5 Installation und Betrieb

5.1 Installation

In der folgenden Installationsanleitung bezeichnet $MCNTP den Pfad im Dateisystem, in dem sich das ausgepackte *Mcntp* Paket befindet. $RSA bezeichnet analog das Verzeichnis, in dem die RSA Bibliothek sich befindet.

Installationsschritte sind:

1. Die librsa holen[36] und übersetzen Die rsaref Bibliothek wurde dem *Mcntp* Paket nicht beigelegt, da laut J.S.A. Kapp die RSAEURO Bibliothek in den USA Probleme mit Copyrightbestimmungen haben kann, während die Verwendung der RSAREF Bibliothek von RSA Data Security Inc. außerhalb der USA illegal ist. Aus dem Dokument "licence.txt" des RSAEURO Pakets:

   > 6. RSAEURO is a publication of cryptographic techniques. Applications developed with RSAEURO may be subject to export controls in some countries. If you are located in the United States and develop such applications using RSAEURO, you are advised to obtain a copy of RSAREF from RSADSI, as you may using RSAEURO infringe on Patents held by Public Key Partners of Sunnydale California.

   Die RSAEURO Bibliothek kann u.a. via anonymem ftp u.a. vom Server ftp.cert.dfn.de bezogen werden.

   Zum Auspacken des RSAEURO Archivs ist das Programm unzip nötig. Hierbei ist zu beachten, daß unzip mit der Option -a mit angegeben wird, da das Zeilenvorschubzeichen ^M, das in den Dateien vorhanden ist, viele Compiler verwirrt. Die notwendigen Schritte zum Erzeugen der Bibliothek sind:

   (a) cd $RSA/install

   (b) vi unix/Makefile

   (c) vi source/global.h

   (d) make -f unix/Makefile

   Im Verzeichnis $RSA/install befinden sich neben der Funktionenbibliothek, einige Beispielprogramme, die genutzt werden können, um festzustellen, ob die Bibliothek korrekt übersetzt wurde. Diese Beispielprogramme können mit den Scripten in $RSA/scripts/ automatisch ausgeführt werden; Referenzausgaben der Programme liegen bei, um die korrekte Funktion feststellen zu können.

2. *Mcntp* Software holen und auspacken. Die *Mcntp* Software ist via anonymem ftp vom Rechner ftp.xlink.net im Verzeichnis /pub/news/tools/ als Datei mcntp.tar.gz erhältlich.

3. Die Datei README im Verzeichnis $MCNTP lesen.

4. Die Datei `rsaref.a` aus $RSA/install/ und die Header-Dateien (*.h) aus $RSA/source/ nach $MCNTP/rsa kopieren

5. Die Datei `config.h` anlegen. Die Datei `config.h` kann durch Kopieren der Datei `config.h.dist` erzeugt werden. Die Datei `config.h` wurde nicht direkt beigelegt, um zu verhindern, daß Aktualisierungen der *Mcntp* Distribution eine vorhandene Version von `config.h` überschreiben.

[36] Siehe Abschnitt 4.7.

6. Die Datei `config.h` editieren. Die Werte in `config.h` stimmen nicht für jedes System und müssen vor dem Übersetzen von *Mcntp* angepasst werden. Hier folgt eine Beschreibung der Präprozessorsymbole in `config.h`, die angepasst werden müssen:

   BLITTLEEND/BBIGEND Dient zur Unterscheidung, ob der Prozessor des Rechners eine sog. BigEndian (z.B. Motorola 680x0 oder Sun Sparc) oder Little-Endian Maschine (z.B. Intel ix86) ist (siehe auch Abschnitt 4.4.1). Beispiel:

   ```
   #define BBIGEND
   ```

   DONTWANTZLIB Wenn keine Datenkompression via Zlib (s.u.) gewünscht wird, kann der entsprechende Code auskommentiert werden. Dies bedeutet aber auch, daß keine komprimierten Artikel akzeptiert werden können. Dieses Präprozessorsymbol sollte nur in Ausnahmefällen definiert werden. Beispiel:

   ```
   /* #define DONTWANTZLIB */
   ```

   HAVE_STATFS Wenn mcrcv Artikel auf der Platte ablegt (siehe 4.5.2), kann mittels des statfs() Betriebssystemaufrufs festgestellt werden, ob noch genug Platz auf der Festplatte vorhanden ist. Ist diesel Parameter nicht definiert, werden die Artikel solange geschrieben, bis das Dateisystem voll ist. Beispiel:

   ```
   #define HAVE_STATFS
   ```

   HAVE_MMAP Mit dem mmap() Systemaufruf können bei vielen Unix Systemen Dateien schneller in den Arbeitsspeicher geladen werden, als mittels read(). Mcxmit kann dies ausnutzen, um die Artikel zur Verarbeitung in den Speicher zu laden. Da nicht alle Systeme eine funktionierende Implementierung von mmap() haben, wird dieser Code nur bei Bedarf eingesetzt. Beispiel:

   ```
   #define HAVE_MMAP
   ```

   SPOOL_PATH Damit mcxmit die Artikel, die versendet werden sollen, finden kann, muß hier der Pfad angegeben werden, unter dem der Artikelbaum zu finden ist (gemeinhin auch als “Artikelspool” bezeichnet). Beispiel:

   ```
   #define SPOOL_PATH "/var/spool/news"
   ```

   SPOOL_PATH_IN Damit mcrcv Artikel, die nicht via NNTP weitergeleitet werden können, als Batches im Dateisystem ablegen kann, muß dieser Parameter definiert werden. Der Parameter bezeichnet das Verzeichnis, in dem Programm rnews die Batches erwartet. Bei den meisten Newssystemen ist dies SPOOL_PATH/in.coming. Beispiel:

   ```
   #define SPOOL_PATH_IN "/var/spool/news/in.coming"
   ```

   BATCH_SIZE Dieser Parameter definiert die maximale Größe der von mcrcv produzierten Batches in Bytes. Wenn der Wert auf -1 gesetzt wird, werden keine Batches produziert. Beispiel:

   ```
   #define BATCH_SIZE 50000
   ```

   SPOOL_LIMIT Dieser Wert definiert, falls der Parameter HAVE_STATFS ebenfalls definiert ist, den freien Platz im Dateisystem an, ab dessen Unterschreitung keine weiteren Artikel mehr im Dateisystem abgelegt werden. Dieser Wert muß größe sein, als der von BATCH_SIZE. Beispiel:

   ```
   #define SPOOL_LIMIT 100000
   ```

SPOOL_RECHECK Wenn mcrcv keine Artikel mehr via NNTP einliefern kann und Artikel gespoolt werden, wird nach der hier angegebenen Zeit in Sekunden nachgeschaut, ob der NNTP-Server wieder verfügbar ist. Wird dieser Wert auf -1 gesetzt, unterbleiben diese Tests. Beispiel:

```
#define SPOOL_RECHECK 2*60
```

CHECKPOINT Mcxmit und mcrcv geben nicht nur bei Terminierung, sondern auch während der Verarbeitung regelmäßig Daten über Syslog(3) aus. Dieser Parameter definiert, nach wievielen bearbeiteten Artikeln die Programme diese Daten ausgeben sollen. Für Server, die viele Artikel versenden oder empfangen ist hier ein Wert von 200 oder 500 sinnvoll. Beispiel:

```
#define CHECKPOINT 50
```

NNTP_PORT Normalerweise werden NNTP-Verbindungen zu Port 119 aufgebaut. Dieser Port wurde von der IANA für diesen Zweck offiziell zugewiesen. Dieser Port wird sowohl von Newsleseclients, als auch von anderen Transferagents für das Einliefern der Artikel genutzt. Aus Performancegründen werden hierfür auf manchen Systemen unterschiedliche Ports verwendet. Dieser Wert gibt an, auf welchem Port mcrcv die Artikel einliefern soll[37]. Beispiel:

```
#define NNTP_PORT 119
```

LIB_PATH Dieser Parameter gibt das Verzeichnis an, in dem die Konfigurationsdatei `yawgmoth.conf`, der Keyring und der private Schlüssel standardmäßig gesucht werden. Beispiel:

```
#define LIB_PATH "/usr/local/news/mcntp/"
```

PER_CLOSE Newstransferagents geben teilweise erst dann Statistiken über eingehende Verbindungen aus, wenn diese beendet worden sind. Da Mcrcv ein langlaufender Prozeß ist, wird hiermit dem NTA die Möglichkeit gegeben, diese Statistiken zu generieren. Der angegebene Wert in Sekunden beschreibt die Zeitspanne, nach der mcrcv die NNTP-Verbindung zum NTA schließt und wieder öffnet. Wenn der angegebene Wert größer als 86400 ist (ein Tag), wird er auf 86400 gesetzt. Ist der Wert kleiner als 60, wird er auf 60 gesetzt, um den NTA nicht unnötig zu belasten. Beispiel:

```
#define PER_CLOSE 60*60
```

Nach der Definition dieser Parameter werden noch einige Tests und Voreinstellungen in `config.h` durchgeführt. Hier sollte der Anwender keine Änderungen vornehmen.

7. Zlib übersetzen. Die Zlib unterliegt keinen solchen Beschränkungen, wie die librsa und ist deshalb im Verzeichnis $MCNTP/zlib vorhanden. Die Zlib zu übersetzen ist einfach:

   (a) cd zlib/zlib

   (b) ./configure

   (c) make

   (d) cd ../../

   Falls die Zlib nicht allgemein zugänglich auf dem System installiert wird, müssen evtl. die Makefiles für mcxmit und mcrcv angepasst werden, wie dies in diesen Makefiles angegeben ist.

8. *Mcntp* übersetzen. Dies geschieht entweder durch Aufruf von make für Systeme, die ein 4.4BSD kompatibles make haben. Ansonsten ist make mit der Option -f Makefile.<Systemtyp> aufzurufen.

[37] Dies muß bei einer Zweiteilung der Port, auf dem die Artikel von anderen Servern ankommen und nicht der, den die Newsleseclients benutzen.

9. Die Programme müssen in den Standardsuchpfad des Benutzers “news” kopiert werden, damit sie gefunden werden, ohne daß zum Aufruf jedesmal der komplette Pfad mit angegeben werden muß.

10. Das Verzeichnis LIB_PATH, das in der Datei `config.h` definiert wurde, muß bei Bedarf noch angelegt werden.

5.1.1 Portierungen auf andere Plattformen

Mcntp wurde bereits auf die in Abschnitt 4.8 beschriebenen Systeme portiert. Wenn *Mcntp* auf andere Systeme portiert werden soll und nicht die BSD-Version von make, pmake zum Einsatz kommt[38], muß im Hauptverzeichnis ein neues Makefile für dieses System mit dem Namen *Makefile.<System>* angelegt werden. Dieses Makefile erhält man am besten, indem man eines der bestehenden Makefiles kopiert. In diesem Makefile sind Änderungen an folgenden Eintragen zu machen:

CC Der Pfad zum verwendeten C-Compiler. Beispiel:

```
CC=gcc
```

AR Der Pfad zum verwendeten ar Programm. Mit ar werden einzelne Dateien zu einer Bibliothek zusammengefügt. Beispiel:

```
AR=/usr/ccs/bin/ar
```

RANLIB Der Pfad zum verwendeten ranlib Programm. ranlib kreiert eine Tabelle der externen Referenzen einer Bibliothek und wird nicht auf allen Systemen benötigt. Beispiel

```
RANLIB=/bin/ranlib
```

ELIBS Bibliotheken, die zusätzlich eingebunden werden müssen. Beispiel:

```
ELIBS="-lfoo -lbar -L/lib/such/pfad -lbaz"
```

RSAREF Die Linker Anweisungen, die nötig sind, um die rsaref Bibliothek in mcrcv und mcxmit einzubinden. Wenn die Eurorsa Bibliothek wie oben beschrieben installiert wurde, ist dies:

```
RSAREF="-L../rsa/ -lrsaref"
```

Wenn *Mcntp* auf ein neues System portiert wurde, sollten die dazu nötigen Änderungen an

<mcntp-bugs@pilhuhn.de>

geschickt werden, damit sie in spätere Versionen von *Mcntp* einfließen können.

[38] pmake bietet neben der Möglichkeit, daß das Übersetzen der Quellen parallel erfolgen kann, noch eine weitere Besonderheit. pmake kann Steuerdateien und andere Makefiles via einer `.include` Direktive einbinden. Die diese Dateien enthalten dann die Regeln, die pmake benutzt, um Programme zu übersetzen. Diese Regeln gelten systemweit. Dadurch sind die Makefiles meist sehr klein.

5.2 Betrieb

Nach dem die *Mcntp* Software übersetzt und installiert ist, kann *Mcntp* in Betrieb gehen. Die folgenden Abschnitte zeigen, wie die einzelnen Module in der Praxis eingesetzt werden. Der Betrieb von *Mcntp* setzt eine vorhandene multicastfähige Infrastruktur voraus. Wie eine solche Infrastruktur bereitgestellt wird, ist nicht Teil dieser Arbeit[39].

Wenngleich auf einem Host *Mcntp* sowohl als Sender, als auch als Empfänger eingesetzt werden kann, ist diese Konstellation eher unüblich. Im Folgenden wird deswegen zuerst der Betrieb als Sender und dann der als Empfänger begesprochen. Die Beispiele kommen alle vom Betrieb bei Xlink.

5.2.1 Betrieb als Sender

Um Artikel via *Mcntp* versenden zu können, muß auf dem Senderhost neben den Multicastsendern mcxmit auch ein Directoryserver yawgmoth betrieben werden. Außerdem muß ein Schlüsselpaar erzeugt werden, welches zur Generierung und Überprüfung der digitalen Signaturen benötigt wird.

Directory Server starten

Der Directory-Server yawgmoth wird beim Systemstart gestartet. Dies kann z.B. aus einem der Scripten heraus geschehen, die das System beim Start automatisch ausführt (`/etc/rc`). Da yawgmoth nicht als Benutzer “Root” gestartet werden muß, ist `rc.news` ein geeigneter Platz.

```
> yawgmoth &
```

Zu Debuggingzwecken kann mit der Option -d eine ausführlichere Ausgabe auf dem Bildschirm ausgegeben werden.

```
> yawgmoth -d 1
Sat Aug 23 22:35:41 MET DST 1997
REQ:pilhuhn.test
USE:228.1.1.1:50001:8:
QUIT:1:0:0:0:
OK
```

So sieht man am Beispiel, welche Anfragen nach Multicastgruppen von den Sendern erfolgen (REQ), welche Gruppe und Port der Client erhält (USE) und welche Abschlußmeldungen der Client liefert (QUIT). Siehe auch Abschnitt 4.3.2 auf Seite 42.

Für die Funktion von yawgmoth ist die Konfigurationsdatei `yawgmoth.conf` notwendig, welche standardmäßig im Verzeichnis LIB_PATH gesucht wird (siehe auch 4.3.1). Eine minimale Version sieht wie folgt aus:

```
ME:8:blackbush
*:228.1.1.1:250:16:
```

[39] Ein multicastfähiger Rechner, auf dem *Mcntp* überstetzt wird, ist bereits eine solche (minimale) Infrastruktur und kann für Tests der Software bereits genutzt werden.

In der “ME” Zeile wird die Sender-ID auf *blackbush* und die TTL auf 8 gesetzt. Der einzige vorhandene Gruppeneintrag trifft auf alle Gruppen zu und instruiert, yawgmoth die Multicastgruppen 228.1.1.1 bis 228.1.1.250 zu verwenden. Da der TTL-Eintrag größer ist, als der in der “ME” Zeile, wird er auf den dortigen Wert gesetzt.

Sollen Newshierarchien mit unterschiedlichen geographischen Ausdehnungen verteilt werden, kann dies über die TTL Werte gesteuert werden. Im folgenden Beispiel wird noch die für Karlsruhe lokale Hierarchie *ka.** verteilt, wobei die Reichweite der Verteilung beschränkt ist (TTL von vier):

```
ME:16:blackbush
# Die Ka.*  Gruppen sollen in Karlsruhe bleiben
ka.*:239.1.1.1:4:
# Die anderen Gruppen im gesamten Backbone verteilen
*:228.1.1.1:250:16:
```

Die Zeilen, die mit dem “#” Zeichen beginnen, sind Kommentarzeilen; ihr Inhalt wird von yawgmoth nicht beachtet.

Auswahl der passenden Multicastgruppenbereiche

In Abschnitt 2.2.4 auf Seite 11 wurde die Unterteilung der Multicastadressen in global und lokal gültige Adressen beschrieben. Wie in diesem Abschnitt und in 2.2.3 erläutert, ist die Angabe der TTL nötig, aber nicht ausreichend, um die Ausbreitung zu beschränken und die Gruppen lokal zu halten.

Newsgruppen, die nicht im globalen MBone verteilt werden, sollten deshalb im Bereich der administrativ limitierten Gruppen verteilt werden, um die Ausbreitung dieser Gruppen zu limitieren. Dies gilt nicht nur für Gruppen, die für ein geographisches Gebiet lokal sind (z.B. für eine Stadt), sondern auch wenn die Gruppen nur für Kunden eines Internetserviceproviders (ISP) verfügbar sein sollen.

Im letzten Beispiel wird die *ka.** Hierarchie bereits in einer der administrativ begrenzten Multicastgruppen verteilt, während alle anderen Gruppen im globalen MBone sichtbar sind[40].

Key generieren

Damit mcxmit funktionieren kann, muß vor dem ersten Versenden ein Schlüsselpaar generiert werden. Dieses Paar besteht aus einem privaten Schlüssel zur Bildung der digitalen Signaturen *mcntp-key.priv* und dem öffentlichen Schlüssel *mcntp-key.pub*, der bei den Empfängern zur Überprüfung der Signaturen verwendet wird.

```
> keygen -b 512 -i snert
Generating keys with 512 Bit length .. may take a while
Ripe-md160 fingerprint for the public key is :
71d897cff15e4edb7a3f10e592a4579739aa2e5a
> ls -ls m*
1 -rw-r--r--  1 hwr  pilhuhn  712 Aug 23 19:48 mcntp-key.priv
1 -rw-r--r--  1 hwr  pilhuhn  264 Aug 23 19:48 mcntp-key.pub
```

[40] Durch die TTL von 16 allerdings maximal 16 Hops vom Sender entfernt.

Bei der Schlüsselgenerierung muß neben der Sender-ID noch die Länge des zu generierenden Schlüssels mit angegeben werden. Je länger der Schlüssel ist, desto sicherer ist er gegenüber Versuchen, ihn kryptographisch zu brechen. Allerdings bedeutet eine größere Schlüssellänge auch ein deutlich höher Aufwand [41] bei der Bildung und Verifikation der digitalen Signatur:

Länge in Bit	Signieren in Sekunden	Überprüfen in Sekunden
512	0.35	0.04
768	1.11	0.07
1024	2.62	0.13

Tabelle 9: Einfluß von Schlüssellängen auf die Geschwindigkeit des Signierens

Da die digitale Signatur bei *Mcntp* nur benutzt wird, um zu verhindern, daß auf einfache Weise Massenwerbepostings oder gefälschte Artikel in Umlauf kommen, sind in der Praxis Schlüssellängen von 512 Bit ausreichend. Im Usenet gibt es viele Server, die für jedermann zugängliche sind. Die Benutzung dieser Server ist mit weniger Aufwand versehen, als das kryptographische Brechen des Schlüssels.

Der generierte private Schlüssel muß vor Benutzung durch mcxmit nach LIB_PATH/mcntp-key.priv kopiert werden.

Ein Senderhost kann mehr als einen Schlüssel haben. So wäre es beispielsweise möglich, eine private Newsgruppe mit einer anderen Sender-ID zu versenden, als die allgemein zugänglichen Gruppen. Wenn nun der öffentliche Schlüssel dieser privaten Gruppe nur an berechtigte Empfänger verteilt wird, werden nur diese in der Lage sein, diese Artikel via mcrcv zu empfangen. Da die Artikel selbst nicht verschlüsselt werden, ist diese Methode aber nicht vollkommen sicher – die Daten können immer noch durch Programme, die auf der Netzwerkschicht Pakete "mithören", empfangen werden[42].

Sender starten

Um den Sender mcxmit zu starten, muß diesem der Name der Gruppe als Parameter mitgegeben werden. Der Sender wartet dann auf der Standardeingabe auf Daten vom NTA.

```
> mcxmit -g pilhuhn.test
```

Die Eingabe besteht aus dem Pfad in Dateisystem, unter dem der Artikel im Dateisystem gefunden werden kann[43], und der Message-ID. Die Angabe der Message-ID ist nicht unbedingt nötig, erspart mcxmit den Aufwand, diese herauszufinden, während diese von den NTAs ohne Aufwand mitgeliefert werden.

Wenn die Artikel vor dem Versand komprimiert werden sollen, ist die Option -c anzugeben.

```
> mcxmit -g pilhuhn.test -c
```

[41] Die Zeiten in Tabelle 9 wurden auf einem 486dx2/80 PC ermittelt.

[42] Wenn die versendeten Daten vor dem Versand komprimiert werden, wird bei der Erzeugung der digitalen Signatur ein Teil des Kompressionsbaums, den zlib benutzt, ebenfalls verschlüsselt, so daß es schwieriger wird, die Nachricht zu lesen, wenn der Schlüssel zum Entschlüsseln der Signatur nicht bekannt ist. Dies ist allerdings kein kryptographisch sicheres Verfahren zur Datenverschlüsselung! Vergleiche auch Abschnitte 4.4.2 und 2.4.1.

[43] Dies kann ein relativer Pfad sein. in diesem Fall wird dem Pfad der Wert von SPOOL_PATH vorangestellt.

Im Betrieb ist es aus Performancegründen[44] besser, mcxmit direkt vom NTA aus mit den Artikelpfaden zu versorgen. Hierzu ist es nötig, daß der NTA selbst Prozesse starten kann, die ihre Informationen dann über die Standardeingabe erhalten. Beim NTA INN ist dies über einen sogenannten *Channel-Feed* möglich. Ein Eintrag in der Datei `Newsfeeds`[45] wird dann wie folgt aussehen:

```
de-cast:de.*:Tc,Wnm,<30000:mcxmit -g de -c
```

Die Felder dieses Eintrags, die voneinander durch einen Doppelpunkt ":" getrennt sind, bestehen aus einem internen Namen des Empfängers (de-cast), der Liste der Gruppen, die der Empfänger erhalten möchte (de.*), einer Liste von Optionen und einem Parameter, der von den Optionen im dritten Feld abhängt. In diesem Fall bedeutet "Tc," daß dieser Eintrag ein *Channel-Feed* ist und der Parameter im vierten Feld ein Programm ist, welches auf dem Standardeingabekanal auf Daten wartet. Die Option "Wnm" gibt an, daß ein Artikelpfad relativ zum Newsspool und die Message-ID ausgegeben werden soll; "<3000" gibt an, daß nur Artikel weitergeleitet werden sollen, die kleiner als 30.000 Bytes groß sind. Mcxmit und mcrcv sind in der Lage, bis zu 60 kBytes große Artikel zu verarbeiten, jedoch ist es in der Praxis sinnvoll, den Betrieb auf Artikel bis maximal 30.000 Bytes zu beschränken, da die Empfangspuffer der Empfängersysteme oftmals nicht groß genug sind, mehrere große Artikel zu halten und dadurch erhaltenen Pakete verworfen werden. Diese Beschränkung verringert die Anzahl der transportierten Artikel nur geringfügig (siehe auch Abschnitt 2.3.1 auf Seite 14).

Dieser Feed verteilt nur Artikel in der Hierarchie *de.**, die vor dem Versand komprimiert werden. Damit INN das mcxmit Programm finden kann, muß es im Standardsuchpfad zu finden sein. Ist dies nicht der Fall, ist beim Aufruf der gesamte Pfad zu mcxmit anzugeben.

Mcxmit gibt über Syslog(3) zur Laufzeit Daten aus, wieviele Artikel bisher vom NTA erhalten und via mcxmit verschickt wurden. Außerdem enthalten diese Daten Angaben, wie groß die Artikel vor dem Versand waren und wieviele Bytes verschickt wurden.

```
Aug 24 18:17:46 blackbush mcxmit[6708]: mcxmit: 149 received 149
 sent 386356 art_bytes 258115
```

In diesem Beispiel wurden 149 Artikel vom Server angeboten. Alle 149 Artikel konnten im Dateisystem gefunden werden[46] und waren kleiner als 60kBytes. Die Artikel hatten vor dem Versand zusammen 386356 Bytes; 258115 Bytes wurden über das Netz verschickt (Artikeldaten und *Mcntp* Protokollheader gesamt).

5.2.2 Betrieb als Empfänger

Für dem Empfang von NetNews via *Mcntp* wird neben dem Multicastempfänger mcrcv noch der öffentliche Schlüssel des Senders benötigt. Dieser Schlüssel kann z.B. vom Sender auf einem World-Wide-Web Server abgelegt worden sein, wie dies bei Xlink der Fall ist.

[44] Wenn die Artikelpfade in eine Datei geschrieben und erst später durch mcxmit verarbeitet werden, ist die Wahrscheinlichkeit höher, daß die Artikel bereits auf anderem Wege beim Empfänger angekommen sind und damit die Übertragung vergeblich war.

[45] Dies ist beim NTA INN die Datei, in der festgelegt wird, welcher Empfänger welche Gruppen erhalten soll.

[46] Dies muß nicht notwendigerweise der Fall sein, da der Artikel vor dem Weiterversand bereits mittels *Cancel* Kontrollnachricht (siehe 2.3) gelöscht worden sein kann.

Keyring erzeugen

Der Keyring wird von *Mcntp* standardmäßig in LIB_PATH gesucht; keygen nutzt diesen Pfad nicht automatisch, um zu verhindern, daß bei Tests der private Schlüssel oder der Keyring überschrieben wird. Falls noch kein Keyring vorhanden ist, muß ein leerer Keyring angelegt werden:

```
> touch keyring
```

Der neue Schlüssel kann dann dem Keyring mittels des Kommandos keygen -r *keyring* -a *Schlüssel* hinzugefügt werden.

```
> keygen -a ./snert-key.pub -r ring
> keygen -a ./blackbush-key.pub -r ring
```

Sollte bereits ein Schlüssel mit gleicher Sender-ID vorhanden sein, meldet keygen dies und bricht den Vorgang ab:

```
> keygen -a ./snert-key.pub -r ring
Warning: key for <snert> already exists. Not replacing it. (use -f )
```

Mit der Option -l kann man mit keygen feststellen, welche Schlüssel auf einen Keyring vorhanden sind:

```
> keygen -l -r ring
id        fingerprint
-------------------------------------
snert     f3c967d39347abe76497ab3b1d625d6b255fe295
blackbush eac6614400992e4c0306fb80417b7979cc3984d8
```

Für jeden Schlüssel werden neben der Sender-ID ein RIPEMD-160 Message-Digest des Schlüssels mit ausgegeben. Durch Vergleich dieses Fingerabdrucks mit dem beim Sender kann festgestellt werden, ob die Schlüssel korrekt übertragen wurden.

Welche Gruppen gibt es?

Normalerweise erhält man vom Sender eine Liste der Gruppen, die dieser verteilt. Ist dies nicht der Fall, kann diese Liste mit groupdump erhalten werden.

```
> groupdump
------ Sat Aug 23 16:37:30 1997
 >>193.141.89.2<<
Packet Version: 1.1 Sender-id: >snert<
pilhuhn.test:228.1.1.1:50001:1
------ Sat Aug 23 16:37:36 1997
 >>193.141.40.8<<
Packet Version: 1.1 Sender-id: >blackbush<
rec:228.1.1.50:50050:16
de:228.1.1.52:50052:16
hss:228.1.1.53:50053:16
```

In diesem Beispiel gibt es zwei Sender, *Snert* und *Blackbush*. Snert verteilt nur eine Gruppe “pilhuhn.test”, während Blackbush mehrere verteilt. Während “de” und “rec” die Namen von Newshierarchien sind, ist “hss” eine Abkürzung für “humanities, soc, sci”. Diese Zuordnung wurde willkürlich gewählt.

Empfänger starten

Um den Multicastempfänger mcrcv zu starten, muß mit der Option -g dem Empfänger die zu erhaltende Newsgruppe mitgegeben werden.

```
> mcrcv -g de &
```

Der Empfänger testet, ob er Artikel auf dem NNTP-Server des Rechners “localhost”[47] einliefern kann. Ist dies der Fall, werden die Artikel dort eingeliefert, ansonsten auf Platte abgelegt. Mit der Option -s kann ein anderer Server und mit der Option -p ein anderer Port, als der in NNTP_PORT voreingestellte, gewählt werden:

```
> mcrcv -g rec -s news.xlink.net -p 1234&
```

Der Keyring wird von mcrcv standardmäßig in LIB_PATH/keyring erwartet. Mit der Option -K kann auch ein anderer Keyring verwendet werden:

```
> mcrcv -g comp -K /pfad/zu/einem/keyring &
```

Normalerweise gibt mcrcv seine Ausgaben via Syslog(3) aus. Mit der Option -d wird die Ausgabe auf den Bildschirm ausgegeben; je größer der mitgegebene Parameter ist, desto detaillierter ist die Ausgabe. Mit -d 5 wird z.B. für jeden erhaltenen Artikel das Ankunftsdatum, der Sender, die Message-ID und Größe, sowie der Antwortcode des NNTP-Servers angegeben:

```
> mcrcv -d 5 -g de
Aug 23 19:29:59. blackbush ==> <5tn3qa$lbe$5@gwdu19.gwdg.de>  239 1130
Aug 23 19:30:00. blackbush ==> <5tn450$lbe$6@gwdu19.gwdg.de>  239 1352
Aug 23 19:30:01. blackbush ==> <5tn60j$lbe$7@gwdu19.gwdg.de>  239 2248
^Cmcrcv: 3 offered 3 accepted 0 rejected 0 refused 0 spooled
```

Im Normalfall wird beim Systemstart eine Reihe von mcrcv Prozessen gestartet, die die gewünschten Newsgruppen bearbeiten. Wie bereits bei yawgmoth ist `rc.news` ein geeigneter Platz, die Empfänger zu starten. Diese laufen dann bis zum Herunterfahren des Systems.

Mcrcv gibt ebenfalls Betriebsdaten über Syslog(3) aus. Diese Daten geben an, wieviele Artikel mcrcv erhalten hat und wie diese weiterverarbeitet wurden.

```
Aug 24 18:24:01 snert mcrcv[21498]: check: 21950 offered 21291
    accepted 659 rejected 0 refused 0 spooled
```

Hier wurden 21950 Artikel vom Netz empfangen. Davon wurden 21291 vom Newsserver akzeptiert, während 659 vom Server als bereits vorhanden zurückgewiesen wurden. Es wurde kein Artikel vom Server nachträglich abgewiesen und auch kein Artikel auf Platte abgelegt, weil der Server nicht verfügbar war.

[47] localhost ist bei Unix Systemen der Name der sog. Loopbackadresse, 127.0.0.1.

5.3 Möglicher Betrieb im globalen MBone

Der Betrieb von *Mcntp* innerhalb des abgeschlossenen Bereichs eines einzelnen Internetserviceproviders (ISP) unterscheidet sich von Betrieb im globelen MBone:

- Innerhalb eines ISP kann der Bereich der administrativ limitierten Adressen verwendet werden (siehe 2.2.4) über den ein ISP volle Kontrolle hat.
- Der ISP kann eine oder mehrere Maschinen als Multicastsender bestimmen und hat eine Kontrolle darüber, welche Gruppen verteilt werden.
- Das Monopol eines einzelnen Providers als Sender im MBone wird von der Internetgemeinde nicht anerkannt werden. Wenn mehrere Provider als Sender auftreten, besteht die Gefahr, daß Artikel mehrfach übertragen werden und damit Bandbreite verschwendet wird.
- Multicastgruppen im MBone werden dynamisch verteilt. Es gibt, wie in 2.2.4 beschrieben, eine Reihe von Multicastadressen, die von der IANA (siehe auch Abschnitt 2.6) zugeteilt werden. Bis auf diese werden die Gruppen von Programmen, die IP-Multicast nutzen, dynamisch und nach einem Pseudozufallsprinzip vergeben. Damit kann nicht ausgeschlossen werden, daß zwei unterschiedliche Applikationen dieselben Multicastgruppen nutzen.

Um NetNews auf dem MBone zu verteilen, bietet sich folgendes Vorgehen an:

- Es werden entweder bei IANA eine Reihe von Multicastadressen für diesen Zweck reserviert oder in Zusammenarbeit mit den Entwicklern anderer Software versucht, einen Bereich für den Newstransport freizuhalten.
- Die großen Backboneserver, die News via *Mcntp* versenden wollen, dürfen nur die Artikel via *Mcntp* weiterleiten, die von ihnen selbst oder von Teilnehmern stammen, die ihre Artikel nur über diesen Sender verteilen können. Artikel, die über andere Backbonenewsserver erhalten werden, dürfen nicht via *Mcntp* weiterverteilt werden.
- Die Artikel werden mit einer TTL von 128 weltweit verteilt.
- Die Zustellung der Artikel via NNTP wird beibehalten.

Durch diese Vorgehensweise werden keine Artikel doppelt über *Mcntp* verteilt, da jeder Artikel nur von dem Backboneserver eingespeisst wird, der dafür zuständig ist. Durch die Beschränkung der Sender auf die Backboneserver, ist die Menge der Schlüssel, die bei den Empfängern bekannt sein muß noch relativ überschaubar.

Damit das aufgezeigte Szenario funktioniert, muß allerdings yawgmoth noch dahingehend modifizert werden, daß yawgmoth für den Versand von Artikeln Gruppen nutzt, die bereits von anderen Sites annonciert werden.

5.4 Zusammenfassung

Dieser Kapitel beschreibt, wie *Mcntp* installiert wird, welche Anpassungen an das lokale System nötig sind und wie *Mcntp* auf neue Systeme portiert werden kann. Danach wird gezeigt, wie die einzelnen Komponenten von *Mcntp* für den Newstransport bei einem Internetserviceprovider eingesetzt werden können. Zuletzt wurde ein Szenario beschrieben, um *Mcntp* auf dem globalen MBone einzusetzen.