser Werkzeuge aufgestellt. Im Anschluß werden zwei derartige Werkzeuge exemplarisch nach den aufgestellten Kriterien bewertet.
Nach diesen einleitenden Worten soll dargelegt werden, warum eigentlich an der Realisierung eines Abrechnungsmanagements im obigen Sinne in heterogenen Datennetzen ein zunehmendes Interesse besteht.
Vor einigen Jahren, als das Rechenzentrum der einzige und *zentrale* DV-Dienstanbieter war, konnten sehr einfach die Dienste, die der einzelne Benutzer in Anspruch nahm, erfaßt und bei Bedarf seiner Organisationseinheit oder dem Benutzer selbst in Rechnung gestellt werden. In den letzten fünf bis zehn Jahren ist man jedoch immer mehr vom Rechenzentrum auf Netztechnologien umgestiegen (*Down-, Rightsizing*), was einerseits zu einer größeren Funktionalität und gesteigerten Effizienz geführt, aber andererseits für das Netzmanagement eine Menge Probleme mit sich gebracht hat. Diese Probleme finden sich vor allem in den Bereichen Datenschutz, Datensicherheit, Fehlermanagement und das in dieser Arbeit behandelte Abrechnungsmanagement.
Zur Zeit wird ein Abrechnungsmanagement gar nicht oder nur sehr selten realisiert. Die Ausgaben für Netze werden meistens als „allgemeine Infrastruktur des Unternehmens" (siehe [22]) betrachtet und diese Kosten werden nur pauschal abgerechnet, was vor allem für externe, firmenfremde Dienstnutzer untragbar ist (*Outsourcing*). Aber auch durch den Budgetdruck innerhalb der Unternehmen wird sich diese Situation sehr bald ändern müssen.
Es besteht ein gesteigerter Bedarf an geeigneten Methoden und Werkzeugen um eine **verursachergerechte Abrechnung von Netzdiensten** durchführen zu können. Die dazu benötigte, möglichst **detaillierte** Nutzungserfassung wird dazu in dieser Arbeit näher untersucht.

1.1 Accounting und Accounting Policy

Das Abrechnungsmanagement, und damit das **Accounting**[1], wird von der bestehenden Abrechnungspolitik, **Accounting Policy**, bestimmt. Diese wird in Unternehmen von der Geschäftsführung vorgegeben. Sie bestimmt unmittelbar die **Granularität** der Datenerfassung (und damit die Detailliertheit der erfaßten Daten). Die zwei Extremfälle für eine Abrechnungspolitik sind:

1. Die durch den Betrieb und die Erweiterung des Netzes insgesamt enstehenden Kosten werden **pauschal** aufgeteilt und den Organisationseinheiten, OEs, zugeordnet.

2. Die entstehenden Kosten werden möglichst **detailliert** erfaßt und dem **einzelnen Benutzer** in Rechnung gestellt.

Zwischen diesen beiden Extremfällen existiert ein breit gefächertes Spektrum. Durch die Realisierung eines Abrechnungsmanagements entsteht stets ein Mehraufwand, **Accounting Overhead**, der direkt proportional zur gewählten Granularität ist.

Die Wahl der Abrechnungspolitik ist abhängig von der **Rolle der IV-Abteilung** bzw. des Unternehmens selbst: Ein Unternehmen, das anderen Unternehmen Netz-Dienstleistungen anbietet, benötigt eine möglichst detaillierte Nutzungserfassung, um seinen Kunden eine genaue Auflistung der von ihnen verursachten Kosten geben zu können. Das gleiche gilt für eine IV-Abteilung, die die anderen Abteilungen als Kunden innerhalb des eigenen Unternehmens betrachtet (bzw. dazu gezwungen ist, sie als solche zu betrachten).

In Abbildung 1 sind einige Abrechnungsverfahren, **Billing Techniques**, beschrieben, durch die jeweils eine Accounting Policy aus dem zuvor erwähnten, breit gefächerten Spektrum realsiert wird. (siehe [9])

Divide and Charge ist dabei die pauschalste Variante: Die Gesamtkosten werden einfach pauschal auf die einzelnen Organisationseinheiten, OEs, aufgeteilt. Bei **Sliding Scale** werden die Gesamtkosten anteilig berechnet; so ist der prozentuale Anteil einer OE an den Gesamtkosten z.B. um größer, je mehr Rechner die OE im Einsatz hat. **Metrics** bedeutet, daß eine Nutzungserfassung durchgeführt wird: hier wird die Nutzung von Diensten und / oder die Nutzung von Ressourcen erfaßt. Ziel ist eine verursachergerechte Abrechnung. Grundlage ist hiebei z.B. eine Erfassung der Verbindungszeit, **Connect Time**, der empfangenen bzw. gesendeten Pakete, **Packets sent / received**, oder am detailliertesten:

[1]Accounting soll als Funktion des Accounting Managements betrachtet werden: Accounting ist dabei gleichzusetzen mit Usage Metering und Reporting zu Abrechnungszwecken

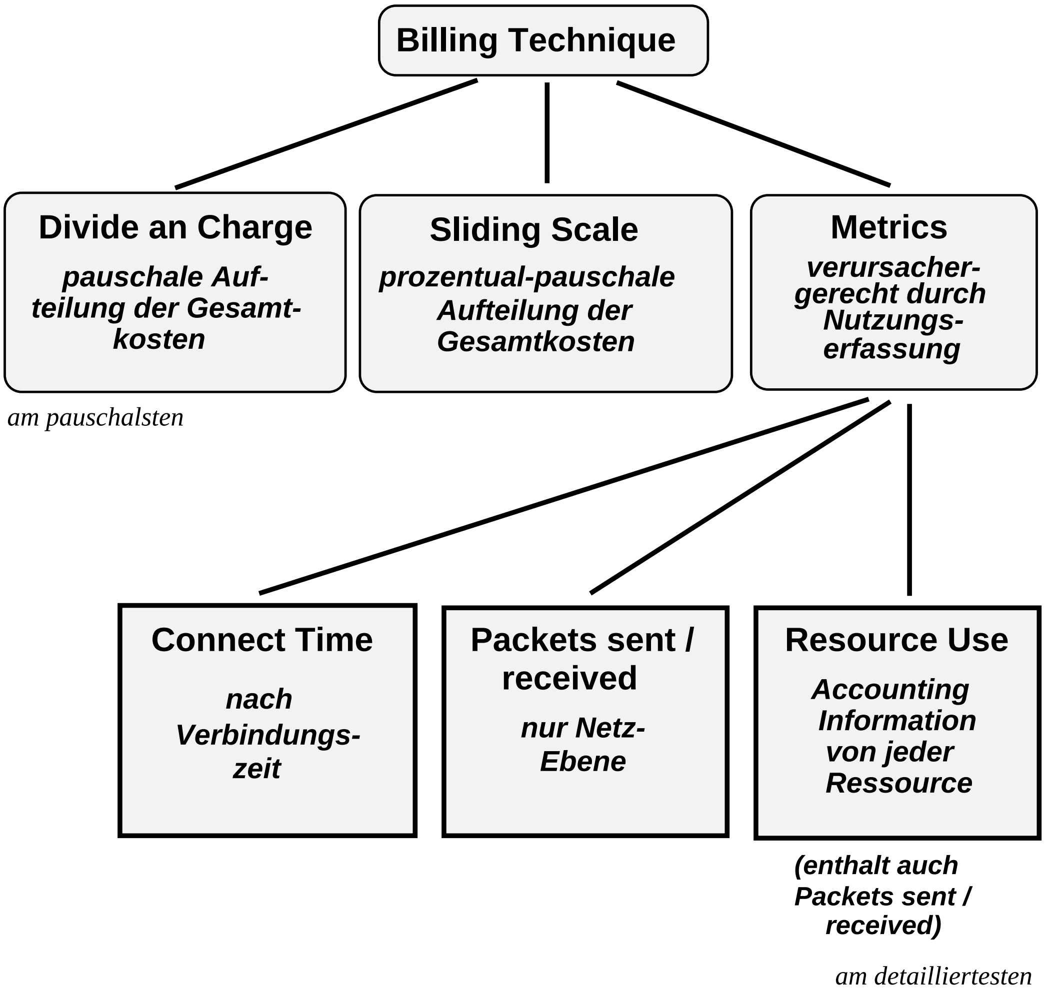

Abbildung 1: Abrechnungsalternativen

die Erfassung des Nutzungsgrades einer jeden Ressource, **Resource Use**, durch einen einzelnen Benutzer.
Diese Abbildung soll zunächst nur einen groben Überblick geben. Der interessanteste Fall der Nutzungserfassung wird in der folgenden Arbeit unter verschiedenen Gesichtspunkten eingehend behandelt werden.

Es wird somit nur die **verursachergerechte** Abrechnung von Netzdiensten betrachtet. An ein Abrechnungsmanagement, das diese Art von Abrechnung realisieren soll, müssen die folgenden Anforderungen gestellt werden.

1.2 Anforderungen an ein Abrechnungsmanagement

Es folgt eine Liste von Anforderungen an ein Abrechnungsmanagement. Ein Abrechnungsmanagement, und damit auch das Accounting, sollte möglichst vielen der genannten Anforderungen gerecht werden. Sie können gleichzeitig auch als Kriterien für die Bewertung von Methoden und Werkzeugen zur detaillierten Erfassung von Netzverkehr herangezogen werden, da durch sie der technische Teilbereich des Abrechnungsmanagements realisiert wird.

- **Einfachheit**: Sowohl die Nutzungserfassung als auch die Weiterverrechnung sollte auf möglichst einfachen Methoden basieren.

- **Verständlichkeit**: Sowohl die Methoden des Abrechnungsmanagements als auch die resultierenden Rechnungen sollten für *alle* Beteiligten leicht verständlich und *nachvollziehbar* sein.

- **Genauigkeit**: Die Abrechung sollte möglichst detailliert sein. Jedoch sollten bei der Festlegung der Genauigkeit neben technischen auch wirtschaftliche Gesichtspunkte berücksichtigt werden: Der Aufwand darf nicht in die Größenordnung des erzielbaren Effekts kommen.
  Auch rechtliche Gesichtspunkte spielen hierbei eine große Rolle. Eine verursachergerechte Nutzungserfassung kann auch eine erhebliche Kontrolle des einzelnen Benutzers bedeuten. Mehr Genauigkeit bedeutet damit auch gesteigerte Kontrollmöglichkeiten (Datenschutz!).

- **Gerechtigkeit**: Die Kostenzuteilung sollte verursachergerecht sein: Jeder Benutzer sollte nur genau die von ihm verursachten Kosten tragen müssen. Kosten für eine neue Netzanbindung, einen neuen Arbeitsplatzrechner oder für einen Umzug sollten genauso dem einzelnen Benutzer bzw. seiner OE in Rechnung gestellt werden wie z.B. der vom einzelnen Benutzer verursachte Netzverkehr.
  Kosten für Erweiterungen am unternehmensweiten Datennetz, die allen oder mehreren Benutzern zugute kommen, wie z.B. eine weitere Standleitung oder ein Internet-Anschluß, sollten je nach Intensität der Nutzung diesen Benutzern anteilig berechnet werden. Das gleiche Prinzip kann für allgemeine Betriebskosten, wie für Stromverbrauch oder Wartung, Anwendung finden.

- **Wiederholbarkeit**: Die Methoden und Werkzeuge des Abrechnungsmanagements sollten unter *gleichen Bedingungen* stets die gleichen Ergebnisse liefern.

- **Planbarkeit**: Die vom Benutzer in Anspruch genommene Leistung muß im einzelnen aus der Abrechnung ersichtlich sein. Dadurch wird gewährleistet, daß diese Informationen für Zwecke der Planung und der Investitionsbeurteilung verwendet werden können.

- **Datensicherheit, Datenschutz**: Sowohl die Methoden und Werkzeuge des Abrechnungsmanagements als auch die gewonnenen Daten müssen gegen Mißbrauch oder Manipulation geschützt werden. Nur die den einzelnen Benutzer betreffenden Abrechnungsdaten sollten auch nur diesem zugänglich sein.

- **Flexibilität**: Das Abrechnungsmanagement sollte sich leicht an sich ändernde Bedingungen (z.B. neue Dienste, neue Ressourcen, Änderung der Abrechnungspolitik) anpassen lassen.

- **Integrierbarkeit**: Die an unterschiedlichen Orten im Netz erfaßten Daten sollten problemlos (automatisch) zusammenzuführen sein. Dies ist insbesonders in heterogenen, verteilten Umgebungen besonders wichtig.
  Auch im Sinne eines integrierten Netzmanagements sollten die Werkzeuge des Abrechnungsmanagements einfach in die bestehende Netzmanagementplattform zu integrieren sein.

- **Auswertbarkeit**: Die mit Hilfe des Abrechnungsmanagements gewonnene Information sollte unterschiedlich auswertbar sein. So kann sie auch zur Ermittlung von Benutzer- bzw. Nutzungsprofilen verwendet werden und damit der Optimierung des Netzbetriebs oder der Optimierung von Arbeitsabläufen dienen (Benutzerverhalten!).

- **Abstraktion von der Netzinfrastruktur**: Damit ist die *Vereinheitlichung* der zu erfassenden Daten auf einer möglichst unteren, technischen Ebene gemeint. Hier sollte *unabhängig von der verwendeten Netztechnologie* eine einheitliche Schnittstelle existieren. Diese Anforderung sollte in heterogenen Umgebungen eine Vorraussetzung sein.

- **Minimierung der Netzbelastung**: Ein Teil des durch das Abrechnungsmanagement entstehenden Overheads ist der *Netz-Overhead*. Dieser ist besonders kritisch, da sich durch ihn die bestehende Netzbelastung (wesentlich) erhöhen und sich der *Quality of Service* deutlich verschlechtern kann. Der Netz-Overhead sollte daher so gering wie möglich gehalten werden.

- **Benutzerbezogenheit**: Die durch Accounting gewonnenen Daten sollten benutzerbezogen sein, d.h. es sollte eine Zuordnung *Benutzer — verursachte Kosten* (z.B. durch den Netzverkehr, den der Benutzer verursacht hat) existieren, damit eine detaillierte, verursachergerechte Abrechnung überhaupt möglich ist.

- **Dienstbezogenheit**: Wie bei der Benutzerbezogenheit sollte ebenfalls eine Zuordnung *Dienst — verursachte Kosten* bestehen, um den einzelnen Benutzer verursachergerecht, dienstbezogen abrechnen zu können.

Da das Gebiet des Abrechnungsmanagements im allgemeinen und des Accountings im speziellen sehr weitreichend ist und in dieser Arbeit nur ein sehr kleiner Ausschnitt betrachtet und behandelt werden kann, soll bereits an dieser Stelle die Fragestellung eingegrenzt werden, um die Schwerpunkte der vorliegenden Arbeit deutlich zu machen.

1.3 Eingrenzung der Fragestellung

Schwerpunkt ist die Analyse von Methoden und Werkzeugen zur Erfassung von Netzverkehr (Erfassung von Netz-Ressourcen) in einem heterogenen Datennetz, in dem verteilte Anwendungen bereitgestellt werden. Der Netzverkehr soll dabei zum Zwecke des Accountings erfaßt werden. Es sollen dabei Kriterien für die Bewertung von Erfassungs-Werkzeugen gefunden werden. Anhand dieser Kriterien sollen dann bestehende, zur Verfügung stehende Werkzeuge bewertet werden.

Das betrachtete Szenario ist ein auf TCP/IP basiertes Datennetz. Das darunterliegende Protokoll ist Ethernet nach 802.3. Die betrachteten Werkzeuge laufen auf UNIX-Workstations (HP-UX und SunOS Systeme). Die Erfassung des Datenverkehrs beschränkt sich auf die OSI-Schichten 3 und 4. Es werden vornehmlich die den Erfassungs- Werkzeugen zugrundeliegenden Methoden analysiert. Sämtliche Betrachtungen werden im wesentlichen vor dem Hintergrund eines am Lehrstuhl entwickelten, unter Mitarbeit des Autors erstellten, OSI-konformen Objekt-Modells für Accounting gemacht. Sämtliche jetzt genannten Begriffe werden im folgenden genau erklärt. Da TCP/IP zusammen mit Ethernet die Kommunikationsplattform bildet, soll zuerst ein einleitendes Kapitel über TCP/IP als (technische) Grundlage für die weiteren Betrachtungen folgen.

2 Die Kommunikationsplattform: TCP/IP

In diesem Kapitel werden kurz wesentliche Grundbegriffe und elementare Zusammenhänge der Protokolle der TCP/IP - Protokollfamilie[2] dargestellt, sowie die Verbindung zum betrachteten Betriebssystem, UNIX, hergestellt.

Die TCP/IP - Architektur besteht aus vier Schichten: Die unterste Schicht, die **Netzwerk Schicht** (Network Level), übernimmt die Bitübertragung sowie die Sicherung, und entspricht somit den OSI-Schichten[3] 1 und 2. Sie heißt Netzwerk Schicht, weil sie auf sogenannten Subnetzwerken, wie z.B. X.25, SLIP[4] oder IEEE 802.x, aufsetzen kann. Dadurch ist TCP/IP in der Benutzung physikalischer Medien sehr flexibel.

Die zweite Schicht ist die **Internet Schicht** (Internet Level). Sie entspricht der OSI-Schicht 3 und wird allein durch das **Internet Protocol, IP** betrieben. Es wird durch ein Hilfsprotokoll, das **Internet Control Message Protocol, ICMP**, das **Address Resolution Protocol, ARP**, sowie durch verschiedene Routing-Protokolle ergänzt. Zudem können in dieser Schicht noch weitere Protokolle, wie z.b. VAX, das Trailer Encapsulation[5] verwendet, vorhanden sein. Durch das IP kann jeder Knoten innerhalb eines TCP/IP-Netzes *direkt* mit einem anderen Knoten kommunizieren.

Die dritte Schicht ist die **Transport Schicht** (Transport Level). Sie entspricht der OSI-Schicht 4 und wird durch zwei Protokolle betrieben: Das **Transmission Control Protocol, TCP**, das virtuelle Verbindungen realisiert und das **User Datagramm Protocol, UDP**, das wie das IP-Protokoll verbindungslos arbeitet.

Die oberste Schicht schließlich ist die **Applikations Schicht** (Application Level). Sie entspricht den OSI-Schichten 5 bis 7 und bietet eine Reihe von Protokollen an, die standardisierte Applikationen, wie z.B. Terminal-Emulation oder Dateitransfer, zur Verfügung stellen. Sie können anderen Applikationen als **Basisapplikationen** dienen.

Zur Unterscheidung der einzelnen Schichten sollen die Datenpakete im weiteren wie folgt benannt werden:

---

[2] Transmission Control Protocol / Internet Protocol

[3] ISO/DIS 7498, Open System Interconnection Basic Reference Model

[4] Serial Line Internet Protocol, für Punkt-zu-Punkt-Verbindungen zwischen zwei TCP/IP-Hosts; Definition eines Hosts siehe weiter unten.

[5] TCP- bzw. UDP- und IP-Header werden am Ende des Datenblockes angehängt

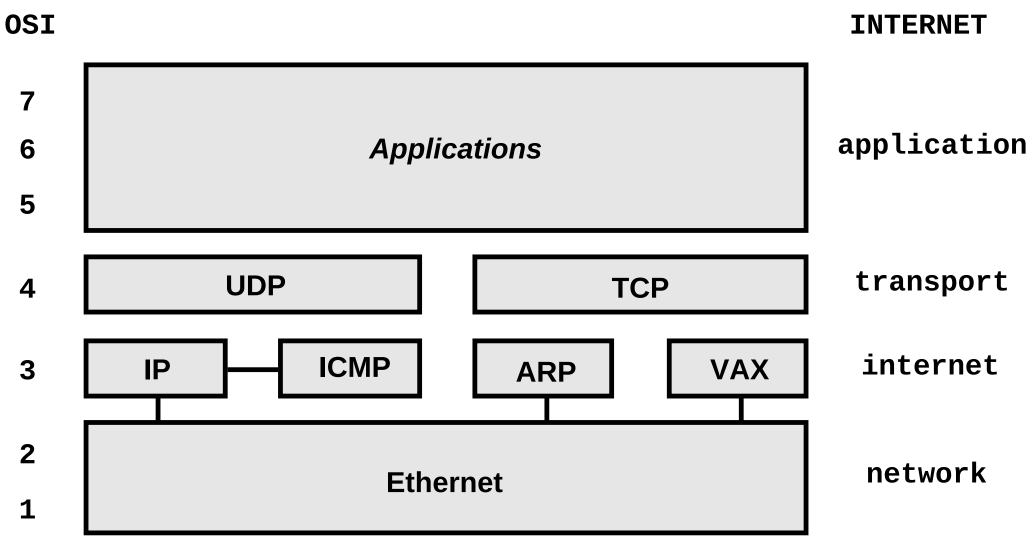

Abbildung 2: Die wichtigsten Protokolle der TCP/IP - Protokollfamilie und ihre Zuordnung zu TCP/IP-Levels bzw. OSI-Schichten

- Transport-Schicht: **Segmente** bzw. **Datagramme**
- Internet-Schicht: **Datagramme**
- Netzwerk-Schicht: **Rahmen, Frames**

Dabei soll in der Transport-Schicht der Begriff Segment im Zusammenhang mit TCP und der Begriff Datagramm im Zusammenhang mit UDP verwendet werden.

Im Weiteren soll näher auf die Protokolle der einzelnen Schichten eingegagen werden. Als Protokoll der Netzwerk Schicht soll exemplarisch Ethernet (nach IEEE 802.3) betrachtet werden, da Ethernet von allen später betrachteten Werkzeugen unterstützt wird und am weitesten verbreitet ist:

2.1 Die Netzwerk-Schicht

Ethernet ist eine LAN[6] - Technologie. Es arbeitet nach dem CSMA/CD[7] - Verfahren nach IEEE 802.3 bzw. ISO 8802-3.

[6]Local Area Network
[7]Carrier Sense Multiple Accsess / Collision Detection

Jeder Knoten in einem Ethernet hat eine weltweit eindeutige **Ethernet Adresse**. Sie besteht aus 6 Oktetten und kann wie folgt dargestellt werden:

$$\text{Ethernet Address} = \text{XX:XX:XX:XX:XX:XX}$$

Dabei repräsentiert XX ein Oktett zwischen 00 und ff (hexadezimal). Da Ethernet Adressen vom Hersteller eindeutig einem Gerät zugeordnet werden, nennt man sie auch **physikalische Adressen** (oder Hardwareadressen). Da sich die Ethernet Adresse eines Gerätes z.B. beim Austausch eines Interfaces, ändern kann, ist es notwendig auf der nächsthöheren Schicht, der Internet-Schicht, dem Gerät eine feste, **logische Adresse** zuzuordnen. Eine Ethernet Adresse kann neben einer einzelnen physikalischen Adresse auch eine **Broadcast Adresse** (alle Knoten[8] sollen adressiert werden) oder eine **Multicast Adresse** (mehrere Knoten sollen adressiert werden) spezifizieren.

Im Ethernet werden die Daten in sogenannten **Frames** (Rahmen) übertragen. Ethernet-Frames haben eine *variable Länge*, jedoch ist kein Frame länger als 1518 Bytes[9]. Ein Ethernet Frame (nach IEEE 802.3) besteht (im einfachsten Fall[10] aus folgenden Bestandteilen:

- Präambel (zur Synchronisation)
- Zieladresse
- Quelladresse
- Länge (Anzahl der Bytes im Datenfeld)
- Daten
- FCS[11] (Prüfsumme)

Für Accounting könnten auf dieser Schicht die Quell- und Zieladressen, die sich jedoch beim Austausch von Interfaces ändern, sowie der Typ, der angibt, welche Daten transportiert werden (z.B. IP, ARP, Netware), interessant sein. Der Typ ist allerdings bei IEEE 802.3 Frames nur bei Verwendung von IEEE 802.2 LLC und eventuell zusätzlich von SNAP zu ermitteln. Die Anzahl der übertragenen Bytes (Oktette) kann einfach dem Längen- Feld zu entnommen werden.

[8] Ethernet Interface eines Hosts
[9] Ohne Präambel, siehe unten
[10] Ohne IEEE 802.2 LLC und SNAP Encapsulation, siehe dazu [27], Seiten 162ff
[11] Frame Check Sequence

2.2 Die Internet-Schicht

Das **Internet** ist ein weltweites (Daten-)Netz das aus einer Vielzahl von Teil-Netzen gebildet wird. Diese (Teil-)Netze werden mit Hilfe von Gateways miteinander verbunden. Man betrachte dazu zunächst die folgenden Definitionen:

Ein **Host** ist ein Endgerät oder ein Koppelelement, das über eine Internet Adresse (Adresse der Internet Schicht, siehe weiter unten) angesprochen werden kann. Ein Host, der mehrere Interfaces mit Internet Adressen besitzt wird auch Multi Home Host genannt.

Ein **Gateway** ist ein Host, der an zwei oder mehr (Teil-)Netze angebunden ist und Datagramme zur Versendung an lokale Hosts bzw. Datagramme von lokalen Hosts zur Versendung an andere Hosts oder Gateways erhält. Ein Gateway ist somit ein Router auf der Internet Schicht, der Datagramme routet und dabei auch in der Lage ist, Protokollwechsel (z.B. von Ethernet auf Token Ring) durchzuführen.

Ein **Router** ist ein Gerät, das Verbindung zu zwei oder mehr (Teil-)Netzen hat und für die Vermittlung von Paketen zuständig ist. Es arbeitet auf der Netzwerk Schicht und nimmt *keinen* Protokollwechsel vor.

Da im Internet bzw. innerhalb eines TCP/IP-Netzes jeder Knoten direkt angesprochen werden kann, muß jeder Knoten eine eindeutige, feste Adresse beitzen, die sogenannte **Internet Adresse**. Eine Internet Adresse ist 32 bit lang und besteht aus einer **netid**, zur Adressierung eines (Teil-)Netzes, und einer **hostid**, zur Adressierung eines Hosts *innerhalb* eines (Teil-)Netzes. Es gibt vier Klassen (Class A bis D) von Internet-Adressen, je nach der Gröse der netid bzw. der hostid. In einem Class D Netz, das ein isoliertes, privates TCP/IP-Netz darstellt, gibt es nur die hostid. Eine Internet-Adresse hat folgendes Aussehen:

$$\text{Internet-Adresse} = X.X.X.X$$

Dabei repräsentiert jedes X eine Integer - Zahl zwischen 0 und 255. Neben der Möglichkeit, mit der Internet Adresse einzelne Hosts ansprechen zu können, gibt es noch zwei weitere Arten von Internet-Adressen: Die **Broadcast-Adresse** und die **Loopback-Adresse**.

Mit einer Broadcast Adresse werden alle Hosts angesprochen. Die globale Broadcast Adresse lautet: 255.255.255.255.
Zusätzlich gibt es noch eine (teil-)netzinterne Broadcastadresse. In einem Class A Netz (das erste Byte repräsentiert dabei die netid und das erste bit ist auf 0 gesetzt) hat sie das Aussehen: 0.X.X.X.

Die Loopback-Adresse dient der Versendung von Daten zum eigenen Host. Hierdurch können z.B. TCP/IP-Protokolle oder -Applikationen getestet werden. In einem Class A Netz hat sie das Aussehen: 127.X.X.X.

Falls aus organisatorischen oder technischen Gründen eine Unterteilung eines Netzs in Subnetze erforderlich ist, bietet IP die Möglichkeit der **Subnetz Adressierung**. Dazu wird ein Teil der Internet Adresse mit einer sogenannten **Subnet Mask** verglichen. Dadurch ist es möglich, alle Hosts eines so definierten Subnetzes (z.B. eine Abteilung) gleichzeitig anzusprechen (Multicast).

IP übernimmt die Segmente bzw. Datagramme aus der Transport Schicht, versieht sie mit einem Header und versendet sie mit Hilfe des Ethernet (**Encapsulation**). Falls die Datenblöcke aus der Transport Schicht zu lang sind werden sie unterteilt (**fragmentiert**) und auf mehrere Frames aufgeteilt. Dabei erhält jedes Teilstück seinen eigenen Header. Die Fragmente werden auf dem Ziel-Host vom Ziel-IP dann wieder zusammengesetzt (reassembliert).

IP realsiert einen verbindungslosen, unzuverlässigen Dienst: Die Datagramme können verloren gehen, dupliziert werden (wenn mehrere Wege existieren) und in der falschen Reihenfolge ankommen.

Ein (IP-)Datagramm besteht im wesentlichen aus folgenden Bestandteilen:

- Länge
- Lebenszeit
- Protokolltyp
- Prüfsumme
- Quelladresse
- Zieladresse
- Daten

IP Datagramme können eine beliebige Länge haben. In der Regel wird die Länge jedoch so gewählt, damit ein Datagramm problemlos von *jedem* Frame der Netzwerk Schicht aufgenommen werden kann und somit keine Fragmentierung notwendig ist. Auf die Lebenszeit wird im Zusammenhang mit ICMP noch eingegangen. Interessant für Accounting mit Blick auf den Header sind die Datagramm-Länge, der Protokolltyp, der angibt welches Protokoll (z.B. UDP) den Datagrammdienst in Anspruch nimmt, sowie die Quell- und Zieladresse.

Das **Internet Control Message Protocol, ICMP** dient der Übermittlung von Kontroll-Meldungen auf der Internet-Schicht (es handelt sich dabei um vordefinierte Meldungen) und stellt eine notwendige Ergänzung zum IP dar. Die Meldungen des ICMP werden mit Hilfe des IP selbst gesendet. Dazu wird die komplette **ICMP Message** in ein Internet Datagramm eingepackt. Der Protokoll-Typ (ICMP) wird entsprechend im IP-Header vermerkt.

Mit Hilfe des ICMP kann zum Beispiel ermittelt werden, ob bestimmte Hosts erreichbar und empfangsbereit sind, welche Internet Adresse sie haben oder ob die Lebenszeit eines IP-Datagramms abgelaufen ist. Dazu ein Beispiel: Die Transport-Schicht übergibt an die Internet Schicht Daten zur Versendung. Da bei Beginn der Versendung nicht gewährleistet werden kann, ob der Ziel-Host auch erreichbar und empfangsbereit ist (er könnte z.B. ausgeschaltet sein), versieht IP ein Datagramm mit einer Lebenszeit. Wenn IP das Datagramm abschickt, wird es von Gateway zu Gateway weitergereicht. Ist ein Gateway das Ziel-Gateway, nimmt es das Paket entgegen und sendet es direkt an den Ziel-Host. Falls es nicht das zuständige Ziel-Gateway ist, so verringert es die Lebenszeit des Datagramms um Eins und schickt es weiter. Wird dabei die Lebenszeit aber gleich Null, so verwirft es das Datagramm und verwendet das ICMP, um den Absender darüber zu informieren. Das Absender-IP reicht wiederum diese Information an die Transport Schicht weiter. Diese übermittelt sie der entsprechenden Applikation.

Wenn auch durch die Internet-Schicht eine Ende-zu-Ende-Verbindung realisiert wird, so erfolgt die tatsächliche, physikalische Kommunikation auf der Netzwerk Schicht. Dies geschieht mit Hilfe des Ethernets. Die dazu nötige Zuordnung von Internet Adressen zu Ethernet Adressen leistet das **Address Resolution Protocol, ARP**. Es baut unter Einsatz von Broadcast-Adressen dynamisch eine Umsetzungtabelle auf. Soll ein Datagramm versendet werden, so prüft ARP, ob die Internet Adresse des Ziel-Hosts in der ARP-Tabelle vorhanden ist. Ist dies der Fall, erfolgt die Versendung unter Angabe der dort gefundenen Ethernet Adresse. Wird die Internet Adresse nicht aufgefunden, so werden von ARP unter Aussendung einer Broadcast Adresse alle angeschlossenen Hosts in einer Anfrage ersucht, die dort angegebene Internet Adresse mit der ihren zu vergleichen und bei Übereinstimmung zusammen mit ihrer Ethernet Adresse zurückzusenden. Die ARP-Tabelle wird i.a. alle 15 min auf die gleiche Weise aktualisiert. ARP arbeitet direkt mit Hilfe des Ethernets.

Bemerkung: Beim umgekehrten Vorgang, der bei Diskless-Workstations nötig ist, die ja beim Booten ihre Internet Adresse noch nicht kennen, wird das Reverse Address Resolution Protocol, RARP, verwendet.

Eine wichtige Aufgabe der Internet-Schicht ist das **Routing** zur Realisierung eine Ende-zu-Ende-Kommunikation. Zu diesem Zweck gibt es mehrere Routing

- Protokolle: Das ist zum einen das **Routing Information Protocol, RIP**, das innerhalb eines autonomen Systems (= eine Gruppe von (Teil-)Netzen samt zugehöriger (interior) Gateways, die der selben Administration unterstellt sind) dynamisch Routing-Information ermittelt und damit entsprechende Routing - Tabellen verwaltet. Als Maß für den besten Weg verwendet es die sog. **Hop Counts**, das ist die Anzahl der Gateways auf dem Weg vom Absender zum Empfänger. Ein ähnliches Protokoll ist das **Distributed Computer Network, DCN, Local-Network Protocol, HELLO**, das jedoch die Laufzeit als Maß für den besten Weg verwendet. Schließlich gibt es noch das **Exterior Gateway Protocol, EGP**, das verwendet wird, um Routing-Information zwischen zwei autonomen Systemen, die direkt über externe Gateways verbunden sind, auszutauschen. (Es gibt noch weitere Routing-Protokolle, die aber hier nicht angesprochen werden; siehe dazu [27], Kapitel 16).

Nach all den eben vorgestellten Protokollen, sieht man leicht ein, daß der Datenverkehr auf der Internet-Schicht zu einem nicht unbeachtlichen Teil aus Kontroll- und Steuerinformation besteht. Für das Accounting kann bestenfalls ausschließlich der reine IP-Verkehr betrachtet werden, obwohl auch dieser durch ICMP-Meldungen verfälscht wird. Die folgende Betrachtung der Tansport-Schicht wird zeigen, daß sich die dort verwendeten Protokolle UDP und TCP wesentlich besser zu Accounting-Zwecken eigenen.

2.3 Die Transport-Schicht

Die Internet Protocol Software sendet und empfängt IP-Datagramme mit Hilfe von Internet Adressen. Es benutzt das Address Resolution Protocol um die Internet Adressen in Ethernet Adressen zu übersetzen. Mit einer Internet Adresse kann jedoch nur ein bestimmter Host adressiert werden, nicht jedoch ein bestimmter Prozeß, der auf dem Host läuft. Dieser Mangel wird mit dem **User Datagram Protocol, UDP** aus der Transport Schicht behoben. Durch das UDP können einzelne Prozesse mit Hilfe von **Port-Nummern** angesprochen werden.

Durch die Angabe der IP-Adresse und des UDP-Ports kann somit ein Kommunikationsendpunkt innerhalb eines Hosts direkt adressiert werden. Ein solcher Kommunikationsendpunkt wird auch **Socket** (siehe weiter unten) genannt. Eine Applikation, die mittels UDP Daten versenden will, gibt einfach die Ziel-IP-Adresse und den Ziel-UDP-Port an. UDP schickt sie in Form von **Datagrammen** ab:

(Das UDP-Datagramm wird mit Hilfe des IP und letzendlich mit Hilfe des Ethernet übertragen.) Auf dem Ziel-Host angekommen, werden die UDP-Datagramme von einem dort laufenden Protokollprozeß entgegengenommen und unter der angegebenen Port-Nummer in eine entsprechende Warteschlage gestellt, aus der sie dann bei Bedarf von der Applikation, für die sie auf dem Ziel-Host bestimmt sind, entnommen werden.

Ein UDP-Datagramm besteht aus einem sehr einfachen Header und den zu übermittelden Daten. Der Header enthält lediglich die Quell- und Ziel-Port-Nummer, die Länge des Datagramms und eine Prüfsumme.

UDP ist wie IP ein verbindungsloses Protokoll. Die Datagramme können also gar nicht oder in beliebiger Reihenfolge beim Empfänger ankommen, außerdem können Duplikate entstehen. Derartige Fehler müssen von den darüberliegenden Protokollen behoben werden.

Soll der einmalige Erhalt der Daten in der richtigen Reihenfolge bereits auf der Transport Ebene gewährleistet sein, so muß das folgende, alternative Transport Protokoll benutzt werden:

Im Gegensatz zum UDP ist das **Transmission Control Protocol, TCP** ein verbindungsorientiertes Protokoll. Es schließt Datenverlust, Datenduplikation und Reihefolgevertauschung aus.

Es dient der blockweisen Versendung von Datenströmen, zum Beispiel im Rahmen eines Filetransfers. TCP unterteilt die zu versendenden Daten in **Segmente** und versendet sie dann mit Hilfe des IP. Die Segmentgröße ist variabel. Dadurch kann eine gute Netz- und Ressourcenauslastung erreicht werden.

Mit Hilfe von Sequenznummern kann das Empfänger-TCP ermitteln, in welcher Reihenfolge es die ankommenden Daten zu interpretieren hat. Dadurch können auch Duplikate identifiziert werden. Empfangene Daten werden mittels Quittungsnummer quittiert. Bleibt eine Quittung aus, so werden die entsprechenden Daten erneut versendet. Anhand der Sequenznumern können Segmente einer virtuellen Verbindung zugeordnet werden. Der erhöhte Aufwand für die Zuverlässigkeit des Protokoll schlägt sich in der Größe des TCP-Headers und in der verminderten Datenübertragungsrate nieder. Die Adressierung funktioniert analog zu UDP. Relevant für Accounting sind dabei lediglich die Quell- und Ziel-TCP-Portnummer und ggf. die Segmentlänge (tritt jedoch nicht im Header auf).

Wenn man also UDP- und TCP-Verkehr erfaßt, kann man theoretisch den gesamten IP-Verkehr *minus* dem Verkehr, der auf der Internet Schicht, der der Übermittlung von Kontroll- und Steuerinformation dient (ICMP-Meldungen, ARP-Broadcasts etc.), ermitteln, und den entsprechenden Applikationen zuordnen. Eine Zuordnung zu Benutzern kann erst in der Applikations-Schicht bzw. durch Auslesen des UNIX Kernels (Zuordnung Socket — Prozeß und Prozeß — User;

ein Tool zur Ermittlung dieser Information, genannt `mapinfo`, wird gerade am Lehrstuhl entwickelt) erfolgen.

2.4 Die Applikations Schicht

Für die Betrachtung der Applikations Schicht sollen exemplarisch zwei Basisapplikationen besprochen werden: TELNET und FTP.

Zuvor sollen noch kurz die Begriffe Client und Server erklärt werden: Ein **Client** ist ein Prozeß, durch den Dienste eines Servers in Anspruch genommen werden können. Die Kommunikation mit dem Server wird bei TCP/IP in der Regel über die Socket-Schnittstelle abgewickelt. Ein **Server** ist in diesem Zusammenhang ein Prozeß, der Dienste zur Verfügung stellt, die von einem Client-Prozeß angefordert und in Anspruch genommen werden können. Bei TCP/IP wird der Server zuerst über einen fest definierten und den Clients bekannten Port, den sog. **Well-Known Port** angesprochen. Über diesen fordern die Clients einen Server - Dienst (**Service**) an. In der Anforderung (**Request**), in Form eines UDP-Datagramms oder eines TCP-Segments, übermittelt der Client zugleich auch einen lokalen Port, über den er mit dem Server kommunizieren will. Der Server beantwortet die Anforderung (**Response**) und übermittelt, wenn er den angeforderten Dienst leisten kann bzw. leisten darf, dem Client eine Bestätigung. Diese enthält eventuell wiederum einen lokalen Port des Hosts, auf dem der Server-Prozeß läuft, und über den die Kommunikation Server-seitig abgewickelt werden soll. (Somit wird der Well-Known Port wieder für weitere Clients frei). Diese Art der Interaktion wird allgemein auch als **Client/Server-Prinzip** bezeichnet.

Aufgrund der engen Beziehung zwischen UNIX und TCP/IP wird der Server üblicherweise als **Daemon** bezeichnet. Ein TELNET-Server wird daher mit **telnetd** (d für daemon) bezeichnet, ein FTP-Server analog mit **ftpd**.

Das TELNET-Protokoll bietet eine Terminal-Emulation zwischen zwei Hosts. Dazu wird auf dem entfernten Host ein TELNET-Server in Form eines Dämon-Prozesses, der `telnetd` heißt, gestartet. Der TELNET-Client, der auf dem lokalen Host durch ein Programm namens `telnet`[12] unter der Angabe der Adresse des entfernten Hosts gestartet wird, nimmt über den Well Known Port für TELNET (Port 23, auf der Seite des Servers) Verbindung mit dem TELNET-Server via TCP auf. Nach einem `login` auf dem lokalen Host (Client) kann mit dem entfernten Host (Server) wie über ein normales Terminal, wie z.B. VT100 oder ANSI, kommuniziert werden. Durch einen `quit` Befehl wird die TELNET-Sitzung

[12]Das Programm kann auch anders heißen, z.B. bei einem TELNET-Server, der auf einem MVS/VM-Host läuft, heißt es `tn3270`

beendet und die Verbindung wieder abgebaut.

Das **File Transfer Protocol, FTP** ermöglicht einen Datenaustausch zwischen Hosts, die auch unterschiedliche Dateisysteme, Datei- oder Zeichenformate benutzen können. So ist zum Beispiel eine Umwandlung von EBCDIC[13] nach ASCII[14] ebenso möglich wie ein Transfer von UNIX nach DOS.

Die Benutzerschnittstelle zu FTP bietet das gleichnamige Kommando `ftp`. Dadurch wird FTP als Client-Prozeß auf dem lokalen Host gestartet. Auf dem entfernten Host, mit dem eine FTP-Sitzung durchgeführt werden soll, muß der Server-Prozeß `ftpd` laufen (bzw. gestartet werden).

Nach einem `login` und der Eingabe des Passwortes kann mit Hilfe von FTP-Kommandos wie `cd`, `ls`, `put` und `get` ein Filetransfer vom Client zum Server und umgekehrt durchgeführt werden.

Zur Kommunikation zwischen Client und Server werden von FTP auf der Seite des Servers *zwei* Well Known Ports benutzt: Port 21 dient der Übermittlung von Kontroll-Meldungen und Port 20 für der eigentlichen Datenübertragung (Outband Signaling).

Zum Abschluß dieses Kapitels soll die Rolle der Socket-Schnittstelle noch einmal genauer betrachtet werden, da sie eine besondere Bedeutung bei der Korrelation der Netz- und System-Daten hat:

Ein **Socket** ist, wie bereits angesprochen, ein Kommunikationsendpunkt. Er befindet sich innerhalb des Betriebssystem-Kerns (UNIX Kernel). Sein Pendant auf der Seite des Netzes ist der **Port**.

Ein Socket auf der Seite des Clients besteht aus einem Identifikator für die vorliegende Protokollfamilie (hier `AF_INET` für TCP/IP) und der entfernten Adresse des Servers (**Foreign Address**). Die Adresse besteht aus der Host Adresse (hier: IP-Adresse) und einer Port-Nummer (hier: UDP- oder TCP-Port). Der Socket des Clients wird implizit an die lokale Adresse **Local Address**, bestehend aus lokaler IP-Adresse und lokaler Portnummer, gebunden.

Der Server bindet explizit einen oder mehrere Sockets an eine lokale Adresse, bestehend aus der lokalen IP-Adresse und dem bzw. den Well-Known-Port(s). Sodann wartet er darauf, das Clients mit ihm Kontakt aufnehemen.

Einem Socket sind zwei Warteschlangen zugeordnet: Eine **Sending Queue** und eine **Receiving Queue**. Ein Socket befindet sich stets in einem fest definier-

[13]Extended Binary Coded Decimal Interchange Code

[14]American Standard Code for Information Interchange

ten **Zustand**. Der Server befindet sich z.B. im Zustand `LISTEN`, wenn er darauf wartet, daß Clients mit ihm Kontakt aufnehmen.

Möchte nun ein Client mit einem Server (remote) eine Verbindung aufbauen, so erzeugt er, falls noch nicht vorhanden, zunächst einen Socket. Dieser Socket wird an einen freien Port gebunden. In den Socket wird durch das Binden die lokale Adresse eingetragen. Jetzt kann über den Socket eine Verbindung zum Server aufgenommen werden. Der Server wird über den Well Known Port adressiert. Über den zugehörigen Socket auf der Seite des Servers, der sich im Zustand `LISTEN`[15] befand, wird nun eine Verbindung aufgebaut. Der Socket auf der Server-Seite erhält als Foreign Address die Adresse des Clients, die dieser beim Request mit übermittelt hat, und sendet an den Client eine Bestätigung (Response). Die Foreign Address auf der Client-Seite ist die Adresse des angesprochenen Servers. Beide Sockets befinden sich nun im Zustand `ESTABLISHED`, die Verbindung ist aufgebaut. Der Verbindungsabbau geht ähnlich von statten. Ein nicht (mehr) benutzter Socket befindet sich im Zustand `CLOSED`. Wird er nicht mehr benötigt, so kann er wieder gelöscht werden.

Bei UDP wird keine Verbindung aufgebaut, die Kommunikation erfolgt jedoch ebenfalls über die Socket-Schnittstelle. Die versendeten UDP-Datagramme werden einfach in der Receiving-Queue aufgenommen und von dort von einer Applikation entnommen.

Bevor die in dieser Arbeit behandeten Methoden und die entsprechenden Werkzeuge diskutiert werden und ein eigenes (am Lehrstuhl entwickeltes) Accounting Modell vorgestellt wird, sollen zuerst bestehende Accounting Modelle sowie die Bedeutung des Monitoring für das Accounting besprochen werden.

Als bestehende Accounting Modelle werden das OSI Accounting Modell und das Internet Accounting Modell vorgestellt. Dazu wird zunächst einführend ein (sehr) kurzer Einblick in das OSI Network Management gegeben.

[15] Der Server hört den Well Known Port ständig ab

3 Accounting Modelle

3.1 OSI Network Management

Im folgenden Abschnitt werden kurz architekturelle Eigenschaften des OSI Network Management beschrieben. Es werden dabei einige grundlegende Begriffe der OSI Network Managementterminologie erläutert.

OSI Network Management ist vom Basic Reference Model, Management Framework, ISO/IEC 7498-4 in der Anwendungsschicht des OSI Basisrefernzmodells nach ISO 7498-1 spezifiziert worden. Neben diesem Standard ist der Systems Management Overview, ISO/IEC 10040, Grundlage für OSI Network Management.

Der Standard ISO/IEC 10040 stellt die folgenden Modelle bereit:

- Organisationsmodell:
  Die ISO beschreibt Organisationsformen im Sinne von Rollen; für OSI Network Management werden die Rollen Manager und Agent unterschieden: Ein steuerndes und überwachendes System hat die Rolle eines **Managers**, ein gesteuertes und überwachtes System hat die Rolle eines **Agenten**. Der Agent verwaltet seine Managed Objects; diese werden im Informationsmodell spezifiziert. Der Manager führt Operationen auf den Managed Objects des Agenten aus. Die Managed Objects erzeugen ihrerseits beim Eintreten von Ereignissen **Notifications**. Diese Meldungen werden an den Manager übermittelt.

- Informationsmodell:
  Ein **Managed Object, MO** beschreibt ein zu verwaltendes, zu überwachendes oder zu steuerndes Betriebsmittel. Managed Objects sind somit Abstraktionen von Ressourcen. Beispiele für Ressourcen sind Netzkomponenten wie Hosts oder Router, aber auch logische Objekte wie Protokollinstanzen, logische Verbindungen und Filter. Alle managementrelevanten Daten und Informationen werden unter dem Begriff **Managementinformation** zusammengefaßt. Der konzeptionelle Behälter für die Managementinformation ist die **Management Information Base, MIB**. Die Beschreibung der Eigenschaften von Managed Objects werden in **Objekt-Klassen** festgelegt. Eigenschaften eines Managed Objects sind z.B. seine Attribute, sein Verhalten und seine Beziehungen zu anderen Managed Objects. Eine **Objekt-Instanz** ist die konkrete Repräsentation einer realen Ressource, die die Eigenschaften ihrer Objekt-Klasse aufweist.

- Funktionsmodell:
  Managementaktivitäten lassen sich verschiedenen Funktionsbreichen zugeordnen. Das OSI Network Management unterteilt sich in fünf Funktionsbereiche: Configuration Management, Fault Management, Security Management, Performance Management und das in dieser Arbeit behandelte **Accounting Management**. Jeder Funktionsbereich gliedert sich wiederum in einzelne Funktionen, die in den ISO-Standards als Systems Management Functions bezeichnet werden.

- Kommunikationsmodell:
  Das Kommunikationsmodell legt die Konzepte zum Austausch von Managementinformation zwischen Manager und Agenten fest. Zur Kommunikation zwischen Manager- und dem Agentsystem werden das Protokoll **CMIP, Common Management Information Protocol** und die dazugehörigen Dienste **CMIS, Common Management Information Services** verwendet. Die Kommunikation zwischen Agent und Managed Objects ist nur semantisch spezifiziert.

An dieser Stelle soll nur noch kurz auf das Informationsmodell näher eingegangen werden. Konzepte und Begriffe aus den anderen drei Modellen werden bei Bedarf an geeigneter Stelle angesprochen.

Das Informationsmodell ist eine Abstraktion der Gesamtheit aller modellierten Network Management Ressourcen. Jede Ressource hat einen Nutzungsaspekt, das ist der bereitgestellte Dienst der Ressource, und einen Managementaspekt, das ist das Managed Object der Ressource.

Um diesen Managementaspekt zu standardisieren, hat die ISO/IEC Objekt-Klassen geschaffen, die in Objektkatalogen[16] spezifiziert sind. Um eine Objekt-Klasse zu beschreiben wird eine einfache Tamplate-Metasprache[17] verwendet. In jeder Template-Definition kann auf weitere Template-Definitionen Bezug genommen werden.

In der Hierarchie dieser Bezüge steht der Template **Managed Object Class** an oberster Stelle. Er besteht aus einem **Object Identifier**, das ist der weltweit eindeutige Registrierungsnahme im ISO-Registrierungsbaum für die beschriebene Objekt-Klasse, und mindestens einem **Mandatory Package**, das sind die obligatorischen Attribute einer Objekt-Klasse. Weiterhin *kann* er Verweise auf Super-Klassen, die ihre Attribute vererben, und auch sog. **Conditional Packages**, das sind die optionalen Attribute, enthalten.

---

[16] siehe z.B. ISO/IEC 10165-2, Structure of Management Information oder ISO/IEC 10737, Elements of Management Information Related to OSI Transport Layer Standard

[17] beschrieben in ISO/IEC 10165-4, Guidelines for the Definition of Managed Objects

In den bereits erwähnten **Packages**, die in der Hierarchie der Template-Bezüge unter dem Template Managed Object Class angesiedelt sind, werden Attribute, Operationen auf Attribute, Gruppen von Attributen, Notifications und das verbal beschriebene Verhalten des Packages zusammengefaßt.

Zwischen den Managed Objects werden somit zwei Arten von Beziehungen (Relations) definiert: **Inheritance** (Vererbung; eine Objekt-Klasse erbt die Templates der Super-Klasse(n)) und **Containment** (eine Objekt-Klasse enthält (andere, weitere) Templates). Die Containment Beziehungen werden, neben den Attributen für die Instanzen-Namen und den Bedingungen für das Erzeugen und das Löschen von Objekt-Instanzen, im **Name Binding**-Template spezifiziert. Objekt-Instanzen von Objekt-Klassen müssen einen eindeutigen Instanzen-Namen besitzen. Instanzen-Namen sind gemäß der Containment Hierarchie hierarchisch aufgebaut.

Der Zugriff auf die Attribute einer Objekt-Instanz wird mit Hilfe der Dienste der Common Management Information Services, CMIS[18] durch das Common Management Information Protocol, CMIP[19] bewerkstelligt. Folgende Dienste stehen zur Verfügung:

- **M-EVENT-REPORT:** Dieser Dienst benachrichtigt den Dienstnutzer über das Eintreten eines beliebigen Events bezüglich eines Managed Objects.
- **M-GET:** Dieser Dienst ermittelt Werte von Attributen einer oder mehrerer Objekt-Instanzen.
- **M-CANCEL-GET:** Dieser Dienst löscht einen zuvor aufgerufenen M-GET noch während dessen Ausführungszeit.
- **M-SET:** Dieser Dienst verändert die Werte von Attributen einer oder mehrerer Objekt-Instanzen, falls dies zulässig ist.
- **M-ACTION:** Dieser Dienst kann eine nicht weiter festgelegte Information für eine oder mehrere Objekt-Instanzen enthalten.
- **M-CREATE:** Dieser Dienst erzeugt eine Objekt-Instanz einer Objekt-Klasse.
- **M-DELETE:** Dieser Dienst löscht eine Objekt-Instanz.

[18] siehe ISO/IEC 9595
[19] siehe ISO/IEC 9596

3.2 Das OSI Accounting Modell

In diesem Abschnitt wird das OSI Model for Accounting skizziert, so wie in ISO/IEC 10164-10.2 beschrieben. Dabei wird in diesem Draft International Standard vor allem auf die **Usage Metering Function** des Systems Management eingegangen.

3.2.1 Der Accounting Prozeß

Accounting ist ein Prozeß, der aus drei Subprozessen besteht. Diese Subprozesse sind:

- Der **Usage Metering Process**:
  Dieser Prozeß ist für das Kreieren von **Accounting Records**, sobald **Accountable Events** in Kommunikationssystemen auftreten, und für das Logging dieser Accounting Records zuständig. Dabei können mehrere Accountable Events in einem einzigen Accounting Record aufgezeichnet werden. Normalerweise werden für eine einzige Dienstinanspruchnahme (**Service Call**) von mehreren Systemen Accounting Records erzeugt.

- Der **Charging Process**:
  Dieser Prozeß sammelt Accounting Records, die zu einer bestimmten Dienstinanspruchnahme gehören und fügt sie zu **Service Transaction Records** zusammen. Hier wird also die Zuordnung zum genutzen Dienst realisiert. Den Service Transaction Records wird dann anschließend noch Kosteninformation (**Pricing Information**) hinzugefügt. Überdies ist der Charging Process für das Logging der Service Transaction Records zuständig.

- Der **Billing Process**:
  Dieser Prozeß sammelt während einer bestimmten Zeitspanne Service Transaction Records, die zu einem bestimmten Dienstnutzer gehören und erzeugt mit deren Hilfe die Rechnungen. Hier wird die Zuordnung zum Dienstnutzer realisiert.

3.2.2 Der Usage Metering Prozeß

In ISO/IEC 10164-10.2 wird ausschließlich der Usage Metering Prozess modelliert. Da gerade dieser Prozeß hauptsächlich Gegenstand der vorliegenden Arbeit ist, wird zunächst nur dieser Prozess näher betrachtet.

Im Zusammenhang mit Usage Metering sind folgende Objekte von besonderer Bedeutung:

- **Accountable Object:**
  Ein Accountable Object ist ein Managed Objekt, das eine Ressource repräsentiert, für die **Usage Data** (Nutzungsdaten) erfaßt werden.

- **Usage Metering Control:**
  Die Usage Metering Control ermöglicht die Kontrolle über die Sammlung der Usage Data von einem Accountable Object; sie startet und stoppt die Sammlung durch Management-Operationen. Zudem bestimmt die Usage Metering Control, welche Usage Data gesammelt und unter welchen Umständen sie berichtet werden sollen.

- **Usage Metering Data:**
  Die Usage Metering Data repräsentieren die erfaßte Nutzung (Accounted Use) eines Accountable Objects durch einen Nutzer. Diese Daten enthalten zusammen mit anderen geeigneten Daten Informationen zur Identifikation des Dienstnutzers, zur Identifikation des erbrachten Dienstes und ein Maß, das Auskunft über die Quantität (und, wenn nötig, auch über die Qualität) der Nutzung gibt.
  Die Usage Metering Data können entweder durch Usage Metering Data Parameters, durch speziellen Usage Metering Notifications oder mit Hilfe der GET-Operation aus den entsprechenden Meter Data Attribut-Werten gewonnen werden.

3.2.3 Strukturelle Relationen zwischen den MOs

Ein Beispiel soll die strukturellen Beziehungen zwischen diesen drei Objekten verdeutlichen: Ein System Object enthält mehrere Accountable Objects. Diese werden von einem Usage Metering Control Object, das ebenfalls im System Object enthalten ist, kontrolliert. Die Accountable Objects enthalten ihrerseits ihre entsprechenden Usage Metering Data Objects. Die Usage Metering Data Objects werden ebenfalls durch das Usage Metering Control Object kontrolliert.

Mit Hilfe der Usage Metering Data Objects wird ein Usage Record Object erzeugt. Dieses Usage Record Object ist in einem Log Object enthalten, das wiederum Teil des System Objects ist. (Beispiel-Instanzen dazu siehe Abbildung 3).

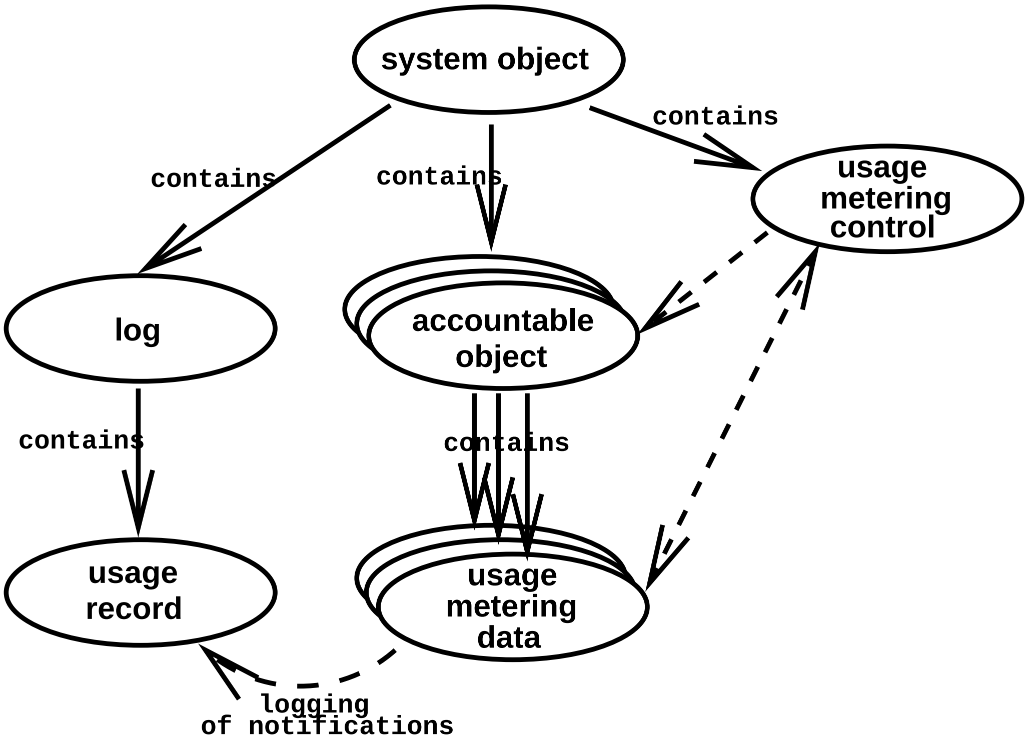

Abbildung 3: Beispiel-Instanzen zur Veranschaulichung der strukturellen Relationen zwischen den Managed Objects

3.2.4 Steuerung des Usage Meterings

Um das Usage Metering zu aktivieren, muß zuerst (mindestens) eine Control Object-Instanz durch eine CREATE-Operation erzeugt werden. Ein Control Object sollte Information enthalten

- zur Identifikation der **Accounting Units** eines Accountable Objects, die für das Usage Metering benutzt werden sollen.
- über **Reporting Triggers**; durch sie werden **Events** spezifiziert, bei deren Auftreten ein Usage Data Object eine **Usage Data Notification** aussenden soll. Analog werden sog. **Recording Triggers** definiert, die die Aufzeichnung der Metering Data steuern.

Für jede Accountable Object-Instanz wird eine Usage Data Object -Instanz erzeugt. Falls ein Accountable Object nur zum Zecke des Accounting existiert, so ist es sinnvoll das Accountable Object und das Usage Data Object in einem Objekt zusammenzufassen. Usage Data Object-Instanzen werden, wenn sie nicht mehr benötigt werden, durch eine DELETE-Operation wieder gelöscht.

Das Usage Metering wird über die Control Object-Instanzen durch Events gesteuert. In ISO/IEC 10164-10.2 sind folgende Events definiert:

- `Start Metering`
- `Recording Trigger` (geben an, unter welchen Umständen die Usage Data aufgezeichnet werden sollen)
- `Reporting Trigger` (geben an, unter welchen Umständen eine Usage Data Notification ausgesendet werden soll)
- `Report Complete`
- `Suspend Metering`
- `Resume Metering`
- `Delete Request` (führt zu einem `Delete Object`; falls sich das Usage Metering im Zustand `Active` befindet wird zuerst eine `Usage Meter Data Report` Notifcation ausgesendet und dann in den Zustand `Completed` übergegangen (siehe unten))

Je nach eintretendem Event und abhängig vom aktuellen Zustand des Usage Meterings kann sich das Usage Metering in vier Zuständen befinden:

- Not Active
- Usage Collection Is Active;
  `meteringActive` ist `TRUE`
- Usage Collection Is Completed;
  `meteringActive` ist `TRUE`
- Usage Collection Is Active;
  `meteringActive` ist `FALSE`
  (Das Usage Metering ist vorübergehend nicht aktiv)

3.2.5 Die Usage Records

Die **Usage Record Class** ist eine Unterklasse der **Event Log Record Class**. Die Definitionen einer Log und ihrer Kontrolle sind in der **Log Control Function**, ISO/IEC 10164-6 beschrieben.

Die Daten, die in einer Usage Record Instanz enthalten sind, setzen sich aus der Information der Notifications, die von Instanzen von Usage Metering Data Objects ausgesendet werden, zusammen. Eine Usage Information Instanz besteht aus einer Folge von **Basic Information Blocks**. Folgende Blöcke sind möglich:

- **Registration**:
  - Identity of User
  - Time Stamp
- **Request**:
  - Identity of Remote Parties (Kommunikationspartner)
  - Service Variant (Dienstvariante, z.B. spezielle Bandbreite)
  - Meter Information (Identifizierung und Zuordnung der Units of Usage — Count of Usage)
- **Accept**:
  - Identity of Remote Parties (Kommunikationspartner; Übereinstimmung: gerufener Partner — tatsächlicher Partner)
  - Service Variant (Dienstvariante; Übereinstimmung mit angeforderter Dienstvariante)
  - Meter Information (Identifizierung und Zuordnung der Units of Usage — Count of Usage)
- **Complete**:
  - Meter Information (Identifizierung und Zuordnung der Units of Usage — Count of Usage); akkumulierte Werte ab der Zeit des Zustandekommens der Kommunikationsbeziehung bis zu ihrer Beendigung
  - Reporting Trigger (Wert des Reporting Triggers der den **Usage Report** initiiert hat)

  - Completion Cause

- **Corresponding**:
  - Service Transaction (eindeutiger Identifier zur Korrelation der Usage Records, die zu ein und der selben Service Transaction gehören)

- **Bulk**: Dieser Block dient zur Messung von Nicht-Event-Bezogender Nutzung von Ressourcen, wie z.B. die Messung des Datenverkehrs über eine Standleitung in einem bestimmten Zeitintervall.
  - Meter Information (Identifizierung und Zuordnung der Units of Usage — Count of Usage); akkumulierte Werte ab der Zeit des Zustandekommens der Kommunikationsbeziehung bis zu ihrer Beendigung
  - Reporting Trigger (Wert des Reporting Triggers der den **Usage Report** initiiert hat)

- **Interruption**:
  - Meter Information (Identifizierung und Zuordnung der Units of Usage — Count of Usage); akkumulierte Werte ab der Zeit des Zustandekommens der Kommunikationsbeziehung bis zu ihrer (nicht ordnungsgemäßen) Unterbrechung bzw. ihrem Abbruch.
  - Reporting Trigger (Wert des Reporting Triggers der den **Usage Report** initiiert hat)
  - Interrupt Cause

Näher soll an dieser Stelle auf die Usage Metering Function nicht eingegangen werden. Falls im folgenden genauere Information nötig sein sollte, so wird sie an der entsprechenden Stelle nachgereicht.

Alternativ zum OSI Accounting Modell wird nun das Internet Accounting Modell vorgestellt. Es handelt sich dabei um einen sehr speziellen, pragmatischen Ansatz. Für das Internet Accounting existiert bereits eine entsprechende MIB. Diese Internet Accounting MIB wird im Anschluß an das Internet Accounting Modell kurz angesprochen.

3.3 Das Internet Accounting Modell

Dieser Abschnitt stellt kurz das Internet Accounting vor, so wie es in RFC 1272, Internet Accounting: Background beschrieben und in der Internet Accounting Management Information Base bereits syntaktisch realisiert ist.

Internet Accounting basiert auf ISO 7498-4, OSI Reference Model Part 4: Management Framework und beschränkt sich zunächst auf das **Usage Reporting**. Das OSI Accounting Modell, so wie in ISO 7498-4 beschrieben, definiert drei Basis-Einheiten:

- Einen **Meter**, der Messungen durchführt und die Ergebnisse seiner Messungen ansammelt.
- Einen **Collector**, der Meter-Daten sammelt und für die Integrität und die Sicherheit der Meter-Daten bei der Zwischenspeicherung und bei der Übertragung zuständig ist.
- Eine **Application**, die die Meter-Daten verarbeitet, formatiert, speichert und die Meter selbst implizit verwaltet.

In der Accounting MIB werden (zunächst) nur die Meters spezifiziert.

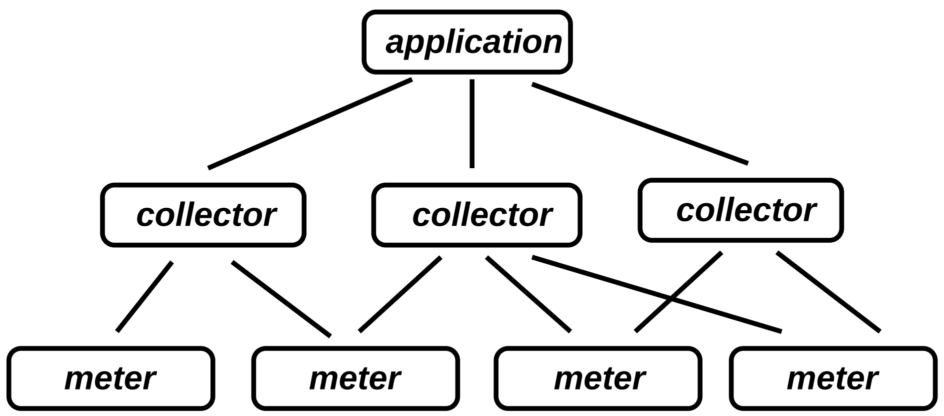

Abbildung 4: Das Internet Accounting Modell

3.3.1 Accounting im Internet

Das Internet ist in **Administrative Domains** unterteilt, wobei jede Domain wohldefinierte Grenzen hat und von einer **Network Administration** bzw. einem

Network Administrator verwaltet wird. Für den Network Administrator sind nun zwei Fälle von Interesse:

- Der Datenverkehr **innerhalb** seiner Domain.
- Der Datenverkehr **durch** seine Domain.

Innerhalb seiner Domain kann ihn der Datenverkehr zwischen zwei End-Systemen oder, weniger detailliert, der Datenverkehr zwischen zwei Abteilungen interessieren.

Was den Datenverkehr anbelangt, der Domain-Grenzen überschreitet, so ist hier das sog. **Recursive Accounting** vorgesehen: Die Benutzung von Ressourcen wird der entsprechenden **Adjacent Domain** in Rechnung gestellt. Wird z.B. ein Dienst in Domain A über Domain B von Domain C aus in Anspruch genommen, stellt zunächst die Domain A der benachbarten Domain B eine Rechnung dafür aus. Domain B ermittelt, daß ja ihre Nachbar-Domain C die Kosten verursacht hat und stellt ihrerseits eine Rechnung für Domain C aus, die die von Domain A in Rechnung gestellten Kosten **plus** der angefallenen Kosten aus Domain B selbst enthält.

Nach dem eben skizzierten Modell muß das Internet Accounting zwei Arten von Accounting unterstützen: Das **End System Accounting** und das **Adjacent System Accounting**.

3.3.2 Accounting Meters

Ein **Meter** ist ein Prozeß, der einen Strom von Paketen auf einem Kommunikationsmedium bzw. zwischen zwei Medien untersucht.

Der Meter zeichnet die Anzahl der Pakete, die zu einem **Flow** gehören, auf; ein Flow ist eine Folge von Paketen zwischen zwei kommunizierenden **Entities** (z.B. zwischen zwei Hosts oder zwischen zwei Domains). Die Zuweisung von Paketen zu Flows geschieht mit Hilfe von **Rules** (Regeln).

Meter können sinnvollerweise als dedizierte Monitore, in Routern (Bridges / Gateways) oder auf Arbeitsstationen implementiert werden. In einem LAN-Segment ist die Plazierung der Meter unkritisch, da man i.d.R. von jedem Punkt aus jedes Paket mitbekommen kann. Bei der LAN/WAN-Anbindung oder bei der Kopplung von LANs sind die Router (Bridges) die geeignetsten Erfassungspunkte.

Meter benötigen zur Speicherung der Flows Speicherplatz. Da dieser mehr oder weniger begrenzt zur Verfügung steht, muß das **Reporting Interval** hinreichend kurz gewählt werden. Ein zu kurz gewähltes Reporting Interval belastet das Datennetz jedoch zu sehr, wohingegen bei einem langem Reporting Interval der Speicherplatz in den Meters sehr groß sein muß.

Für die Steuerung des Reporting Intervals stehen Notifications in Form von Traps zur Verfügung. Ein Trap warnt davor, daß der Meter-Speicher schon sehr voll ist, ein weiterer Trap meldet den Verlust von Flow-Daten, falls der Speicher ganz voll geschrieben wurde.

3.3.3 Accounting Collectors

Die **Collectors** sind nicht näher spezifiziert. Für die Spezifikation von Collectors sollte man zunächst zwei Punkte berücksichtigen:

- Die **Matrix of Communicating Entities** in den Meters ist zwar recht groß, aber i.d.R. relativ schwach besetzt, so daß man für ihre (weitere) Speicherung Listen verwenden kann.
- Die Meters können entweder statisch oder dynamisch konfiguriert werden. Eine statische Konfiguration wäre eine fest vorgegebene Anzahl an Zählern, deren Werte entsprechend erhöht werden. Eine dynamische Konfiguration läge vor, wenn sich die Anzahl und / oder die Art der Einträge (Zählerstand *und* Spezifikation des Zählers) ändert. In diesem Fall müssen diese Einträge periodisch gesammelt und anschließend wieder aus dem Meter-Speicher entfernt werden.

Weiterhin sollten die Collectors die folgenden Anforderungen erfüllen:

- Schutz der Daten gegen unberechtigten Zugriff.
- Integrität der Daten.
- Die Steuerung der Sammlung sollte sowohl lokal als auch remote möglich sein.

Für den Datenschutz bieten sich die **Security Services** des SNMP [20] an. Die Datenintegrität kann durch eine bestätigte Übertragung, durch ein redundantes Reporting von einem Meter zu meheren Collectors und zusätzlichen Backup-Speicher in den Meters gewährleistet werden.

[20] Simple Network Management Protocol

3.3.4 Die Internet Accounting MIB

Die Internet Accounting Management Information Base ist in vier Gruppen unterteilt:

- **Control Group**
- **Flow Group**
- **Rule Group**
- **Action Group**

In der **Control Group** wird der Meter-Speicher überwacht und das `Sampling-Interval` (für statistisches Sampling) festgelegt. Zudem enthält sie u.a. Information über die verwendeten Regeln und über Notfallregeln (falls ein Störfall eintritt).

Die **Flow Group** besteht aus der `FlowTable`, in der die Meßdaten, die `FlowTableEntries`, gespeichert werden. Da ein Flow gerichtet ist, enthält jeder FlowTableEntry Zähler für beide Richtungen (Source — Dest, Dest — Source). Zusätzlich besteht die Flow Group aus einer `CreationTable` (Teilmenge der neu erzeugten FlowTableEntries), einer `ActivityTable` (Teilmenge der zuletzt veränderten FlowTableEntries) und einer `ActivityColumnTable` (Teilmenge der zuletzt veränderten FlowTableEntries; ausserdem gibt es die Möglichkeit, Werte eines bestimmten Attributs (z.B. `FlowSourceAdjacentAddress`) eines jeden aktiven Flows mit Hilfe dieser Tabelle abzufragen.).

Die **Rule Group** besteht aus der `RuleTable`; die RuleTable besteht wiederum aus den `RuleTableEntries`. Ein RuleTableEntry besteht aus einem `RuleSelector` (= das zu Überprüfende Attribut), einer `RuleMask`, einem `Rule-MatchedValue` und einer `RuleAction`, die ausgeführt wird, falls gilt:

```
RuleSelector AND RuleMask == RuleMatchedValue
```

Es gibt acht RuleActions wie z.B. `Count` oder `Goto` (gehe zur nächsten Regel).

Die **Action Group** besteht aus der `ActionTable`, die wiederum aus `Action-TableEntries` besteht. Mit einem ActionTableEntry wird, sofern noch nicht vorhanden, ein FlowTableEntry in der FlowTable erzeugt.

Zusätzlich existieren drei Traps: `declareHighWater` warnt vor einem Meter-Speicherüberlauf, `declareDataLoss` meldet einen Datenverlust (z.B. in Folge eines Meter-Speicherüberlaufs) und mit `declareFlood` können bei vollem Meter-

Speicher die Flow Records, die nicht mehr gespeichert werden können, übertragen werden.

An dieser Stelle soll die Bestrachtung bestehender Accounting Modelle beendet werden. Die betrachteten Accounting Modelle dienen als Basis für die Konstruktion eines eigenen (am Lehrstuhl entwickelten) Accounting Modells. Bevor darauf eingegangen wird, wird zunächst noch die Bedeutung des Monitorings für das Accounting sowie das Remote Monitoring und seine Realisierung durch RMON (siehe weiter unten) betrachtet.

4 Accounting und Monitoring

4.1 Monitoring

Monitoring umfaßt alle Methoden, die zur Überwachung eines Systems dienen. Dazu werden i.d.R. kontinuierlich alle *relevanten* Systemdaten ermittelt und für die Analyse geeignet abgespeichert und / oder weitergeleitet.

Das in diesem Fall zu überwachende System ist das **Netz** (**Network Monitoring**). Im gesamten Netz sind (je nach geforderter Granularität) **Monitoring Agents** verteilt, die jeweils einen Teilbereich des Netzes (wie z.B. einen LAN / WAN - Übergang oder ein Segment) überwachen und aufzeichnen. Diese Information kann von **Management Agents** abgefragt werden bzw. sie wird automatisch an Management Agents weitergeleitet (durch **Polling** bzw. **Event-Driven**). Ein Management Agent kann einem übergeordneten Manager als Agent dienen, eine spezielle Netzmanagement Applikation oder ein Netzmanagement System sein.

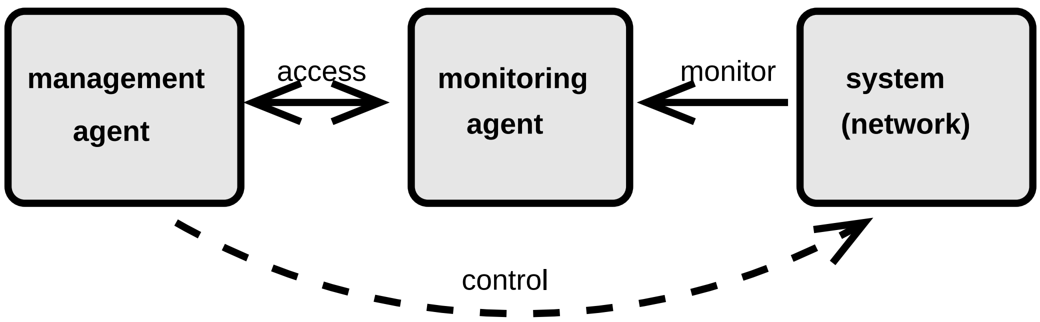

Abbildung 5: Beziehungen zwischen Management Agent, Monitoring Agent und dem zu überwachenden System

Der bereits vorgestellte Accounting Meter ist somit ein spezieller Monitoring Agent; er zeichnet abrechnungsrelevante Daten auf, speichert diese Daten (eventuell (vor-)analysiert) ab und leitet sie an einen Manager, den Accounting Collector, weiter.

Beim Monitoring wird grob **internes** (**Internal Monitoring**) und **externes Monitoring** (**External Monitoring**) unterschieden: Beim internen Monitoring befindet sich der Monitoring Agent am selben Ort wie die Objekte, die überwacht werden sollen (Z.B. ein Monitor für eine Bridge in der zu überwachenden Bridge). Beim externen Monitoring wird der Monitoring Agent außerhalb der zu überwachenden Objekte an das Kommunikationsmedium angeschlossen (zum Beispiel als eigener Host mit eigenem LAN-Interface) Ein Monitoring Agent der keine Kommuniaktion mit einem Management Agent unterhält und für sich alleine steht, wird als **Monitor** bezeichnet.

Das Monitoring kann als Datenlieferant für mehrere Netzmanagementbereiche dienen. Dabei benötigt das Abrechnungsmanagement die bei weitem detailliertesten Daten.

Für das Leistungsmanagement (Performance Management) werden z.B. auch Pakete gezählt, wobei es jedoch in der Regel uninteressant ist, wem die einzelnen Pakete zuzuordnen sind. Für das Fehlermanagement (Fault Management) ist etwa nur noch das Verhältnis der insgesamt übertragenen Pakete zu den fehlerhaften Paketen interessant. Die mit Hilfe des Monitorings gewonnenen Daten werden lediglich nach oben hin, und letztendlich in den Netzmanagement Applikationen, unterschiedlich verarbeitet und präsentiert.

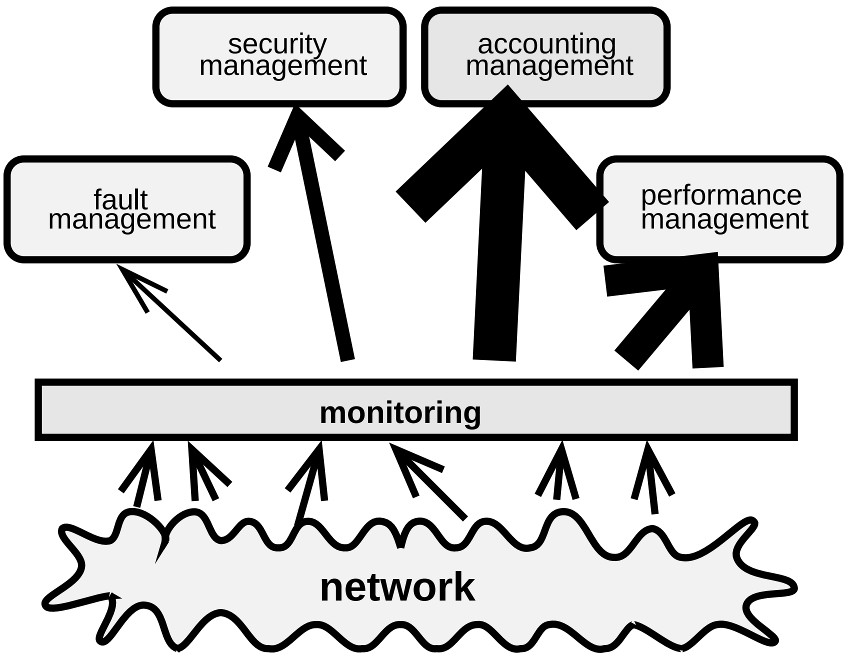

Abbildung 6: Monitoring als Informationslieferant für verschiedene Netzmanagementbereiche

Durch ein hinreichend detailliertes Monitoring wäre eine einfache Integration des bisher meist vernachlässigten Abrechnungsmanagements in ein integriertes Netzmanagement möglich. Dabei kann ein vereinheitlichtes, das Gesamtnetz umfassendes Monitoring als Informationsplattform für mehrere Netzmanagement Applikationen dienen

Eine spezielle Form von Monitoring ist das sog. Remote Monitoring. Dazu existiert eine entsprechende Remote Monitoring MIB, mit der ein verteiltes Monitoring realisert werden kann. Diese MIB ist in den zu überwachenden Koppelementen bzw. in Probes aktiv und kann *remote* verwaltet werden. Sie wird an dieser Stelle besprochen, weil viele Werkzeuge zur Erfassung von Netzverkehr[21] ihre Daten über diese RMON MIB beziehen.

[21] Wie z.B. auch das in dieser Arbeit behandelte Tool NetMetrix

4.2 Remote Network Monitoring

Mit Hilfe des **Remote Network Monitoring, RMON** kann eine einheitliche Monitoring-Plattform für das Management von TCP/IP-basierten Datennetzen realsiert werden.

Es wird zusammen mit der RMON MIB im RFC 1271 beschrieben und ist dort zunächst für Ethernet spezifiziert. Eine Erweiterung für Token-Ring findet man im RFC 1513.

Das Remote Network Monitoring entstand aus den folgenden Grundgedanken:

- Eine NM-Station[22] soll nicht in ständigem Kontakt mit ihren **Remote Monitoring Devices** stehen; das bedeutet einerseits geringere Kommunikationskosten, wenn Remote Monitoring Devices z.B. nur über einen teuren WAN-Link erreichbar sind, andererseits können die Remote Monitoring Devices weitgehend unabhängig von der NM-Station Daten sammeln, was insbesondere bei Störungen (im Datennetz oder gar bei einem Ausfall der NM-Station) von Vorteil ist. Das gleiche Konzept wird auch beim Internet Accounting mit den Accounting Meters verfolgt.

- Die Remote Monitors zeichnen selbstständig die Netz-Performance und den Netzverkehr auf und führen Diagnosen durch; falls besondere Umstände eintreten, benachrichtigen sie ihre NM-Station(en). Die aufgezeichnete **Historical Information** dient dann als Basis für die weitere Diagnose. Auch hier besteht eine Parallele zum Internet Accounting. Beim Internet Accounting wird jedoch ausschließlich Netzverkehr in einer wesentlich detaillierteren Form aufgezeichnet.

- Da die Remote Monitoring Devices Daten direkt vor Ort Daten erfassen, können mit ihrer Hilfe Störungen genauer lokalisiert werden. Für das Accounting bedeutet dieser Umstand die Möglichkeit, den Weg, den die einzelnen Datenpakete nehmen, zu rekonstruieren.

- Die Remote Monitoring Devices können mit meheren NM-Stationen bzw. Managers kommunizieren. Dadurch können Management-Aufgaben aufgeteilt werden und das Netzmanagement selbst ist weniger störanfällig. Im Zusammenhang mit Accounting können zusätzlich Accounting-Manager Daten von den Remote Monitoring Devices abfragen und zu Accounting-Zwecken weiterverarbeiten.

[22] Netzmanagement-Station

Die RMON MIB besteht aus neun Gruppen: Statistics, History, Alarm, Host, HostTopN, Matrix, Filter, PacketCapture und Event. Dabei ist für das Accounting vor allem die Gruppe **Matrix** von Interesse:

Für jedes Interface des Remote Monitoring Devices kann der Datenverkehr, der über dieses Interface läuft, in zwei Verkehrs- Matrizen (für beide Richtungen) aufgezeichnet werden. Ein Eintrag in einer solchen Matrix enthält:

- Die Quell- und Zieladresse (MAC Layer[23] Address).
- Einen Zähler für die Pakete,
- einen Zähler für die Oktette und
- einen Zähler für die Fehler zwischen der Quell- und Zieladresse.

Diese Information könnte bereits direkt für Accounting benutzt werden bzw. müßte geeignet erweitert werden, z.B. um die IP-Adressen (die z.B. durch DNS[24] einfach ermittelt werden könnten, ohne daß eine weitere Analyse der einzelnen Frames nötig wäre[25]) und die TCP-Ports (für die Zuordnung zu Diensten).

Sobald eine neue Kommunikation zwischen zwei Partnern stattfindet, wird in jeder Matrix ein neuer Eintrag erzeugt; reicht der Speicherplatz nicht aus, werden die am wenigsten benutzen Einträge einfach gelöscht. Falls dies für Accounting nicht geduldet werden kann, müssen die Matrizen entweder häufig genug abgefragt werden, oder es muß ein Trap existieren, der – wie bei der Accounting MIB – vor einem bevorstehenden Datenverlust warnt. Für die Berücksichtigung des Quality of Service kann die Anzahl der Fehler herangezogen werden; diese Information fehlt in der Accounting MIB.

[23]Medium Access Control Layer, entspricht Schicht 2a des OSI-Schichtenmodells

[24]Domain Name Service Protocol

[25]Man erhält aber auf diese Weise nicht den IP-Verkehr sondern den gesamten Datenverkehr zwischen zwei Hosts, die eine IP-Adresse besitzen.

5 Methoden zur Erfassung von Netzverkehr

5.1 Anwendungs-, System- und Netzdaten

Die Gesamtheit der **abrechnungsrelevanten Daten**, die man durch eine detaillierte Nutzungserfassung gewinnen kann, läßt sich grob in drei Klassen einteilen:

- **Anwendungsdaten**:
  Das sind die abrechnungsrelevanten Daten, die direkt aus den Anwendungen, die im betrachteten Datennetz zum Einsatz kommen, selbst gewonnen werden können. Viele Anwendungen bieten Information über genutzte Ressourcen, wie Plattenplatz oder CPU-Zeit pro Benutzer. Von besonderer Bedeutung sind hierbei die Anwendungen, die *remote* als Server (wie z.B. ein FTP-Server, `ftpd`) laufen.

- **Systemdaten**:
  Das sind die abrechnungsrelevanten Daten, die auf den einzelnen Hosts von Seiten des Betriebssystems gewonnen werden können. Diese können entweder mit Hilfe des Prozeß-Accountings durch den Befehl `accton` oder durch Befehle wie `netstat` oder `rpcinfo` gewonnen werden. Zur Aufbereitung und Komprimierung der `accton`-Daten wurde am Lehrstuhl ein PERL-Script im Rahmen eines FoPra's entworfen. Da Programme wie `netstat` oder `rpcinfo` keine vollständige Information (benutzer- prozeßbezogen) liefern, wird derzeit am Lehrstuhl ein Tool namens *mapinfo* entwickelt, das eine Zuordnung von Benutzer zu Prozeß, von Prozeß zu verwendeten Sockets und zu den Daten des `accton` realisieren soll.

- **Netzdaten**:
  Das sind die abrechnungsrelevanten Daten, die aus dem Netz selbst, also mit Hilfe von Koppelementen oder Probes, gewonnen werden können. Die Schnittstelle zwischen System- und Netzdaten besteht in der bereits angesprochenen Socketschnittstelle: Die Sockets auf der Systemseite entsprechen den Ports auf der Netzseite. Diese Arbeit befaßt sich ausschließlich mit der Gewinnung von Netzdaten.

Durch ein geeignetes Zusammenführen der Daten aus diesen drei Bereichen (Korrelation) kann ein Abrechnungsmanagement für das gesamte Datennetz (unter Einbeziehung der (Betriebs-)Systemdaten und der Anwendungsdaten) realisiert werden. Hierbei kann das Datennetz im Prinzip beliebig **heterogen** sein, da die Protokolle der TCP/IP - Protokollfamilie als Schicht 3 und Schicht 4 Protokolle verwendet werden. Ebenso können auch die immer wichtiger werdenden **verteilten Anwendungen** damit erfaßt werden (RPC - Information).

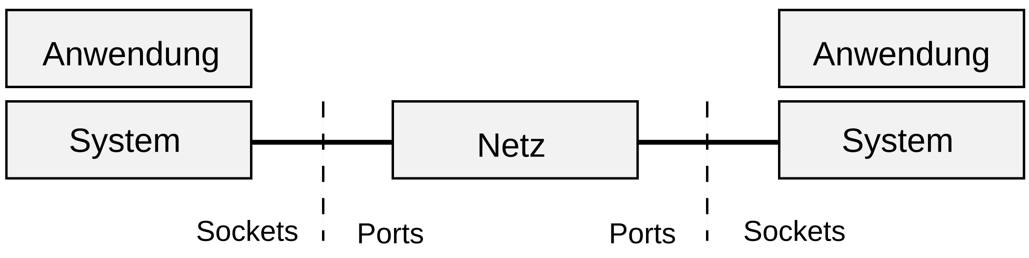

Abbildung 7: Die Bereiche Anwendung, System und Netz, aus denen die abrechnungsrelevanten Daten gewonnen werden

Die **abrechnungsrelevanten Daten** lassen sich wie folgt klassifizieren:

- **Accountable Units**:
  Das sind die Daten, die der Gegenstand der Abrechnung sind. Accountable Units sind z.B. Plattenplatz, CPU-Zeit oder Anzahl der gesendeten bzw. empfangenen Pakete.

- **Abbildungsinformation**:
  Das sind die Daten, die der Zuordnung der Daten aus den drei Bereichen (Anwendung, System und Netz) dienen: Für die Abbildung von einer TCP-Verbindung zum zugehörigen Prozeß z.B. lautet die Abbildungsinformation: Prozeß-ID, Socket (auf der Seite des Systems) *und* Port, TCP-Verbindung (auf der Seite des Netzes). Dadurch kann die Zuordnung Prozeß-ID — TCP-Verbindung gemacht werden, da der Socket den verwendeten Port enthält.

- **Abrechnungsrelevante Zusatzinformation**:
  Das sind die Daten, die zusätlich noch erfaßt werden müssen, um eine detaillierte benutzer- und dienstbezogene Abrechnung ermöglichen zu können. Beispiele dafür wären der Benutzername, der Prozeßname sowie Start- und Endzeiten von Datenübertragungen.

Bei der Betrachtung größerer, räumlich verteilter (durch Backbone oder WAN-Links gekoppelter) Datennetze kann es sehr hilfreich sein, das Netz in **Bereiche** einzuteilen. Hierbei muß natürlich die Information aus den einzelnen Bereichen korreliert werden. Diese Korrelation kommt zu der oben beschriebenen Korrelation von Anwendungs-, System- und Netz- Daten dann noch hinzu.

Ein wesentlicher Vorteil dieses **hierarchischen Ansatzes** ist die Möglichkeit, den Weg zu ermitteln, den ein einzelnes Datenpaket nimmt. Das ist nicht nur für das Performance- und Faultmanagement interessant, sondern auch für das Accountingmanagement, weil so für unterschiedliche Routen verschiedene Ko-

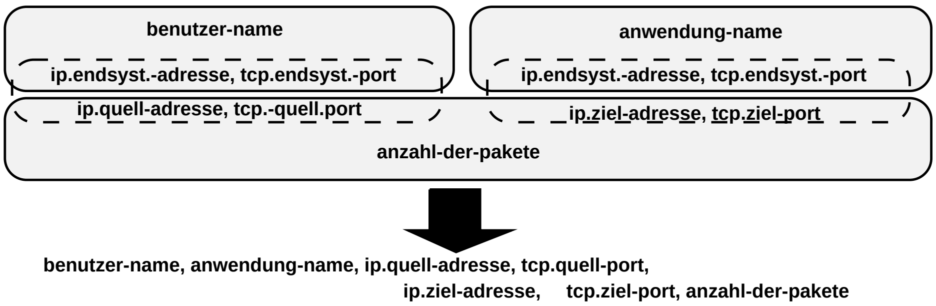

Abbildung 8: Das Zusammenführen der Daten aus den Bereichen Anwendung, System und Netz

stensätze angesetzt werden können. Der bedeutende Vorteil besteht jedoch darin, daß die CPU- und Speicherplatzbelastung, die durch das Accounting zwangsweise entsteht (Accounting Overhead), **verteilt** werden kann. In Anlehnung an das Internet Accounting Modell erfaßt jeder Meter nur die Daten in seinem Segment, d.h. er muß i.d.R. deutlich weniger Daten speichern[26] und eine eventuelle (Vor-) Analyse ist nicht so rechenintensiv. Die erfaßten Daten können in komprimierter Form in bestimmten Zeitintervallen durch die Collectors gepollt werden. Für den Fall, daß ein Meter ausfällt, müssen zusätzlich (redundante) Meters zu Verfügung stehen.

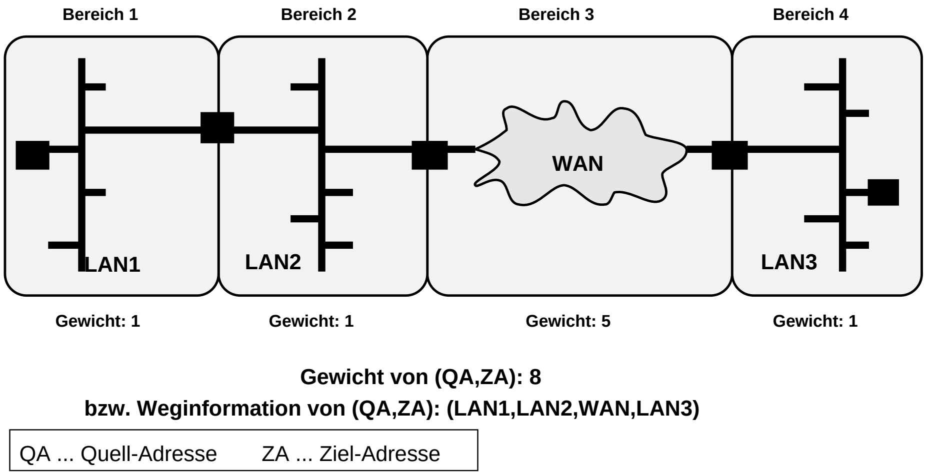

Abbildung 9: Einteilung eines Datennetzes in Bereiche

[26] Der segmentinterne Datenverkehr der anderen Segmente fällt weg

Um die Verteiltheit der Erfassung und die Komprimierung der anfallenden Massendaten beim hierarchischen Ansatz zu verdeutlichen, soll dieses Beispiel gegeben werden:

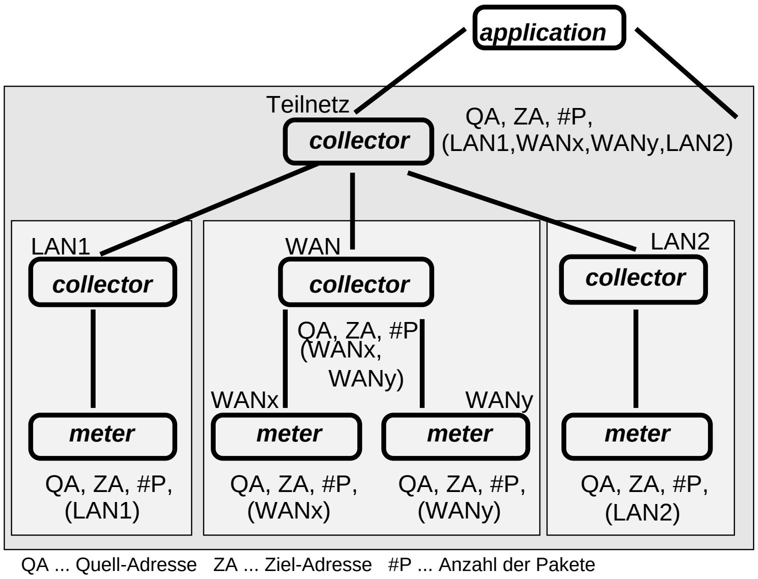

Abbildung 10: Beispiel für einen hierarchischen Ansatz mit abstrakten Beispieldaten

Ein Teilnetz besteht aus drei Bereichen. Zwei LAN-Segmente sind durch einen WAN-Link miteinander gekoppelt. In den beiden LAN-Segmenten wird je ein Meter, und an jedem LAN/WAN-Übergang ebenfalls je ein Meter plaziert. Die Meter in den LAN-Segmenten werden jeder von einem Collector abgefragt. Für den WAN-Link existiert ein Collector, der die beiden Meters an den LAN/WAN-Übergängen abfragt. Über dieser Collector-Ebene gibt es einen Collector für das Teilnetz, der die drei untergeordneten Collectors abfragt. (Bem.: eine Collector — Collector Beziehung ist im Internet Accounting Modell *nicht* vorgesehen, sie kann jedoch in sehr großen Datennetzen durchaus sinnvoll sein). Dieser Teilnetz - Collector wird zusammen mit anderen Teilnetz - Collectors von einer Application (z.B. ein Accounting Modul eines NM-Systems) verwaltet. (Bem.: die Application verwaltet auch die untergeordneten Collectors sowie die Meters; siehe dazu das Internet Accounting Modell im Abschnitt 3.3). Abstrakte Beispieldaten, die hierbei erfaßt werden könnten sind in Abbildung 10 dargestellt. Interessant ist hierbei, daß die Menge der Daten auf der untersten Ebene beim Weiterleiten durch **Unifikation** erheblich **verdichtet** wird, so daß auf der obersten Ebene nur noch ca. die Hälfte der Datenmenge ankommt, diese jedoch alle abrechnungsrelevanten Daten des erfaßten Teilnetzes in korrelierter Form enthält.

5.2 Fragestellungen bei der Nutzungserfassung

In diesem Abschnitt werden Fragestellungen diskutiert, die sich vor allem bei der Planung einer Nutzungserfassung in einem Datennetz zum Zwecke des Accountings ergeben. Dieser Abschnitt soll einen Überblick über die in diesem Zusammenhang auftretenden Probleme geben.

Wünschenswert wäre eine möglichst detaillierte Nutzungserfassung für eine *verursachergerechte* Abrechnung von Netzdiensten. Dazu ist eine sehr feine Granularität bei der Nutzungserfassung nötig, sowohl im Netz selbst (d.h. es muß an mehreren Orten möglichst genau erfaßt werden) als auch in den Endgeräten (aus den einzelnen Anwendungen und mit Hilfe des Betriebssystems). Bei der Betrachtung der im folgenden genannten Fragestellungen soll von dem Fall einer sehr feinen Granularität ausgegangen werden.

Die Nutzungserfassung soll die zentrale Frage beantworten: **Wer** nutzt **was** in **welchem Umfang** ? Das *was* sind die weiter oben beschriebenen **Accountable Units**, die Teil der abrechnungsrelevanten Daten sind. Diese werden grundsätzlich durch technische Randbedingungen und im Endeffekt aber durch die bestehende Abrechnungspolitik bestimmt. Im betrachteten Szenario ist das *was* die Anzahl der übertragenen Datenpakete (Datagramme oder Segmente) oder besser: das übertragene Datenvolumen in Byte, da Datenpakete in einem heterogenen Netz i.d.R. stark variierende Längen haben (u.a. durch Fragmentierung von IP-Datagrammen). Das *wer* ist im detailliertesten Fall der einzelne Benutzer. Weiterhin wird nicht nur pro Benutzer, sondern auch pro Dienst erfaßt und dann auch abgerechnet. Somit wird das in Rechnung gestellte übertragene Datenvolumen zusätzlich noch nach Diensten aufgeschlüsselt.

Im Zusammenhang mit einer detaillierten Nutzungserfassung ergeben sich dann folgende Fragestellungen:

- **Ebene der Erfassung**
  Im betrachteten Szenario bieten sich lediglich *zwei* Erfassungsebenen an: Die Internet-Schicht mit dem Protokoll IP und die Transport-Schicht mit den Protokollen UDP und TCP. Auf beiden Ebenen ist die Erfassung zu Accouningzwecken sinnvoll. Die Erfassung auf der Transport-Schicht setzt die Erfassung auf der Internet-Schicht voraus. Wenn man lediglich an dem Datenverkehr zwischen Hosts interessiert ist, genügt die Erfassung auf der Internet-Schicht; zudem ist diese Art der Erfassung *wesentlich* weniger aufwendig als die Erfassung auf der Transport-Schicht und kann alleine mit Daten aus dem Netz selbst realisiert werden. Die Erfassung auf der Transport-Schicht umfaßt den Datenverkehr zwischen Prozessen, die auf verschiedenen Hosts laufen. Hier müssen nicht nur wesentlich mehr Daten im Netz

selbst erfaßt werden, sondern es fallen i.d.R. gerade in Endgeräten riesige Datenmengen an, da man in diesem Fall ja zugleich eine benutzer- *und* dienstbezogene Abrechnung realisieren will.

- **Detailliertheit der Erfassung**
  Wenn die Ebene der Erfassung bestimmt ist, ist die nächste Entscheidung die Spezifikation der Detailliertheit der Erfassung. Es muß dabei geklärt werden, welche Accountable Units überhaupt erfaßt werden sollen, welche Abbildungsinformation notwendig ist und welche Abrechnungszusatzinformation für die Weiterverarbeitung der Daten benötigt bzw. gewünscht wird. Die Attribute zur Spezifikation der Detailliertheit sind allgemein durch die technische Realisierbarkeit (welche Information kann an welcher Stelle im Datennetz aus einem Protokoll-Header gewonnen werden) und ganz konkret durch die **Konfigurierbarkeit** bzw. Konfiguration der eingesetzten Werkzeuge vorgegeben. Je detaillierter die Erfassung angesetzt wird, desto größer wird der **Accounting Overhead**[27].

- **Genauigkeit der Erfassung**
  Für eine vorgegebene Detailliertheit der Erfassung sollte die Genauigkeit der Erfassung eingehend untersucht werden. Dazu sind (mehrere) Tests nötig[28]. Es muß die Frage beantwortet werden: Liefern die eingesetzen Werkzeuge wirklich die tatsächlichen Verkehrsdaten? Daß dies nicht in allen Fällen gewährleistet werden kann ist leicht einzusehen: Wenn z.B. ein Netz-Monitor (als Software) auf einer Workstation den Datenverkehr auf dem an die Workstation angeschlossenen Medium erfaßt, kann bei Lastspitzen die Verarbeitungskapazität der Workstation (auf der ja noch weitere Prozesse laufen; auch müssen Prozesse zur Kommunikation mit der Workstation und zur Kommunikation mit dem Netz-Monitor wie `rlogin` oder `snmp` vorhanden sein) nicht mehr ausreichend sein, so daß nicht mehr jedes Datenpaket erfaßt werden kann und diese Methode dadurch *ungenau* wird. Eine weitere wichtige Frage lautet: Wie kann die Genauigkeit überprüft werden? Neben dem Zusammenhang zwischen Durchsatz und Genauigkeit besteht auch ein enger Zusammenhang zwischen Detailliertheit und Genauigkeit. Je detaillierter die Erfassung ist, desto mehr Verarbeitungskapazität wird dazu benötigt und desto ungenauer kann dadurch die Erfassung in ungünstigen Situationen (z.B. bei Lastspitzen) werden. Diese Zusammenhänge müssen stets im Auge behalten und eingehend untersucht werden.

- **Wahl von geeigneten Methoden**
  Die Genauigkeit muß für jede Methode in verschiedenen, möglichst realen, Szenarios untersucht werden. Man unterscheidet grundsätzlich zwischen **ex-**

[27]Höhere CPU-Belastung, mehr Speicherplatz, höhere Netzbelastung
[28]Siehe Kriterium Genauigkeit weiter unten im Kapitel 8

akten Methoden und **statistischen Methoden**. Bei exakten Methoden wird *jedes* Frame analysiert; bei statistischen Methoden *jedes n-te* Frame. Zusätzlich gibt es noch statistische Methoden, bei denen *ungefähr jedes n-te* Frame analysiert wird, also noch Zufallsalgorithmen zum Einsatz kommen. Diese Methoden sollen im folgenden als **stochastische Methoden** bezeichnet werden. Je nach geforderter Genauigkeit bzw. zur Verfügung stehender Verarbeitungskapazität, muß eine für den konkreten Einsatzfall geeignete Methode gewählt werden. Natürlich können auch Kombinationen von Methoden verwendet werden. Mehr dazu siehe in den Abschnitten *exakte Methoden / statistische Methoden* weiter unten.

- **Ort der Erfassung**
  Eine zentrale Frage lautet: Wo kann man abrechnungsrelevante Netz-Daten überhaupt gewinnen und welche Orte sind dann für eine gegebene Methode / Detailliertheit / Ebene am geeignetsten, d.h. auch wann ist der entstehende Accounting Overhead am geringsten? Als Erfassungsorte kommen die Koppelelemente, die die einzelnen Segmente miteinander verbinden, spezielle Probes in den einzelnen Segmenten und die Endsysteme selbst in Frage. Die Erfassung in den Endsystemen (z.B. über die Socket-Schnittstelle) wird nicht weiter betrachtet. Die Vor- und Nachteile der Erfassungsorte Koppelelement und spezielle Probe werden im Abschnitt *Koppelelemente und / oder Probes* diskutiert. Der Ort der Erfassung ist auch bei der Wahl der Methode(n) zur Sammlung der erfaßten Netzdaten von entscheidender Bedeutung. Bei einem hierarchischen Ansatz werden die Erfassungsorte u.U. ganz anders gewählt werden als bei einem flachen Ansatz.

- **Benutzeridentifikation**
  Um eine verursachergerechte Abrechnung durchführen zu können, muß eine Identifikation des einzelnen Benutzers (oder zumindest des Nutzers, z.B. eine Abteilung) möglich sein. Die Zuordnung Benutzer zu erfaßter Nutzung von Ressourcen ist auf der betrachteten Ebene (OSI-Schichten 3 und 4) *nicht* möglich. Hier kann nur Abbildungsinformation (die Portnummern) gewonnen werden, mit deren Hilfe dann durch die in den Endsystemen gewonnenen Daten (über die Sockets) diese Zuordnung gemacht werden kann.

- **Anwendungsidentifikation**
  Wenn man den Netzverkehr zusätzlich nach Anwendungen abrechnen will bzw. muß, so muß man auch eine Anwendungsidentifikation realisieren, d.h. man muß eine Zuordnung von Anwendung zu Flow treffen können. Dies ist nicht so ohne weiteres möglich, da man mit Hilfe der einzelnen Frames nicht ermitteln kann, zu welcher Anwendung sie gehören, zumindest nicht bei der Analyse der IP-, UDP- und TCP-Header. Die Anwendungsidentifikation

muß ebenso wie die Benutzeridentifikation durch Korrelation mit Daten aus den Endsystemen bzw. durch Analyse der oberen Protokoll-Schichten (OSI-Schichten 5 bis 7) erfolgen.

- **Prozeßketten**
  Ein besonderes Problem stellen die sogenannten **Prozeßketten** dar: Ein von einem Benutzer gestarteter Prozeß stützt sich auf weitere Prozesse und verursacht somit u.U. ein mehr an Netzverkehr, der dem Benutzer nicht mehr direkt zugeordnet werden kann. Um dieses Problem in den Griff zu bekommen, muß in den Endsystemen der Kernel-Memory überwacht werden. So können unter Umständen die Prozeßketten mitgeloggt werden. Gerade bei **verteilten Anwendungen** sind solche Prozeßketten ständig am Entstehen. Hier besteht jedoch die Möglichkeit durch Analyse von RPC[29]-Headern bzw. ebenfalls durch Überwachung des Kernel-Memory die Prozeßketten mit zu erfassen. (Programme die den Kernel-Memory auslesen sind z.B. `netstat` und `rpcinfo`, die auf jedem UNIX-System zur Verfügung stehen.) Mit Ausnahme einer Analyse von RPC-Headern kann das Aufzeichenen der Prozeßketten ausschließlich in den Endsystemen erfolgen. Die einzige Abbildungsinformation, die Netzdaten dazu beitragen können, sind wiederum die Portnummern. Eine Analyse von RPC-Headern wird in dieser Arbeit nicht betrachtet.

- **Zusammenführen der erfaßten Daten**
  Neben Fragestellungen im Zusammenhang mit der reinen **Erfassung** der abrechnungsrelevanten Netzdaten ergeben sich auch Fragen, die sich mit der **Sammlung** bzw. dem **Zusammenführen** der erfaßten Netzdaten befassen. Zur Zusammenführen der Netzdaten (z.B. aus verschiedenen Teilnetzen) und zum Zusammenführen der Netzdaten mit den Anwendungs- und Systemdaten dient die bereits vorgestellte und des öfteren angesprochene Abbildungsinformation. Bei einem flachen Ansatz erfolgt das Zusammenführen zentral in einem leistungsfähigen Host (z.B. Mainframe). Bei einem hierarchischen Ansatz werden z.B. bereits die innerhalb eines Segments gewonnen Daten miteinander korreliert, wobei die Daten insgesamt dadurch bereits komprimiert werden (Daten werden (teilweise) zusammengefaßt). Bei der Bewertung von Werkzeugen muß die **Korrelierbarkeit** der erfaßten Netzdaten in verschiedenen Szenarien untersucht werden. Näheres dazu siehe im Kapitel 8 unter *Korrelierbarkeit der erfaßten Daten.*

- **Schutz vor Datenverlust**
  Methoden zur Erfassung von Netzverkehr für das Abrechnungsmanagement müssen extrem **zuverlässig** sein, d.h. ein Verlust an erfaßten abrechnungsrelevanten Daten muß so gut wie unmöglich sein. Es müssen daher beson-

[29]Remote Procedure Call Protocol

dere Vorkehrungen zum Schutz vor Datenverlust getroffen werden. Dazu zählen unter anderem redundante Meßpunkte (Mehrfacherfassung, Ersatzmeßpunkte), kurze Reporting-Intervalle und Plattenspeicherplatz in den Meßpunkten.

- **Schutz vor Manipulation**
  Genauso wichtig wie der Schutz vor Datenverlust ist der Schutz vor Datenmanipulation. Besonders Prozesse und Logs in den Endsystemen stellen in diesem Zusammenhang einen Schwachpunkt dar. Hier müssen Vorkehrungen zur Vermeidung, zur Entdeckung von einer Datenmaipulation und zur Schadensbegrenzung nach erfolgter Datenmanipulation getroffen werden. Wenn z.B. auf einem Endsystem ein Port-Monitor-Hintergrundprozeß läuft, der die Socket-Schnittstelle überwacht, und dieser Prozeß versehentlich oder aber auch mit Absicht von einem Benutzer dieses Endsystems abgebrochen (`kill`) wird, so muß dies so schnell wie möglich erkannt werden (z.B. remote durch einen Kontroll-Prozeß auf einem vor Zugriff geschützten Host) und der Port-Monitor so schnell als möglich wieder gestartet werden (Schadensbegrenzung). Tritt dieses Ereignis öfter auf, so muß ein Mitarbeiter zu diesem Endsystem geschickt werden, der der Sache nachgehen soll. Wesentlich schwieriger zu entdecken sind Manipulationen an den Abrechnungsdaten selbst, z.B. die Manipulation einer System-Log in einem Endsystem zu Gunsten eines oder mehrerer Benutzer. Hier sollen aber zunächst die Zugangskontrolle und die Möglichkeit der Vergabe von Rechten innerhalb von UNIX als ausreichend betrachtet werden.

- **Datenschutz**
  Die erfaßten, abrechnungsrelevanten Daten müssen nicht nur gegen Verlust und Manipulation geschützt werden, sondern auch gegen rein lesenden Zugriff durch unberechtigte Dritte. Nur die den einzelnen Benutzer betreffenden Abrechnungsdaten sollten auch nur diesem zugänglich sein. Methoden zur Erfassung von Netzverkehr können nicht nur zur Kontrolle des Netzes und zur Abrechnung von Netzverkehr verwendet, sondern auch zur **Kontrolle des Benutzers** mißbraucht werden. So bedeuten detailliertere Daten auch eine verbesserte Möglichkeit der Benutzerkontrolle. Aufgrund dieser Tatsache scheitert die Einführung eines detaillierten Abrechnungsmanagements in den Unternehmen meist am Widerstand des Betriebsrates. Eine gleichwertige Kontrollmöglichkeit besteht in der Mainframe- orientierten Datenverarbeitung in den Rechenzentren schon lange, jedoch werden dort die Daten als hinreichend geschützt betrachtet (geschlossenes System).

- **Integration in die bestehende NM-Umgebung**
  Die Integration kann auf drei Ebenen erfolgen; dabei wäre eine vollständige Integration (auf allen drei Ebenen) wünschenswert. Die drei Ebenen sind:

  – Integration auf der Ebene der eigentlichen Nutzungserfassung. Z.B. Verwendung von bereits vorhandenen RMON MIBs in Koppelelementen; damit Einreihung in die bereits vorhandenen NM[30]-Werkzeuge.

  – Integration auf der Ebene der Sammlung bzw. Korrelation der erfaßten Nutzungsdaten; vorhandene Kollektoren mitbenutzen bzw. eine einheitliche Schnittstelle zum NM-System bieten.

  – Integration auf der Ebene der NM-Plattform; damit zugleich Integration in die zugehörige Benutzeroberfläche (z.B. Verwendung eines Accounting Management Tool, das unter HP OpenView genutzt werden kann).

- **Berücksichtigung des QoS**
  Der Quality of Service, QoS, darf zum einen durch die Nutzungserfassung und die Sammlung der erfaßten Daten nicht oder nur unwesentlich beeinträchtigt werden (z.B. darf sich der Durchsatz eines Routers durch die Erfassung von Accounting-Daten in diesem Router nicht (wesentlich) verschlechtern), zum anderen sollte ein verschlechterter Quality of Service (nicht durch das Accounting verursacht!) bei der späteren Abrechnung berücksichtigt werden (wenn z.B. eine Standleitung gestört ist und die Kommunikation über eine teure Wählleitung aufrecht erhalten wird). Die Berücksichtigung des QoS wird in dieser Arbeit nicht weiter betrachtet.

[30]Netz Management

5.3 Ein OSI-konformes Objekt-Modell

Am Lehrstuhl wird bzw. wurde unter Mitwirkung des Autors ein OSI-konformes Objekt-Modell für Accounting entwickelt. Dieses bildet den konzeptionellen Rahmen für die weiteren Betrachtungen. Es ist konform mit dem in Kapitel 3 vorgestellten OSI Accounting Modell. Die praxisorientierten Aspekte sind im wesentlichen konform mit dem Internet Accounting Modell. Das liegt nahe, da das Internet mit der Protokollfamile TCP/IP ja Gegenstand der Betrachtung ist.

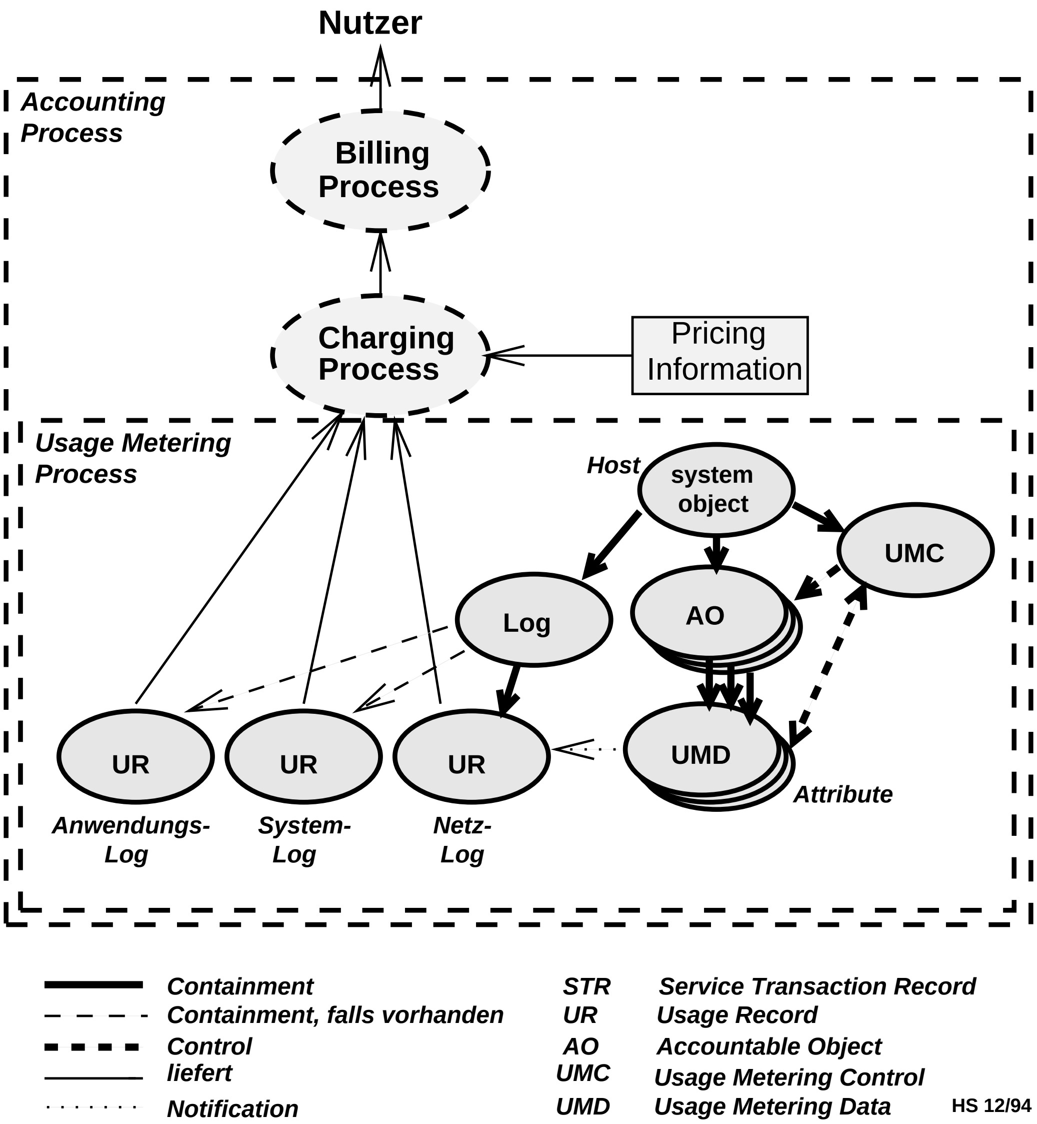

Abbildung 11: Ein OSI-konformes Objekt-Modell mit Beispiel-Instanzen

5.3.1 Accountable Objects

Im betrachteten Szenario, ein auf TCP/IP basierendes Datennetz, sind die **Accountable Objects, AOs**, die Protokollmaschinen der OSI-Schichten 3 und 4. (Accountable Objects sind Managed Objects, MOs, die Ressourcen repräsentieren, deren Nutzung einem Nutzer zugeschrieben werden soll; ein Managed Object ist die Managementinformation[31], die aus Managementsicht ein zu verwaltendes, zu überwachendes oder zu steuerndes Betriebsmittel (= Ressource) beschreibt).

Die Nutzung der Ressource Protokollmaschine kann durch Zählen und Analysieren der gesendeten bzw. empfangenen PDUs[32] ermittelt werden. Die Zuordnung zum Nutzer kann in der Regel auf der betrachteten Ebene (OSI-Schichten 3 und 4) *nicht* erfolgen. (Ausnahme: Es besteht eine *feste* Zuordnung zwischen IP-Adresse und Nutzer (z.B. bei einem PC, der ausschließlich von einem einzigen Benutzer genutzt wird) bzw. zwischen Port-Nummer und Nutzer, die in den seltensten Fällen vorliegt).

Es werden somit drei Arten von Accountable Objects betrachtet:

- IP-Protokollmaschine
- UDP-Protokollmaschine
- TCP-Protokollmaschine

Die Accountable Objects enthalten die für sie spezifischen Usage Metering Data (Containment).

5.3.2 Usage Metering Data

Usage Metering Data sind Daten, die die Nutzung einer Ressource repräsentieren und mit denen **Usage Records** abgeleitet werden können. Bei der Betrachtung der Protokollmaschinen der OSI-Schichten 3 und 4 entsprechen diese Daten **Attributen**, die **Flows** spezifizieren. Jedem Accountable Object Protokollmaschine läßt sich ein Flow Object zuordnen. So wird z.B. durch die Usage Metering Data einer IP-Protokollmaschine ein IP-Flow spezifiziert. Die Usage Metering Data und damit das Flow Object ist in *seinem* Accountable Object enthalten (Containment).

[31] siehe Kapitel 3, OSI Network Management: das Informationsmodell
[32] Protocol Data Units

Die Usage Metering Data bestehen aus *allen* Attributen, die eine bestimmte Klasse von Flows (z.B. IP-Flows) *maximal* spezifizieren. Dabei müssen *nicht alle* Attribute zur Spezifikation von Flows herangezogen werden.

5.3.3 Flows

Ein **Meter** ist ein *Prozeß*, der einen Strom von Paketen (*Stream of Packets*) auf einem Kommunikationsmedium oder zwischen zwei Kommunikationsmedien untersucht. Der Meter speichert die Anzahl der Pakete, die zu einem **Flow** (Datenstrom) zwischen zwei kommunizierenden Einheiten (*Communicating Entities*), wie Prozesse, Hosts oder Domains, gehören. Die Zuweisung von Paketen zu Flows kann durch die Ausführung von **Regeln** geschehen.

Soweit die Definition von Flows im RFC 1272, Internet Accounting: Background. Eine Betrachtung der auf diesem RFC[33] basierenden **Accounting MIB** ergibt, daß ein Flow aus den **PDUs** besteht, die dieselbe **Quell-** und **Zieladresse** besitzen. Zudem ist ein Flow durch seine **Anfangs-** und **Endzeit** bestimmt: Sobald eine PDU mit einer Quell- und Zieladresse auftritt, die noch keinem Flow zugeordnet werden kann, wird ein neuer Flow-Eintrag[34] erzeugt. Dabei wird die Anfangszeit gesetzt. Die Endzeit ist die aktuelle (System-)Zeit, falls der Flow als noch nicht beendet gilt. Ein Flow *kann* sinnvollerweise als beendet gelten, wenn während eines bestimmten Zeitintervalls (evtl. zu ermitteln über `InactivityTimeout`; in der Accounting MIB wird hierfür ein Default-Wert von 10 Minuten vorgeschlagen) keine entsprechende PDU mehr auftritt. Die Quell- und Zieladresse kann auf den OSI-Schichten 2 bis 5 (bzw. 7) spezifiziert sein, muß aber mindestens auf einer Schicht angegeben sein.

Angewendet auf das in dieser Arbeit betrachtete Szenario, kann der Begriff Flow somit wie folgt definiert werden:

- Im Rahmen von TCP-Verbindungen:
  Alle innerhalb einer TCP-Verbindung übertragenen Segmente (PDU = Segment). Der Anfang eines Flows ist der Moment des Zustandekommens einer TCP-Verbindung, das Ende ist der Abbruch der Verbindung. Dementsprechend werden Anfangs- und Endzeit gesetzt. Quell- und Zieladresse sind die IP-Quell- und IP-Ziel-Adresse zusammen mit der jeweiligen TCP-Quell- und TCP-Ziel-Portnummer. Die kommunizierenden Einheiten sind Prozesse, die über eine TCP-Verbindung miteinander kommunizieren. Da auf der betrachteten Ebene (OSI-Schichten 3 und 4) ohne weiteres keine

[33] Request For Comments

[34] Ein Flow-Eintrag repräsentiert einen Flow; er ist Teil einer Flow-Tabelle, in Form eines Datensatzes einer rel. Datenbank bzw. in Form einer Text-Zeile.

TCP-Verbindungen identifiziert werden können, muß der TCP-Verkehr wie UDP-Verkehr behandelt werden. (Ausnahme: Zusatzinformation aus dem UNIX-Kernel-Memory, Zustände der zugehörigen Sockets bzw. eventuell über TCP-Flags[35]

- Im Zusammenhang mit UDP:
  UDP bietet nur einen einfachen, verbindungslosen Transportdienst. Daher ist der Anfang eines Flows durch das erstmalige Auftreten eines UDP-Datagramms (PDU = Datagramm) mit einer bestimmten Quell- und Zieladresse[36] definiert. Analog zur TCP-Verbindung besteht hier eine Adresse aus IP-Adresse und UDP-Portnummer. Alle weiteren UDP-Datagramme mit dieser Quell- und Zieladresse werden diesem neu erzeugten Flow-Eintrag zugeordnet (d.h. der zugehörige Zähler wird entsprechend erhöht). Tritt während eines längeren Zeitintervalls kein derartiges UDP-Datagramm mehr auf, so gilt der Flow als beendet.

- Im Zusammenhang mit IP:
  IP-Datagramme können analog zu den UDP-Datagrammen behandelt werden. Eine Adresse besteht hier jedoch nur aus der IP-Adresse selbst. Auf dieser Ebene (OSI-Schicht 3) kann nur der Verkehr zwischen zwei Netzwerkinterfaces (also zwischen zwei Hosts) erfaßt werden. Dies kann jedoch in vielen Fällen bereits ausreichend sein.

Accountable Events[37] sind in diesem Zusammenhang die Anfangs- und Endzeit eines Flows. Weitere Accountable Events treten bei TCP-Verbindungen während des Verbindungsauf- und abbaus auf.

Die Zuordnung von Flows zu Benutzern und zu Diensten (Prozessen) kann auf der betrachteten Ebene[38] nicht stattfinden. Sie muß unter Zuhilfenahme der darüberliegenden Ebenen[39] oder durch Information aus den beteiligten Hosts (aus dem Kernel-Memory) erfolgen.

5.3.4 Attribute und Usage Records

An dieser Stelle sollen alle Attribute vorgestellt werden, mit denen ein IP-, ein UDP- und ein TCP-Flow spezifiziert werden kann.

[35]siehe weiter unten unter *Attribute zur Spezifikation von TCP-Flows*
[36]Es existiert noch kein entsprechender Flow-Eintrag
[37]siehe: OSI Accounting Modell
[38]TCP/IP-Transportebene = OSI-Schicht 4
[39]OSI-Schichten 5 bis 7

Attribute zur Spezifikation von IP-Flows:

- `IP.SourceAddress`:
  Die 32 Bit lange IP-Quelladresse, meist im Format X.X.X.X dargestellt. (X repräsentiert dabei eine Integer-Zahl zwischen 0 und 255).

- `IP.DestinationAddress`:
  Die 32 Bit lange IP-Zieladresse.

- `IP.Protocol`:
  Identifiziert das übergeordnete Transport Layer Protocol. Im betrachteten Szenario sind dies UDP und TCP; dabei hat UDP den Wert 17 und TCP den Wert 6.

- `IP.TotalLength`:
  Gesamtlänge des Datagramms. Wenn ein Datagramm fragmentiert werden mußte, so gibt dieser Wert die Gesamtlänge eines Fragments an.

- `IP.HeaderLength`:
  Länge des IP-Headers; diese ändert sich, wenn Optionen verwendet werden. Normalerweise ist der IP-Header (ohne Optionen) 20 Bytes lang (der entsprechende Wert hierbei ist 5)
  Die Länge des Datenfeldes ist damit durch `IP.TotalLength` – `IP.HeaderLength` zu ermitteln.

- `IP.TypeOfService`:
  — gibt die *Priorität* eines Datagramms an (3 Bits, damit acht Prioritäten möglich). Zusätzlich existieren vier Service-Bits:
  — Service: `LowDelay`, möglichst geringe Verzögerung (D-Bit).
  — Service: `HighThrouput`, möglichst hoher Durchsatz (T-Bit).
  — Service: `HighReliability`, möglichst wenig Datagramme sollen verworfen werden (R-Bit).
  — Service: `LowCost`, möglicht geringe Kosten (C-Bit).
  Die Service-Bits sind in erster Linie für die verwendeten Router bzw. Routing-Protokolle relevant. Viele Router unterstützen jedoch nicht alle bzw. keine der Services.

Ein IP-Flow wird *minimal* durch die Attribute `IP.SourceAddress` und `IP.DestinationAddress` spezifiziert.

Zusätzliche Attribute zur Spezifikation von UDP-Flows[40]:

- `UDP.SourcePort`:
  16 Bit lange Quell-Portnummer, Ingeter-Zahl zwischen 0 und 65535; normalerweise sind die Portnummern 0 bis 255 für TCP/IP Standardapplikationen *reserviert*[41]. Über eine Portnummer werden einzelne Prozesse *innerhalb* eines Host adressiert.

- `UDP.DestinationPort`:
  16 Bit lange Ziel-Portnummer, Integerzahl zwischen 0 und 65535.

- `UDP.DatagramLength`:
  Länge des UDP-Datagramms (maximal 65535 Bytes lang).

Zusätzliche Attribute zur Spezifikation von TCP-Flows:

- `TCP.SourcePort`:
  wie `UDP.SourcePort`.

- `TCP.DestinationPort`:
  wie `UDP.DestinationPort`.

- `TCP.Flags`:
  Zur Identifizierung von *TCP-Verbindungen* könnten die folgenden Flags dienen:
  — `SYN`, wird beim Verbindungsaufbau zur Synchronisation verwendet.
  — `FIN`, wird verwendet, um eine Verbindung abzubauen.
  Zu Accountingzwecken könnten auch noch diese Flags untersucht werden:
  — `URG`, d.h. das Urgent Pointer Field ist gültig. Der Urgent Pointer zeigt auf das letzte Byte der Urgent Data; diese Daten müssen vor allen anderen Daten verarbeitet werden (wird meist verwendet, um Programme zu unterbrechen).
  — `PSH`, d.h. die empfangende TCP-Schicht soll das Segment sofort an die Applikations-Schicht weiterreichen, also nicht zuerst in einen größeren Buffer schreiben.

Im TCP-Header gibt es *kein* Längen-Feld. Das liegt daran, daß TCP die zu versendenden Daten selbst segmentiert. (Bei UDP muß die Segmentierung des Datenstroms von den Protokollen der Applikations-Schicht durchgeführt werden; hier wird die Datagramm-Länge explizit angegeben).

Ein **Usage Record** entspricht nach ISO/IEC DIS 10164-10.2 einem Datensatz,

[40]Jeder UDP- bzw. TCP-Flow ist zugleich auch ein IP-Flow
[41]Das sind die bereits besprochenen Well-Known-Ports

der Information über die Ressourcennutzung durch *einen* bestimmten Nutzer innerhalb eines bestimmten Zeitraums enthält. Da auf der betrachteten Ebene *keine* Benutzeridentifikation möglich ist, ist es sinnvoll, daß das Usage Record *alle* Flows in Form einer **Flow-Tabelle** enthält. Der Aufbau eines Flow-Tabellen Eintrags ergibt sich direkt aus der Spezifikation eines Flows. Dabei ist zu beachten, daß alle Usage Records ein einheitliches Format haben sollten, da i.d.R. mehrere Usage Records zu einem Service Transaction Record zusammengefaßt werden. Die Usage Records sollten bzw. könnten zusätzlich zu den angeführten Attributen noch die folgende Information enthalten.

- `NumberOfPackets`:
  Anzahl der Datagramme bzw. Segmente eines Flows.

- `NumberOfBytes`:
  Anzahl der Bytes eines Flows. Berechnung über `NumberOfPackets`, `IP.TotalLength` und eventuell `IP.HeaderLength`.

- `GeographicalInformation`:
  Gibt den Ort der Erfassung eines Flows an.

- `StartTime`:
  Zeitpunkt des ersten Auftretens einer PDU eines Flows.

- `StopTime`:
  Zeitpunkt des letzen Auftretens einer PDU eines Flows.

- `SuspendTime`:
  Summe aller Zeitintervalle zwischen einem `SuspendMetering` und einem `ResumeMetering` Befehl (siehe weiter unten).

Ein Usage Record wird durch **Notifications** erzeugt bzw. verändert. Das Usage Record ist im Log Object enthalten (Containment). Das Log Object ist wiederum im System Object enthalten (Containment). Das System Object soll im betrachteten Szenario (zunächst) einen Host repräsentieren.

5.3.5 Usage Metering Control

Die **Usage Metering Control** steuert zum einen das Usage Metering, also die Erfassung des Netzverkehrs, zum anderen spezifiziert sie, was überhaupt aufgezeichnet werden soll. In diesem Abschnitt werden daher Befehle bzw. Parameter zur Steuerung des Usage Meterings vorgeschlagen.

Befehle zur Steuerung des Usage Meterings:

- `StartMetering`:
  Erstmaliger Start der Erfassung.

- `StopMetering`:
  Endgültiges Ende der Erfassung.

- `SuspendMetering`:
  Verübergehendes Aussetzen der Erfassung.

- `ResumeMetering`:
  Wiederaufnahme der Erfassung.

Parameter zur Steuerung des Usage Meterings:

- `SamplingInterval`:
  Bei einem Wert von 1 wird jedes Frame erfaßt und analysiert, bei einem Wert von n, jedes n-te Frame.

- `SamplingRandomOn`:
  Bei TRUE *und* bei `SamplingInterval` größer oder gleich m, wobei m viel größer als 1, wird ein Zufalls-Algorithmus aktiviert, wobei dann nicht mehr genau jedes n-te Frame erfaßt wird, sondern *ungefähr* jedes n-te[42].

- `NetFilter`:
  Filter zum Herausfiltern der Daten eines bestimmten Netzes (über die netid, die Bestandteil einer IP-Adresse ist).

- `SubNetFilter`:
  Filter zum Herausfiltern der Daten eines bestimmten Subnetzes (Subnetz-Adressierung).

- `HostFilter`:
  Filter zum Herausfiltern von bestimmten Host-Adressen.

- `PortFilter`:
  Filter zum Herausfiltern von bestimmten Port-Adressen.

- `NumberOfFlows`:
  Anzahl der Flows, die in der Flow-Tabelle gespeichert werden können. Warnung vor einem Speicherüberlauf durch Notifications.

[42] Siehe mehr dazu unter Statistische Methoden

5.3.6 Usage Metering und Charging

Die durch das Usage Metering erfaßten Nutzungsdaten werden in den Usage Records mitgeloggt. Der Charging Process sammelt (Collection) die Usage Records, die zu einer *bestimmten Dienstinanspruchnahme* gehören, ein. Im betrachteten Szenario holt sich der Charging Process die Usage Data, die zu einer bestimmten Dienstinanspruchnahme, wie einer FTP-Session, gehören, und verknüpft sie mit Hilfe der Abbildungs-Zusatzinformation: Verknüpfung über Prozeß-ID bzw. Sockets.

Die Usage Data kommen dabei i.d.R. aus *drei verschiedenen Arten von Logs*: einer **Anwendungs-Log**, einer **System-Log** und einer **Netz-Log**. Das Ergebnis wird in einer weiteren Log in Form von Service Transaction Records, STR, gespeichert und kann i.d.R. bereits mit *Kosteninformation* versehen werden. Ein Service Transaction Record entspricht damit der Beschreibung eines **Dienstes** aus Accounting-Sichtweise. Die Service Transaction Records sind die Grundlage für den Billing Process, der / durch den dem einzelnen Nutzer eine Rechnung stellt / gestellt wird.

5.4 Koppelelemente und / oder Probes

In diesem Abschnitt wird ein Überblick über die verschiedenen Meßpunkte (Erfassungspunkte) in einem Datennetz gegeben. In Kapitel 4 wurde das *Monitoring* bereits einführend vorgestellt. Nun sollen die *Monitoring Agents*, die Daten aus dem Netz zur Verfügung stellen, näher betrachtet werden. Dabei sollen die Vor- und Nachteile der einzelnen Arten von Meßpunkten diskutiert werden.

Ein Datennetz besteht im wesentlichen aus Segmenten und aus Koppelelementen, mit denen die einzelnen Segmente miteinander verbunden werden. Hier soll der Begriff Segment wie folgt definiert werden:

Ein **Segment** umfaßt alle Hosts, die an einem Bus (z.B. Ethernet nach IEEE 802.3 oder Token Bus nach IEEE 802.4) oder an einem Ring (z.B. Token Ring nach IEEE 802.5 oder FDDI Ring) *direkt*, d.h. ohne Einsatz von Koppelelementen, angeschlossen sind. In dieser Arbeit werden mit Ethernet nach IEEE 802.3 nur Bus-Segmente betrachtet; ein Fan-Out samt angeschlossener Hosts ist dabei Teil eines Bus-Segments. Ein Ethernet-Segment wird somit durch Koppelelemente (Hubs, Bridges, Router oder Gateways) und durch zwei Abschlußwiderstände begrenzt.

Der Datenverkehr, der innerhalb eines Segments auftritt, soll unterschieden werden in:

- **Segmentinternen Datenverkehr**; das ist der Datenverkehr, der dadurch entsteht, daß Hosts *innerhalb* eines Segments miteinander kommunizieren.

- **Segmentübergreifenden Datenverkehr**; das ist der restliche Datenverkehr: Datenverkehr, der ein Segment über ein Koppelelement betritt und / oder verläßt.

Die zu betrachtenden **Koppelelemente** sind Brigdes, Router und Gateways. Bridges filtern den Datenverkehr auf der Ebene der Hardware-Adressen. Dadurch werden die angeschlossenen Segmente deutlich entlastet, da der jeweils segmentinterne Datenverkehr im jeweiligen Segment bleibt und nicht andere Segmente belastet. Sehr leistungsfähige Bridges, die ein Backbone ersetzen und i.d.R. mehr als zwei Interfaces besitzen, werden auch als Hub Bridges (Hubs) bezeichnet. Router haben ebenfalls diese Filter-Funktionalität, sind jedoch zudem für die Wegewahl zuständig. Sie operieren auf der Ebene der Internet- Adressen. Gateways operieren im betrachteten Szenario ebenfalls auf der Internet- Schicht, führen jedoch zudem eine Protokollumsetzung (z.B. von Ethernet nach FDDI) durch.

Man hat somit zwei Orte für die Erfassung von Netzdaten zur Verfügung: die Koppelelemente und die Segmente. Innerhalb der Segmente kann der Datenverkehr mit Hilfe von Probes erfaßt werden. Der Begriff Probe soll dabei wie folgt definiert werden:

Eine **Probe** ist Teil eines Monitoring Agents, der innerhalb eines Segments als Hardware, mit eigener Hardware-Adresse, oder als Software, auf einem Host des Segments, den passierenden Datenverkehr erfaßt. Dabei kann sowohl der segmentinterne als auch der segmentübergreifende Datenverkehr erfaßt werden. Verfügt die Probe zudem über eine standardisierte Netzmanagement-Schnittstelle (wie SNMP oder CMIP), so ist sie ein Monitoring Agent (und nicht nur Teil eines Monitoring Agents).

Um den gesamten, erfaßbaren Datenverkehr in einem Datennetz mitzuloggen, muß pro Segment mindestens eine Probe zur Verfügung stehen, da die betrachteten Koppelemente den Datenverkehr filtern. Um das Accounting möglichst zuverlässig zu machen, sollten weitere, redundante Probes eingesetzt werden, damit bei einem Ausfall einer Probe das Accounting möglichst genau bleibt.
Die redundanten Probes sollten dabei am besten gleichzeitig den Datenverkehr miterfassen[43]. Vorteil von Probes ist, daß sie i.a. die Performanz des Datennetzes nicht vermindern, da sie das verwendete Medium nur von aussen abhören. Ein Nachteil kann die große Anzahl von Probes sein, die eingesetzt werden muß (z.B. wenn an einer Hub Bridge 10 Segmente angeschlossen sind, sollten auch 10, besser jedoch 20 Probes eingesetzt werden).

Die Alternative zu der Erfassung mittels Probes ist die Erfassung in den Koppelelementen. Nachteil dabei ist, daß die Erfassung, (Vor-)Analyse, (eventuell) Speicherung und Weiterleitung der erfaßten Daten die Leistungsfähigkeit des Koppelelemets verschlechtert; die Performanz des Datennetzes könnte somit herabgesetzt werden. Ein Vorteil ist die geringere Anzahl der vorhandenen Meßpunkte (im Beispiel der Hub Bridge, an der 10 Segmente angeschlossen sind, hat man nur einen Meßpunkt, nämlich die Hub Bridge selbst; wenn die Hub Bridge ganz ausfällt, können natürlich keine Verkehrsdaten erfaßt werden; die Probes in den Segmenten, die über die Hub Bridge gekoppelt sind, könnten lediglich segmentinternen Datenverkehr erfassen).

In einigen Fällen kann es sinnvoll sein, die Erfassung in den Koppelelementen mit der Erfassung mittels Probes zu kombinieren, sofern dies von den eingesetzten Werkzeugen unterstützt wird. So kann z.B. der Datenverkehr in LAN-Segmenten in den Koppelelementen erfaßt werden und der Datenverkehr am LAN/WAN-

[43]Bis der Ausfall einer Probe angenommen werden kann, vergehen i.d.R. mehrere Minuten; diese Lücke bei der Erfassung kann bei einer sehr genauen Nutzungserfassung meist nicht toleriert werden

Übergang nicht *im* Gateway, sondern mittels einer Probe *vor* dem Gateway, um das Gateway zu entlasten und um so einen maximalen Durchsatz zu ermöglichen.

Ein Meßpunkt muß bzw. kann folgende Aufgaben erledigen:

- Er muß das angeschlossene Medium abhören und die spezifizierten Frames (alle oder (ungefähr) jedes n-te) vom Datennetz in einen Buffer kopieren bzw. sofort (z.B. in einem Trap) versenden. (Minimalanforderung)

- Er kann bereits die Protokoll-Header (vor-)analysieren und die Ergebnisse der Analyse (zwischen-)speichern.

- Er sollte zudem eine standardisierte NM-Schnittstelle besitzen, über die ein Manager mit ihm kommunizieren kann. Es muß aber zumindest eine Möglichkeit bestehen, über die die Daten aus dem Buffer bzw. die bereits (vor-)analysierten Daten abgefragt (geholt) werden können.

So kann ein Meßpunkt (eine Probe oder ein Koppelelement) über eine mehr oder minder reichhaltige Funktionalität verfügen: Im einfachsten Fall versendet ein Meßpunkt die spezifizierten Protokoll-Header in einer Notification (z.B. in einem Trap), im aufwendigsten Fall analysiert der Meßpunkt bereits die Protokoll-Header, speichert die Ergebnisse in komprimierter Form (in einer Datenbank) ab (Schutz vor Datenverlust!) und berichtet seine Ergebnisse sowie besondere Ereignisse (Events) über eine standardisierte NM-Schnittstelle an seine(n) Manager.

Der Ort und die Art der Meßpunkte wird durch die verwendete Methode bestimmt. In den beiden folgenden Abschnitten werden dazu verschiedene Arten von Methoden diskutiert.

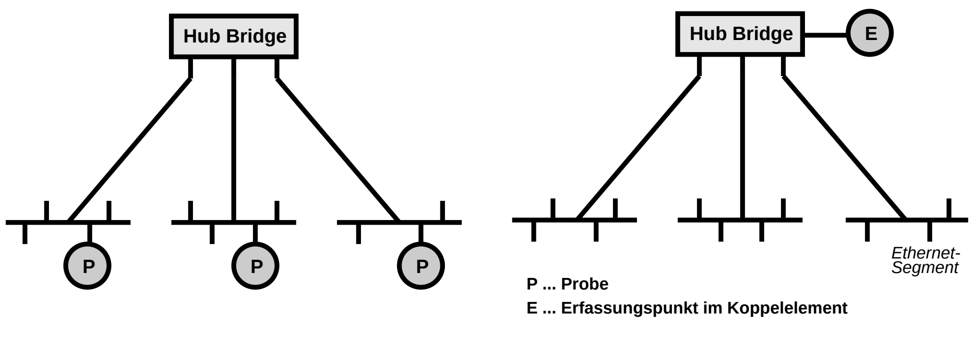

Abbildung 12: Erfassung mittels Probes bzw. im Koppelelement

5.5 Exakte Methoden

In diesem Abschnitt sollen sog. **exakte Methoden** näher erklärt und definiert werden. Die Alternative, die statistischen Methoden, werden im nächsten Abschnitt besprochen.

Eine **exakte Methode** zur Erfassung von Netzverkehr zeichnet sich dadurch aus, daß in den Meßpunkten (Erfassungspunkten) **jedes** erfaßbare Frame[44] vom Netz kopiert und analysiert und / oder weitergeleitet wird. Im vorgestellten OSI-konformen Objekt-Modell ist dabei folgender Parameter zur Steuerung des Usage Meterings wie folgt gesetzt:

$$\texttt{SamplingInterval} = \texttt{1}$$

Der Wert von `SamplingRandomOn` ist in diesem Fall irrelevant und wird nicht ausgewertet. Exakte Methoden sollen untergliedert werden in

- **permanent** exakte Methoden und
- **zeitweise** exakte Methoden.

Bei permanent exakten Methoden werden über einen längeren Zeitraum (z.B. einen Monat) alle erfaßbaren Frames betrachtet; bei zeitweise exakten Methoden ist dieser Zeitraum wesentlich kürzer (z.B. eine Stunde), die Messungen können jedoch über einen längeren Zeitraum periodisch wiederholt werden. Z.B. kann jeden Tag von 9 bis 10 Uhr, von 12 bis 13 Uhr und von 15 bis 16 Uhr ein sog. **Snapshot** (eine Momentaufnahme des aktuellen Datenverkehrs) aufgezeichnet werden. Die Aufzeichnung wird dabei durch `SuspendMetering` und `ResumeMetering`-Befehle gesteuert. Die `SuspendTime`, das ist die Zeit, in der kein Datenverkehr aufgezeichnet wird, kann somit relativ groß werden (Bei einer permanent exakten Methode sollte die `SuspendTime` idealerweise den Wert 0 besitzen). Diese Methoden eignen sich jedoch nur sehr bedingt für Accounting-Zwecke und gehören zudem eigentlich eher zu den statistischen Methoden. Da aber in einem bestimmten Zeitraum (auch wenn es sich nur um eine Stunde handelt) jedes Frame erfaßt wird, handelt es sich nach obiger Definition dennoch um eine exakte Methode.

Sinnvoll ist der Einsatz von zeitweise exakten Methoden z.B. wenn man **Nutzerklassen** einführen will. Jeder Nutzer einer bestimmten Nutzerklasse zahlt z.B. monatlich einen festen Betrag für die Nutzung von DV-Dienstleistungen. Um eine Klassifizierung machen zu können, muß ein **Nutzerprofil** erstellt werden. Für den Bereich Netz kann dies mit dem zuvor angesprochenen Snapshot-Verfahren

[44]Bei ausreichender Verarbeitungs- und Speicherkapazität sollte wirklich jedes Frame erfaßt werden können.

realisiert werden. Anhand der durchschnittlichen Nutzungsintensität kann dann eine Zuordnung Nutzer — Nutzerklasse getroffen werden. Zeitweise exakte Methoden eignen sich damit lediglich für eine *bedingt verursachergerechte* Nutzungserfassung.

Bei exakten Methoden kopieren die Erfassungspunkte jedes erfaßbare Frame eines spezifizierten Protokolls (bzw. aller spezifizierten Protokolle) zur Analyse vom Netz. Dabei ist die Plazierung der Erfassungspunkte von entscheidender Bedeutung. Sie sollten so plaziert werden, damit auch der gesamte abrechnungsrelevante Datenverkehr des abzurechnenden (Teil-)Netzes erfaßt werden kann. Nur so ist eine exakte Methode wirklich eine exakte Methode und kann ein Maximum an Genauigkeit und Vollständigkeit bieten. Im betrachteten Szenario werden also alle Ethernet-Frames durch Probes (Hardware oder Software) vom Netz kopiert und ihren Flows zugeordnet. Somit können alle Flows vollständig erfaßt werden.

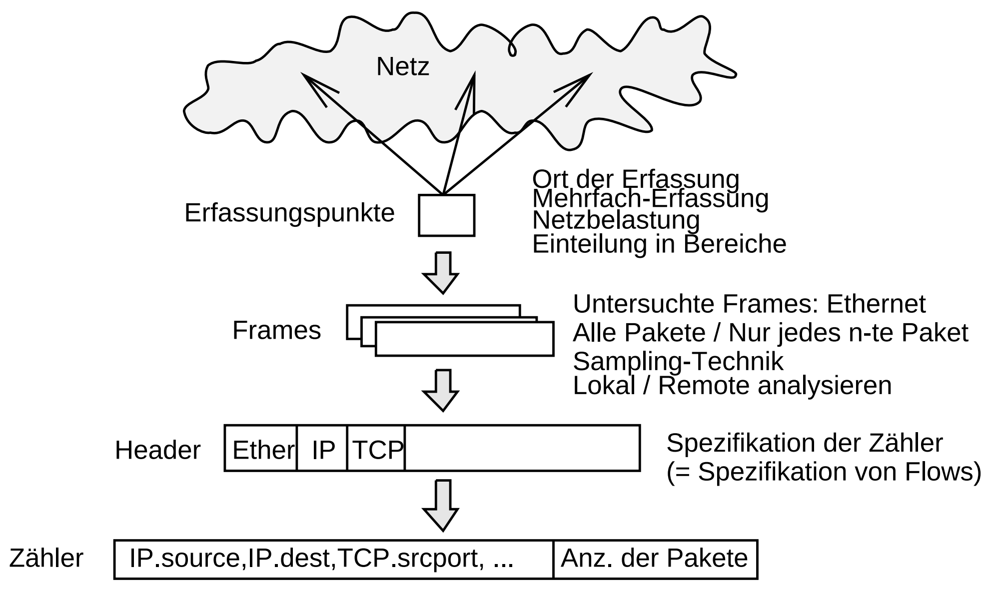

Abbildung 13: Erfassung von Netzverkehr

Permanent exakte Methoden sind eigentlich das Mittel der Wahl für ein verursachergerechtes, detailliertes Abrechnungsmanagement. Sie benötigen in der Regel jedoch eine sehr hohe Verarbeitungskapazität[45]. Daher kann man – gerade in Netzen mit einem sehr hohem Aufkommen an Datenverkehr – die Verwendung von statistischen Methoden in Erwägung ziehen.

[45]Hohe CPU-Leistung, großer Hauptspeicher, geringe Plattenzugriffszeit, ...

5.6 Statistische Methoden

Bei statistischen Methoden wird in den Erfassungspunkten *nicht* jedes Frame erfaßt, sondern nur jedes n-te bzw. ungefähr jedes n-te Frame. Das macht statistische Methoden weniger abhängig von einer sehr hohen Verarbeitungskapazität jedoch auch ungenauer. Inwieweit diese Ungenauigkeit für das Accounting akzeptiert werden kann, wird im Kapitel 10, *Bewertung von HP EASE*, genauer untersucht.

Zuerst sollen die statitischen Methoden unterschieden werden in

- **Streng statistische Methoden** und
- **Stochastische Methoden**

Eine **streng statistische Methode** zur Erfassung von Netzverkehr ist eine Methode, bei der in den Erfassungspunkten **genau** jedes n-te Frame vom Netz kopiert, der Frame-Header eventuell (vor-)analysiert und weitergeleitet wird. Die entsprechenden Parameter des OSI-konformen Objekt-Modells haben somit die folgenden Werte:

$$\texttt{SamplingInterval} = \texttt{n}$$
$$\texttt{SamplingRandomOn} = \texttt{FALSE}$$

Bei einer **stochastischen Methode** hingegen wird nicht genau jedes n-te Frame betrachtet, sondern **ungefähr** jedes n-te Frame. Dadurch kann das Ergebnis der statistischen Berechnung nicht durch zyklische Prozesse verfälscht werden. Es wird dazu ein Zufallsalgorithmus verwendet. Meist werden dazu Werte um den ganauen Wert `n` berechnet, deren Durchschnitt genau `n` ist. Diese Werte dienen als **SkipCounter**, d.h. der aktuelle Wert wird jedes Mal, wenn am Netz-Interface ein Frame erkannt wird, um eins vermindert. Hat der aktuelle SkipCounter den Wert 0, so wird das gerade erkannte Frame in einen Buffer kopiert. Dieses Verfahren wird für alle berechneten SkipCounter-Werte wiederholt. Nachdem der letzte SkipCounter-Wert benutzt wurde, werden wieder, wie oben beschrieben, neue SkipCounter-Werte berechnet. Die entsprechenden Parameter des OSI-konformen Objekt-Modells besitzen bei einer stochastischen Methode somit die Werte:

$$\texttt{SamplingInterval} = \texttt{n}$$
$$\texttt{SamplingRandomOn} = \texttt{TRUE}$$

Wie bei den exakten Methoden kann sowohl bei den streng statistischen Methoden als auch bei den stochastischen Methoden eine Unterscheidung in **permanent** streng statistische bzw. permanent stochastische und in **zeitweise** streng ststistische bzw. zeitweise stochastische Methoden gemacht werden. Dabei sind die beiden letzteren noch ungenauer und weniger für Accounting-Zwecke geeignet als die zeitweise exakten Methoden. Die Genauigkeit hängt jedoch in diesem Zusammenhang eng mit dem Betrag der `SuspendTime` zusammen. Je kürzer die `SuspendTime`, desto genauer die Methode, wobei hier eine hinreichende Genauigkeit von statistischen Methoden vorausgestzt werden soll.

Die Genauigkeit einer statistischen Methode ist von folgenden Größen abhängig:

- Vom **Erfassungszeitraum** und
- vom **Sampling-Intervall**.

Je länger der Erfassungszeitraum gewählt wird und je kürzer das Sampling-Intervall gesetzt wird, d.h. je kleiner der Wert `SamplingInterval` ist, desto genauer ist das Ergebnis der Nutzungserfassung. Die Genauigkeit ist *nicht* von der tatsächlichen Anzahl der Frames auf dem untersuchten Medium, sondern von der Anzahl der herausgegriffenen Frames (Samples) abhängig. Mit Hilfe von statistischen Methoden kann so der Netzverkehr in Netzen bzw. Segmenten mit sehr hohem Datenverkehrsaufkommen (z.B. Backbone) sehr genau zum Zwecke des Accountings erfaßt werden. Dabei kann eine hinreichend hohe Genauigkeit mit einer vergleichbar geringen Verarbeitungskapazität erreicht werden. Exakte Methoden würden in vergleichbaren Situationen leicht „in die Knie gehen“ bzw. zwangsweise zu statistischen Methoden entarten. Näheres zum Thema Sampling und Genauigkeit findet man im Abschnitt 7.2.2 unter *Theoretische Betrachtung der Genauigkeit*. Die sechs verschiedenen Arten von Methoden zur Erfassung von Netzverkehr sind in der folgenden Abbildung noch einmal veranschaulicht.

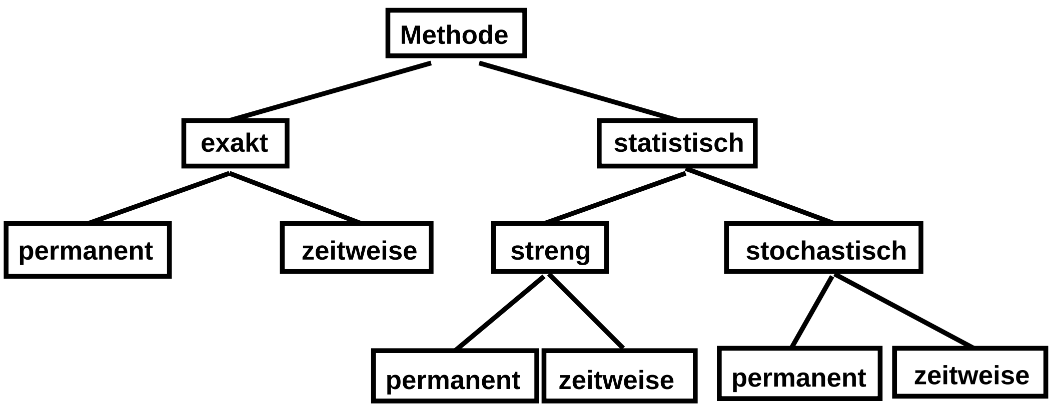

Abbildung 14: Arten von Methoden zur Erfassung von Netzverkehr

6 Modellierung des Charging Process

Im vorigen Kapitel, im Abschnitt 5.3, wurde ein OSI-konformes Objekt-Modell für die OSI Usage Metering Function für den Bereich Netz vorgestellt. Dieser Teil des Accounting Modells (das am Lehrstuhl entwickelt wird bzw. entwickelt wurde) befaßt sich im wesentlichen mit dem Aufbau und dem Zustandekommen von Usage Records. Bevor nun im weiteren (nach der Vorstellung der zur Verfügung stehenden Werkzeuge) die Aufstellung von Kriterien für die Bewertung von Werkzeugen bzw. die Bewertung von zur Verfügung stehenden Werkzeugen zur Erfassung von Netzverkehr folgt, soll noch kurz ein Modell für den Charging Process nach OSI vorgestellt werden. Dieses Modell, so wie auch das Teil-Modell für den Bereich Netz im Abschnitt 5.3, kommt im wesentlichen aus der Feder des Autors und ist grundsätzlich kritisch zu betrachten. Es bildet, zusammen mit dem Teil-Modell aus Abschnitt 5.3, den modellhaften Rahmen für die weiteren Ausführungen und soll zudem als Diskussionsgrundlage für weitergehende Betrachtungen (ausserhalb dieser Arbeit) dienen.

Der Charging Process sammelt Usage Records, die zu einer bestimmten Dienstinanspruchnahme (Service-Call) gehören und fügt sie zu Service Transaction Records zusammen. Der Charging Process heißt deshalb *Charging* Process, da die Service Transaction Records zusätzlich mit Pricing Information versehen werden.

Der Charging Process soll nun in drei Sub-Prozesse unterteilt werden:

- Integration Process
- Service Projection Process
- Pricing Process

Der **Integration Process** ist für die Sammlung von verteilt erfaßten, abrechnungsrelevanten Netzdaten zuständig; dabei kann ein flacher Ansatz[46] oder ein hierarchischer Ansatz[47] realsiert werden. Die Information, wo welche Daten erfaßt werden sollen, erhält er aus den **Integration Information Data**, die Teil der **Administrative Data**[48] sind. Die Integration Information Data können dabei auch über das Datennetz verteilt gehalten werden. In den Integration Information Data sind z.B. auch redundante Erfassungspunkte enhalten, die automatisch

[46]Verteilte Erfassung in den Erfassungspunkten; zentrale Sammlung in einem Server oder in einer NM-Station

[47]Durch den zusätzlichen Einsatz von verteilten Sammlungspunkten

[48]Die Administrative Data enhalten Information über Dienstabonenten, vertragliche Vereinbahrungen, usw.; sie enthalten aber auch neben den Integration Information Data die Service Information Data und die Pricing Information Data.

benutzt werden können, wenn reguläre Erfassungspunkte ausfallen.

Der **Service Projection Process** ist für die Zusammenfassung von Usage Records zu Service Transaction Records zuständig. Dabei stützt er sich auf den Integration Process. Die Information, welche Usage Records zu welchem Dienst gehören entnimmt der Service Projection Process den **Service Information Data**[49], die ebenfalls Teil der Administrative Data sind. Die Service Information Data sollten zentral gehalten werden.

Der **Pricing Process** schließlich fügt den durch den Service Projection Process erzeugten STRs noch die nötige Kosteninformation hinzu. Diese Kosteninformtion ist in den **Pricing Information Data** (Teil der Administrative Data; sollten ebenfalls zentral gehalten werden) enhalten.

Diese Modellierung des Charging Process ist in der folgenden Abbildung noch einmal grafisch veranschaulicht.

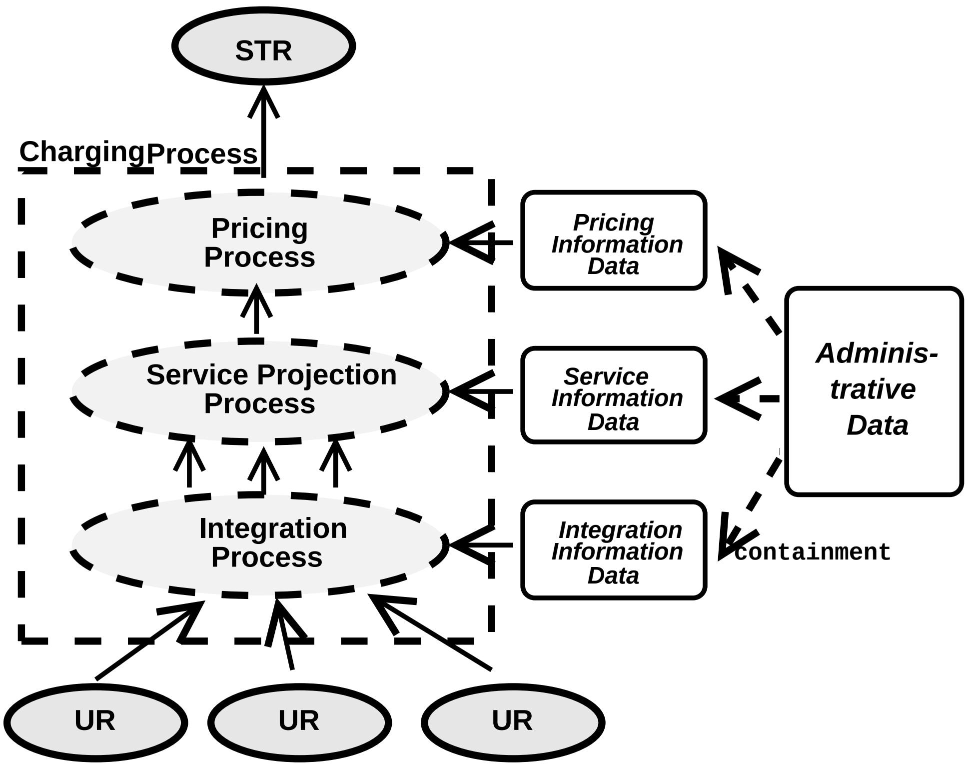

Abbildung 15: Modellierung des Charging Process

[49]Sie enthalten die Beschreibung von Diensten aus Accounting-Sichtweise

7 Werkzeuge zur Erfassung von Netzverkehr

In diesem Kapitel werden kurz die zur Verfügung stehenden Werkzeuge zur Erfassung von Netzverkehr für ein Abrechnungsmanagement in verteilten, heterogenen Umgebungen vorgestellt. Es handelt sich hierbei um das Netz-Satistik Paket NNStat (public domain) und um die HP[50]-Produkte HP EASE und HP NetMetrix.

7.1 NNStat

NNStat (Release 3.0, 1991), das Internet Statistics Collection Package von Robert T. Braden und Annette L. DeSchon vom USC / Information Sciences Institute in Marina del Rey, California, ist ein Werkzeug-Paket zur **verteilten** Erfassung von Internet Verkehrsdaten für die Erstellung von Netz-Statistiken.

Die Verkehrsdaten werden mit Hilfe des Programms **statspy** erfaßt. statspy läuft dazu pro zu erfassendem Segment auf mindestens einem Host des jeweiligen Segments und zeichnet die gewünschten Verkehrsdaten auf. Dazu analysiert es **jedes Frame**, das am lokalen Netz-Interface empfangen wird. Es können zur Zeit jedoch ausschließlich folgende Frame-Header analysiert werden:

- Network-Level: Ethernet
- Internet-Level: IP und ICMP
- Transport-Level: UDP und TCP

Da statspy grundsätzlich jedes Frame analysiert und erfassen kann, handelt es sich bei NNStat um die Realisierung einer **exakten Methode**. Die von statspy zu erfassenden Verkehrsdaten werden mit einer C-ähnlichen Konfigurationssprache spezifiziert. Beim Start von statspy wird dazu als Argument der File-Name des gewünschten Konfigurations-Files angegeben.

Mit Hilfe von **Filter Objects** können die gewünschten Ethernet-Frames, die bestimmte IP-Datagramme, UDP-Datagramme oder TCP-Segmente transportieren, herausgefiltert werden. Die Werte der relevanten Header-Felder dieser Frames werden in **Recorder Objects** zusammen mit einem Zähler für die Anzahl der herausgefilterten Frames (und evtl. zusätzlich einem Zähler für die in diesen Frames transportierten Bytes) im Hauptspeicher der statspy-Hosts lokal gespeichert.

[50] Hewlett Packard

Dazu ein kleines Beispiel:

```
attach{
        if TCP.dstport is setf(20, 21)
           record IP.srchost, IP.dsthost in ftp.hosts matrix-all;
      }
```

Mit dem Kommando `attach` wird die aktuelle statspy-Konfiguration um die Konfiguration in den geschweiften Klammern erweitert. So kann die Konfiguration von statspy *dynamisch* erweitert bzw. geändert werden (Das Ändern geschieht mit Hilfe des Kommandos `detach`). Die Anweisung in den geschweiften Klammern überprüft, ob im TCP-Header des gerade analysierten Frames im Header-Feld `TCP.dstport`[51] der Wert 20 oder 21 (FTP) steht. Dazu wird das Filter Object `setf` verwendet, das eine Menge (set) von Filter-Werten enthalten kann. Wenn nun im Header-Feld `TCP.dstport` der Wert 20 oder 21 steht, so werden die Zähler im Recorder Object `ftp.hosts`, das der Klasse `matrix-all` angehört, für das Paar `IP.srchost`, `IP.dsthost`[52] entsprechend erhöht: Der Frame-Zähler um den Wert 1 und der Byte-Zähler um die Anzahl der in diesem Frame übertragenen Bytes. Falls ein entsprechender Eintrag noch nicht existiert, so wird dieser erzeugt.

Mit Hilfe des Programms **collect**, können die Daten in den einzelnen Recorder Objects in den einzelnen statspy-Hosts abgefragt und in Log- Files abgespeichert werden. Dazu wird pro Recorder Object und pro statspy- Host ein eigenes Log-File auf dem collect-Host angelegt. collect kann als Hintergrundprozeß periodisch die erfaßten Daten aller statspy-Hosts abfragen und in den entsprechenden Log-Files speichern.

collect kann zusammen mit statspy auf einem einzigen Host laufen; es können ebenso mehrere collect-Prozesse gleichzeitig (auf mehreren Hosts) laufen — so kann u.U. eine hierarchische Sammlung der Verkehrsdaten realisiert werden. Ein Log-File für das obige Recorder Object `ftp.hosts` hat in etwa folgendes Aussehen:

```
Log created on Wed Dec 14 13:00:05 1994, for host 131.159.12.30.
    ...
OBJECT: ftp.hosts  Class=matrix-all  [Created: 12:50:00  12-14-94]
  ReadTime: 13:00:04  12-14-94, ClearTime: 12:50:00  12-14-94
  Total Count= 513 (+0 orphans)
  #bins = 1
```

[51]entspricht dem Attribut `TCP.DestinationPort` zur Spezifikation von TCP-Flows

[52]entspricht den Attributen `IP.SourceAddress` und `IP.DestinationAddress` zur Spezifikation von IP-Flows

```
[129.187.13.35 : 131.159.12.30] = 513    (100%) @-516sec
```

Durch die im Konfigurations-File angegebenen Header-Felder werden Flows spezifiziert. Im obigen Beispiel werden so FTP-Flows definiert, die eine Teilmenge aller TCP-Flows (nämlich alle TCP-Flows mit TCP-Ziel-Portnummer 20 und 21) darstellen.
Leider können in den zur Verfügung stehenden Recorder Objects nur einzelne Werte oder Paare von Werten gespeichert werden. Dies ist ausreichend für die Erfassung von IP-Flows, die ja explizit nur durch die Attribute `IP.SourceAddress` und `IP.DestinationAddress` spezifiziert werden. Möchte man hingegen zusätzlich die Portnummern explizit abspeichern, so würde man Recorder Objects benötigen, die 4-Tupel speichern können.
NNStat eignet sich insofern auch für ein Accounting auf dem Transport Level, falls Paare wie `IP.dsthost, IP.dstport` erfaßt werden und zugleich zusätzlich Information aus den Endsystemen zur Verfügung steht, *oder* wenn explizit nur einzelne Endsysteme (z.B. spezielle Server) betrachtet werden sollen.
Implizit, d.h. durch die Angabe der Portnummern über die Filter Objects, können natürlich spezielle UDP- bzw. TCP-Flows spezifiziert werden, nur muß so für jede einzelne Flow-Klasse ein eigenes Recorder Object und damit ein eigenes Log-File existieren, was i.a. nur für statistische Zwecke jedoch nicht für Accounting geeignet ist. Für ein Accounting auf der Internet-Ebene eignet sich NNStat hingegen theoretisch uneingeschränkt.

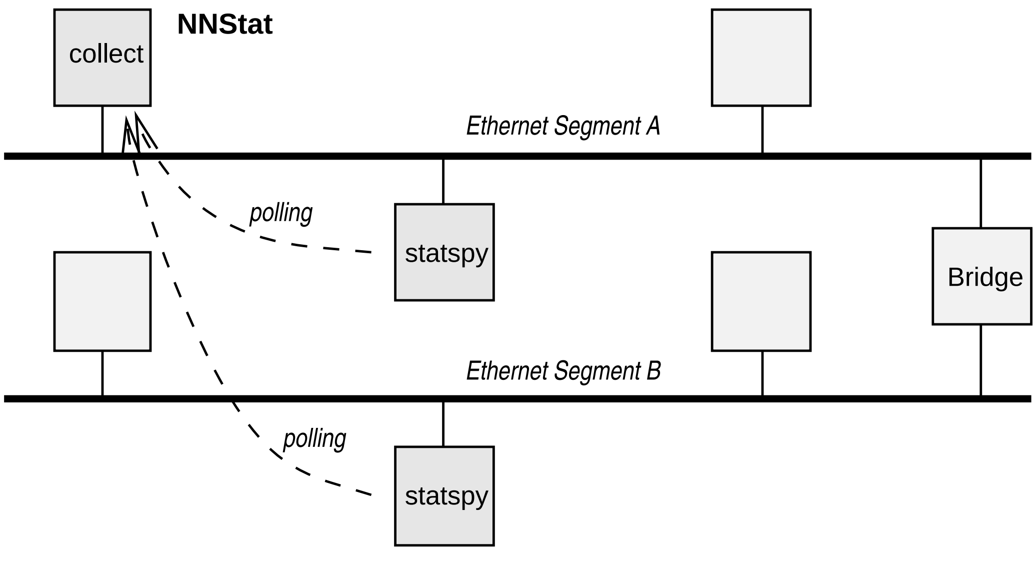

Abbildung 16: NNStat

7.2 HP EASE

7.2.1 Beschreibung des Programmpakets

HP EASE[53] (Release 1.0) ist ein Programmpaket der Firma Hewlett Packard zur **verteilten** Erfassung der Netznutzung. Zur Gewinnung der Verkehrsdaten wird in den Erfassungspunkten, den sog. **Sampling Devices**, eine **stochastische** Methode realisiert: Es wird *ungefähr* jedes n-te Frame erfaßt; falls genau jedes n-te Frame erfaßt werden würde, könnten zyklische Prozesse – wie zum Beispiel ein `ping` – das Ergebnis wesentlich verfälschen. Diese Erfassungsmethode nennt HP **Sampling**, wobei n der Wert der Samplingrate (vgl. `SamplingInterval` im OSI-konformen Objekt-Modell) ist. Der zugrundeliegende Zufallsalgorithmus wurde bereits im Abschnitt 5.6 kurz skizziert. HP EASE, das auf einer HP-UX oder einem SunOS System läuft, ist nach dem Client - Server Prinzip aufgebaut und besteht aus den folgenden Bestandteilen:

- Die Sampling Devices (Agenten):
  Das sind spezielle HP Traffic Probes bzw. bestimmte HP Hubs und Bridges. Sie setzen nach ca. n empfangenen Frames einen SNMP Trap ab, in den der Header des n-ten Frames eingepackt und an den HP EASE Server geschickt wird (Encapsulation). Dabei kann n die Werte 50, 100, 200, 400 und 800 annehmen.

- Der HP EASE Server (Manager):
  Hier werden die von den Sampling Devices gesendeten Frame-Header analysiert. Die aus dieser Analyse gewonnenen Daten (wie `IP.SourceAddress`, `IP.DestinationAddress`, `UDP.SourcePort` , ... , Anzahl der Frames, Anzahl der Bytes) werden in einer Datenbank in verschiedenen Tabellen (ErrorTable, TrafficTable, ...) stündlich abgespeichert. Diese Daten werden **History Information** genannt. Zusätzlich hält sich der Server Tabellen, in denen die aktuellen Verkehrdaten, die sog. **Real Time Information**, gespeichert wird.

- Die HP EASE Clients (Applications):
  Sie laufen auf einer NM-Satation, können von HP OpenView aus bedient werden, sind aber grundsätzlich unabhängig von HP OpenView. Es handelt sich hierbei um drei Werkzeuge:

  - Der Resource Manager:
    Mit diesem Tool kann man sich über die aktuellen Verkehrsdaten informieren. Durch Festsetzen von Schwellwerten definiert man Warnungen

[53] Embedded Advanced Sampling Environment

und Alarme. Hier kann man sich zusätzlich die am häufugsten benutzen Verbindungen ansehen. Ein Update erfolgt jede Minute.

- Der History Analyzer:
  Mit seiner Hilfe stellt man Anfragen (Queries) an die Datenbank des HP EASE Servers. Sämtliche Daten lassen sich grafisch darstellen. Die Historical Information wird stündlich abgespeichert.

- Der Traffic Expert:
  Dieses Tool besteht zum einem aus einem Report-Generator, mit dessen Hilfe tägliche Reports erstellt werden können. Zum anderen besteht er aus einem Workgroup-Analyzer, der Verkehrsflüsse grafisch veranschaulicht. Zudem können mit dem Workgroup-Analyzer Nutzungsprofile für einzelne Segmente aber auch für einzelne Benutzer analysiert werden.

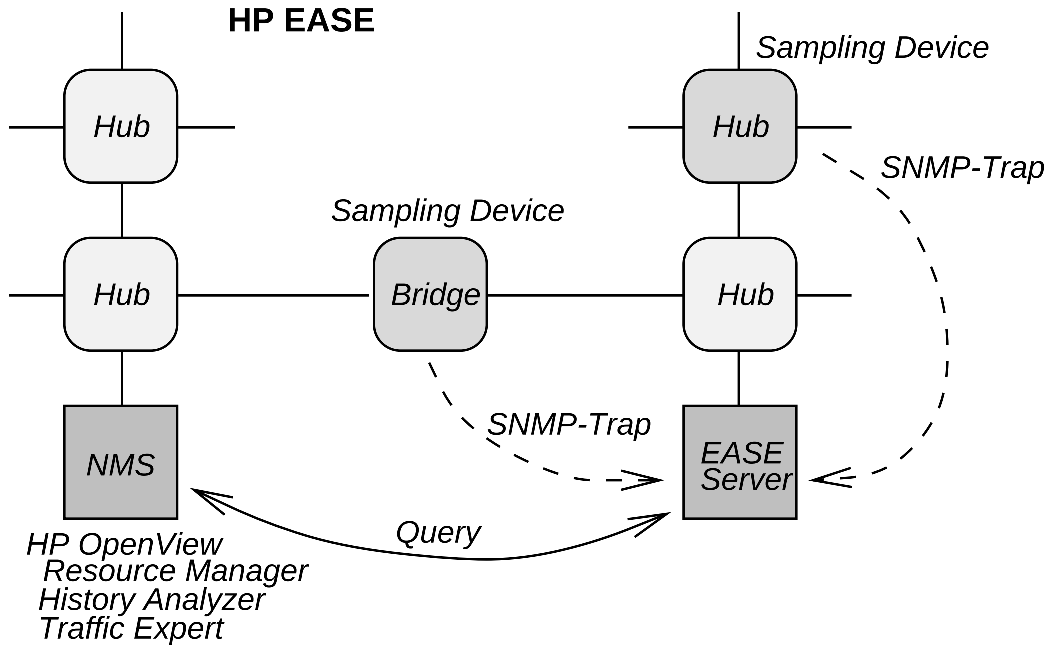

Abbildung 17: HP EASE

Für das Accounting ist aber vor allem die Möglichkeit interessant, unabhängig von den Clients auf die Datenbank des Servers zugreifen zu können. Dies geschieht mittels *Commandline Programs*, die zusammen mit dem History Analyzer geliefert werden. Damit lassen sich leicht Skripts für die Generierung von Accounting-Logs erstellen. Speziell kann das Kommando `easetraffic` verwendet werden, das zum Auslesen der Tabellen der History Information dient. Jede Tabelle[54] wird jede Stunde in einem eigenen File abgespeichert. Eine Traffic Table zum Beispiel hat dabei den folgenden Aufbau:

```
SourceAddress DestAddress SourcePort DestPort Bytes Frames
```

Das Kommando `easetraffic` liefert in etwa den folgenden Output (Adress-Typ ist IP, Protokoll-Typ ist IP, es findet keine Umwandung der IP-Adressen in Namen statt und es wird lediglich eine einzige Traffic Matrix (eine Stunde) abgefragt):

```
Entries(211) ... SrcAddr, SPort, DstAddr, DPort, Bytes, Frames
  131.159.4.5, 0, 131.159.12.35, 0, 4.15215e+06, 97294.1
  131.159.12.35, 0, 131.159.4.5, 0, 4.98034e+06, 91312.9
  ...
```

HP EASE erkennt und dekodiert die verschiedensten Protokolle wie IP, UDP, TCP, IPX und DEC. Vorerst wird jedoch nur Ethernet als darunterliegendes Protokoll unterstützt.

Durch das eingestzte Samplingverfahren kann der Anteil des Netzverkehrs, der durch die Erfassungs-Methode zusätzlich erzeugt wird (Netzoverhead), am gesamten Netzverkehr weniger als 0,25 Prozent[55]betragen. Pro Segment sollte mindestens ein Sampling Device eingestzt werden. Es können (in größeren Netzen *müssen*) gleichzeitig mehrere HP EASE Server zum Einsatz kommen. Jeder Server hat eine Verarbeitungskapazität von etwa 200 Samples/s. Über die Clients oder über die Commandline Programs können mehrere Server angesprochen werden. Durch das Sampling können leicht auch entfernte Netze mit erfaßt werden (beachte Samplingrate[56] !).

Im folgenden Abschnitt wird die Genauigkeit dieser Methode zunächst theoretisch untersucht. Die Ergebnisse der praktischen Untersuchung finden sich im Kapitel 10. Dort wird auch ein Vergleich zwischen theoretischen und praktischen Ergebnissen gemacht.

[54] In der vorliegenden Version sind es 15 Tabellen

[55] Nur bei einer Samplingrate von 800; bei einer Samplingrate von 50 beträgt er theoretisch 2%.

[56] Gefahr der Sättigung eines Low-Speed WAN-Links!

7.2.2 Theoretische Betrachtung der Genauigkeit

Obwohl durch ein Sampling Device immer nur ungefähr jedes n-te Frame herausgegriffen wird, so wird doch in einem bestimmten Zeitintervall im Mittel *genau* jedes n-te Frame analysiert. Das liegt daran, daß der zugrundeliegende Zufallsalgorithmus eine Anzahl von Werten um den exakten Wert n berechnet deren Durchschnittswert genau n beträgt.

Folglich sei X eine diskrete, *standardnormalverteilte* Zuvallsvariable mit dem Erwartungswert μ und der Standardabweichung σ. Da X standardnormalverteilt ist, gilt $\mu = 0$ und $\sigma = 1$.

Man nehme nun eine *Sicherheitswahrscheinlichkeit* von $p = 0{,}95$ (95 Prozent) an. Damit erhält man ein Signifikanzniveau von $\alpha = 1 - p = 0{,}05$.

Daraus ergibt sich für die *beidseitige* p-Grenze λ_p der Standardform:

$$\lambda_p = u_{(1+p)/2} = \Phi^{-1}\left(\frac{1+p}{2}\right) = \Phi^{-1}(0,975) = 1,96$$

Dabei ist u die einseitige p-Grenze und Φ die Verteilungsfunktion der Standardnormalverteilung. Eine Hypothese H_0 ("die gesampleten Werte liefern das exakte Ergebnis") wird bei einem Signifikanzniveau von α abgelehnt, falls

$$\lambda_p \leq \frac{\mid \bar{X} - \mu \mid}{\sigma/\sqrt{n}} = \frac{e}{1/\sqrt{n}}$$

($\sigma = 1$) wobei e den Betrag des mittleren Fehlers repräsentiert. Damit ergibt sich für $\lambda_p = 1{,}96$:

$$e = 1,96 \cdot \sqrt{1/n}$$

wobei n die Anzahl der betrachteten Samples ist (= der Umfang der Stichprobe).

Daraus ergeben sich (analog) Formeln wie:

$$p\% = 90\% \Rightarrow e\% = 164 \cdot \sqrt{1/n}$$

$$p\% = 95\% \Rightarrow e\% = 196 \cdot \sqrt{1/n}$$

$$p\% = 99\% \Rightarrow e\% = 258 \cdot \sqrt{1/n}$$

Man sieht (wie nicht anders zu erwarten ist): Je größer n, desto geringer ist die Fehlerwahrscheinlichkeit. Um also eine größere Sicherheitswahrscheinlichkeit und damit eine größere Genauigkeit zu erreichen, muß man mehr Samples pro Sekunde erfassen und / oder eine größere Zeitspanne betrachten.

Hat man nun aber den (relativen) Fehler gegeben und will daraus auf die Sicherheitswahrscheinlichkeit schließen, so muß man zunächst λ_p berechnen:

$$e = \lambda_p \cdot \sqrt{1/n} \Leftrightarrow \lambda_p = \frac{e}{\sqrt{1/n}}$$

Mit λ_p kann man nun folgendermaßen die Sicherheitswahscheinlichkeit p berechnen:

$$\Phi^{-1}\left(\frac{1+p}{2}\right) = \lambda_p \Leftrightarrow \frac{1+p}{2} = \Phi\left(\lambda_p\right) \Leftrightarrow p = 2 \cdot \Phi\left(\lambda_p\right) - 1$$

Durch Einsetzen erhält man für die Sicherheitswahscheinlichkeit p bzw. für die prozentuale Sciherheitswahlscheinlichekit $p\%$:

$$p = 2 \cdot \Phi\left(\frac{e}{\sqrt{1/n}}\right) - 1$$

$$p\% = \left(2 \cdot \Phi\left(\frac{e}{\sqrt{1/n}}\right) - 1\right) \cdot 100\%$$

Um diesen Abschnitt abzurunden sollen noch kurz ein paar Zahlen genannt werden: Im betrachteten Zeitraum T treten 3.600 Frames auf. Bei der Default-Samplingrate von 400 werden dabei gerade mal 9 Samples erfaßt. Bei einer geforderten Sicherheitswahrscheinlichkeit von 95% erhält man eine Fehlerwahrscheinlichkeit von 65% (bei einer Sicherheitswahrscheinlichkeit von 99% erhält man sogar eine Fehlerwahrscheinlichkeit von 86%!), d.h. das durch ein statistisches Verfahren ermittelte Datenvolumen beträgt 1260 bis 5940 Frames (bzw. 504 bis 6696); man hat es also in diesem Fall mit einer in keinster Weise akzeptierbaren Abweichung zu tun.

Ganz anders sieht es bei einem Datenvolumen von 1.000.000 Frames im betrachteten Zeitraum T aus: Hier hat man 2.500 Samples, die analysiert werden können; man erhält eine Fehlerwahrscheinlichkeit von 3,92% (bzw. 5,16%), das bedeutet das berechnete Datenvolumen beträgt 960.800 bis 1.039.200 Frames (bzw. 948.400 bis 1.051.600 Frames), was bereits eine gute Näherung darstellt.

7.3 HP NetMetrix

Bei NNStat wird der Netzverkehr an verschiedenen Orten im Netz durch die statspy-Hosts verteilt erfaßt. Dabei werden die einzelnen Frames bereits vor Ort analysiert und den spezifizierten Flows zugeordnet. Hier haben wir eine verteilte Erfassung *und* Analyse. Bei HP EASE wird an verschiedenen Orten im Netz ungefähr jedes n-te Frame herausgegriffen und der Frame-Header an einen zentralen EASE Server geschickt; dort werden die eingehenden Frame-Header analysiert und die Ergebnisse gespeichert. Hier haben wir eine verteilte Erfassung, eine Sammlung über ein standardisiertes NM-Protokoll (SNMP) und eine zentrale Analyse. Der Vollständigkeit halber soll an dieser Stelle noch ein weiteres Werkzeug zur Erfassung von Netzverkehr angesprochen werden, da es eine weitere Methode realisiert: HP NetMetrix.

Die Besprechung ist jedoch nur rein **konzeptioneller** Natur, da dieses Werkzeug zwar verfügbar war, jedoch dennoch nicht zum Einsatz gekommen ist: Auf einer HP-UX hätte man zur Installation von NetMetrix den Kernel zweimal neu binden müssen, auf den zur Verfügung stehenden SunOS-Systemen war leider nicht genug Plattenspeicherplatz[57] vorhanden. Rein konzeptionell ist NetMetrix jedoch sehr interessant.

Wie bei NNStat wird bei NetMetrix der Netzverkehr **verteilt erfaßt** und auch **verteilt analysiert**; die Ergebnisse werden in der Probe in einer **MIB** gespeichert. Diese MIB besteht im wesentlichen aus der RMON MIB (siehe dazu Kapitel 4.2) und einer proprietären Erweiterung der RMON MIB in Form einer eigenen Gruppe, in der die interessanten Verkehrsdaten (ab OSI-Schicht 3) gespeichert werden. Somit ist die Aussage, daß NetMetrix RMON-basiert arbeitet, nur die halbe Wahrheit!

Als Probes können hardware-basierte Probes, wie die *HP LanProbe II*, oder auch eine software-basierte Probe, der *HP Power Agent*, eingesetzt werden. Es wird pro Segment mindestens eine Probe plaziert. Die MIBs der Probes werden zentral von einer NM-Station aus über das standardisierte NM-Protokoll SNMP abgefragt. Die Konfiguration und die Steuerung der Probes geschieht ebenfalls über SNMP. Auf der NM-Station laufen Anwendungen wie der *Internetwork Monitor*, der *Load Monitor* oder der *Protocol Analyzer*, die die erfaßten Daten weiterverarbeiten und grafisch aufbereiten.

Im LRZ[58] wurde NetMetrix versuchsweise eingestzt, um seine Verwendbarkeit für die Erstellung von Netzstatistiken zu überprüfen. Dabei zeigte sich, daß NetMetrix eine ungeheure Systembelastung erzeugt und auf normalen Workstations eine

[57] Es wäre ein Minimum von 300MB nötig gewesen.
[58] Leibnitz RechenZentrum

sehr unbefriedigende Performanz aufweist. Auch aus der Sicht von Accounting erscheint NetMetrix bereits bei theoretischer Betrachtung ungeeignet zu sein. Es können zwar accountingrelevante Daten wie IP-Adressen und TCP- bzw. UDP-Ports erfaßt werden, jedoch besteht keine Möglichkeit an diese Datenbasis ausserhalb der Benutzeroberfläche heranzukommen, wie dies z.B. bei HP EASE der Fall ist[59]. Das heißt, daß die erfaßten Daten wahscheinlich nicht zu Accounting-Zwecken weiterverarbeitet werden können.

Das Konzept von NetMetrix ist jedoch höchst interessant und ist in etwa mit der dem Internet Accounting Modell zugrundeliegenden Methode (Realisierung durch die Internet Accounting MIB) zu vergleichen: Der Netzverkehr wird verteilt erfaßt, analysiert *und* gleich vor Ort in einer Monitor-MIB gespeichert. Von dort aus könnten dann die abrechnungsrelevanten Daten gesammelt werden.

Die Methode von NetMetrix ist noch einmal in vereinfachter Form in der folgenden Abbildung grafisch dargestellt:

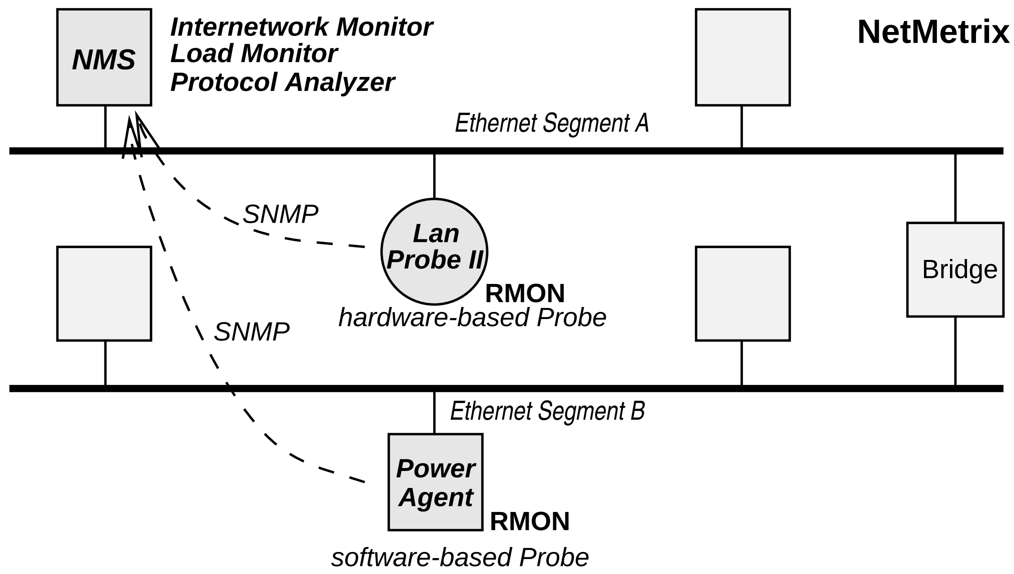

Abbildung 18: NetMetrix

[59]Laut NetMetrix-Handbuch; angeblich existiert aber eine Möglichkeit, ASCII - Dumps auszugeben.

8 Kriterien zur Bewertung von Werkzeugen

In diesem Kapitel werden Kriterien aufgestellt, nach denen Werkzeuge für die Erfassung von Netzverkehr bezüglich ihrer Eignung für den Einsatz im Rahmen eines Abrechnungsmanagements bewertet werden können. Die meisten Kriterien entstehen dabei als logische Schlußfolgerungen aus den Anforderungen an ein Abrechnungsmanagement. Ein zu überprüfendes Werkzeug sollte diesen Kriterien so weit wie möglich gerecht werden. Es ist jedoch sehr unwahrscheinlich, daß ein Werkzeug *allen* Kriterien in ausreichendem Maße gerecht wird. Im Anschluß an dieses Kapitel werden die Werkzeuge NNStat und HP EASE anhand dieser Kriterien exemplarisch bewertet. Die Bewertung von NetMetrix enfällt (siehe dazu Abschnitt 7.3).

8.1 Genauigkeit

Ein Maximum an Genauigkeit bei der Erfassung aber auch bei der Sammlung und Weiterverarbeitung der erfaßten Daten ist die Vorraussetzung für eine verbrauchsbezogene, verursachergerechte Abrechnung von Netzdiensten.

Erfassungs-Werkzeuge die eine exakte Methode realisieren, sollten im Normalfall wirklich jeden Accountable Event und damit auch wirklich jedes zu erfassende Frame registrieren, analysieren und aufzeichnen können. Die Genauigkeit ist abhängig von der Belastbarkeit einer Hardware-Probe bzw. eines Hosts, auf dem eine Software-Probe läuft (vor allem wenn auf diesem Host zusätzlich (Benutzer-) Prozesse laufen!) und der Intensität des Datenverkehrs auf dem Medium mit den zu erfassenden Frames. Um die Genauigkeit von exakten Werkzeugen zu überprüfen, muß **kontrolliert Netzverkehr erzeugt** werden. Es sollten dabei Flows von unterschiedlicher Länge (auch mit unterschiedlicher Frame-Länge) und unterschiedlicher Intensität erzeugt werden. Auf diese Weise können auch Lastspitzen simuliert werden. Dazu sollten verschiedene Szenarien untersucht werden.

Ein denkbares Szenario wäre die Erzeugung eines Flows von einem Host A zu einem Host B über einen bestimmten UDP-Port[60]. Auf Host A wird die Anzahl der zu übertragenden Frames bzw. Bytes angegeben. Host B gibt nach dem Empfang aller Frames dieses Flows eine Rückmeldung darüber, wieviele Frames er tatsächlich empfangen hat. Hat er alle Frames ordnungsgemäß empfangen, sollte auch eine Probe oder ein Koppelement, das als Erfassungspunkt im betrachteten Segment oder zwischen den betrachteten Segmenten dient, all diese

[60]UDP-Ports eignen sich insofern besser als TCP-Ports, da die Anzahl der Frames innerhalb einer TCP-Verbindung je nach Anzahl der gesendeten Frames im Rahmen des Verbindungsauf- bzw. -abbaus variieren kann

Frames erfaßt haben. Die Frames sollten von Host A aus in bestimmten, sehr kurzen Zeitabständen (wenige ms), zu Host B gesendet werden. Damit können gezielt Lastspitzen erzeugt werden. Stellt sich ein exaktes Werkzeug auf diese Weise tatsächlich als ein exaktes Werkzeug heraus, so kann dieses Werkzeug für die Überprüfung der Genauigkeit eines statistischen Werkzeugs herangezogen werden.

Natürlich kann ein statistisches Werkzeug auch durch die gezielte Erzeugung von Netzverkehr überprüft werden. Jedoch hat die Überprüfung anhand eines exakten Werkzeugs i.d.R. einen entscheidenden Vorteil: Man kann für die Überprüfung längere Erfassungszeiträume[61] betrachten. Denn erst bei der Betrachtung eines längeren Erfassungszeitraums werden die Ergebnisse der Erfassung eines statistischen Werkzeugs hinreichend genau. Ausserdem kann so ein reales Szenario, der reale Datenverkehr mit seinen speziellen Charakteristika, wie erhöhte Last in Spitzen-Zeiten, betrachtet werden.

Anhand der Meßergebnisse kann nicht nur die Genauigkeit als solche überprüft werden, sondern man kann anhand dieser Daten auch einfacher die Werte für das jeweilige Sampling-Intervall pro Segment und die Länge der Erfassungszeiträume bestimmen.

8.2 Konfigurierbarkeit und Flexibilität

Neben der Genauigkeit eines Erfassungswerkzeugs ist auch die Konfigurierbarkeit von besonderer Bedeutung. Ein erstes Kriterium ist dabei: **was** kann eigentlich überhaupt mit diesem Werkzeug erfaßt werden?

Grundsätzlich könnten alle Header-Felder aller Protokolle aller sieben OSI-Schichten als Erfassungsgrundlage dienen. Protokolle der Schicht 2 sind abhängig vom verwendeten Medium und von der verwendeten Übertragungstechnik. Daher werden i.d.R. nur ein oder zwei Protokolle dieser Schicht von einem Erfassungswerkzeug unterstützt. In den meisten Fälle ist es aufgrund seiner großen Verbreitung das Ethernet.

Auf Schicht 3 und 4 werden dann meist eine Vielzahl von Protokollen unterstützt. Beispiele dafür sind IP, UDP, TCP, DEC, IPX und die Novell Netware Protokolle IPX/SPX. Auf Schicht 4 kann bereits eine Identifikation von Diensten erfolgen, z.B. über die TCP-Ports. Eine Identifikation von Benutzern ist erst ab Schicht 5 möglich. Jedoch werden die OSI-Schichten 5 bis 7 nur von einigen, wenigen

[61] Wenn über längere Zeiträume auf die eben beschriebene Art und Weise intensiv Netzverkehr erzeugt wird, so leiden die eigentlichen Benutzer des Datennetzes spürbar unter dieser zusätzlichen, erhöhten Netzbelastung

Werkzeugen unterstützt und dann meist nur in Form von Protokoll-Analysatoren, die immer nur einige wenige (so um die 1.000) Frames analysieren können.

Wünschenswert wäre ein Werkzeug, das alle sieben OSI-Schichten für möglichst viele Protokolle analysieren und nach Bedarf die Inhalte der interessierenden Header-Felder in einer Datenbank ablegen kann. Der Hauptgrund dafür, daß meist nur die Protokolle der OSI-Schichten 2 bis 4 analysiert werden, ist, neben einer zusätzlichen, erhöhten Systembelastung, daß so gut wie alle Werkzeuge nicht für ein Accounting sondern für die Aufgaben des Performance Managements bzw. für statistische Zwecke entwickelt worden sind.

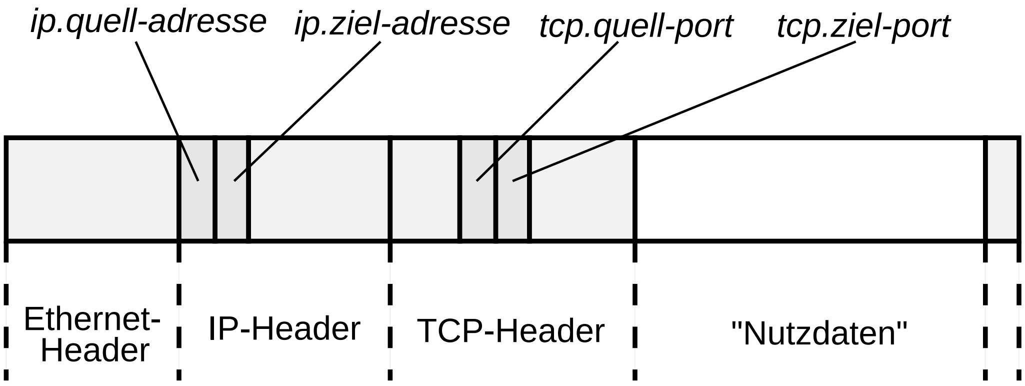

Abbildung 19: Ethernet-Frame mit Header-Feldern

Ein zu bewertendes Werkzeug sollte alle Protokolle unterstützen, die in seinem möglichen zukünftigen Einsatzgebiet gefahren werden. Das Werkzeug sollte grundsätzlich alle abrechnungsrelevanten Header- Felder anaylsieren und aufzeichenen können (in Form eines n-Tupels). Zusätzlich neben dieser **ungefilterten** Erfassung des Netzverkehrs sollte auch eine **gefilterte** Erfassung möglich sein. Die dazu nötigen Filter sollten dazu **frei konfigurierbar** sein. Damit kann z.B. auch eine Aufzeichnung des Netzverkehrs nach bestimmten Diensten erfolgen. Dies macht jedoch nur insofern Sinn, wenn dazu alle verwendeten Ports bekannt sind bzw. wenn aus den Endsystemen Information über die Identität von bestimmten Ports geliefert werden kann.

Neben der Forderung nach einer freien Konfigurierbarkeit sollte auch die Forderung nach einer leichten und flexiblen Konfigurierbarkeit berücksichtigt werden (Neben dem *was* ist auch das *wie* von besonderem Interesse). Es ist dabei nicht nur die Konfigurierbarkeit auf der Ebene der Erfassung (d.h. konkret die Analyse der Header-Felder), sondern auch die Konfigurierbarkeit auf der Ebene der Sammlung der erfaßten Netzdaten Gegenstand der Untersuchung. So sollten u.a.

mehrere Erfassungspunkte angegeben werden können, die Intervalle zur Sammlung der erfaßten Daten sollten beliebig gesetzt werden können, und es sollte angegeben werden können was gesammelt werden soll (Filterfunktionalität der Sammlung).

Eine freie, leichte und flexible Konfigurierbarkeit der eingesetzen Erfassungswerkzeuge gewährleistet eine hinreichende Flexibilität der Erfassungs-Methode. Diese muß an sich ändernde Bedingungen, wie eine Veränderung der Topologie (z.B. das Umsetzen eines Servers zur besseren Verteilung der Netzlast) oder eine Änderung in der Abrechnungspolitik, leicht angepaßt werden können. Dazu sollten zum einen die Erfassungspunkte (Probes, Koppelemente) und zum anderen die Kollektoren (verteilte Sammlungspunkte, Erfassungs-Server, NM-Stationen) möglichst beliebig konfiguriert werden können. Parameter, die konfiguriert werden können (bzw. könnten) finden sich im Abschnitt 5.3 (OSI-konformes Objekt-Modell).

8.3 Speicherplatzbedarf und Archivierung

Die hinreichend genau erfaßten Daten müssen (zwischen-)gespeichert werden. Dazu muß in ausreichendem Maße Hauptspeicher- bzw. Plattenplatz zur Verfügung stehen. Bereits in kleineren Datennetzen und bei der Betrachtung nur eines Teils des Datenverkehrs hat man es sehr schnell mit einer beachtlichen Menge an Daten zu tun (Massendaten!), die geeignet gespeichert werden müssen.

Um einen Einblick in die Größenordnung dieser Massendaten zu geben, soll an dieser Stelle kurz ein theoretisches Beispiel betrachtet werden: Man nehme ein Datennetz mit einer Gesamtzahl von 100 Hosts an. Der **Connectivity Factor**[62] betrage 20. Alle 100 Hosts sind an einem Ethernet-Bus angeschlossen. Eine Probe erfaßt den gesamten Datenverkehr. Erfaßt werden nur die IP-Adressen (2 mal 4 Bytes) und die Port-Nummern (2 mal 2 Bytes). Zu diesem 4-Tupel kommt noch ein Zähler für die Anzahl der Frames (4 Bytes) und ein Zähler für die Anzahl der übertragenen Bytes (6 Bytes) hinzu[63]. Um einen Flow zu speichern, werden somit genau 22 Bytes benötigt. Im Mittel werden 2.000 Flows, also 44.000 Bytes abgespeichert. Diese sollen stündlich gespeichert werden. Es ergibt sich pro Tag ein Datenvolumen von ca. 1031 Kilobytes, pro Monat sind es rund 32 Megabytes. Bereits bei diesem kleinen Beispiel erhält man ein beachtliches Datenvolumen. In realen, mittleren Datennetzen kommt man pro Monat leicht auf ein Datenvolumen von 300 Megabytes.

[62]Ein Durchschnittswert, der angibt, mit wievielen anderen Hosts ein bestimmter Host im Mittel kommuniziert

[63]*Minimaler* Speicherplatzbedarf; der *reale* Speicherplatzbedarf ist i.d.R. wesentlich größer.

Diese abrechnungsrelevanten Daten müssen mindestens einen Monat umfassen und anschließend archiviert werden (Nachweispflicht!). Die Archivierung nach einem erfaßten Monat kann heutzutage problemlos, kosten- und platzsparend mit Magnetbändern (z.B. DAT) realisiert werden. Die laufenden Erfassungdaten müssen i.d.R. auf Festplatten gehalten werden. Hinzu kommt noch der Hauptspeicherplatz. Auch dieser kann in beachtliche Größenordnungen kommen, wenn z.B. eine Software-Probe auf einem Host läuft und den passierenden Datenverkehr erfaßt, analysiert und die Ergebnisse bis zum Zeitpunkt der Sammlung im Hauptspeicher hält.

Aber nicht nur der Speicherplatz für die zu erfassenden und zu archivierenden Daten muß berücksichtigt werden, sondern auch der Speicherplatzbedarf (v.a. Hauptspeicherplatz) für die Werkzeuge muß beachtet werden. 150 MB Plattenplatz und 64 MB Hauptspeicher als Systemanforderungen sind in diesem Zusammenhang keine Seltenheit.

8.4 Netz- und Systembelastung

Ein weiteres, wichtiges Kriterium ist die zusätzliche Netz- und Systembelastung, die durch den Einsatz von Werkzeugen zur Erfassung von Netzverkehr entsteht. Sie sollte möglichst gering gehalten werden und ist unmittelbar von der gewählten Methode abhängig.

In den Erfassungspunkten ist die Systembelastung durch das ständige Analysieren der Frame-Header und die andauernde Speicherung der Ergebnisse sehr hoch. Hohe CPU-Leistung und geringe Zugriffszeiten auf den Hauptspeicher, vor allem aber auf die Platte(n), sind hier Vorraussetzung, damit im exakten Fall auch wirklich jedes Frame erfaßt und analysiert werden kann. Aber nicht nur die Hardware kann die Performanz des Systems erhöhen. Optimierte Software ist selbstverständlich leistungsfähiger als umständlich programmierte Software. Vor allem der Leistungsfähigkeit der Erfassungssoftware (nicht nur der schönen Benutzeroberfläche) sollte ein besonderes Augenmerk gewidmet werden.

Die hohe Systembelastung ist der Hauptgrund, warum die Koppelelemente (noch) nicht als vollwertige Erfassungspunkte eingesetzt werden können.

Neben der Systembelastung in den Erfassungspunkten und natürlich auch an den Orten der Sammlung und der Weiterverarbeitung (weitere Analyse, Filterung, Zusammenfassung, Anreicherung der Information mit Zusatz- Information usw.) ist die Netzbelastung durch die Sammlung der Netzdaten ein weiteres Kriterium zur Bewertung der Werkzeuge. Die erfaßten Daten sollten in einer möglichst komprimierten Form gesammelt werden. Das bedeutet für den exakten Fall, daß die

Inhalte der Frame-Header schon vor Ort so weit wie möglich analysiert werden und nur noch Tabellen von Ergebnissen periodisch übermittelt werden sollten.

Unter der Annahme, daß im obigen Beispiel jeder der 100 Hosts pro Stunde nur 1 Megabyte an Daten austauscht und daß stündlich die erfaßten Daten von der Probe gepollt werden (Protokoll-Overhead soll vernachlässigt werden), so beträgt der Anteil der Erfassungsdaten am Netzverkehr nur 0,04% (Gesamtverkehr = 100 * 1.024 * 1.024 = 104.857.600 Byte; Größe der Tabelle der erfaßten Daten = 44.000 Byte). Wird hingegen eine statistische Methode verwendet, wobei jedes 50-te Frame erfaßt und an einen zentralen Server geschickt wird, dann beträgt der Anteil der Erfassungsdaten am Netzverkehr 2%. Generell sollte der durch die Erfassung von Netzdaten entstehende Netzverkehr maximal 5% des normalen Datenverkehrs betragen. Die Untersuchung der Netz- und Systemlast hängt eng mit der Untersuchung des Speicherplatzbedarfs zusammen.

8.5 Korrelierbarkeit der erfaßten Daten

Die in den Erfassungspunkten erfaßten Daten müssen früher oder später zusammengeführt werden. Dazu sollten die Daten bereits in den Erfassungspunkten in eine Form gebracht werden, die einfach weiterverarbeitet werden kann. Dies geschieht am einfachsten in Form von Tabellen mit einem festen Format (Übereinstimmung in Anordnung und Beschaffenheit der Spalten).

Das Zusammenführen über Zeitstempel ist zum einen wegen des dadurch wesentlich größeren Datenvolumens[64] und zum anderen aufgrund der Problematik einer einheitlichen Netz-Zeit[65] sehr schwer und sehr uneffizient realsierbar.

Das Zusammenführen geschieht so i.d.R. durch einfache **Unifikation**; d.h. Flows mit gleichen Tabellen-Einträgen (gleiche IP-Adressen, gleiche Portnummern, usw.) werden einfach zusammengefaßt; die Zählerstände werden addiert. Die Sammlung der verteilt erfaßten Daten impliziert das korrekte Zusammenführen dieser Daten. In diesem Zusammenhang sollte neben der Korrektheit des Zusammenführens auch die Effizienz des Zusammenführens betrachtet werden. Die Effizienz ist u.a. abhängig von der Art der Sammlung der erfaßten Netzdaten. Dazu sollen grundsätzlich zwei Ansätze zur Sammlung von Netzdaten unterschieden werden: Der **flache Ansatz** und der **hierarchische Ansatz**.

[64] Keine oder nur eine eingeschränkte Summenbildung über die Zeit führt zu einer wesentlich größeren Anazhl an Flows

[65] Dazu müßte ein spezielles Protokoll zur Synchronistaion der System-Zeiten aller Hosts eingesetzt werden, das jedoch auch nicht so genau ist (unterschiedliche Signallaufzeiten, Ausfall von Hosts).

Beim flachen Ansatz sind die Erfassungspunkte über das Datennetz hinweg verteilt. Ein oder mehrere **zentrale** Hosts sammeln die erfaßten Daten und verarbeiten sie weiter. Man spricht in diesem Fall auch von einer *verteilten* Erfassung mit einer *zentralen* Sammlung. In kleineren Datennetzen ist dieser Ansatz durchaus sinnvoll.

Beim hierarchischen Ansatz befinden sich zwischen den Erfassungspunkten und den zentralen Hosts noch ein oder mehrere Kollektoren (verteilte Sammlungspunkte). So kann es einen Kollektor geben, der die erfaßten Daten aller Probes für ein bestimmtes Segment, z.B. eine Abteilung, sammelt. Ein Kollektor für ein Teil-Netz, z.B. für einen bestimmten Standort, sammelt wiederum die bereits von den Kollektoren für die einzelnen Segmente gesammelten und zusammengefaßten Daten, faßt sie seinerseits zusammen und stellt sie dem oder den zentralen Hosts (aufgrund der hohen Systembelastung sollten am besten Mainframes verwendet werden) zur Verfügung. Man spricht hier von einer *verteilten* Erfassung mit einer *verteilten* Sammlung. Der hierachische Ansatz ist in mittleren, vor allem aber in großen Datennetzen das Mittel der Wahl.

Es muß also die Korrelierbarkeit, die Korrektheit des Zusammenführens und die Effizienz des Zusammenführens untersucht werden. Die Effizienz des Zusammenführens ist natürlich in erster Linie von der Leistungsfähigkeit der eingesetzen Hard- und Software anhängig.

8.6 Auswertbarkeit und Weiterverarbeitung

Die durch die verteilte Erfassung und durch die verteilte bzw. zentrale Sammlung der abrechnungsrelevanten Daten gewonnene Information sollte in einer Form gespeichert werden, so daß sie unterschiedlich auswertbar ist. So sollten einerseits anhand dieser Information ohne großen Aufwand Rechnungen gestellt werden können. Andererseits sollten sich aus den erfaßten Daten auf einfache Art und Weise Nutzerprofile erstellen lassen, die zudem noch graphisch dargestellt werden könnten.
So dienen die erfaßten Daten nicht nur der Rechnungstellung sondern auch der Planung und der Investitionsbeurteilung: Veränderung der Netztopologie und der logischen Strukturierung des Netzes (z.B. Positionierung von Servern); neuer Einsatz und Austausch von Software; Veränderungen an der Hardware der Endsysteme; Steigerung der Übertragungskapazität; Rentabilität der eingestzten Ressourcen. Durch (weitere) Summenbildung bei den Accountable Units können die Daten, wie bereits diskutiert, auch sehr gut für die Aufgaben des Performance Managements verwendet werden. Die Form der erfaßten Daten ist somit in erster Linie abhängig von der Art der Weiterverabeitung.

Unabhängig von der Art der Weiterverarbeitung sollte die Information auf jeder Ebene in Form von Tabellen gehalten werden. Sie können so einfach in weiterverarbeitende bzw. archivierende Applikationen wie z.B. relationale Datenbanken oder Tabellenkalkulationen importiert werden; mit Blick auf das (integrierte) Netzmanagement könnten sie genauso einfach in (verteilt gelegene) Accouning-MIBs importiert werden, von wo sie dann über ein standardisiertes Netzmanagement-Protokoll unter verschiedensten Aspekten abgefragt werden könnten.

Um die verschiedenen Erfassungsebenen und die jeweilige, mögliche Form der erfaßten Daten zu illustrieren, soll folgendes, theoretisches, Beispiel dienen:

- Ebene der Datenerfassung vor Ort (Usage Metering Process):
  Erfaßt werden sollen: IP-Quell- und Zieladresse, TCP-Quell- und Zielport, Anzahl der übertragenen Bytes sowie geographische Information. Die Daten werden im Segment buha (= Buchhaltung) erfaßt, sie werden in folgender Form gespeichert[66]:

ip.src	ip.dst	tcp.src	tcp.dst	#bytes	segment
122.123.10.12	128.100.10.20	21	3059	10000	buha
122.123.10.12	128.100.10.20	20	3065	90000	buha
...					

  In einem anderen Segment (eink = Einkauf) werden analog folgende Daten erfaßt und *in der gleichen Form* gespeichert:

ip.src	ip.dst	tcp.src	tcp.dst	#bytes	segment
122.123.10.12	128.100.10.20	21	3059	10000	eink
122.123.10.12	128.100.10.20	20	3065	90000	eink
...					

- Ebene der verteilten Sammlung (Integration Process):
  Im Bereich geb3 (= Gebäude 3) werden die verteilt erfaßten Daten gesammelt und zusammengefaßt; sie werden in der folgenden Form gespeichert:

ip.src	ip.dst	tcp.src	tcp.dst	#bytes	segmente
122.123.10.12	128.100.10.20	21	3059	10000	buha,eink
122.123.10.12	128.100.10.20	20	3065	90000	buha,eink
...					

- Ebene der zentralen Sammlung:
  Die Daten werden aus den verschiendenen Bereichen (ber = Bereiche) ge-

[66] Der Einfachheit halber wird nur der Datenverkehr für eine Richtung (vom ftp-Server zum Client) betrachtet.

sammelt, u.U. weiter zusammengefaßt (Integration Process), den entsprechenden Diensten (ser = Service) zugeordnet (Service Projection Process) und mit Kosteninformation versehen (Pricing Process). Zudem werden die IP-Adressen in Namen umgewandelt. Die Daten haben schließlich die folgende Form:

ip.src	ip.dst	p.src	p.dst	ser	#bytes	ber	segmente	cost
maier	huber	21	3059	ftp	10000	geb3	buha,eink	0,03
maier	huber	20	3065	ftp	90000	geb3	buha,eink	0,03
...								

Diese Daten müssen archiviert werden, schon allein um der Nachweispflicht zu genügen.

- Ebene der Weiterverarbeitung:
  Hier soll lediglich die Rechnungstellung (Billing Process) betrachtet werden. Dazu werden die Daten weiter zusammengefaßt und nur die in der Rechnung auftretenden Daten werden in einer Tabelle gespeichert:

name	partner	service	volumen	kostensatz	betrag
huber	maier	ftp	100000	0,03 DM/kB	2,93 DM
...					

In dieser Tabelle treten nur der Dienstnutzer (huber; ftp-client), der Kommunikationspartner (maier; ftp-server), das übertragene Datenvolumen, der Kostensatz und die Kosten dieser Datenübertragung auf. Diese Daten sind ausreichend für die Rechnungstellung.

Sollen die Daten unter mehreren Gesichtspunkten weiterverarbeitet werden, so muß u.U. die Form der erfaßten Daten auf allen Ebenen geändert werden. Die einfache und problemlose Abänderung der Form der Daten sollte bereits anhand des Kriteriums *Konfigurierbarkeit und Flexibilität* untersucht worden sein.

8.7 Integrierbarkeit

Die Werkzeuge sollten idealerweise auf drei Ebenen integrierbar sein:

- Integration auf der Ebene der verteilten Erfassung und Sammlung von Netzverkehr: Damit ist die Vereinheitlichung der Form der erfaßten Daten gemeint. Auf diese Weise ist auch eine Abstraktion von der Netzinfrastruktur möglich (z.B. wenn in einem Bereich in einem Segment FDDI-Verkehr erfaßt wird und in einem anderen Segment Ethernet-Verkehr; die Daten von beiden

Segmenten werden von einem Host für den Bereich gesammelt, zusammengefaßt und in eine einheitliche Form gebracht (alle anderen Bereiche halten ihre Daten in der gleichen Form; unabhängig von der verwendeten Netz-Technologie stehen so die Daten einheitlich zur Verfügug). Die Integration auf dieser Ebene besteht mit Blick auf ein (integriertes) Netzmanagement darin, die vor Ort erfaßten Daten auch vor Ort in Form von MIBs zu halten.

- Integration in eine Netzmanagement-Plattform: Hier müssen die Daten in entsprechenden MIBs zur Verfügung stehen. Bei einer Integration z.B. in das Netzmanagement-System SPECTRUM[67] könnte die Integration darin bestehen, daß verschiedene Accounting-Views erzeugt werden, über die das Accounting gesteuert und überwacht werden kann. Vorraussetzung ist die Integration auf der Ebene der verteilten Erfassung und Sammlung (alle abrechnungsrelevanten Daten sowie Mechanismen zur Steuerung des Accountings in Form von MIBs).

- Integration in eine (grafische) Benutzeroberfläche zur Erleichterung der Steuerung und Überwachung des Accountings (setzt die Integration auf den beiden unteren Ebenen *nicht* voraus).

8.8 Dienst- und Benutzerbezogenheit

Die Daten sollten sowohl dienst- als auch benutzerbezogen erfaßt werden können. Im betrachteten Szenario kann der Benutzer in der Regel jedoch nicht erfaßt werden (Ausnahme: Jeder Host wird immer nur ausschließlich von einem einzigen Benutzer benutzt). Für die Identifikation des Benutzers ist die Analyse der OSI-Schichten 5 bis 7 oder aber Zusatzinformation aus den Endsystemen nötig. Auf den OSI-Schichten 2 bis 4 ist generell keine Benutzeridentifikation möglich. Als ein abgeschwächtes Ersatzkriterium sollte zumindest die (optionale) Umwandlung von IP-Adressen in Namen berücksichtigt werden.

Auch die Identifikation von Diensten ist auf den OSI-Schichten 2 bis 4 nur sehr eingeschränkt möglich. Die einzige Identifikationsmöglichkeit sind die Well-Known Ports (siehe Anhang). Aber es können i.d.R. nur TCP-Verbindungen auf diese Weise erfaßt werden. Jeglicher (zusätzlicher) UDP-Verkehr kann ohne Zusatzinformation aus den Endsystemen nicht zugeordnet werden. Viele Dienste benutzen aber TCP und UDP gleichzeitig. Nur einige wenige Dienste wie FTP können auf diese Art und Weise vollständig erfaßt werden. Wenn die Erfassung so nicht möglich ist, so kann wenigstens ein Umschlagen des nicht zuordnunsfähigen

[67]von der Firma Cabletron

Verkehrs anhand der über die Well-Known Ports ermittelten Daten erfolgen[68]. Diese Vorgehensweise pauschalisiert das Abrechnungsverfahren jedoch deutlich. Als ein abgeschwächtes Ersatzkriterium sollte die Erkennung möglichst vieler Well-Known Ports sowie die Möglichkeit, eigene Well-Known Ports zu definieren, berücksichtigt werden.

8.9 Zuverlässigkeit, Schutz vor Manipulation

An dieser Stelle sollte zum einen die Ausfallsicherheit aller an der Erfassung und Sammlung von Verkehrsdaten beteiligten Komponenten untersucht werden. Dazu gehört auch die Möglichkeit redundante Erfassungs- und Sammlungspunkte einsetzen zu können. Zum anderen sollten diese Komponenten (Hardware-Probes, Software-Probes auf Workstations, Kollektoren (auf Workstations), NM-Stationen, ...) gegen Manipulation und Mißbrauch (Datenschutz!) geschützt sein.
In Punkto Manipulationssicherheit ist dedizierte, abgeschlossene Erfassungs-Hardware eindeutig das Mittel der Wahl. Abgeschlosse Hardware bedeutet, daß die jeweilige Komponente keine *unmittelbare* Eingabemöglichkeit (z.B. Tastatur) und am besten auch so gut wie keine unmittelbare Ausgabemöglichkeit (außer etwa den LEDs zur Zustandsanzeige einer Probe) besitzt.
Der mittelbare Zugriff auf die Komponenten z.B. über SNMP sollte durch Sicherheitsmechanismen (wie die Vergabe von Passwörtern) auf den authorisierten Personenkreis eingeschränkt werden können (Es besteht i.d.R. auch keine Möglichkeit eines `rlogin`).

[68]Prozentuale Aufteilung des Restverkehrs anahand der Intensität der Nutzung der Well-Known ports

9 Bewertung von NNStat

In diesem Kapitel wird das Werkzeug NNStat anhand der zuvor aufgestellten Kriterien bewertet. Eine analoge Bewertung des Werkzeugs HP EASE folgt im nächsten Kapitel.

9.1 Genauigkeit

Bevor die Genauigkeit des Werkzeugs HP EASE (stochastische Methode) unter Zuhilfenahme des Werkzeugs NNStat (exakte Methode) überprüft werden kann, muß zunächst erst einmal die Genauigkeit von NNStat selbst untersucht werden; dabei muß die Frage beantwortet werden: Ist NNStat selbst hinreichend genau?

Um die Genauigkeit isoliert untersuchen zu können, muß dazu gezielt Netzverkehr erzeugt und erfaßt werden. Es wurde folgendes Szenario gewählt: Zwischen zwei Hosts wurde kontrolliert Netzverkehr erzeugt. Die Erzeugung des Netzverkehrs geschah mit dem Programm `spray` (Näheres zu `spray` siehe weiter unten). Spray wurde auf dem Host `sunhegering5` gestartet und sendete eine gewisse Anzahl von Frames zum Host `hphegering1`. Gleichzeitig wurde der Netzverkehr zwischen diesen beiden Hosts von einem dritten Host[69] aus, dem Host `sunhegering6`, durch das Programm `statspy` erfaßt. Erfaßt wurden dabei alle UDP-Ports zwischen den beiden betrachteten Hosts. Die Frames wurden je nach betrachtetem Ziel-Host i.d.R. immer an einen festen UDP-Port gesendet (z.B. 1042 beim Host `sunhegering3`, 1046 beim Host `hphegering1`) und konnten so leicht identifiziert werden.

Das Programm `spray` (Teil der *SunOS Networking Software Installation Option*) befindet sich normalerweise im Verzeichnis `/usr/etc` und kann von jedem Benutzer gestartet werden. Es muß nur der Ziel-Host angegeben werden. Weiterhin können folgende Optionen verwendet werden: Die Option `-c` gibt die Anzahl der Frames an, die gesendet werden sollen; die Option `-d` gibt die Verzögerungszeit in ms an, die zwischen dem Senden von zwei Frames eingehalten werden soll; wird die Option `-i` verwendet, so wird ICMP anstelle von RPC benutzt; mit der Option `-l` kann schließlich noch die Länge der User Data im Ethernet-Frame und damit die Frame-Länge angegeben werden. Die Länge der User Data ist normalerweise 86 Bytes, das entspricht der Länge der RPC und UDP/IP Header zusammen. Spray benutzt zum Versenden der Frames normalerweise RPC, das sich wiederum auf UDP stützt.

[69] NNStat erfaßt nur die am jeweiligen statspy-Host *empfangenen* Frames

Für die Verzögerungszeit wurde in den meisten Fällen `-d 1` verwendet, da bei einer Verzögerungszeit von 0 ms i.d.R. über 40% der Frames verlorengingen (und so nur teilweise erfaßt werden konnten). Die Anzahl der verlorengegangenen Frames ist abhängig von der aktuellen System- und Netzbelastung und ist daher starken Schwankungen unterworfen. Mit der Option -d 1 erreicht man eine deutliche Verzögerung gegenüber der Option -d 0. Der Unterschied der beiden Verzögerungszeiten ist sehr viel größer als nur 1 ms. Es wurden 1.000, 10.000 und 100.000 Frames gesendet; die Frames hatten dabei eine Länge von 86 bzw. 1026 Bytes[70]. Die Ergebnisse einiger Messungen sind in den folgenden Tabellen aufgelistet. Ein typischer Aufruf von `spray` sieht in etwa so aus:

```
/usr/etc/spray -c 10000 -d 0 -l 1026 131.159.12.39
```

Es folgen die Meßergebnisse in Form von zwei Tabellen: eine Tabelle für die Verzögerungszeit 0ms und eine Tabelle für die Verzögerungszeit 1ms[71]

Verzögerungszeit 0ms (-d 0):

Anz.Frames	**Frame-Länge**	**Frames/s**	**Fr.dropped**	**NNStat**
1.000	86	96	572	626
1.000	1.026	1.602	393	701
10.000	86	1.535	4.168	7.174
10.000	1.026	1.631	3.595	6.620

Verzögerungszeit 1ms (-d 1):

Anz.Frames	**Frame-Länge**	**Frames/s**	**Fr.dropped**	**NNStat**
1.000	86	88	keine	1.000
1.000	1.026	88	4	999
10.000	86	88	1	9.984
10.000	1.026	87	22	9.994
100.000	86	87	18	99.890
100.000	1.026	87	7	99.986

Da ein variierender Anteil der verlorengegangenen Frames dennoch von NNStat erfaßt wird und es aber nicht ausgeschlossen werden kann, daß NNStat von `spray` erfolgreich versendete Frames *nicht* erfaßt, sollten ausschließlich die Fälle betrachtet werden in denen keine oder nur sehr wenige Frames verlorengehen: Das sind die Fälle mit Verzögerungszeit; in diesen Fällen werden die Frames also nicht extrem burst-artig gesendet.

[70] Diese Zahlen ergeben sich aufgrund der Längen der RPC und UDP/IP Header
[71] Sicher wesentlich größer als 1ms !!!

Beachte:Die längere, extrem burst-artige Versendung von Frames erzeugt auf Seiten von NNStat ein so hohe Systembelastung, daß andere Prozesse, wie z.B. ein X-Server, extrem ausgebremst werden, was zu einiger Verärgerung bei anderen Benutzern führen kann! NNStat erfaßt aber auch in diesen Fällen, wie man auch in der ersten Tabelle sehen kann, recht zuverlässig.

Wie man sehr schön aus der zweiten Tabelle ersehen kann, kann mit Hilfe von NNStat tatsächlich (fast) jedes Frame erfaßt werden. Diese Aussage gilt natürlich nur dann, wenn keine extremen Lastspitzen[72] auftreten. Im Falle von extremen Lastspitzen kann NNStat sicherlich zu einer statistischen Methode entarten. Im Rahmen der Messungen dieser Arbeit konnten weder derartige Lastspitzen erzeugt noch an irgendeiner Stelle registriert werden. Man kann also im weiteren davon ausgehen, daß in einem durchschnittlich benutzten, nicht grob fehlerhaften, Netz NNStat wirklich eine exakte Methode realisiert.

9.2 Konfigurierbarkeit, Flexibilität

Mit NNStat können immer nur Paare von Header-Feldwerten in einem Recorder-Object der Klasse `matrix` aufgezeichnet werden. So eignet sich NNStat im allgemeinen nur für ein Accounting auf der Ebene von IP-Adressen (zwischen Endsystemen, jedoch nicht zwischen miteinander über das Datennetz kommunizierenden Prozessen). Accountingrelevante Header-Feldwerte sind dabei: `IP.length`, `IP.srchost`, `IP.dsthost`, `IP.srcnet`, `IP.dstnet`, `IP.srcsubn`, `IP.dstsubn`, `TCP.srcport`, `TCP.dstport`, `UDP.srcport` und `UDP.dstport`.
Mit Hilfe von Filter-Objects können damit beliebig Flows definiert und in Recorder-Objects der Klasse `matrix` aufgezeichnet werden. Beispiele für Filter-Objects sind: `eqf` und `setf`. Die Konfiguration für die Erfassungspunkte, die `statspy`-Hosts, kann durch ein Konfigurationsfile (in einer C-änhlichen Sprache), das wiederum weitere Konfigurationsfiles einschließen kann (`include`), spezifiziert werden. Der `collect` Prozeß kann als Hintergrundprozeß periodisch mehere `statspy`-Hosts auslesen und die Meßergebnisse (zentral) mitloggen (in Log-Files). Welche Objekte und in welchen Zeitabständen diese Objekte ausgelesen (und auch zurückgesetzt) werden sollen, wird mit Hilfe von Commandline-Optionen angegeben. Es können grundsätzlich beliebig viele `statspy`- und `collect`-Prozesse eingesetzt werden.

So ist NNStat, mit den zu Verfügung stehenden Header-Feldwerten und der Einschränkung auf ein- oder zweistellige Recorder-Objects, frei konfigurierbar. Die Konfiguration kann leicht — auch online — geändert werden (`attach` und `detach`). So kann NNStat leicht auf veränderte Bedinungen, wie eine Änderung in

[72] Vor allem Netzlast, aber auch Systemlast

der Abrechnungspolitik, angepaßt werden; die zugrundeliegende, exakte Methode besitzt somit ein hohes Maß an Flexibilität.

9.3 Speicherplatzbedarf, Archivierung

Der Speicherplatzbedarf für NNStat selbst ist sehr gering: `statspy` benötigt 140 Kilobyte Plattenplatz und etwa 400 Kilobyte Hauptspeicher; `collect` 100 Kilobyte Plattenplatz und ebenfalls etwa 400 Kilobyte Hauptspeicher[73]. In den `statspy`-Hosts werden die spezifizierten Flows in den entsprechenden Recorder-Objects lokal im Hauptspeicher gehalten; ein Recorder-Object, das ca. 1.000 Paare von IP-Adressen samt Zähler für Frames und Bytes speichert, wird in etwa nur 16.000 Bytes[74] belegen[75]. Wird ein Recorder-Object eines `statspy`-Hosts ausgelesen, so werden die gespeicherten Werte in ein Log-File auf der Festplatte des `collect`-Hosts geschrieben; ein Log-File, das (wie oben) ca. 1.000 Paare von IP-Adressen samt Zählerwerte für Frames und Bytes enthält, benötigt etwa 70.000 Bytes auf der Festplatte. NNStat benötigt also ein Minimum an Hauptspeicher- und Plattenplatz. Dieser Umstand dürfte sich auch bei der notwendigen Archivierung der erfaßten Netzdaten als sehr günstig herausstellen.

9.4 Netz-, Systembelastung

Die zusätzliche Netzbelastung durch den Einsatz von NNStat wird am besten mit Hilfe von NNStat selbst bestimmt. Dazu werden drei Hosts benötigt. Auf dem *ersten* Host läuft ein `statspy`-Prozeß, der den erfaßbaren Netzverkehr (z.B. IP-Verkehr) im betrachteten Segment aufzeichnet. Auf dem *zweiten* Host läuft ein `collect`-Prozeß, der die vom ersten Host erfaßten Daten periodisch (in kurzen Abständen) sammelt und in ein Log-File schreibt. Auf dem *dritten* Host läuft zusätzlich ein zweiter `statspy`-Prozeß, der den Netzverkehr zwischen den beiden anderen Hosts aufzeichnet. Dieser Datenverkehr wird ausschließlich über den Well-Known-Port 2222[76] abgewickelt.

Konkret wurde folgendes Szenario betrachtet: Der erste Host (`statspy`) war der Host `sunhegering5`. Sein `collect`-Host war der Host `sunhegering6`. Der Host, der den durch NNStat in diesem *isolierten* Szenario verursachten Netzverkehr erfaßt hat, war der Host `sunhegering3`. Das Konfiguarions-File von `statspy` für diesen Host hatte dabei folgendes Aussehen:

[73]ermittelt mit dem Kommando `top`

[74]4 Bytes pro IP-Adresse und 4 Bytes pro Zähler

[75]wurde nicht untersucht

[76]es kann auch ein anderer Port konfiguriert werden

```
attach
{
  if TCP.srcport is eqf(2222) or TCP.dstport is eqf(2222)
    record IP.srchost IP.dsthost in belastung matrix-all-bytes;
}
```

Der `collect`-Host sammelte in einem Zeitraum von 20 Minuten jede Minute die erfaßten Daten von seinem `statspy`-Host; dabei wurden insgesamt 487 Frames bzw. 291.424 Bytes ausgetauscht. Insgesamt wurden im betrachteten Zeitraum 88.817 Frames bzw. 18.731.949 Bytes registriert. Damit ergibt sich eine zusätzliche Netzbelastung von 0,5% bzw. 1,6%, je nachdem ob man die Anzahl der Frames oder die Anzahl der Bytes als Berechnungsgrundlage heranzieht. Das Sammeln der Daten wurde jede Minute durchgeführt; werden die Daten nur jede Stunde gesammelt, so erhält man lediglich eine zusätzliche Netzbelastung von etwa 0,01%. Auch wenn deutlich mehr Flows erfaßt werden sollten (bei dieser Messung waren es 309), so kann man ohne Bedenken behaupten, daß die Netzbelastung durch NNStat i.d.R. wirklich **sehr gering** ist.

Die Systembelastung, insbesondere die CPU-Belastung, ist bei NNStat, gerade bei hohem Datenverkehrsaufkommen, besonders groß. Wie die Messungen mit Hilfe des Programms `spray` im Abschnitt 9.1 gezeigt haben, kann die Systembelastung derart groß werden, so daß der `statspy`-Host zu stehen scheint, da er fast ausschließlich mit der Analyse der empfangenen Frames beschäftigt ist. Hosts, auf denen ein `statspy`-Prozeß läuft, können daher in der Regel nicht für weitere Aufgaben verwendet werden und dienen ausschließlich als Erfassungspunkte (Ausnahme: strukturierte Verkabelung, relativ geringer Datenverkehr im jeweiligen Segment).

9.5 Korrelierbarkeit der erfaßten Daten

Die erfaßten Daten in den `statspy`-Hosts liegen im Hauptspeicher als Tabelle vor. Diese Tabelle kann entweder durch einen `read`-Befehl über das Commandline-Interface von `statspy` bzw. `rspy` oder besser: durch einen `collect`-Prozeß ausgelesen werden. Bei der Verwendung von `collect` wird ein entsprechendes Log-File angelegt bzw. auf den neuesten Stand gebracht. Eine Zeile dieses Log-Files hat in etwa die folgende Gestalt (Beispiel-Log-File siehe Anhang):

```
[131.159.12.31 : 131.159.12.45]= 132487 &10178924B (20.9%) @-4sec
```

Diese Form läßt sich nicht direkt weiterverarbeiten. Die relevante Information

```
131.159.12.31   131.159.12.45   132487   10178924
```

muß erst extrahiert werden. Dafür existiert jedoch kein Werkzeug. Genausowenig existiert ein Werkzeug zum Zusammenfassen von mehreren Log-Files (Teilaufgabe des Integration Process). NNStat stellt somit *keine* Werkzeuge zur Korrelation der erfaßten Daten zur Verfügung und eignet sich bestenfalls für die Realisierung eines flachen Ansatzes.

9.6 Auswertbarkeit, Weiterverarbeitung

Im vorigen Abschnitt wurde bereits die Form der erfaßten Daten bemängelt. Erst nach einer Extraktion der relevanten Daten (durch einen Parser) aus den entsprechenden Log-Files könnten die erfaßten Daten ausgewertet und weiterverarbeitet werden. Dazu existieren leider keine Werkzeuge. Der Grund für die fehlenden Werkzeuge ist der, daß NNStat nicht für das Accounting- oder das Performance-Management, sondern in erster Linie zur Erstellung von Netz-Statistiken konzipiert wurde.

9.7 Integrierbarkeit

NNStat ist nach den in Abschnitt 8.7 aufgestellten Kriterien auf keiner Ebene integrierbar. Zum einen kann es nur in Datennetzen, die auf Ethernet basieren, eingesetzt werden, zum anderen besteht keine Möglichkeit der Korrelation der verteilt erfaßten Daten (außer manuell mit erheblichem Aufwand). Es gibt keine Möglichkeit, die erfaßten Daten in eine MIB zu importieren. Es existiert keine grafische Benutzeroberfläche.

9.8 Dienst- und Benutzerbezogenheit

NNStat analysiert nur Ethernet-, IP-, UDP- und TCP-Header, untersucht also nur die OSI-Schichten 2 bis 4. Daher ist *keine* Benutzeridentifikation möglich. Falls jedoch nur Endsysteme betrachtet werden sollen, bietet NNStat die Möglichkeit, die IP-Adressen in Namen umzuwandeln. Dazu stellt es das Programm `lookupnames` zur Verfügung. Es erzeugt für die angegebenen Log-Files (durch `collect` erzeugt) ein neues Log-File, wobei es nach jeder IP-Adresse den zugehörigen Namen (soweit dieser mit Hilfe des DNS oder der lokalen Namenstabelle zu ermitteln ist) einfügt.

Eine Identifikation von Diensten ist bei NNStat nur über die Portnummern möglich. Aber auch aus einem anderen Grund eignet sich NNStat nicht für eine dienstbezogene Erfassung von Netzverkehr: Die Recorder-Objects von `statspy` können immer nur Paare von Werten aufzeichnen. So können i.a. nur IP-Adressen aufgezeichnet werden. Die Portnummern können nur zur Filterung herangezogen werden. Ausnahme: Betrachtung eines einzelnen Rechners (z.B. besonderer Server) und Aufzeichnung z.B. des Paares `IP.src, TCP.src` (`IP.dst` und `TCP.dst`=Well-Known-Port sind fest).

9.9 Zuverlässigkeit, Schutz vor Manipulation

Die Hosts, auf denen `statspy` bzw. `collect` läuft, müssen selbstverständlich immer eingeschaltet bleiben. Bei einem Ausfall eines solchen Hosts geht natürlich eine Menge an erfaßten bzw. zu erfassenden Daten verloren. Deshalb sollten redundante Erfassungspunkte existieren, d.h. jeder `statspy`-Prozeß sollte in einem Datennetz auf je zwei verschiedenen Hosts des zu erfassenden Segments laufen (was ohne weiteres möglich ist; ein Ausfall muß jedoch erkannt werden und die fehlenden Daten müssen vom zweiten Erfassungspunkt manuell ausgelesen werden, d.h. durch Ausführen eines `collect`-Prozesses unter der Anagabe der Adresse des Hosts, auf dem der zweite `statspy`-Prozeß läuft). Für den Fall eines Stromausfalls, sollten sowohl die `statspy`- als auch die `collect`-Hosts durch eine unterbrechungsfreie Stromversorgung, USV, abgesichert sein.

Was den Schutz vor Manipulation anbelangt, so bietet NNStat zwar die Möglichkeit, den Zugriff auf die `statspy`-Prozesse einzuschränken (durch den Befehl `restrict`), jedoch sind die Prozesse und die Daten an sich nur durch die Sicherheitsvorkehrungen von UNIX (Login- Einschränkungen, Datei-Attribute, Prozeß-Attribute) geschützt. Die Gefahr einer Manipulation (z.B. beenden eines Prozesses (`kill`) oder Verändern bzw. Löschen eines Log-Files) ist somit recht groß.

10 Bewertung von HP EASE

In diesem Kapitel wird HP EASE anhand der im Kapitel 8 aufgestellten Kriterien zur Bewertung von Werkzeugen zur Erfassung von Netzverkehr (für das Abrechnungsmanagement) bewertet.

10.1 Genauigkeit

Wie in Abschnitt 9.1 festgestellt wurde, kann das Werkzeug NNStat zur Überprüfung der Genauigkeit anderer Werkzeuge zur Erfassung von Netzverkehr herangezogen werden. An dieser Stelle soll die Genauigkeit von HP EASE untersucht werden. Dadurch kann auch die Ganauigkeit von statistischen Methoden allgemein eingeschätzt werden.

In Abschnitt 7.2 wurde die Genauigkeit von HP EASE bereits theoretisch untersucht. Nach der praktischen Untersuchung der Genauigkeit wird ein Vergleich mit theoretisch ermittelten Ergebnissen gezogen.

Für die Untersuchung der Genauigkeit von HP EASE wurden drei verschiedene Messungen durchgeführt:

- Messung über zwei Stunden am Lehrstuhl
- Messung über drei Tage am Lehrstuhl
- Messung über drei Tage am LRZ

Alle Messungen wurden auf der Ebene der IP-Adressen durchgeführt. Für die Messungen am Lehrstuhl wurde als Sampler eine HP Traffic Probe am Fan-Out im Rechnerraum verwendet; NNStat lief auf dem Host `sunhegering5`. (Der von diesem Host gesendete Datenverkehr wurde zusätzlich vom Host `sunhegering6` mit Hilfe eines *zweiten* `statspy`-Prozesses ermittelt, da `statspy` nur den *eingehenden* IP-Verkehr erfaßt. Auch die HP Traffic Probe erfaßt ihren ausgehenden IP-Verkehr nicht; daher wurden die durch die Traffic Probe ermittelten Werte für die Gesamtanzahl der Flows, der Frames und der Bytes um die mit NNStat ermittelten Werte für den ausgehenden IP-Verkehr der Traffic Probe korrigiert.

Für die Messung im LRZ wurde als Sampler eine HP Traffic Probe am Gateway zum Wissenschaftsnetz verwendet; die NNStat-Daten stammen vom Host `dfvgate` im LRZ — dieser Host befindet sich ebenfalls am Gateway zum Wissenschaftsnetz und dient ausschließlich der Erstellung von Netzstatistiken mittels NNStat für das LRZ.

Es werden nun zuerst einfach einige Meßergebnisse vorgestellt und dann anschließend erörtert. Es treten bei Messungen am Lehrstuhl meist nur Adressen des Subnets 131.159 auf. Aus Platzgründen werden daher die IP-Adressen wie folgt abgekürzt: 12.24 steht z.B. für 131.159.12.24.

Bei der Messung im LRZ werden nur Subnets betrachtet, da das einzige NNStat-Objekt der Klasse `matrix` die `Rnet.matrix`[77] ist, in der lediglich der Datenverkehr zwischen Subnets aufgezeichnet wird. Hier werden aus Platzgründen nur die Subnet-Adressen angegeben: 129.187 steht somit für 129.187.0.0.

Die ersten beiden Meß-Szenarien untersuchen das übertragene Datenvolumen zwischen zwei Hosts (Lehrstuhl), das dritte Meß-Szenario untersucht (zwangsweise) das Datenvolumen zwischen zwei Subnets. Mit dem Commandline Program `easetraffic` von HP EASE können leider keine Subnets wie bei NNStat herausgefiltert werden. Deshalb mußte *jedes* Paar von Subnetzen separat vom EASE-Server abgefragt werden. Die Abfrage hatte in etwa das folgende Aussehen:

```
/usr/HPEASE/bin/easetraffic -atype=ip,src=^129.187,dst=^132.199
-ptype=ip -i -v 12300195 12010295
```

Dies mußte für beide Richtungen (src–dst, dst–src) durchgeführt werden, damit der Vergleich mit den Werten aus der `Rnet.matrix` gemacht werden konnte.

Da die `Rnet.matrix` maximal 2048 Einträge hat (mehr dazu siehe weiter unten), tritt eine nicht unbeachtliche Anzahl an **orphans** (= Waisen; Frames, die registriert aber nicht erfaßt wurden) auf[78]. Die Summe aus diesen `orphans` und dem `Total Count` ergibt die Gesamtanzahl der Frames; die Gesamtanzahl der Bytes wurde mit Hilfe des Verhältnisses Gesamtanzahl der Frames — `Total Count` hochgerechnet.

Nun aber zu den Meßergebnissen; es werden immer zuerst der gesamte erfaßte Datenverkehr und dann die ersten zwei bzw. drei Flows[79] (geordnet nach der Anzahl der übertragenen Frames) und ein bzw. zwei Flows mit einer deutlich geringeren Abzahl an übertragenen Frames bzw. Bytes betrachtet. (N bedeutet NNStat und E bedeutet HP EASE):

[77] Dieses Objekt gehört der Klasse `matrix-sym-bytes` an, d.h. der Datenverkehr von A nach B und von B nach A wird in einem Eintrag zusammengefaßt; dieser Umstand wurde bei den Ergebnissen von HP EASE durch Summenbildung berücksichtigt.

[78] Siehe dazu das Log-File im Anhang E

[79] Flow bezeichnet im folgenden abkürzend einen Eintrag in einer Flow-Tabelle (z.B. in einem `collect`-Log-File), der angibt, wieviele Frames bzw. Bytes von einer Quell-IP-Adresse zu einer Ziel-IP-Adresse übertragen wurden.

- **Szenario 1**: Messung am Lehrstuhl über zwei Stunden.
  Flows, Frames, Bytes = Anzahl der im Erfassungszeitraum *insgesamt* erfaßten Flows / Frames /Bytes)

E-Flows	N-Flows	E-Frames	N-Frames	E-Bytes	N-Bytes
266	463	846.556	918.568	116.816.605	123.689.842

(Frames, Bytes = Anzahl der im Erfassungszeitraum für den durch IP.source und IP.dest spezifizierten Flow erfaßten Frames / Bytes)

IP.source	IP.dest	E-Frames	N-Frames	E-Bytes	N-Bytes
12.24	12.36	143.103	128.306	11.884.400	12.310.986
12.31	12.45	140.861	132.487	9.606.520	10.178.924
12.36	12.24	104.609	96.132	12.543.600	13.083.745
...	...	...	...	...	...
12.40	12.36	20.518	18.608	1.693.490	1.922.237
...	...	...	...	...	...

- **Szenario 2**: Messung am Lehrstuhl über drei Tage.
  (Flows, Frames, Bytes = Anzahl der im Erfassungszeitraum *insgesamt* erfaßten Flows / Frames /Bytes)

E-Flows	N-Flows	E-Frames	N-Frames	E-Bytes	N-Bytes
2020	2048	15.953.788	16.621.105	5.292.871.415	5.212.911.682

Bemerkung: Es könnten sicher mehr als 2048 Flows registriert werden; NNStat speicherte jedoch nur *maximal 2048* Einträge pro Recorder Object[80].

(Frames, Bytes = Anzahl der im Erfassungszeitraum für den durch IP.source und IP.dest spezifizierten Flow erfaßten Frames / Bytes)

IP.source	IP.dest	E-Frames	N-Frames	E-Bytes	N-Bytes
12.39	12.30	979.933	951.224	1.397.740.000	1.375.511.607
12.36	12.30	952.993	934.620	1.033.360.000	1.030.530.875
.....	...	...	...	...	...
12.31	12.30	886.145	870.375	834.342.000	842.954.040
...	...	...	...	...	...
12.40	12.30	453.622	428.067	285.385.000	275.548.836
...	...	...	...	...	...

[80] Dieser Maximalwert kann beliebig geändert werden: einfach den Wert nach `#define BIN_MAX` in jedem Source File entsprechend abändern und erneut `make` ausführen.

- **Szenario 3**: Messung im LRZ über drei Tage.
  (Flows, Frames, Bytes = Anzahl der im Erfassungszeitraum *insgesamt* erfaßten Flows / Frames /Bytes)

E-Flow	N-Flow	E-Frames	N-Frames	E-Bytes	N-Bytes
67664	2048	66.017.300	65.558.358	16.668.100.000	17.475.841.000

(Frames, Bytes = Anzahl der im Erfassungszeitraum für den durch IP.source und IP.dest spezifizierten Flow erfaßten Frames / Bytes)

IP.source	IP.dest	E-Frames	N-Frames	E-Bytes	N-Bytes
129.187	132.199	3.341.200	3.198.317	1.037.040.100	1.041.184.195
129.187	141.78	2.066.250	2.034.629	360.971.400	394.016.886
129.187	130.83	2.109.879	2.017.622	664.907.000	676.150.715
...	...	...	...	...	...
129.187	141.44	620.146	619.802	199.889.280	209.046.781

Anhand des *relativen* Fehlers wird nun versucht, die Genauigkeit von HP EASE zu beurteilen; dazu sind in der folgenden Tabelle sämtliche absoluten und relativen Fehlerwerte aus den obigen Meßergebnissen der Reihe nach aufgelistet (F = Frames, B = Bytes):

Szenario	abs.Fehler(F)	rel.Fehler(F)	abs.Fehler(B)	rel.Fehler(B)
1	72.012	7,84 %	6.873.237	5,56 %
1	14.797	11,53 %	426.586	3,47 %
1	8.374	6,32 %	572.404	5,62 %
1	8.477	8,82 %	540.145	4,13 %
1	1.910	10,26 %	228.747	11,90 %
2	667.317	4,01 %	79.959.733	1,53 %
2	28.709	3,02 %	22.228.393	1,62 %
2	18.373	1,97 %	2.829.125	0,28 %
2	15.770	1,81 %	8.612.040	1,02 %
2	25.555	5,97 %	9.836.164	3,57 %
3	458.942	0,70 %	807.741.000	4,62 %
3	142.883	4,47 %	4.144.095	0,40 %
3	31.621	1,55 %	33.045.486	8,39 %
3	92.257	4,57 %	11.243.715	1,66 %
3	344	0,06 %	9.157.501	4,38 %

Wie erwartet, werden die von HP EASE errechneten Werte um so genauer, je länger der Erfassungszeitraum ist (Vergleich Szenario 1 mit 2 u. 3). Der relativ große Fehler bei der Gesamtanzahl der Bytes im dritten Szenario könnte von der zuvor erwähnten (notwendigen) Hochrechnung kommen. Bemerkenswert ist, daß der Datenverkehr am Lehrstuhl deutlich homogener zu sein scheint wie am LRZ;

diese Annahme könnte aus der Inhomogenität der relativen Fehler von Szenario 3 und der doch relativ homogenen Werte aus Szenario 2 geschlossen werden. Weitere Unterschiede in der Datenverkehrs-Charakteristik sind zum einen der unterschiedliche Connectivity Factor: Er beträgt am Lehrstuhl in etwa 50, am LRZ hingegen ca. 260 (auf Subnetz-Ebene!). Zum anderen sind die Frames am Lehrstuhl hauptsächlich auf einige wenige Top-Flows verteilt, am LRZ sind die Frames in Großen und Ganzen relativ gleich verteilt (wahrscheinlich weil nur Subnetze betrachtet werden).
Die auf diese Art und Weise ermittelten Ergebnisse sind auch bei NNStat lediglich Näherungswerte, da NNStat nur 2048 Einträge berücksichtigte; man kann jedoch davon ausgehen, daß diese Näherungswerte jedoch hinreichend genau sind, da die nicht berücksichtigten Einträge ein sehr geringen Prozentsatz (zusammen deutlich kleiner als 0,1%) ausmachen.
Bemerkung: Die jeweils erste Seite der Logs des dritten Szenarios findet der interessierte Leser im Anhang. Der vom Host `dfvgate` gesendete Netzverkehr wurde aus zwei Gründen nicht berücksichtigt: 1. kein Zugang zu diesem Host, 2. dieser Host dient ausschließlich der Erfassung des ein- und ausgehenden Netzverkehrs; er produziert selbst – zumindest im Verhältnis zum erfaßten Netzverkehr – kaum zusätzlichen Netzverkehr (Anteil am Gesamtverkehr sehr viel kleiner als 0,1%).
Zu guter Letzt soll noch ein Vergleich zwischen den Werten für den relativen Fehler für die Frames aus Szenario 2 und den entsprechenden theoretisch ermittelten Werten für den mittleren Fehler gezogen werden; dabei wird eine Sicherheitswahrscheinlichkeit von 95 % angenommen[81]. Das Samplingintervall betrug 50. Zudem soll ausgehend vom tatsächlichen relativen Fehler auf die Sicherheitswahrscheinlichkeit rückgeschlossen werden. (Die Formeln und ihre Herleitung zur Berechnung der theoretischen Werte befinden sich im Abschnitt 7.2.2).
Erklärung: $e\%$ ist der prozentuale mittlere Fehler, $r\%$ ist der praktisch ermittelte, prozentuale relative Fehler und $p\%$ ist die zugehörige prozentuale Sicherheitswahscheinlichkeit.

Anz.Frames	Anz.Samples	$e\%$	$r\%$	$p\%$
16.621.105	332.422	0,34%	4,01%	$\approx$ 100%
951.224	19.024	1,42%	3,02%	$\approx$ 100%
934.620	18.692	1,43%	1,97%	99,28%
870.375	17.408	1,49%	1,81%	98,30%
428.067	8.561	2,12%	5,97%	$\approx$ 100%

Ergebnis: Die theoretisch ermittelten Werte liegen (teilweise deutlich) unter den durch Messung ermittelten Werte; die prozentuale Sicherheitswahrscheinlichkeit beträgt bei diesen schlechten Werten für den Fehler erwartungsgemäß durchweg nahezu 100%[82].

[81] HP geht bei seinen Genuigkeitsbetrachtungen von $p = 95\%$ aus.
[82] 100% Sicherheitswahrscheinlichkeit weist auf reale Werte für den Fehler hin.

10.2 Konfigurierbarkeit, Flexibilität

HP EASE analysiert Header-Felder von Protokollen der OSI-Schichten 2 bis 4. Als Schicht-2-Protokoll wird derzeit jedoch nur Ethernet bzw. Ethernet nach IEEE 802.3 unterstützt. Weitere Schicht-2-Protokolle sollen folgen. Auf den OSI-Schichten 3 und 4 können zur Zeit die Header der folgenden Protokolle analysiert werden: SNAP, HPXSAP, IP, ICMP, TCP, UDP, IPX und DEC. Im betrachteten Szenario sind dabei IP, TCP und UDP Gegenstand der Analyse.
Im Zusammenhang mit Accounting sind die folgenden Header-Felder von Interesse: `IP.SourceAddress`, `IP.DestinationAddress`, `TCP.SourcePort`, `TCP.DestinationPort`, `UDP.SourcePort`, `UDP.DestinationPort`, `NumberOf Frames` und `NumberOfBytes`. Diese Information wird in der EASE-Server-Datenbank in Tabellen der folgenden Form (binär) abgespeichert:

```
SrcAddr   SrcPort   DstAddr   DstPort   Bytes   Frames
```

Diese Daten können über die grafische Benutzeroberfläche mit dem History Analyzer oder mit Hilfe des Commandline Programs `easetraffic` beliebig abgefragt werden (Beispiele siehe Anhang). Im Gegensatz zu NNStat liegt hier die gespeicherte Adress-Information als 4-Tupel vor. Neben `easetraffic` gibt es noch weitere Commandline Programs z.B. zur Abfrage der Tabelle der verfügbaren erfaßten Daten (`easedata`), der Tabelle der registrierten Fehler (`easeerrors`) und der Tabelle der Segmentstatistik (z.B. Anzahl der IP-Datagramme pro Segment; `easesegcount`).

Die Sampler kopieren ungefähr jedes n-te Paket vom Netz, packen die gesamte Header-Information (Ethernet, IP und UDP/TCP) in einen SNMP-TRAP ein und versenden diesen an den zuständigen EASE-Server. Es kann hier sowohl das Sampling-Intervall (n = 50, 100, 200, 400, 800) als auch der zuständige EASE-Server konfiguriert werden. Der EASE-Server analysiert die eingehende Header-Information und schreibt die entsprechenden Header-Feld-Werte in die Server-Datenbank. Von dort aus kann die gewonnene Information beliebig abgefragt werden (siehe oben und die Beispiele im Anhang; *ungefilterte Erfassung*, Filterung erst bei der Abfrage der Daten aus der Server-Datenbank). Die Konfiguration eines jeden Servers und für die zugehörigen Sampler steht im File `eased.ini` im Verzeichnis `/usr/HPEASE/config`. Sämtliche Ereignisse, wie das Hochfahren des EASE-Servers oder der Ausfall eines Samplers, werden in das Log-File `easelog` im Verzeichnis `/usr/HPEASE/tmp` geschrieben.

HP EASE ist mit der Einschränkung auf die unterstützen Protokolle frei konfigurierbar. Zur Änderung der Konfiguration eines EASE-Servers muß dieser jedoch beendet werden (`kill`), das File `eased.ini` abgeändert und der EASE-Server erneut gestartet werden (`eased&`). Dazu kann aber das Konfigurations-File zuvor

abgeändert und der Server automatisch in der Nacht (default ist 3 Uhr nachts) mit der neuen Konfiguration gestartet werden.
Dadurch daß HP EASE frei (und auch einfach) konfigurierbar ist, kann es auch leicht an sich ändernde Bedingungen (wie z.B. Einsatz eines weiteren EASE-Servers, weil die Kapazität der vorhandenen EASE-Server aufgrund eines gesteigerten Datenverkehrsaufkommens nicht mehr ausreicht) angepaßt werden und besitzt somit im Sinne des Kriteriums *Konfigurierbarkeit und Flexibilität* aus Kapitel 8 ein großes Maß an Flexibilität.

10.3 Speicherplatzbedarf, Archivierung

HPEASE benötigt mindestens 32 MB Hauptspeicher; Plattenspeicherplatz wird in folgendem Umfang benötigt: 7 MB für den EASE-Server, 6 MB für den Resource Manager, 10 MB für den Traffic Expert und 12 MB für den History Analyzer. Für die Historical Data des EASE-Servers (= die Server-Datenbank) werden in einem Datennetz mit 1.000 Hosts ca. 300 MB an Plattenplatz für einen Erfassungszeitraum von einem Monat benötigt. Dieser Wert ist jedoch vom Connectivity Factor[83] und auch von der Anzahl und Art der verwendeten Protokolle (Protokoll-Zusatz-Information) abhängig.

Bei den Messungen am Lehrstuhl stellte sich jedoch heraus, daß im Mittel ca. nur 250 Hosts mit einem Connectivity Factor von 2 bis 3 erfaßt werden mußten[84]. Der Plattenplatz, der zur Speicherung der Historical Data für einen Monat auf dem Host `hphegering1` (= EASE-Server) benötigt wurde betrug in etwa nur 20 bis 30 MB[85].

Die erfaßten Daten werden auf jedem EASE-Server auf der Festplatte archiviert. Sie liegen dort in binärer Form vor. Sie werden pro Stunde in einem jeweils eigenen Unterverzeichnis `/usr/HPEASE/HistoryAnalyzer/JJJJMM/TT/NN`[86] in mehreren verschiedenen Dateien (z.B. `TMTCP` = TrafficMatrixTCP) gespeichert. Sie können mit dem Commandline Program `easetraffic` einfach in ASCII umgewandelt werden.

[83] Bei 1.000 Hosts nimmt HP einen Connectivity Factor von 10 an

[84] Erfassungszeitraum je eine Stunde.

[85] Die Angabe des Plattenspeicherplatzes ist nur ein Schätzwert, da *nicht* einen Monat lang nur der Datenverkehr des Lehrstuhls aufgezeichnet wurde; immer wieder wurde gar kein Datenverkehr, zusätzlich der Datenverkehr im LRZ oder ausschließlich der Datenverkehr im LRZ (Gateway zum Wissenschaftsnetz) aufgezeichnet; der Richtwert für den Plattenspeicherplatz für die im LRZ erfaßten Daten dürfte in etwa bei 100 bis 150 MB pro Monat liegen

[86] J = Jahr, M = Monat, T = Tag und N eine fortlaufende Nummer für die Stunde

10.4 Netzbelastung, Systembelastung

Die Netzbelastung kann auf einfache Weise mit Hilfe von NNStat ermittelt werden. Dazu läßt man beide Werkzeuge über einen bestimmten (längeren) Zeitraum gleichzeitig den Verkehr in einem bestimmten Segment erfassen. Wichtig ist, daß auch wirklich beide Erfassungspunkte, die HP Traffic Probe und der `statspy`-Host, am selben Segment angeschlossen sind und daß die HP Traffic Probe der *einzige* Sampler für den betrachteten EASE-Server ist (dazu entsprechender Eintrag im File `eased.ini`).

Zur Ermittlung der Netzbelastung wurden zwei Szenarien untersucht: Gleichzeitige Messung 1. über zwei Stunden und 2. über drei Tage im Segment `Lehrstuhl`. Dazu diente die HP Traffic Probe `HpTpHeger1` (`131.159.12.53`) als *einziger* Sampling Point für den EASE-Server `hphegering1` (`131.159.12.39`). Als `statspy`- und `collect`-Host diente der Host `sunhegering5`, der am gleichen Fan-Out wie die Traffic Probe angeschlossen war. Der Netzverkehr, den die Traffic Probe durch das Versenden von SNMP-TRAPs verursacht hat, läßt sich einfach aus dem `collect`-Log-File ablesen. Der gesamte IP-Verkehr läßt sich ebenfalls diesem Log-File entnehmen; aber **Vorsicht**: das ist nicht wirklich der gesamte IP-Verkehr, denn es fehlt der von dem Host `sunhegering5` *gesendete* Netzverkehr, da `statspy`, wie die meisten Werkzeuge zur Erfassung von Netzverkehr, nur den *Receive*-Buffer ausließt, nicht aber den *Send*-Buffer. Dieser fehlende Netzverkehr muß von einem zweiten `statspy`-Host (`sunhegering6`; ebenfalls am gleichen Fan-Out wie die Traffic Probe) ermittelt werden. Die Sampling-Rate betrug 50 (d.h. es wurde im Mittel die Header-Information jedes 50-ten Frames in einem SNMP-TRAP versendet). Die Ergebnisse dieser Messungen waren wie folgt:

Szenario	IP-Verkehr Probe	IP-Verkehr gesamt	Netzbelastung
1	7.230.605 Bytes	123.689.842 Bytes	5,85 %
1	18.608 Frames	918.568 Frames	2,03 %
2	135.541.415 Bytes	5.212.911.682 Bytes	2,60 %
2	345.288 Frames	16.621.105 Frames	2,08 %

Theoretisch müßte die Netzbelastung bei einer Samplingrate von 50 genau 2% betragen, denn die Header-Information, eingepackt in einen SNMP-TRAP, paßt bequem in ein einzelnes IP-Datagramm. In der Tat beträgt die Netzbelastung, wenn man die *Frames* betrachtet, annähernd 2%. Wenn man jedoch die Anzahl der übertragenen *Bytes* betrachtet, erhält man eine unterschiedliche Netzbelastung; das liegt daran, daß die mittlere Länge der IP-Datagramme und damit der sie transportierenden Ethernet-Frames variiert[87].

[87]Am Lehrstuhl z.B. ist die mittlere Länge aller Frames meist (deutlich) geringer als die

Die Systembelastung wurde nicht so eingehend untersucht. Der Host, auf dem der EASE-Server lief (`hphegering1`), mußte (gefühlsmäßig) bei allen Szenarien kaum an Leistung einbüßen. Lediglich jede Stunde, wenn die Server-Datenbank um die aktuellen Tabellen für die vergangene Stunde erweitert wurde, war kurz ein geringer Leistungsabfall, der sich dadurch äußerte, daß sich das Arbeiten mit dem Host `hphegering1` etwas verlangsamte, festzustellen. Auch eine Überprüfung mit dem Kommando `top`, mit dem die Prozesse betrachtet werden können, die den Host am meisten belasten, hat ergeben, daß der EASE-Server zwar immer unter den ersten 10 Prozessen zu finden war, aber kaum an der Spitze stand.

Anders sieht es hingegen bei EASE-Server-Abfragen durch den History Analyzer oder durch `easetraffic` aus, wenn im angegebenen Zeitraum eine wirklich große Zahl an Verkehrsdaten (ab. 10.000 Einträge pro Stunde) erfaßt wurde, wie dies zum Beispiel am LRZ der Fall ist. Für die Abfrage aller IP-Flows, die innerhalb von drei Tagen am LRZ aufgezeichnet wurden, benötigte der EASE-Server (`hphegering1`) ganze 10 Minuten. Die Systembelastung wurde von `top` mit 100% angegeben (Vor und nach dieser Abfrage betrug die Systembelastung laut `top` nur etwa 3 bis 4 Prozent).

10.5 Korrelierbarkeit der erfaßten Daten

In jedem EASE-Server wird in der Regel die Header-Information von *mehreren* Sampling Points analysiert und zusammengeführt; das Zusammenführen der erfaßten Netzdaten findet also nach der Analyse im jeweiligen EASE-Server statt. Weiterhin kann von jedem beliebigen Host aus auf alle verfügbaren EASE-Server zugegriffen werden. Hier kann eine weitere Korrelation auf dem entsprechenden Host stattfinden (z.B. durch mehrfache `easetraffic`-Aufrufe unter Angabe der gewünschten EASE-Server). Auch die HP EASE Clients (z.B. der History Analyzer) korrelieren ihre Daten auf diese Art und Weise.

Wenn also mehrere EASE-Server zum Einsatz kommen, dann werden die Netzdaten *verteilt* erfaßt (mehere Sampler senden ihre erfaßten Daten an einen bestimmten EASE-Server) und können dann auch *verteilt* gesammelt werden (von den verschiedenen EASE-Servern). Die verteilten Daten können von Applikationen einfach zusammengefaßt werden, da sie auf jedem EASE-Server in einer geeigneten, einheiltlichen Form vorliegen. Dadurch ist auch durch eine Erweiterung der Methode (z.B. durch Erstellen von Skripten, mit denen die Daten von mehreren EASE-Servern zusammengefaßt werden können) ein hierarchischer Ansatz ohne größere Probleme realisierbar.

mittlere Länge der Frames, die die SNMP-TRAPs der Traffic Probe transporieren.

10.6 Auswertbarkeit, Weiterverarbeitung

Die Daten stehen in den EASE-Servern in einer geeigneten, einheitlichen Form zur Verfügung und können beliebig abgefragt werden. So können sie ohne Probleme unterschiedlich weiterverabeitet werden. Wenn z.B. ein Abrechnungsmanagement auf der IP-Ebene realsiert werden soll, so können die abrechnungsrelevanten Netzdaten einfach aus allen EASE-Servern abgefragt, zusammengefaßt und für die Rechnungstellung weitergeleitet werden. Auch lassen sich auf einfache Weise, ggf. durch Summenbildung, Nutzerprofile erstellen.

Auch die HP EASE Clients verarbeiten ein und dieselben Daten aus der jeweiligen Server-Datenbank unterschiedlichst weiter. Die jeweils unter einem besonderen Aspekt abgefragten Daten werden in den HP EASE Clients graphisch anspruchsvoll präsentiert. Die Clients, auch wenn sie noch so schöne Grafiken produzieren, sind für Accounting-Zwecke jedoch nicht zu gebrauchen; lediglich wenn man (mal kurz) einen Blick auf Nutzungs- Profile werfen will, so sind die Clients ein angenehmes Mittel dafür.

10.7 Integrierbarkeit

HP EASE ist auf zwei der in Kapitel 8 vorgestellten Ebenen integriert: auf der Ebene der verteilten Erfassung und Sammlung und auf der Ebene der Benutzeroberfläche. HP EASE ist jedoch nicht in eine NM-Plattform integrierbar.

Die Daten liegen auf jedem EASE-Server in einer einheitlichen Form vor. Die Sampling Devices werden in Zukunft auch andere Netztechnologien als Ethernet unterstützen (geplant ist zunächst FDDI). Man hat also somit eine Abstraktion von der Netzinfrastruktur. Auch ist so ein hierarchischer Ansatz realisierbar. Die EASE Clients können zwar auch unabhängig vom Netzmanagementsystem HP OpenView laufen (auch unter derselben grafischen Benutzeroberfläche), sind aber von vorherein bereits mit in die Benutzeroberberfläche von HP OpenView integriert (eigene Menüpunkte zum Starten der Clients).

Obwohl HP EASE in die *Benutzeroberfläche* von HP OpenView integriert ist, so ist sie jedoch *nicht* in die Netzmanagement- Plattform HP OpenView integriert. Die Daten der jeweiligen EASE-Server-Datenbank stehen nicht in der Form einer MIB zur Verfügung. Zur Zeit bestehen jedoch bei HP Bestrebungen, eine EASE-MIB als Bestandteil der RMON MIB zu realisieren.

10.8 Dienst- und Benutzerbezogenheit

Auch HP EASE analysiert nur die OSI-Schichten 2 bis 4; somit ist ohne weiteres *keine* Benutzeridentifikation möglich. Eine Umwandlung von IP-Adressen in Namen und eine Umwandlung von Well-Known-Ports in Dienstbezeichnungen (wie `ftp`) ist von vornherein vorgesehen; wenn diese Umwandlung nicht gewünscht wird, so kann sie unterdrückt werden (z.B. mit der Option `-i` des Commandline Programs `easetraffic`). Zudem ist es auch möglich eigene Well-Known Ports zu definieren.

Eine Identifikation von Diensten kann nur über die Portnummer erfolgen. So kann aber nur der Datenverkehr, der über die Well-Known-Ports abgewickelt wird, erfaßt werden. Ein Großteil des Datenverkehrs kann somit aber nicht zugeordnet werden; dieser Anteil kann bestenfalls pauschal anteilsmäßig zugeordnet werden (anhand der Nutzungsintensität der Well-Known-Ports).

10.9 Zuverlässigkeit, Schutz vor Manipulation

Die HP Traffic Probes erwiesen sich als äußerst ausfallsicher. Im gesamten Zeitraum (ca. 3 Monate) in dem Messungen durchgeführt wurden ist keine der beiden eingesetzten Traffic Probes ausgefallen. Wenn ein Host, auf dem ein EASE-Server läuft, bootet, so wird auch der EASE-Server beendet[88]. Damit er beim Booten wieder automatisch gestartet wird, muß dies in der Datei `/etc/netlinkrc` (HP-UX) entsprechend (siehe Handbuch) eingetragen werden.

Da die HP Traffic Probes, so wie auch die anderen Sampling Devices Hardware sind und keine direkte Ein- und Ausgabemöglichkeit (auch ein `rlogin` ist nicht möglich) besitzen, sind sie gut gegen Manipulation geschützt (Ausnahme: Drücken des Reset-Knopfes; dadurch gehen aber nur ein paar Samples verloren, weil die Probe ja schnell wieder aktiv ist). Eine HP Traffic Probe wird einmalig über das BOOTP konfiguriert (v.a. Zuweisung der IP-Adresse), jede weitere Konfiguration geschieht vom jeweiligen EASE-Server aus.

Der EASE-Server ist genauso geschützt wie (fast) jeder andere UNIX-Prozeß auch: durch die Sicherheitsvorkehrungen von UNIX (Login-Einschränkungen, Datei-Attribute, Prozeß-Attribute (der `eased` läuft unter `root`)). Auch der Zugriff auf die erfaßten Daten kann nur durch ebendiese Sicherheitsmechanismen eingeschränkt bzw. verhindert werden.

[88]Der EASE-Server beendet und startet sich normalerweise selbst jede Nacht (3 Uhr), damit der von ihm belegte Hauptspeicherplatz wieder frei wird.

11 Zusammenfassung und Ausblick

Zur Realisierung eines Abrechnungsmanagements in einem Datennetz werden Verbrauchsdaten aus den Bereichen Netz, System (= Endsysteme) und Anwendung benötigt. Welche Daten benötigt werden und wie genau diese Daten erfaßt werden müssen, wird indirekt durch die Abrechnungspolitik vorgegeben. Die Abrechnungspolitik wiederum ist in erster Linie von der Rolle der IV-Abteilung bzw. von der Rolle des gesamten Unternehmens abhängig. In Abschnitt 1.6 wurden unabhängig von der jeweils vorliegenden Abrechnungspolitik Anforderungen an ein Abrechnungsmanagement aufgestellt.

Diese Arbeit befaßte sich hauptsächlich mit der Erfassung von Verbrauchsdaten aus dem Bereich Netz. Diese Erfassung wird im allgemeinen durch die Heterogenität des Datennetzes und durch die Verteiltheit der Anwendungen erschwert. Damit der Rahmen der Diplomarbeit nicht gesprengt wurde, wurden nur TCP/IP basierte Datennetze untersucht; wurden Endsysteme betrachtet, so handelte es sich hierbei um UNIX-Workstations (HP-UX und SunOS Systeme). Daher wurde im Kapitel 2 die TCP/IP-Protokollfamile und als darunterliegendes Protokoll das Ethernet vorgestellt.

Es OSI-konformes Accounting Modell bildete den modellhaften Rahmen für diese Diplomarbeit. Dazu wurden im Kapitel 3 bestehende Accounting Modelle besprochen. Da weder das OSI Accounting Modell noch das Internet Accounting Modell für den betrachteten Teilbereich speziell genug gefaßt sind, wurde im Kapitel 5, Abschitt 3, und im Kapitel 6 ein vom Autor am Lehrstuhl entwickeltes, OSI-konformes Objekt-Modell vorgestellt, das diesen Mangel beseitigen sollte.

Accounting ist demnach ein Prozeß, der aus drei Subprozessen besteht. Der Usage Metering Process erzeugt als Folge von Accountable Events benutzerbezogen die Usage Records. Der Charging Process, der aus den Usage Records die Service Transaction Records erzeugt, besteht wiederum aus drei Subprozessen: Der Integration Process ist für die Sammlung der Usage Records zuständig. Der Service Projection Process faßt die Usage Records dienstbezogen zu Service Transaction Records zusammen. Der Pricing Process schließlich fügt den Service Transaction Records noch die nötige Kosteninformation hinzu. Diese Service Transaction Records verwendet der Billing Process dann zur Rechnungstellung.

Die Verbrauchsdaten, die durch den Usage Metering Process erfaßt werden, können praktisch durch Monitoring gewonnen werden. Aus diesem Grund wurde in Kapitel 4 das Monitoring an sich und eine spezielle Form des Monitorings, das sogenannte Remote Monitoring und seine Realisierung in der RMON MIB, besprochen. Das Werkzeug NetMetrix, das im Kapitel 7, Abschnitt 3, konzeptionell vorgestellt wurde, arbeitet teilweise RMON-basiert und global betrachtet, durch die proprietäre Erweiterung der RMON-MIB, RMON-orientiert. Ein Monitoring kann als Datenlieferant für mehere Funktionsbereiche des Netzmanagements dienen; dabei benötigt das Abrechnungsmanagement jedoch die bei weitem detailliertesten Daten. Somit kann ein hinreichend detailliertes, einheitliches, das gesamte Datennetz umfassendes Monitoring als Informationsplattform für die verschiedensten Netzmanagement-Applikationen dienen.

Durch das Monitoring werden im Bereich Netz in erster Linie Flows aufgezeichnet. In dieser Arbeit wurden drei Arten von Flows betrachtet: IP-Flows, UDP-Flows und TCP-Flows. Praktisch, d.h. bei den Messungen, wurden nur IP-Flows betrachtet. Der Hauptgrund dafür ist, daß NNStat nur einzelne Attributwerte und Paare von Attributwerten aufzeichnen kann, jedoch keine 4-Tupel[89]. Zur Spezifikation von UDP- bzw. TCP-Flows sind jedoch i.a. vier Attribute nötig (2 IP-Adressen und 2 Ports). Attribute zur Spezifikation wurden im Kapitel 5, Abschnitt 3 im Rahmen des OSI-konformen Objekt-Modells besprochen. Die Flows werden als Flow-Einträge, die Teil einer Flow-Tabelle sind, gespeichert. Diese Flow-Einträge enthalten neben den Attributwerten, die den jeweiligen Flow spezifizieren, noch einen Zähler für die Frames und einen Zähler für die Bytes, die dem jeweiligen Flow angehören. Ein Flow-Eintrag enthält im betrachteten Szenario jedoch keinerlei benutzer- oder dienstbezogene Information (außer den Portnummern). Zur Gewinnung dieser Information allein aus dem Bereich Netz müßten zusätzlich die OSI-Schichten 5 bis 7 analysiert werden. Dies ist eine wichtige Anforderung an (zukünftige) Werkzeuge zur Erfassung von Netzverkehr.

Wie eingangs bereits erwähnt, läßt sich die Gesamtheit der abrechnungsrelevanten Daten in drei verschiedene Arten von Daten einteilen: Die Netzdaten, die Systemdaten (aus den einzelnen Endsystemen) und die Anwendungsdaten (abrechnungsrelevante Daten, die die Anwendungen von sich aus zur Verfügung stellen). Jede dieser drei Arten von abrechnungsrelevanten Daten wurde klassifiziert in Accountable Units, wie Plattenplatz oder CPU-Zeit, in Abbildungsinformation, wie Socket oder Port, und in abrechnungsrelevante Zusatzinformation, wie Benutzername oder Dienstbezeichnung.

Die Methoden zur Erfassung dieser abrechnungsrelevanten Daten wurden grob in exakte und statistische Methoden eingeteilt. Vorraussetzung für die Analyse dieser Methoden war die Untersuchung, wann ein hierarchischer oder ein flacher Ansatz geeigneter erscheint, auf welcher Ebene (z.B. OSI-Schicht) erfaßt werden soll, wo die Erfassung stattfinden soll, ob Probes oder Koppelelemente dazu benutzt werden sollen, wo eine Benutzer- und wo eine Anwendungsidentifikation

[89] Ein weiterer Grund ist, daß man auf IP-Ebene meist schon mehrere tausend Einträge in die Flow-Tabelle erhält. Werden nun auch noch die Ports erfaßt, so vergrößert sich diese Anzahl um den Faktor `n`, wobei `n` die mittlere Anzahl benutzer Ports zwischen zwei Hosts ist. Ein realsitischer Wert für `n` wäre am Lehrstuhl etwa 30 bis 60.

möglich ist, wie Prozeßketten behandelt werden können und vieles andere mehr. Diese Themen wurden so weit wie möglich in Kapitel 5 behandelt.

Im Kapitel 7 wurden zunächst die zur Verfügung stehenden Werkzeuge vorgestellt. Entsprechend ihrer zugrundeliegenden Methode handelte es sich hierbei um das exakte Werkzeug NNStat und um das statistische Werkzeug HP EASE. Als drittes Werkzeug wurde das ebenfalls exakte, RMON-orientierte Werkzeug HP NetMetrix besprochen, das aber aus den im Abschnitt 3 vorgetragenen Gründen anschließend nicht bewertet wurde.

Um die Werkzeuge bewerten zu können, wurden in Kapitel 8, ausgehend von den Anforderungen an ein Abrechnungsmanagement aus Kapitel 1 und den Fragestellungen bei der Nutzungserfassung aus Kapitel 5, insgesamt neun Kriterien aufgestellt. Diese Kriterien umfaßten Themen wie die Genauigkeit der Erfassung, die Konfigurierbarkeit der Werkzeuge, die zusätzliche Netz- und Systembelastung und das Zusammenführen der erfaßten Daten. Durch die Aufstellung dieser Kriterien wurde zum einen implizit eine weitere Analyse von Methoden zur Erfassung von Netzverkehr für das Abrechnungsmanagement in verteilten, heterogenen Umgebungen unternommen. Zum anderen konnten die Werkzeuge NNStat und HP EASE anhand dieser Kriterien in den Kapiteln 9 und 10 exemplarisch bewertet werden. Weitere, praktisch bedeutsame Informationen finden sich im Anhang.

Das zentrale Ergebnis dieser Diplomarbeit ist, daß sich die betrachteten Werkzeuge zur Realisierung eines benutzer- und dienstbezogenen Abrechnungsmanagements, so wie durch das OSI Accounting Modell gefordert, nur sehr bedingt eignen. Sie können in eigeschränktem Maße jedoch sehr wohl zu Accountingzwecken eingesetzt werden. Interessant ist hierbei vor allem der Einsatz von Werkzeugen, die auf einer statistischen Methode beruhen; wenn die Erfassung des Netzverkehrs nicht so hundertprozentig genau sein soll, so zeichnen sich diese Werkzeuge vor allem durch eine i.d.R. deutlich geringere Netz- und Systembelastung aus. Ich sehe den Einsatz dieser Werkzeuge und auch den Einsatz von Werkzeugen wie NNStat jedoch nur als Übergangslösung an, bis geeignete Werkzeuge existieren.

Dadurch daß das Interesse an einem detaillierten Abrechnungsmanagement in den Unternehmen, wie auch einige Gespräche mit Unternehmensvertretern deutlich gezeigt haben, immer größer wird und die technischen Vorraussetzungen für eine detaillierte Erfassung und Sammlung von abrechnungsrelevanten Daten jetzt eigentlich (zu einem vernünftigen Preis) vorhanden sind, ist es nur noch eine Frage der Zeit, bis geeignete Erfassungswerkzeuge als Hardware oder Software auf dem Markt zu finden sein werden. Besondere Bedeutung messe ich in Zukunft einer deutlich verbesserten Accountingfunktionalität von Koppelelementen wie Routern oder Bridges zu.

Der Einsatz dieser Werkzeuge für die Realisierung eines Abrechnungsmanagements in den Unternehmen könnte jedoch, wie viele verantwortliche Unternehmensvertreter befürchten, am Einspruch des Betriebsrats scheitern. Dieser Einspruch ist zum einen gerechtfertigt, da eine detaillierte Erfassung im Datennetz auch eine hervorragende Kontrollmöglichkeit über den einzelnen Benutzer, sprich Arbeitnehmer, bietet. Zum anderen besteht eine gleichwertige Kontrollmöglichkeit in der Mainframe-orientierten Datenverarbeitung in den Rechenzentren schon lange Zeit, jedoch werden die Daten dort als hinreichend geschützt betrachtet (geschlossenes System).

Inwieweit eine zukünftige Accountingfunktionalität der Koppelelemente ausreichend ist und wie geeignet ein zukünftiges Erfassungswerkzeug sein wird, kann vielleicht, zumindest zu einem gewissen Teil, anhand der in dieser Arbeit aufgestellten Kriterien mit entschieden werden.

Literatur

[1] F. Arnold, ISDN: Viele Kommunikationsdienste in einem System, R. Müller-Verlag, ISBN 3-481-30621-0, 1987.

[2] U. Arnold, Heterogene Netzwerke, Franzis, ISBN 3-7723-4421-6, 1992.

[3] AT&T, UNIX System V Release 4 Programmer's Guide: Networking Interfaces, Prentice Hall, ISBN 0-13-947078-6, 1990.

[4] C. Brooks, Internet Accounting Management Information Base Rev.2.2, 1993.

[5] Ch. Brown, UNIX Distributed Programming, Prentice Hall, ISBN 0-13-075896-5, 1994.

[6] N. Brownlee, Network Traffic Meter & NeTraMet Manager / Collector, Introductory Documentation, 1993.

[7] D. M. Chiu, R. Sudama, Network Monitoring Explained, Ellis Horwood, ISBN 0-13-614710-0, 1992.

[8] D. Comer, Internetworking with TCP/IP (Volume I - III), Prentice-Hall International, ISBN 0-13-474321-0, 0-13-465378-5, 0-13-020272-X, 1993.

[9] DATAPRO, Accounting Management: Issues, Techniques and Tools, Datapro International Network Management, 6160 Management Issues, McGraw-Hill, 1993.

[10] DATAPRO, Outsourcing IT Services, Enterprise Systems, 1412 Management Issues, McGraw-Hill, 1993.

[11] B.H. Hunter, K. Bradford Hunter, UNIX Networks - An Overview for System Administrators, Prentice Hall, ISBN 0-13-089087-1, 1994.

[12] ISO / IEC DIS 10164-10.2, Systems Management: Usage Metering Function, 1993.

[13] F.-J. Kauffels, Netzwerk-Management: Probleme, Standards, Strategien, Datacom, ISBN 3-89238-047-3, 1992.

[14] M. Mansouri-Samani, M. Sloman, Monitoring Distributed Systems (A Survey), 1993.

[15] C. Mills, D. Hirsch, G. Ruth, Internet Accounting: Background, RFC 1272, 1991.

[16] M. T. Rose, Einführung in die Verwaltung von TCP/IP-Netzen, Hanser, ISBN 3-446-16444-8, 1993.

[17] W. M. Schrier, Capacity Forecasting and Chargeback for a General Purpose Backbone Internetwork.

[18] S. Schwerdtner, OSI Network Management: Standardkonformität, Problemanalyse und Testszenarien, Diplomarbeit, 1993.

[19] M. Sloman, J. Moffett, Managing Distributed Systems, Domino: A1/IC/1.2, 1990.

[20] M. Sloman et al., Domain Management and Accounting in an International Cellular Network, Elsevier Science Publishers B.V., 1993.

[21] M. Sloman, Network and Distributed Systems Management, Addison-Wesley, ISBN 0-201-62745-0, 1994.

[22] K. Terplan, Kommunikationsnetze, Hanser, ISBN 3-446-15303-9, 1989.

[23] K. Terplan, Kommunikationsnetze: Planung, Organisation, Hanser, ISBN 3-446-15303-9, 1989.

[24] K. Terplan, Ch. Voigt, LAN-Management: Funktionen, Instrumente, Perspektiven, Datacom, ISBN 3-89238-042-2, 1993.

[25] S. Waldbusser, Remote Network Monitoring Management Information Base, RFC 1271, 1991.

[26] S. Waldbusser, Token Ring Extensions to the Remote Network Monitoring MIB, RFC 1513, Update für RFC 1271, 1993.

[27] K. Washburn, J.T. Evans, TCP/IP - Running a Successful Network, Addison-Wesley, ISBN 0-201-62765-5, 1993.

[28] P. Wenzel, Einführende Grundlagen zur Kommunikation offener Systeme, Vieweg, ISBN 3-528-24369-4, 1993.

Abbildungsverzeichnis

A Wichtige Well-Known Ports

Port	Abkürzung	Beschreibung
20	`ftp-data`	File Transfer Protocol (Data)
21	`ftp`	File Transfer Protocol (Control)
23	`telnet`	Telnet
25	`smtp`	Simple Mail Transfer Protocol
37	`time`	Time
42	`nameserver`	Host Name Server
43	`whois`	Who Is
49	`login`	Login Host Protocol
53	`domains`	Domain Name Server
67	`bootps`	BootStrap Protocol (Server)
68	`bootpc`	BootStrap Protocol (Client)
69	`tftp`	Trivial File Transfer Protocol
79	`finger`	Finger
103	`X.400`	X.400
104	`X.400-SND`	X.400-SND
111	`sunrpc`	Sun Remote Procedure Call
119	`nntp`	Network News Transfer Protocol
123	`ntp`	Network Time Protocol
137	`netbios-ns`	NetBIOS Name Service
138	`netbios-dgm`	NetBIOS Datagram Service
139	`netbios-ssn`	NetBIOS Session Service
144	`news`	News
161	`snmp`	Simple Network Management Protocol
162	`snmp-trap`	SNMP Trap
163	`cmip-manage`	CMIP/TCP Manager
164	`cmip-agent`	CMIP/TCP Agent
512	`exec`	Exec
513	`rlogin`	RWho, RLogin
514	`shell`	RShell
515	`printer`	RPrinter
6000	`X11`	X-Server

B Was bei den Messungen zu beachten ist

Da die Erfassungspunkte, sowohl die HP Traffic Probe, als auch `statspy` den Verkehr, der von ihrem Host aus *gesendet* wird *nicht* erfassen, muß dieser fehlende Verkehr, der die Meßergebnisse schwerwiegend verfälschen kann, separat erfaßt werden.

Der von der Traffic Probe ausgehende Verkehr kann entweder mit einer *zweiten* Traffic Probe statistisch oder exakt mit `statspy` erfaßt werden.

Der vom `statspy`-Host ausgehende Verkehr `muß` mit einem *zweiten* `statspy`-Prozeß, der auf einem *anderen* Host läuft, erfaßt werden. Die entsprechenden Konfigurationsfiles könnten dann in etwa so aussehen:

```
### statspy-Host 1:

attach
{
  record IP.srchost IP.dsthost
    in lehrstuhl matrix-all-bytes;
}

### statspy-Host 2:

attach
{
  if IP.srchost is eqf( <statspy-Host 1> )
    {
      record IP.srchost in korrektur freq-all-bytes;
    }
}
```

C Beispiel-Konfigurationen und Beispiel-Logs von NNStat

Der Host `sunhegering5` erfaßt den gesamten IP-Verkehr am Lehrstuhl. Der Host `sunhegering6` erfaßt dazu den restlichen IP-Verkehr (= der vom Host `sunhegering5` *gesendete* IP-Verkehr).

Konfiguration vom Host `sunhegering5`:

```
## Konfigurationsdatei fuer statspy:
## Name:    parm.lehrstuhl (statspy-Host #1)
## Aufgabe: Erfasse den gesamten IP-Verkehr, den der
##          SAA mitbekommt.

attach
{
  record IP.srchost IP.dsthost
    in lehrstuhl matrix-all-bytes;
}
```

Anfang des dazugehörigen Log-Files vom Host `sunhegering5`:

```
Log created on Fri Jan 20 12:58:49 1995, for host 131.159.12.30.
    Sample interval = 60 minutes; checkpoint interval = 60 minutes.
    Clear interval = 1000 minutes.
    Object name = 'lehrstuhl'.

OBJECT: lehrstuhl  Class= matrix-all-bytes [Created: 12:21:34 01-20-95]
  ReadTime: 12:58:48 01-20-95, ClearTime: 12:21:34 01-20-95 (@-2234sec)
  Total Count= 96353 (+0 orphans)
  Total Bytes= 23648523B    #bins = 309
[131.159.12.36 : 131.159.12.30]= 20095 &10610457B  (20.9%) @-0sec
[131.159.12.6 : 131.159.12.30]= 7620 &912316B      ( 7.9%) @-9sec
[131.159.12.31 : 131.159.24.22]= 7279 &503442B     ( 7.6%) @-0sec
[131.159.12.40 : 131.159.12.30]= 7081 &2033274B    ( 7.3%) @-46sec
[131.159.12.45 : 131.159.12.31]= 6324 &431380B     ( 6.6%) @-1sec
[131.159.12.31 : 131.159.12.45]= 5926 &1004224B    ( 6.2%) @-1sec
[131.159.24.22 : 131.159.12.31]= 5218 &1047585B    ( 5.4%) @-0sec
[131.159.12.24 : 131.159.12.30]= 3393 &448128B     ( 3.5%) @-4sec
```

Konfiguration vom Host `sunhegering6`:

```
## Konfigurationsdatei fuer statspy:
```

```
## Name:    parm.korrektur (statspy-Host #2)
## Aufgabe: Erfasse den vom statspy-Host #1
##          gesendeten IP-Verkehr.

## hier: statspy-Host #1  sunhegering5
##                        (Erfassung Datenverkehr Lehrstuhl)
##       statspy-Host #2  sunhegering6
##                        (Erfassung restlicher Datenverkehr)

attach
{
  if IP.srchost is eqf(131.159.12.30)
    {
      record IP.srchost
        in korrektur freq-all-bytes;
    }
}
```

Dazugehöriges Log-File vom Host `sunhegering6`:

```
Log created on Fri Jan 20 13:07:28 1995, for host 131.159.12.36.
    Object name = 'korrektur'.

OBJECT: korrektur  Class= freq-all-bytes  [Created: 12:22:17 01-20-95]
  ReadTime: 13:07:27 01-20-95, ClearTime: 12:22:17 01-20-95 (@-2710sec)
  Total Count= 51551 (+0 orphans)
  Total Bytes= 7545171B    #bins = 1
[131.159.12.30]= 51551 &7545171B   (100.0%) @-0sec
```

D Isolierte Betrachtung eines Dienstes

Im folgenden wird ein einzelner Dienst betrachtet. FTP eignet sich dazu besonders gut, da FTP (in der voliegenden Implementierung) ausschließlich über TCP kommuniziert (TCP Ports 20 und 21). Dazu wurde zwischen den Hosts `hphegering1` (FTP-Client) und `sunhegering5` (FTP-Server) eine FTP-Sitzung durchgeführt. Diese konnte isoliert betrachtet werden, da während dieser Siztung sonst *kein* anderer FTP-Verkehr zwischen diesen beiden Hosts auftrat.

NNStat lief auf dem Host `sunhegering6`; der EASE-Server wie gewohnt auf dem Host `hphegering1` (Erfassungspunkt für den EASE-Server war die HP Traffic Probe im Rechnerraum).

Das NNStat Objekt `to30` zeichnete den gesamten TCP-Verkehr vom Host `hphegering1` zum Host `sunhegering5` auf; das Objekt `to39` den Verkehr in der Gegenrichtung. Eine Abfrage der EASE-Server-Datenbank hatte zum Beispiel das folgende Aussehen:

```
/usr/HPEASE/bin/easetraffic -a type=ip -p type=tcp,src=20 -f 20 -i
-d 131.159.12.39 -v 1025011995
```

Die FTP-Sitzung bestand aus fünf Aktionen:

- Initiieren der FTP-Sitzung
- Ausführen des Kommandos `ls`
- Ausführen des Kommandos `get mbox`
- Ausführen des Kommandos `put mbox`
- Beenden der FTP-Sitzung durch `quit`

Es folgt der Ablauf der FTP-Sitzung; nach jeder Aktion wird das jeweilige Meßergebnis von NNStat bzw. HP EASE aufgelistet:

```
:> ftp sunhegering5
Connected to sunhegering5.informatik.tu-muenchen.de.
220 sunhegering5 FTP server (SunOS 4.1) ready.
Name (sunhegering5:schiffel):
331 Password required for schiffel.
Password: ******
230 User schiffel logged in.
ftp>
```

Meßergebnis von NNStat (über den TCP-Port 513 lief das rlogin zum Host `hphegering1`):

```
Acquired 16646 packets in 99 secs => 168(avg) 522(max) 50(inst) /sec

OBJECT: to30  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:48:32 01-25-95, ClearTime: 10:46:53 01-25-95 (@-99sec)
  Total Count= 41 (+0 orphans)
  Total Bytes= 1922B  #bins= 2  Maxchain = 1  SortMoves = 0
[513 : 1022]= 33 &1564B    (80.5%) @-17sec
[1753 : 21]= 8 &358B   (19.5%) @-17sec

OBJECT: to39  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:48:32 01-25-95, ClearTime: 10:46:53 01-25-95 (@-99sec)
  Total Count= 60 (+0 orphans)
  Total Bytes= 2586B  #bins= 2  Maxchain = 1  SortMoves = 0
[1022 : 513]= 53 &2150B    (88.3%) @-17sec
[21 : 1753]= 7 &436B   (11.7%) @-17sec
```

Das Meßergebnis von HP EASE wird an dieser Stelle noch nicht betrachtet, da die EASE-Server-Datenbank nur stundenweise (nach Ablauf einer Stunde) abgefragt werden kann und so im nachhinein nicht mehr gesagt werden kann, bei welcher Aktion welcher Datenverkehr auftrat (Summenbildung!); bei der Betrachtung des Well-Known-Ports 20 ist dies jedoch möglich, da auf der Seite des Clients für jede Aktion immer wieder ein *anderer* Port verwendet wird.

```
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (131.159.12.39,1756) (0 bytes).
total 624
.....
```

NNStat:

```
OBJECT: to30  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:49:18 01-25-95, ClearTime: 10:46:53 01-25-95 (@-145sec)
  Total Count= 72 (+0 orphans)
  Total Bytes= 8739B  #bins= 3  Maxchain = 1  SortMoves = 0
[513 : 1022]= 56 &8025B    (77.8%) @-19sec
[1753 : 21]= 12 &550B  (16.7%) @-20sec
[1756 : 20]= 4 &164B   ( 5.6%) @-20sec

OBJECT: to39  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:49:18 01-25-95, ClearTime: 10:46:53 01-25-95 (@-145sec)
  Total Count= 95 (+0 orphans)
  Total Bytes= 9016B  #bins= 3  Maxchain = 1  SortMoves = 0
```

```
[1022 : 513]= 78 &3161B    (82.1%) @-19sec
[21 : 1753]= 10 &687B  (10.5%) @-20sec
[20 : 1756]= 7 &5168B  ( 7.4%) @-20sec
```

HP EASE:

```
    131.159.12.30, 20, 131.159.12.39, 1756, 301.6, 50.2667
```

HP EASE hat nur das TCP-Port-Paar 20,1756 mitbekommen und eine haarsträubende Hochrechnung gemacht. Das liegt daran, daß HP EASE (auch bei einer Sampling-Rate von 50) wahrscheinlich lediglich *ein* Frame dieses Flows erfassen konnte.

```
ftp> get mbox
200 PORT command successful.
150 ASCII data connection for mbox (131.159.12.39,1758) (88127 bytes).
226 ASCII Transfer complete.
16577 bytes received in 0.79 seconds (20.52 Kbytes/s)
```

NNStat:

```
OBJECT: to30  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:50:33 01-25-95, ClearTime: 10:46:53 01-25-95 (@-220sec)
  Total Count= 90 (+0 orphans)
  Total Bytes= 9704B  #bins= 4  Maxchain = 1  SortMoves = 1
[513 : 1022]= 66 &8629B    (73.3%) @-38sec
[1753 : 21]= 15 &707B  (16.7%) @-39sec
[1758 : 20]= 5 &204B   ( 5.6%) @-39sec
[1756 : 20]= 4 &164B   ( 4.4%) @-95sec

OBJECT: to39  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:50:33 01-25-95, ClearTime: 10:46:53 01-25-95 (@-220sec)
  Total Count= 130 (+0 orphans)
  Total Bytes= 27211B  #bins= 4  Maxchain = 1  SortMoves = 2
[1022 : 513]= 95 &3851B    (73.1%) @-37sec
[20 : 1758]= 15 &17181B    (11.5%) @-39sec
[21 : 1753]= 13 &1011B (10.0%) @-39sec
[20 : 1756]= 7 &5168B  ( 5.4%) @-95sec
```

HP EASE:

HP EASE hat das TCP-Port-Paar 20, 1758 bzw. 1758, 20 NICHT mitbekommen!

```
ftp> put mbox
200 PORT command successful.
150 ASCII data connection for mbox (131.159.12.39,1760).
```

```
226 ASCII Transfer complete.
16577 bytes sent in 0.02 seconds (882.98 Kbytes/s)
```

NNStat:

```
OBJECT: to30  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:51:00 01-25-95, ClearTime: 10:46:53 01-25-95 (@-247sec)
  Total Count= 122 (+0 orphans)
  Total Bytes= 28304B  #bins= 5  Maxchain = 1  SortMoves = 3
[513 : 1022]= 78 &9294B    (63.9%) @-12sec
[1753 : 21]= 19 &904B  (15.6%) @-12sec
[1760 : 20]= 16 &17738B    (13.1%) @-12sec
[1758 : 20]= 5 &204B   ( 4.1%) @-66sec
[1756 : 20]= 4 &164B   ( 3.3%) @-122sec

OBJECT: to39  Class= matrix-all-bytes  [Created: 10:46:53 01-25-95]
  ReadTime: 10:51:00 01-25-95, ClearTime: 10:46:53 01-25-95 (@-247sec)
  Total Count= 165 (+0 orphans)
  Total Bytes= 28742B  #bins= 5  Maxchain = 1  SortMoves = 4
[1022 : 513]= 116 &4700B   (70.3%) @-12sec
[21 : 1753]= 16 &1249B     ( 9.7%) @-12sec
[20 : 1758]= 15 &17181B    ( 9.1%) @-66sec
[20 : 1760]= 11 &444B      ( 6.7%) @-12sec
[20 : 1756]= 7 &5168B      ( 4.2%) @-122sec
```

HP EASE:

```
    131.159.12.30, 20, 131.159.12.39, 1756, 301.6, 50.2667
    131.159.12.30, 20, 131.159.12.39, 1760, 301.6, 50.2667
    131.159.12.39, 1760, 131.159.12.30, 20, 73389.4, 50.2667
    131.159.12.39, 1753, 131.159.12.30, 21, 1306.93, 50.2667
```

Auch hier ist HP EASE – wie auch zu erwarten war – sehr ungenau; Man kann davon ausgehen, daß für jeden der vier obigen Flows jeweils nur *ein* Frame erfaßt wurde. Die unterschiedliche Hochrechnung für die Anzahl der Bytes liegt an der tatsächlichen Anzahl der Bytes im jeweils erfaßten Frame.

```
ftp> quit
221 Goodbye.
```

E Log-Files einer Messung am LRZ

Im folgenden sind die Log-Files des dritten Meß-Szenarios aus Abschnitt 10.1 aufgelistet. Aus Platzgründen wurde jeweils nur der Kopf des jeweiligen Log-Files (mit den Flow-Einträgen, die die Flows mit den meisten PDUs repräsentieren) hier eingebunden.

Erfaßt wurde vom 30.01.1995 12:00 Uhr bis zum 01.02.1995 12:00 Uhr. Das Kommando zum Abfragen (nur IP-Verkehr) der EASE-Serverdatenbank lautete:

```
/usr/HPEASE/bin/easetraffic -ptype=ip -f30 -i -v 12300195 12010295
```

Die Ausführung dieses Kommandos benötigte ca. 10 Minuten (wegen der 67664 zu berücksichtigenden Einträge), die Systembelastung des EASE-Servers `hphegering1` betrug dabei nahezu 100% (laut `top`).

```
Results are:
  RPCCounts:
    Other Bytes         1.29854e+10
    Total Bytes         1.66681e+10
    Other Frames        5.28588e+07
    Total Frames        6.60173e+07
    Total Entries       67664

  Entries(30) ... SrcAddr, SPort, DstAddr, DPort, Bytes, Frames
    129.187.11.2, 0, 132.199.2.1, 0, 9.40295e+08, 1.87459e+06
    129.187.249.15, 0, 130.83.22.1, 0, 6.06149e+08, 1.13973e+06
    132.199.2.1, 0, 129.187.11.2, 0, 2.74222e+07, 1.0449e+06
    141.78.7.252, 0, 129.187.13.35, 0, 2.83176e+07, 1.02208e+06
    129.187.13.35, 0, 141.78.7.252, 0, 3.23441e+08, 982027
    130.83.22.1, 0, 129.187.249.15, 0, 1.56378e+07, 601455
    131.188.2.45, 0, 129.187.10.35, 0, 2.63038e+08, 536774
    130.73.103.51, 0, 131.159.0.176, 0, 1.36411e+08, 523776
    129.187.10.35, 0, 131.188.2.45, 0, 1.16312e+07, 444044
    129.187.11.2, 0, 132.180.92.24, 0, 2.13683e+08, 411536
    131.159.0.176, 0, 130.73.103.51, 0, 2.98177e+07, 385699
    131.159.0.51, 0, 131.188.61.13, 0, 1.04475e+08, 377061
    129.187.249.15, 0, 141.44.20.23, 0, 1.9281e+08, 371722
    131.188.61.13, 0, 131.159.0.51, 0, 2.77029e+07, 303467
    131.159.0.176, 0, 132.231.51.57, 0, 1.23851e+08, 303133
    129.187.123.2, 0, 129.247.161.124, 0, 1.5535e+08, 294037
    132.231.51.57, 0, 131.159.0.176, 0, 6.64322e+06, 252467
    141.44.20.23, 0, 129.187.249.15, 0, 6.21656e+06, 236661
    129.187.11.5, 0, 132.180.92.24, 0, 1.11207e+08, 221102
    132.180.92.24, 0, 129.187.11.2, 0, 5.57006e+06, 200982
```

```
131.188.2.45, 0, 129.187.13.35, 0, 5.98829e+07, 194400
192.245.197.18, 0, 131.159.0.198, 0, 5.02275e+06, 193183
131.234.10.2, 0, 131.159.0.51, 0, 3.28448e+07, 188545
131.159.0.51, 0, 131.234.10.2, 0, 1.45855e+07, 179783
129.187.156.66, 0, 168.143.2.72, 0, 1.07833e+07, 168115
129.187.13.35, 0, 131.188.2.45, 0, 1.90521e+07, 156973
131.159.0.198, 0, 192.94.94.33, 0, 6.25689e+07, 140933
129.187.231.131, 0, 141.30.231.149, 0, 7.19261e+07, 140811
129.187.13.4, 0, 137.193.10.6, 0, 7.29348e+07, 137639
131.220.221.220, 0, 131.159.0.198, 0, 3.41712e+06, 130757
```

Das entsprechende Ergebnis von NNStat:

```
OBJECT:Rnet.matrix Class=matrix-sym-bytes [Created: 12:00:00 01-30-95]
  ReadTime:12:00:00 02-01-95,ClearTime:12:00:00 01-30-95 (@-172800sec)
  Total Count= 47030932 (+18527426 orphans)
  Total Bytes= 12537464479B    #bins = 2048
[129.187.0.0 : 132.199.0.0]= 3198317 &1041184195B  ( 4.9%) @-1sec
[129.187.0.0 : 131.188.0.0]= 2658012 &647152208B   ( 4.1%) @-1sec
[129.187.0.0 : 141.78.0.0]= 2034629 &394016886B    ( 3.1%) @-0sec
[129.187.0.0 : 130.83.0.0]= 2017622 &676150715B    ( 3.1%) @-1sec
[129.187.0.0 : 134.245.0.0]= 1604925 &239502098B   ( 2.4%) @-1sec
[131.159.0.0 : 131.188.0.0]= 1350587 &377466366B   ( 2.1%) @-4sec
[129.187.0.0 : 132.180.0.0]= 1088578 &398089099B   ( 1.7%) @-235sec
[130.73.0.0 : 131.159.0.0]= 948436 &196907807B     ( 1.4%) @-3sec
[131.159.0.0 : 131.234.0.0]= 920344 &234744545B    ( 1.4%) @-59sec
[131.159.0.0 : 132.231.0.0]= 742661 &196628368B    ( 1.1%) @-2sec
[131.159.0.0 : 132.199.0.0]= 713848 &217010988B    ( 1.1%) @-44sec
[129.187.0.0 : 129.247.0.0]= 622150 &269562682B    ( 0.9%) @-160sec
[129.187.0.0 : 141.44.0.0]= 619802 &209046781B     ( 0.9%) @-14sec
[130.83.0.0 : 131.159.0.0]= 492618 &173665739B     ( 0.8%) @-6sec
[130.75.0.0 : 131.159.0.0]= 439058 &154352856B     ( 0.7%) @-287sec
[129.187.0.0 : 131.234.0.0]= 432826 &150547936B    ( 0.7%) @-7sec
[129.187.0.0 : 132.187.0.0]= 365079 &88262321B     ( 0.6%) @-1sec
[129.187.0.0 : 130.133.0.0]= 346482 &111243808B    ( 0.5%) @-211sec
[131.159.0.0 : 131.246.0.0]= 334615 &98744466B     ( 0.5%) @-3sec
[130.149.0.0 : 131.159.0.0]= 323051 &81797171B     ( 0.5%) @-172sec
[131.159.0.0 : 131.220.0.0]= 321238 &93575129B     ( 0.5%) @-7sec
[128.176.0.0 : 131.159.0.0]= 320262 &103246986B    ( 0.5%) @-171sec
[141.78.0.0 : 141.78.0.0]= 310475 &46550237B   ( 0.5%) @-3sec
[129.187.0.0 : 137.193.0.0]= 309227 &91634857B ( 0.5%) @-77sec
[129.26.0.0 : 129.187.0.0]= 283530 &72782227B  ( 0.4%) @-21sec
[131.159.0.0 : 134.96.0.0]= 274202 &74340104B  ( 0.4%) @-28sec
[131.159.0.0 : 134.109.0.0]= 270578 &85164254B ( 0.4%) @-36sec
[129.217.0.0 : 131.159.0.0]= 262469 &84013083B ( 0.4%) @-0sec
[129.187.0.0 : 168.143.0.0]= 257149 &22045515B ( 0.4%) @-1sec
```

```
[129.187.0.0 : 141.30.0.0]= 243481 &80554477B  ( 0.4%) @-153sec
[131.159.0.0 : 192.94.94.0]= 234814 &67610827B ( 0.4%) @-3sec
[129.187.0.0 : 129.206.0.0]= 229217 &102040446B    ( 0.3%) @-21sec
[128.174.0.0 : 131.159.0.0]= 228262 &69474633B ( 0.3%) @-191sec
[131.159.0.0 : 193.175.25.0]= 218010 &38862060B    ( 0.3%) @-2sec
[129.187.0.0 : 130.149.0.0]= 208644 &26592924B ( 0.3%) @-1sec
[130.235.0.0 : 131.159.0.0]= 200101 &63671923B ( 0.3%) @-16sec
[129.13.0.0 : 131.159.0.0]= 189514 &15061486B  ( 0.3%) @-54sec
[128.6.0.0 : 131.159.0.0]= 186887 &28927865B   ( 0.3%) @-6sec
[129.187.0.0 : 131.246.0.0]= 182334 &59594748B ( 0.3%) @-23sec
[129.187.0.0 : 130.161.0.0]= 182302 &52295996B ( 0.3%) @-1sec
[129.187.0.0 : 137.226.0.0]= 179946 &56740887B ( 0.3%) @-1283sec
[130.237.0.0 : 131.159.0.0]= 178422 &61543301B ( 0.3%) @-868sec
```