Team der Chemnitzer Linux-Tage:

Chemnitzer Linux-Tage 2011

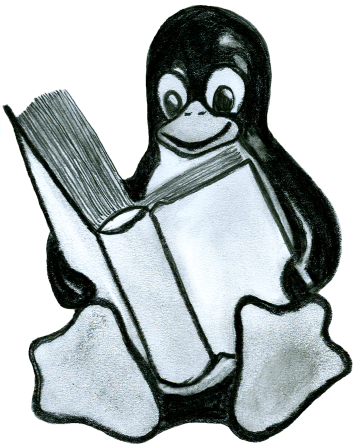
– Tagungsband –

19. und 20. März 2011

Team der Chemnitzer Linux-Tage

Chemnitzer Linux-Tage 2011

19. und 20. März 2011



– Tagungsband –

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Universitätsverlag Chemnitz
2011

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Technische Universität Chemnitz/Universitätsbibliothek
Universitätsverlag Chemnitz

Herstellung und Auslieferung:
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
http://www.mv-verlag.de

ISBN 978-3-941003-29-3

URL: http://nbn-resolving.de/urn:nbn:de:bsz:ch1-qucosa-65020

Satz und Layout: Jens Pönisch, Monique Kosler und Daniel Schreiber
Fotos Cover: Daniel Lucas Hahn (vorn), Petra Pönisch (hinten)
Graphik Innentitel: Petra Pönisch
URL der Chemnitzer Linux-Tage: http://chemnitzer.linux-tage.de

Premium-Sponsoren:

Weitere Sponsoren:

Medienpartner:

Inhaltsverzeichnis

1 Vorwort

Das sind die letzten Chemnitzer Linux-Tage. Wer so etwas hört und schon ein oder mehrere Male in Chemnitz dabei war, erschrickt. Und doch gibt es in jedem Jahr Momente, wo die Aufgaben so groß werden, dass man sich das nicht noch einmal aufbürden möchte. Dann werden alle Kräfte mobilisiert, und sei es mit dem Gedanken «Es ist ja zum letzten Mal!». Das Motto «Freiheit leben» hat dann einen sehr angestrengten Beigeschmack. Und doch gehören diese enormen Anstrengungen dazu. Immer wieder. Ein großer Teil dessen, worüber wir im März diskutieren, wurde auf ebensolche Weise von Leuten geschaffen, die dies mehr oder weniger nebenbei taten. Ohne Entwickler, Tester, Texter, Administratoren und viele andere, die den Raum schufen, um Freiheit leben zu können, gäbe es weder Open Source noch Linux – und schon gar keine Linux-Tage in Chemnitz.

Daher ein Dank an alle, die mitgeholfen haben, und ganz besonders natürlich an die Autoren und Referenten, Layouter, Korrektoren und Setzer. Wir freuen uns sehr, nun zum zweiten Mal einen Tagungsband vorlegen zu können. Er zeigt, wie weitgefächert die Themen unserer Veranstaltung sind. Wenn sich dann im März das Publikum ebenso vielseitig interessiert in die Vortragssäle stürzt, werden zwischen den Organisatoren die ersten Stimmen laut: Na gut, noch einen allerletzten machen wir. Seit Jahren. Wir sind so frei.

Ralph Sontag im Februar 2011

Drahtlose Sensordatenerfassung und -verarbeitung mit Linux

Axel Wachtler, Jörg Wunsch, Matthias Vorwerk
{axel,joerg,matthias}@uracoli.de
http://uracoli.nongnu.org/clt2011

Die Erfassung physikalischer Größen wie Temperatur oder Luftfeuchte ist ein wichtiges Werkzeug zur Prozeßoptimierung und -überwachung. Dabei haben drahtlose Sensoren gegenüber kabelgebundenen Sensoren den Vorteil, daß sie einfach an schwer zugänglichen Orten zu installieren sind. Als Funkschnittstelle eines Temperatursensors eignen sich IEEE-802.15.4-Transceiver wegen der einfachen Ansteuerung und dem geringen Strombedarf. Im Beitrag werden am Beispiel einer verteilten Temperaturerfassung alle Schritte von der Firmwareentwicklung bis hin zur Datenaufbereitung im Webserver demonstriert.

1 Funkstandards zur Übertragung von Sensordaten

Im PC-Umfeld sind mit Wi-Fi (IEEE 802.11) und Bluetooth (IEEE 802.15.1) derzeit zwei Funktechniken verbreitet. Wi-Fi ist als Netzwerkinterface für hohe Datenraten von 54 MBit/s bis 300 MBit/s spezifiziert. Im Gegesatz dazu wurde Bluetooth speziell für die Nahbereichskommunikation mit Datenraten von 732,2 kbit/s bis 2,1 MBit/s spezifiziert.

Der Funkstandard IEEE-802.15.4 [2] wurde speziell für batteriebetriebene Sensoranwendungen mit niedrigen Datenraten im Bereich von 20 bis 250 kbit/s spezifiziert, wodurch mit diesen Funkknoten eine Batterielebensdauer von mehreren Jahren erreicht werden kann.

Alle drei Funkstandards können im 2.4-GHz-ISM-Frequenzband (Industrial, Scientific and Medical Band) [1] arbeiten. Die Reichweite variiert je nach eingesetzter Antenne und den Umgebungsbedingungen im Bereich von 10 bis 200 m.

Im 802.15.4-Standard wird die physikalische Übertragung (Modulationsverfahren, Funkkanäle, Rahmenaufbau) und der Medienzugriff (Kanalbelegung mit CSMA-CA, Verbindungsauf- und -abbau, Datenaustausch) beschrieben. Der ZigBee-Standard setzt auf dem 802.15.4-MAC-Layer auf und definiert die Netzwerkfunktionalität (Routing) und die Anwendungsprofile (z.B. Home Automation, Smart Energy, RF4CE). Als alternative Netzwerkschicht zu ZigBee gibt es noch das von der IETF spezifizierte 6LoWPAN-Protokoll, das kompatiblel zu IPV6 ist. Unabhängig von den standardisierten Protokollen für komplexe Netzwerkszenarien können die Transceiver auch direkt angesteuert werden und mit einem sehr stark vereinfachten Protokoll betrieben werden.

2 Funkmodule

IEEE 802.15.4 Transceiver werden unter anderem von den Firmen AMBER wireless, Atmel, Freescale Semiconductor und Texas Instruments produziert. Die Atmel-Schaltkreise der Familie AT86RF2xx werden über eine SPI-Schnittstelle und wenige digitale IO-Pins angesteuert.

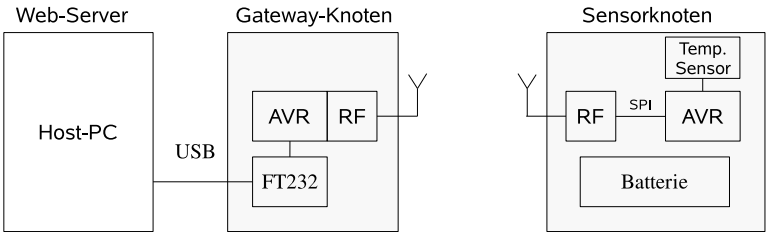


Abbildung 1: Blockschaltbild des Sensornetzwerks

Im Gegensatz dazu hat der Mikrocontroller ATmega128RFA1 den 802.15.4-Transceiver bereits integriert. Die Schaltkreise selbst nutzen einem Bastler ohne SMD-Löt- und UHF-Design-Erfahrung jedoch relativ wenig. Für eigene Exeprimente werden vor allem Module und PCBs eingesetzt.

Bei **Transceiver-Modulen** handelt es sich um kleine Platinen, die neben dem Transceiver und Controller meist auch mit einer keramischen Chip-Antenne oder einer U.FL-Antennenbuchse bestückt sind. An den Rändern der Module befinden sich entweder SMD-Kontakte oder Lötstiftleisten. Damit ist es möglich, ein Modul wie einen großen Schaltkreis auf der eigenen Platine zu verwenden. Für erste Experimente gibt es für die Module meist sogenannte Breakout Boards. Das sind einfache Platinen, die die SMD-Anschlüsse des Moduls auf grössere Stecker, Stiftleisten oder Schraubklemmen umsetzen. Damit kann eine eigene Schaltung mal eben schnell in „fliegender“ Verdrahtung aufgebaut werden. Beispiele sind u.a. die *ANY-Module* der Adaptive Network Solutions GmbH, die *deRFA-Module* der dresden elektronik ingenieurtechnik gmbh, sowie die *ZigBit-Module* von Atmel (ehemals MeshNetics).

Darüber hinaus sind **Printed Circuit Boards** (PCB) verfügbar, die mit einem seriellen oder USB-Interface, Steckleisten und Batteriehaltern ausgerüstet sind. Diese etwas größeren Platinen eignen sich zum direkten Einbau in ein Gerät. Als Beispiele seien hier die *ANY-Bricks* und *ANY-Dongles* der Adaptive Network Solutions GmbH, die *ICRadio-Minis* und *ICRadio-Sticks* der In-Circuit GmbH, die *Radio Controller Boards (RCBs)* der dresden elektronik ingenieurtechnik gmbh, sowie der *AVR-Raven* und der *AVR-RZUSB-Stick* von Atmel genannt.

Ebenfalls gibt es die ersten **Eigenbau-PCBs**, z.B. das *Muse231* vom Ingenieurbüro Dietzsch und Thiele PartG, das *RadioFaro-Board* von Daniel Thiele, das *tiny230 Radio Board* von Jörg Wunsch, sowie *Zigduino* von Logos Electromechanical LLC. Besonderer Augenmerk ist beim Eigenbau auf die rechtlichen Aspekte zu legen; dies ist bei Nutzung fertiger Module oder PCBs etwas einfacher.

3 Ein einfaches Sensornetzwerk

Das vorgestellte Sensornetzwerk besteht aus einem Gateway-Knoten und einem oder mehreren Sensorknoten. Der Gateway-Knoten ist über eine serielle Schnittstelle mit dem PC verbunden.

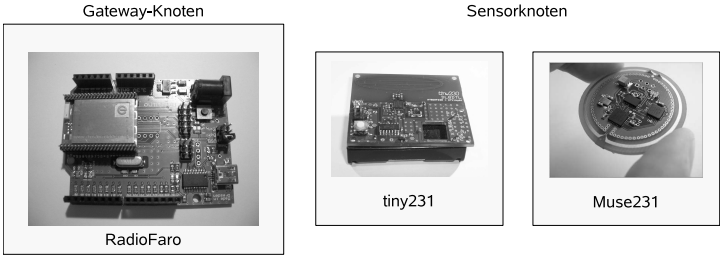


Abbildung 2: Die verwendeten Funkknoten

Die batteriebetriebenen Sensorknoten messen zyklisch die Temperatur und / oder andere physikalische Größen und senden die Messwerte zum Gateway-Knoten, der die empfangenen Daten über die serielle Schnittstelle ausgibt. Von dort können sie im PC, zum Beispiel von einem Python-Script weiterverarbeitet werden.

Am Ende des Artikels wird ein kleines Script vorgestellt, das die empfangenen Sensordaten als HTML-Seite aufbereitet und über einen Webserver im Netz bereitstellt. Das vorgestellte Beispiel soll vor allem das Konzept erklären und ist nicht auf Sicherheit optimiert.

Als **Gateway-Knoten** soll hier das *Radifaro-Board* von Daniel Thiele verwendet werden. Dieses Board ist kompatibel zum Arduino Duemilanove, wobei der ATmega168 gegen ein ATmega128RFA1-Modul von dresden elektronik ausgetauscht wurde. Als **Sensorknoten** werden das *tiny231* von Jörg Wunsch und das *Muse231* vom Ingenieurbüro Dietzsch und Thiele eingesetzt. Beide Boards sind mit einem AT86RF231-Transceiver bestückt, die eingesetzten Microcontroller hingegen sind unterschiedlich, beim *Muse231* ist es ein ATmega88PA und beim *tiny231* ein ATtiny84. Weitere Informationen zu den Boards befinden sich auf der Hompage des Vortrages.

Die nächsten Abschnitte beschreiben die Hard- und Sofftwarevoraussetzungen zur Programmierung der Funkmodule sowie die schrittweise Inbetriebnahme des gesamten Systems.

4 Entwicklungsumgebung

Auswahl eines Programmieradapters

Die Firmware des Microcontrollers kann mit einem Programmieradapter oder mit einem Bootloader geändert werden. Besitzt man Funkmodule mit einem vorprogrammierten Bootloader, kann man zunächst auf die Anschaffung eines Programmieradapters verzichten. Spätestens jedoch, wenn man sich durch einen Programmierfehler aus dem Bootloader ausgesperrt hat oder die Fuses des Controllers geändert werden sollen, kommt man um die Anschaffung eines Programmieradapters nicht herum.

Die Microcontroller der AVR-Familie können über eine 4-Drahtschnittstelle, dem sog. *ISP-Interface* oder über das *JTAG-Interface* programmiert werden. Bekannte ISP-Programmieradapter sind u.a. PonyProg (sp12), USBasp, Atmel AVR-ISP und Arduino. Als JTAG Programmieradapter kommen Atmels AVR-Dragon und JTAG-ICE MkII zum Einsatz.

Der AVR-Dragon Programmieradapter ist für Bastler eine gute Wahl, da er zum einen erschwinglich ist und zum anderen alle Programmier- und Debugmodi von AVR-Controllern unterstützt. Man muss sich allerdings um ein Gehäuse und ggf. die Bestückung einiger Stiftleisten selbst bemühen.

Installation der Software

Die **AVR-Toolchain** für Linux besteht aus dem GCC-Compiler, dem GNU-Linker aus den *binutils* und der AVR-libc. Der Programmieradapter wird mit dem Programm avrdude angesteuert und das Debugging erfolgt mit AVaRICE und avr-gdb. Unter Ubuntu erfolgt die Installation der Pakete mit dem Befehl:

```
> sudo apt-get install avr-libc binutils-avr gcc-avr \
                       avrdude avarice gdb-avr
```

Das **µracoli-Projekt** stellt für Atmel Transceiver und Microcontroller eine Treiber-Bibliothek und Beispielanwendungen bereit. Die Software ist unter der *modified BSD license* veröffentlicht und kann somit für Open- und Closed-Source-Projekte problemlos eingesetzt werden. Mit den bereitgestellten Funktionen kann die Funkkommunikation zwischen zwei oder mehreren Knoten einfach implementiert werden, ohne daß man dazu vorher komplexe Protokollregelwerke gelesen und verstanden haben muß. Die Pakete `uracoli-src-<version>.zip` und `uracoli-doc-<version>.zip` können von http://uracoli.nongnu.org/download.html bezogen werden.

Für die Webserver Applikation wird später die Scriptsprache **Python** und das Modul **pyserial** benötigt und wie folgt installiert:

```
> sudo apt-get install python python-serial
```

Anschließen der Hardware

Nach der Software-Installation kann der Gateway-Knoten an den PC angeschlossen werden. Um die zum Board zugehörige serielle Schnittstelle zu ermitteln, verwendet man den Befehl `dmesg`.

```
 > dmesg | grep tty
00:08: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:09: ttyS1 at I/O 0x2f8 (irq = 3) is a 16550A
usb 2-3.3: FTDI USB Serial Device converter now attached
    to ttyUSB0
```

Wenn das Board mit einen USB Anschluß ausgestattet ist, dann ist der Gerätename entweder `/dev/ttyUSBx` oder `/dev/ttyACMx`. Die *STK541 USB Adapter Boards* von Atmel und die *Sensor Terminal Boards* von dresden elektronik melden sich mit herstellerspezifischen Vendor und Device IDs am Bus an und werden ab der Linux-Kernelversion 2.6.29 automatisch als serielle Schnittstellen erkannt.

Einrichtung der Funkknoten

Zunächst wird aus dem μracoli-Paket die Wireless-UART-Firmware `wuart_<board>.hex` auf den **Gateway-Knoten** geladen. Mit diesem Programm wird eine bidirektionale transparente Datenverbindung zwischen Funk- und UART-Interface hergestellt. Welchen Namen man für *<board>* einsetzt und wie die Fuses richtig programmiert werden, entnimmt man dem Abschnitt „Boards and Modules“ der Dokumentation.

Bei der Inbetriebnahme eines neuen Boards müssen zunächst die Fuses des Controllers richtig programmiert werden. Die Fuse-Werte stellen neben der Taktfrequenz und der Taktquelle weitere Parameter, z.B. die Größe des Bootloaders und die Verfügbarkeit von SPI- oder JTAG-Interface ein. Das Programmieren der Fuses sollte sehr sorgfältig erfolgen, da bei falscher Wahl der Controller u.U. nicht mehr programmierbar ist. Eine wertvolle Hilfe bei der Fuse-Definition ist die Webanwendung Fusecalc [5]. Das Prorammieren der Fuses und der Firmware für ein *RadioFaro*-Board über ein JTAG-ICE MkII erfolgt mit den folgenden Befehlen:

```
> avrdude -P usb -p m128rfa1 -c jtag2 -U lfuse:w:0xff:m\
          -U hfuse:w:0xda:m -U efuse:w:0xf5:m\
> avrdude -P usb -p m128rfa1 -c jtag2 \
          -U bin/wuart_radiofaro.hex
```

Im nächsten Schritt wird ein Terminalprogramm auf der betreffenden Schnittstelle geöffnet. Anstelle vom Klassiker `minicom` kann auch das Script `miniterm.py` [4] aus den Beispielen des pySerial-Modules verwendet werden. Nach einem Reset des Gateway-Knotens sieht man die Startmeldung der Wireless-UART-Firmware im Terminalfenster.

```
 > python miniterm.py -p /dev/ttyUSB1
--- Miniterm on /dev/ttyUSB1: 9600,8,N,1 ---
--- Quit: Ctrl+]  |  Menu: Ctrl+T | Help: Ctrl+T followed
    by Ctrl+H ---
Wuart 0.2 chan=17
```

Ein einfaches Test-Programm `xmpl_linbuf_tx_<board>` für den **Sensorknoten** muß zunächst compiliert werden. Das erfolgt im Verzeichnis `uracoli-src-<version>/xmpl` für ein `Muse231` Board mit dem Befehl:

```
> make -f xmpl_linbuf_tx.mk muse231
```

Danach wird die Firmware `bin/xmpl_linbuf_tx_muse231.hex` zusammen mit den passenden Fuse-Werten auf den Sensorknoten geladen. Wenn nach einem Reset des Boards eine LED

am Gateway-Knoten blinkt und im Fenster des Terminalprogramms ein zyklischer Text ausgegeben wird (*A wonderful bird ...*), besteht eine Funkverbindung zwischen den beiden Knoten.

5 Die Sensoranwendung

Die Sensor-Anwendung erfaßt zyklisch Meßdaten und sendet diese zum Gateway-Knoten. Im Verzeichnis `Src/Sensor` befinden sich die Firmware `bin/sensorapp_<board>.hex`, das GNU-Makefile `sensorapp.mk`, der C-Source-Code `sensorapp.c`, sowie das Headerfile `sensorapp.h`. Die Programmierung des Sensorknotens erfolgt wie oben beschrieben.

Die `main`-Funktion der Sensor-Anwendung führt die Funktion `app_init()` einmal aus und ruft danach in einer Endlosschleife abwechselnd `app_task()` und `do_sleep()` auf.

```
int main(void) {
    app_init();
    timer_start(tmr_transmit, T_TX_PERIOD, 0);
    while(1) {
        app_task();
        do_sleep();
    }
}
```

Die Funktion `app_task()` implementiert einen Zustandsautomaten, der die Messung der Temperatur durchfürt und das Ergebnis per Funk versendet. Nach dem Versenden des Rahmens wird die Funktion `usr_radio_tx_done()` von der Interrupt Service Routine des Transceivers aufgerufen. In der Funktion wird ein Timer gestartet, der seinerseits bei Ablauf das Event `APP_MEASURE` generiert.

```
void app_task(void) {
    app_event_t state;
    cli();
    state = app_event;
    app_event = APP_IDLE;
    sei();
    do {
        switch(state) {
            case APP_MEASURE:
                temp_start();
                state = APP_RESULT;
                break;
            case APP_RESULT:
                sprintf(TxFrame.data,
                        "n:␣0x%04x,␣t:␣%d\n\r",
                        TxFrame.src, temp_get());
                radio_set_state(STATE_TXAUTO);
                radio_send_frame(sizeof(TxFrame),
                      (uint8_t *)&TxFrame, 0);
                state = APP_IDLE;
                break;
```

```
            default:
                state = APP_IDLE;
                break;
        }
    } while(state != APP_IDLE);
}
```

6 Webserver

Über die serielle Schnittstelle des Host-PCs werden nun die Daten der Sensorknoten empfangen und es können auch Daten an diese gesendet werden. Als einfaches Beispiel wird hier das Auslesen und Darstellen der von den Sensorknoten gemessenen Temperaturen mit Hilfe eines Webservers gezeigt. Dieser ist in Python geschrieben und verwendet die Module `SimpleHTTPServer` und `SocketServer`. Das Server wird mit dem Befehl `python webserver.py` gestartet und ist danach unter `http://localhost:8080` erreichbar.

```
import SocketServer
import SimpleHTTPServer
...
class CustomHandler(SimpleHTTPServer.
    SimpleHTTPRequestHandler):
    def do_GET(self):
        global sdev
        self.send_response(200)
        self.send_header('Content-type','text/html')
        self.end_headers()
        txt = sdev.build_sensor_html_table()
        self.wfile.write(header() + txt + footer())

if __name__ == "__main__":
    ...
    httpd = SocketServer.ThreadingTCPServer(
                    ('localhost', PORT),CustomHandler)
    ...
    httpd.serve_forever()
```

In einem separaten Thread werden die ankommenden Sensordaten von der seriellen Schnittstelle gelesen und in einem Dictionary gespeichert. Die Funktion `build_sensor_html_table()` baut daraus eine HTML-Tabelle. Diese Methode wird von der oben beschriebenen Serverfunktion `do_GET()` aus aufgerufen.

```
class Device:
    def __init__(self,port, baud, verbose=0):
        self.sport = serial.Serial(port, baud)
        self.rxThread = threading.Thread(target = self.
            _rx_thread_)
        ...
        self.rxThread.start()
        self.sensor_data = {} # collects sensor data
```

```
    # function running in thread
    def _rx_thread_(self):
        while 1:
            sline = self.sport.readline().strip()
            tim = int(time.time())
            (node, temp) = self.parse_line(sline)
            self.sensor_data[node] = (tim, temp)

    def build_sensor_html_table(self):
        ...
        for node in nodes:
            ...
            (tim,temp) = self.sensor_data[node]
            if (int(time.time()) - tim) > 10:
                ret += '<td style="color:red">'\
                       '%s</td>\n' % temp
            else:
                ret += '<td>%s</td>\n' % temp
        ...
        return ret
```

7 Ausblick

Im Artikel wurde die Einrichtung und Inbetriebnahme einer Entwicklungsumgebung für Atmel-basierte IEEE-802.15.4 Funkmodule beschrieben. Mit der verfügbaren Open-Source-Software ist es möglich, einfache kleine Sensornetzwerke aufzubauen. Die gezeigten Beispiele sollen als Anregung dienen und Lust auf eigene Experimente machen. Über Anregungen, Fragen und aktive Hilfe freut sich das μracoli-Team.

Literatur

[1] *ISM-Band*. Wikipedia
http://de.wikipedia.org/wiki/ISM-Band

[2] *IEEE 802.15 Standards*. IEEE
http://standards.ieee.org/about/get/802/802.15.html

[3] *μracoli - The Micro Controller Radio Communication Library*. Homepage
http://uracoli.nongnu.org

[4] *Examples*. pySerial v2.5 Documentation
http://pyserial.sourceforge.net/examples.html

[5] *Engbedded Atmel AVR Fuse Calculator*. Mark Hämmerling
http://www.engbedded.com/fusecalc

Embedded Linux Distributionen - Das Ende der schwarzen Magie?

Daniel Kriesten krid@hrz.tu-chemnitz.de
Sebastian Kratzert krase@hrz.tu-chemnitz.de
Ulrich Heinkel heinkel@hrz.tu-chemnitz.de

Linux hat einen festen Platz im Markt der Betriebssysteme für eingebettete Systeme. Eine Vielzahl an Systemen, die den Entwickler bei der Erstellung seines embedded Linux unterstützen stehen zur Verfügung. Während alle Systeme die Grundaufgaben eines Buildsystems, also das Erstellen einer Toolchain und eines Root-Filesystems erfüllen, unterscheiden sie sich in anderen Details wie der Patch-Philosophie oder der Unterstützung von Teams gewaltig. Dieser Artikel zeigt einige Faktoren, die neben den unterstützten Targets und der Anzahl angebotener Pakete bei der Auswahl eines geeigneten Buildsystems von Interesse sein können.

1 Einleitung

Den Einsatz von Linux als Betriebssystem (BS) für eingebettete Systeme (ES) muss man eigentlich nicht mehr motivieren. Die Vielzahl an unterstützten Architekturen und Hardware, die große Auswahl an Software und nicht zuletzt die Bandbreite an Einsatzmöglichkeiten, vom hochverfügbaren Server über Desktopsysteme bis zu Echtzeitsystemen sprechen schließlich für sich. Die aktive Nutzer- und Entwicklergemeinde und der damit verbundene Support in teilweise professioneller Qualität runden das Bild im positiven Sinne ab.

Trotzdem nimmt die Entwicklung von Firmware für ressourcenschwache ES noch immer eine Sonderstellung ein. Kommen im Desktop- und Serverumfeld meist Distributionen zum Einsatz, die die verwendete Software als kompilierte Binärpakete ausliefern, wird die Firmware für ein ES oftmals von Hand erstellt. Der Entwickler kann dabei auf die Unterstützung durch kommerzielle oder freie Systeme, im weiteren als *Buildsysteme* bezeichnet, zurückgreifen. Kommerzielle Produkte binden ihn dabei an den Hersteller der jeweiligen Lösung, während es bei den freien Projekten oft an der Dokumentation oder einer einheitlichen Nutzeroberfläche mangelt.

Im Hobbybereich wird gern bestimmte Hardware, wie die Router eines Herstellers oder die Entwicklerkits für ein spezielles System-on-Chip (SoC) verwendet, die dann direkt von den Buildsystemen unterstützt wird. Im kommerziellen Umfeld hingegen sollten Buildsysteme leicht um eigenen Plattformen erweiterbar sein. Abseits der technischen Beurteilung stellt sich mit der wachsenden Anzahl von Entwicklern und Produkten eines Unternehmens auch die Frage nach der Reproduzierbarkeit sowie den Möglichkeiten eines Buildsystems im Team genutzt zu werden. Speziell diesen Punkten soll der vorliegende Artikel Rechnung tragen.

Dazu geht Abschnitt 2 auf verwandte Arbeiten ein, während Abschnitt 3 die für die Untersuchungen verwendete Hardware und Software vorstellt. Abschnitt 4 beleuchtet die Buildsysteme und die damit erreichten Ergebnisse. In Abschnitt 5 finden sich Zusammenfassung und Ausblick.

2 Verwandte Arbeiten

Viele Bücher und Zeitschriften und nahezu unzählige Onlineartikel befassen sich mit Linux für eingebettete Systeme. Oft bieten sie dem Leser einen einstieg in die Softwareentwicklung für eingebettete Systeme, erklären den Aufbau des Betriebssystems, geben Informationen zu den Besonderheiten bei der Hardware von ES und erläutern ausführlich das Erstellen einer Cross-Toolchain sowie den Vorgang des Cross-Kompilierens. Buildsysteme werden in Fachbüchern ebenfalls erwähnt, doch meist nicht im Detail vorgestellt. Zeitschriften liefern hier schon eher Informationen, doch oft bilden Onlineartikel und Internetforen die ergiebigsten Informationsquellen. Dieser Artikel stützt sich hauptsächlich auf eigene Erfahrungen und einige Bücher [YMBYG08, SGD09, Hal06]. Die Erkenntnisse der Studenten um Professor Högel (HS Augsburg), speziell [Kam09, GR09], sowie Artikel in „c't Magazin für Computertechnik“ [SH10] und im „LinuxMagazin“ [Reb08] decken sich mit den Erfahrungen der Autoren.

3 Verwendete Hard- und Software

Der folgende Abschnitt gibt einen Überblick über die für die weiteren Arbeiten verwendete Hard- und Software.

3.1 Hardware

Für die Betrachtung der Buildsysteme wurden verschiedene Hardwareplattformen genutzt. Zum einen das auch in der Community beliebte *ATNGW100* von Atmel [Atm06], dass auf dem AVR32AP7000 SoC basiert. Dieses SoC wird schon seit geraumer Zeit von Linux unterstützt, ist andererseits aber so selten, dass es noch fast als exotisch gelten kann. Dies spiegelt sich vor allem in der noch nicht ausgereiften Unterstützung durch die binutils und die GNU Compiler Collection (GCC) wieder, was das Erstellen der Cross-Toolchain erschwert. Ein ebenfalls nicht ganz alltägliches System ist das *XUP-Board* [Xil05] aus dem Xilinx University Programm, dass anstelle eines Prozessors ein Field Programmable Gate Array (FPGA) mit PowerPC 405 Kern besitzt. Die Unterstützung für PowerPC-CPUs ist weit fortgeschritten und das XUP-Board wird von Linux schon einige Zeit unterstützt. Trotzdem ist die hohe Flexibilität des FPGA nicht alltäglich. Das dritte System ist eine Entwicklung der Firma Unicontrol [Uni10] und basiert auf dem weit verbreiteten TQM5200 System-on-Module (SoM). Kern ist ein Motorola MPC5200 SoC mit PowerPC 603e Kern. Mit dem Silicon Motion SM501 verfügt das TQM5200 über eine im embedded Bereich häufiger zu findende Graphics Processing Unit (Grafikprozessor) (GPU). Auch wenn der Kernel den MPC5200 und das TQM5200 schon seit langem unterstützt, sind für den SM501, der in diesem System nicht per PCI an die CPU angebunden ist, zusätzliche Patches notwendig. Weitere Anpassungen werden durch die Hardwarevorgaben des Unicontrol Systems gemacht.

Die gewählte Hardware stellt eine Auswahl dar, die sich von durchschnittlichen Hobbyplattformen unterscheidet, ohne zu starke Besonderheiten aufzuweisen. Viele Buildsysteme unterstützen inzwischen das ATNGW100, während das XUP-Board trotz guter Unterstützung

im Kernel nur selten zu finden ist. Mit dem System der Unicontrol GmbH kommt ein auf Standardhardware basierendes, nicht unterstütztes System hinzu, dessen Hauptkomponente eine sehr gute Unterstützung im Kernel erfährt.

3.2 Software

Bei der Verwirklichung eigener Projekte kommt es schnell dazu, dass der vom Buildsystem gebotene Softwareumfang nicht ausreicht. Entweder unterstützen angebotene Programme einige Features nicht, oder benötigte Software wird nicht mit dem System mitgeliefert. in beiden Fällen ist eine Anpassung oder Erweiterung notwendig. Im Verlauf der hier dargestellten Untersuchungen traten beide Fälle auf. Weiter kam es dazu, dass Fehler in Programmpaketen durch eigene Patches behoben werden mussten.

Unter anderem waren zusätzliche Kernelpatches für die Integration des SM501 sowie die Nutzung des Synchronous Serial Controller (SSC) des AVR32 notwendig. Für eine eigene grafische Nutzerschnittstelle wurde Python mit Tkinter-Unterstützung benötigt. Ebenso wurden eigene Pakete hinzugefügt. Beim Systemstart sollten außerdem bestimmte Dienste ausgeführt werden. Diese Anforderungen stellen Analog zur Auswahl der Hardware eine Mischung aus alltäglichen und eher seltenen Fällen dar.

4 Buildsysteme

Der folgende Abschnitte stellt Buildsysteme im Allgemeinen kurz vor. Anschließend werden die Punkte, auf denen das Augenmerk der Autoren lag weiter präzisiert und die Erfahrungen mit den Untersuchten Systemen genauer betrachtet.

4.1 Arten von Buildsystemen

Es gibt sowohl freie als auch kommerzielle Software, die den Entwickler bei der Erstellung von Linux Firm- und Software für ES unterstützt. Im Allgemeinen kann man diese Bildsysteme in drei Gruppen unterteilen. Systeme der ersten Gruppe unterstützen beim Erstellen der benötigten (Cross-)Toolchain, meist bestehend aus dem Compiler (z. B. GCC), einer C-Bibliothek (wie glibc, uClibc oder eglibc), den binutils und den Headern des Linux-Kernel. Vertreter dieser Gruppe sind *crossdev* [ohn11], *crosstool* [Keg06], *crosstool-ng* [Mor11] und OSELAS.Toolchain( ) [Pen11]. Die zweite Gruppe liefert, ähnlich den Distributionen für Server und Desktoprechner, kompilierte Binärpakete aus. Die dritte und vermutlich größte Gruppe bilden die Buildsysteme, die ein Linux Firmware-Image aus den Quellpaketen ausgewählter Programme und Bibliotheken erstellt. Die meisten dieser Systeme bringen auch eine eigene Cross-Toolchain mit. Die in diesem Beitrag betrachteten Buildsysteme gehören alle der dritten Gruppe an. Im Einzelnen sind es *Buildroot*, *Embedded Linux Development Kit (ELDK)*, *Gentoo Linux*, *OpenWRT* und *OpenEmbedded* und *PTXdist*.

4.2 Aufgaben eines Buildsystems

Die Grundaufgabe eines Buildsystems ist die Verwaltung von Softwarepaketen zur Erstellung eines Firmware-Images. Dazu sollte es je nach Bedarf eine Toolchain bzw. Cross-Toolchain inklusive Debugger für das gewählte Target ebenso bereitstellen, wie eine Auswahl an Bibliotheken und Anwendungsprogrammen. Weiterhin sollte es dem Nutzer möglich sein, die Paketauswahl sowie weitere Optionen anzupassen. Die in Abschnitt 4.1 vorgestellten Systeme erfüllen diese Anforderungen, wobei sie sich in der Anzahl der unterstützten Targets, der Anzahl der angebotenen Softwarepakete und der Nutzerschnittstelle unterscheiden. Da das Ziel dieses Beitrags nicht die Vorstellung einzelner Buildsysteme sowie ihrer Besonderheiten ist, wird an dieser Stelle auf die bereits zitierte Literatur sowie die Internetauftritte der einzelnen Buildsysteme verwiesen.

Alle Buildsysteme, die die Firmware aus Quellpaketen einzelner Softwareprojekte erstellen, müssen für jedes der unterstützten Softwareprojekte eine „Bauanleitung“ mitbringen, die die Einzelnen Schritte vom Herunterladen des Quellcodes über das Entpacken bis hin zum Kompilieren und zur Installation beschreibt. Diese „Bauanleitungen“ werden gemeinhin als *Recipes* (engl. Rezepte) oder *Rule Files* bezeichnet.

Viele Softwareprojekte beachten die Besonderheiten des Cross-Kompilierens nicht oder nicht ausreichend und müssen vor dem Cross-Kompilieren durch Patches modifiziert werden. Daher ist eine weitere Grundaufgabe und gleichzeitig ein weiteres Bewertungskriterium der Buildsysteme die Verfügbarkeit und der Umgang mit Patches. Die *Patch-Philosophie*, also die Herangehensweise in den einzelnen Projekten, unterscheidet sich dabei deutlich. So versucht beispielsweise PTXdist, mit wenigen Patches zu arbeiten und verstärkt die notwendigen Änderungen direkt in die Projekte zu tragen. OpenWRT hingegen ist für seine vielen Patches bekannt. Auch das *Patch-Management* unterscheidet sich deutlich, wobei es zwei grundsätzliche Herangehensweisen gibt. Ein Weg, wie ihn beispielsweise OpenWRT geht, ist die Nutzung eines Overlay-Mechanismus, bei dem Patches aus mehreren Quellen, z. B. erst allgemeine Patches der Distribution und dann spezielle für ein Board, nacheinander angewendet werden (*additives patchen*). Diesen Ansatz findet man beispielsweise bei Buildroot und OpenWRT. Die Alternative dazu ist die Nutzung nur einer Quelle, auch wenn Patches an mehreren Stellen hinterlegt werden können (*exklusives patchen*). Über eine Priorisierung der Quellen kann so ein Verhalten erzeugt werden, wie man es beispielsweise auch von Konfigurationsdateien kennt und bei dem beispielsweise Patches im Projektverzeichnis die in globalen Verzeichnissen überlagern. Dieser Ansatz wird beispielsweise bei PTXdist verfolgt.

4.3 Erweiterte Anforderungen

Für die Auswahl des geeigneten Buildsystems können neben den unterstützten Targets, der Anzahl angebotener Pakete und der Patch-Philosophie weitere Punkte von entscheidender Bedeutung sein. Bild 1 gibt einen Überblick über die im weiteren Text betrachteten Punkte.

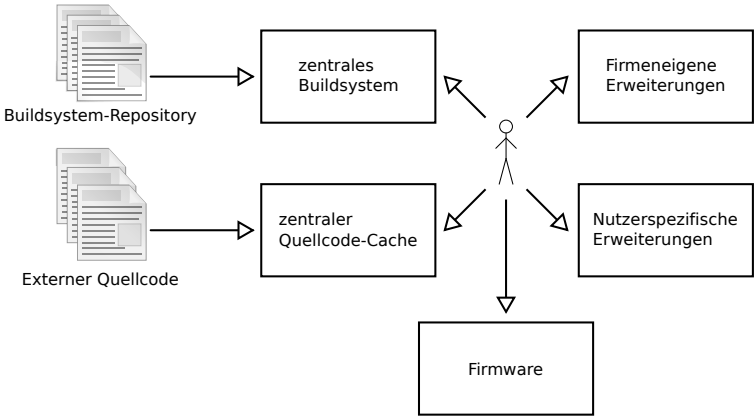


Figure 1: Schmatische Darstellung des idealen Buildsystems

Dokumentation, Support und Aktivität der Community sind drei wesentliche Punkte. Eine gute und aktuelle Dokumentation erleichtert den Einstieg in die Arbeit mit dem Buildsystem. Dem gleichen Punkt trägt auch die Komplexität des Buildsystems Rechnung. Allerdings zeigt sich, dass bei den getesteten Systemen Komplexität und Flexibilität in direktem Zusammenhang stehen. Je einfacher sich ein Buildsystem gibt, desto schwieriger ist die Erweiterung um eigene Pakete und vor allen Dingen um eigene Systeme. Dabei stecken die Probleme vor allen Dingen in Details wie der Patchverwaltung oder der Verzeichnisstruktur. Außerdem zeigte sich während der durchgeführten Arbeiten, dass spezialisierte Buildsysteme, wie OpenWRT als Routerdistribution, nicht ohne weiteres für andere Anwendungsfälle geeignet sind.

Wie bereits in 1 erwähnt, ist die Pflege der Dokumentation bei allen Projekten ein schwieriger Punkt, wobei die Federführend durch eine Firma gepflegten Projekte (ELDK und PTXdist) hier etwas besser abschneiden, als die Community-Projekte. Fragen lassen sich über die Foren, Mailinglisten oder Internet Relay Chat (IRC)-Kanäle aber schnell beantworten. Für den Einstieg in ein Buildsystem sind die verfügbaren Beispiele ebenfalls von Nutzen und werden oft als Ersatz für die Dokumentation referenziert.

Ist ein Produkt erst einmal auf dem Markt, endet meist die aktive Entwicklungsphase. Alle Aufzeichnungen und der Quellcode werden archiviert und die Entwickler wenden sich neuen Aufgaben zu oder verlassen gar die Firma. Kommt nun eine Supportanfrage, ist es notwendig, dass Projekt schnell und unkompliziert zu reaktivieren. Hier spielt die Reproduzierbarkeit der alten Arbeiten eine entscheidende Rolle und ein Buildsystem, dass diesen Punkt berücksichtigt, kann entscheidend zur Vereinfachung dieses Prozess beitragen. Bisherige Lösungen beinhalten oft die Archivierung kompletter Projektverzeichnisse, inklusive aller Binaries und temporärer Dateien. Doch selbst in diesem Fall kann es dazu kommen, dass die Reaktivierung aus unbekannten gründen fehlschlägt. Durch die Nutzung von Versionsverwaltungssystemen (VCS) können die hier betrachteten Buildsysteme, mit Ausnahme von Gentoo, jederzeit

wieder auf den Stand gebracht werden, den sie zum Zeitpunkt der Firmware-Erstellung hatten, so dieser auch dokumentiert wurde. Durch die Nutzung der Funktionen des VCS (Branches, Tags), kann das Wiederherstellen einer Version weiter erleichtert werden. Einen interessanten Ansatz verfolgt PTXdist, bei dem ein Stand des Buildsystems auf dem Host installiert wird (Standardmäßig unter `/usr/local/{lib,bin}`) und somit in unveränderter Form zur Verfügung steht.

Ein weiterer Punkt ist die Verfügbarkeit des Quellcodes, der bei den vorgestellten Buildsystemen standardmäßig aus dem Internet geladen wird. So kann nicht sichergestellt werden, dass eine bestimmte Version einer Software noch verfügbar ist. Dieser Umstand kann durch einen Firmeneigenen Proxy oder ein Firmeneigenes Softwareverzeichnis sichergestellt werden. Alle Buildsysteme unterstützen die priorisierte Nutzung bestimmter Server sowie die Nutzung zentraler Downloadverzeichnisse, was auch den verursachten Traffic bei häufigen Builds verringert.

Mehrere Gründe sprechen für den Einsatz eines zentral organisierten Buildsystems innerhalb einer Firma. Für die Arbeit im Team müssen alle Mitglieder auf einen einheitlichen Stand zurückgreifen können. Auch die parallele Entwicklung mehrerer Produkte kann durch die zentrale Verwaltung von Patches, Recipes und Konfigurationen vereinfacht werden. Hier verfolgen die untersuchten Buildsysteme sehr unterschiedliche Ansätze. OpenWRT sieht von Haus aus nicht vor, dass mehrere Entwickler mit einem Grundsystem arbeiten. Allerdings können zusätzliche Pakete (sog. feeds) bequem hinzugefügt werden. Hier wurde eine eigene Methodik entwickelt, die die Arbeit mit einer Version des Grundsystems vereinfacht. Dazu wird ein eigenes git-Repository [Sch09] gepflegt, das in unregelmäßigen Abständen mit dem Subversion-Repository [Nag05] von OpenWRT abgeglichen wird. Nach jedem Abgleich werden die durch uns gemachten Erweiterungen hinzugefügt, angepasst und getestet, bevor intern auf das so entstandene Basissystem umgestellt wird. Erweiterungspakete werden in einem eigenen feed gepflegt. Buildroot unterstützt mit sogenannten *Out of Tree Builds* das Arbeiten mit einem zentralen Buildsystem. Dabei kann der Entwickler optional ein Verzeichnis angeben, in dem er sein Projekt erstellen will. Da Buildroot auf Makefiles basiert, kann auch der Aufruf der einzelnen Kommandos von einer beliebigen Stelle im System erfolgen. Bei OpenEmbedded und PTXdist ist die Philosophie, das Projektverzeichnis und das Verzeichnis des Buildsystems voneinander zu trennen, fest verankert. Beide Systeme bauen nicht nur auf diesem Konzept auf, sonder unterstützen dieses Vorgehen auch in der Art, wie beispielsweise Patches oder eigene Pakete zum System hinzugefügt werden. Bei beiden Systemen kann die Verwaltung eigener Patches, eigener Pakete und auch eigener Konfigurationen (Abgestimmt auf die firmeneigene Hard- und Software) komplett getrennt vom zentralen Buildsystem erfolgen. Innerhalb eines Unternehmens ist es so möglich, Interna bequem auf den eigenen Servern zu halten und trotzdem bequem auf die Entwicklungen im Community-Repository zuzugreifen.

Ein letzter Punkt ist die Erweiterung der Buildsysteme um eigene Pakete und eigene Targets. Das einbinden neuer Pakete ist in allen Systemen vorgesehen und problemlos möglich. Die meisten Systeme unterstützen den Entwickler durch Mechanismen, die das Hinzufügen von Paketen ohne einen Eingriff in das eigentliche Buildsystem zulassen (OpenWRT-Feeds, OpenEmbedded-Overlays, `ptxdist newpackage`, ELDK per rpm). Mehr Aufwand bedeutet oft die Integration eines neuen Targets. Hier stechen vor allem OpenEmbedded und

PTXdist durch ihre Flexibilität hervor. Alle untersuchten Buildsysteme verweisen auf existierende Architekturen und empfehlen das kopieren und anpassen einer möglichst ähnlichen Architektur.

5 Zusammenfassung und Ausblick

Der Vorliegende Text gibt einige Hinweise, welche Punkte neben den unterstützten Targets, der Anzahl angebotener Pakete und der Patch-Philosophie eines Buildsystems beim Einstieg in die Entwicklung von Embedded Linux noch relevant sein können. Die Grundaufgaben eines Buildsystems erfüllen alle Systeme ohne Beanstandungen. Doch die untersuchten, freien Projekte verfolgen verschiedene Ansätze, die sie auch für einen Einsatz in kleine und mittelständische Unternehmen (KMU) interessant machen. Dabei kommt es nicht zuletzt auf die Interessen und Ziele des KMU und das Vorwissen seiner Mitarbeiter an. Es zeigte sich, dass sehr spezielle Distributionen wie OpenWRT, das auf Router ausgerichtet ist, mehr Aufwand bei der Realisierung alternativer Anwendungsfälle erfordern. Weiterhin gilt, dass jüngere Buildsysteme durch die Umsetzung neuer Ansätze als flexibler gelten dürfen. OpenEmbedded sticht mit seiner aktiven Community und seiner unerreicht großen Paketauswahl ebenso hervor wie ELDK und PTXdist, die durch Firmen betreut und gepflegt werden.

Die Arbeit mit den verschiedenen Buildsystemen und die Auseinandersetzung mit den zugrundeliegenden Konzepten führt unweigerlich zu dem Gedanken, ein eigenes Buildsystem umzusetzen. Da der dafür notwendige zeitliche und personelle Aufwand jedoch nicht zu unterschätzen ist, werden die gewonnen Erkenntnisse in die untersuchten Projekte zurückfließen. Außerdem entstehen gerade im Hype um die Smartphones und Tablet-PCs neue Distributionen (Android, MeeGo), die ebenfalls mit der Umsetzung interessanter Konzepte das Interesse wecken. Doch auch gestandene Distributionen wie LTIB versprechen neue Erkenntnisse.

Danksagung

Die hier vorgestellten Arbeiten entstanden im Rahmen des Projekts *Generalisierte Plattform zur Sensordatenverarbeitung* (Fkz.: 03IP505) und wurden durch das Bundesministerium für Bildung und Forschung (BMBF) innerhalb der Initiative „InnoProfile - Unternehmen Region“ gefördert.

References

[Atm06] Atmel Corporation: *Mature ngw100 network gateway kit*, 2006. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4102, visited on 2011/01/11.

[GR09] Gschossmann, Thomas und Thomas Rothoerl: *Studienarbeit: Buildumgebungen*. Studienarbeit, Hochschule Augsburg, September 2009.

[Hal06] Hallinan, Christopher: *Embedded Linux Primer: A Practical Real-World Approach*. Prentice Hall, 2006, ISBN 0131679848.

[Kam09] Kamleiter, Andreas: *Embedded Linux Build Systeme*. Studienarbeit, Hochschule Augsburg, 2009.

[Keg06] Kegel, Dan: *Building gcc cross-toolchains*, December 2006. http://kegel.com/crosstool/.

[Mor11] Morin, Yann E.: *crosstool-ng*, January 2011. http://ymorin.is-a-geek.org/projects/crosstool.

[Nag05] Nagel, William: *Subversion Version Control: Using the Subversion Version Control System in Development Projects*. Prentice Hall, 2005, ISBN 0131855182.

[ohn11] ohne Verfasser: *Crossdev*, January 2011. http://en.gentoo-wiki.com/wiki/Crossdev.

[Pen11] Pengutronix e.K.: *Oselas.toolchain()*, January 2011. http://www.pengutronix.de/oselas/toolchain/index_de.html, visited on 2011/01/15.

[Reb08] Rebe, René: *Embedded-Linux-Distributionen vorgestellt*. Linux Magazin, März 2008.

[Sch09] Schottelius, Nico: *Git. Dezentrale Versionsverwaltung für Code und Dokumente [Broschiert]*. Open Source Press; Auflage: 1., Aufl., 2009, ISBN 3937514724.

[SGD09] Schröder, Joachim, Tilo Gockel und Rüdiger Dillmann: *Embedded Linux: Das Praxisbuch (X.systems.press) (German Edition)*. Springer, 2009, ISBN 3540786198.

[SH10] Stückjürgen, Christoph und Gernot Hillier: *Linux inside*. c't Magazin für Computertechnik, 19:164–169, August 2010.

[Uni10] Unicontrol Systemtechnik GmbH: *Uniccontrol Systemtechnik GmbH*, Januar 2010. http://www.unicontrol.de/de/index.html, besucht: 2011/01/11.

[Xil05] Xilinx Inc.: *Xilinx university program virtex-ii pro development system*. WebPage, August 2005.

[YMBYG08] Yaghmour, Karim, Jon Masters, Gilad Ben-Yossef, and Philippe Gerum: *Building Embedded Linux Systems*. O'Reilly Media, 2008, ISBN 0596529686.

Fast Startup Linux

Wolfram Luithardt, Daniel Gachet und Guy Morand

Hochschule für Technik und Architektur Freiburg

Bd. de Pérolles 82, CH-1701 Fribourg

wolfram.luithardt@hefr.ch

1. Einleitung

Die Zeit, die ein elektronisches System benötigt, um bis zur vollen Funktionsfähigkeit aufzustarten, ist ein wichtiger Parameter, der die Bedienbarkeit dieses Systems häufig sehr beeinträchtigen kann. Stellen wir uns ein Smartphone vor, das mehrere Minuten benötigt (diese Zeit ist bei PCs teilweise notwendig), bis man damit arbeiten, z.B. telefonieren kann. Solch ein Gerät könnte wohl auf dem freien Markt kaum bestehen. In diesem Paper wollen wir über Untersuchungen berichten, wie die Aufstartzeit von eingebetteten GNU/Linux-Systemen um Faktoren reduziert werden kann. Eingebettete Systeme sind Mikrokontroller- oder Mikroprozessor-Schaltungen, die in Anwendungen eingesetzt werden, bei denen sich der Benutzer häufig nicht bewusst ist, dass er mit einem kleinen (und oft sehr spezialisierten) Computer arbeitet (z.B. Waschmaschine, Fernseher usw.). In den letzten Jahren werden solche eingebetteten Systeme häufig mit GNU/Linux betrieben. Die Gründe für diesen massiven Einsatz von Linux sind offensichtlich: Linux ist frei zugänglich und damit für spezielle Applikationen optimierbar, es ist gut dokumentiert und sehr stabil. In jedem Fall erlaubt Linux, gewisse Verhaltensweisen, wie hier die Aufstartmechanismen, zu studieren und Optimierungen durchzuführen.

Unsere Überlegungen bzw. Messungen und Optimierungen wurden durch mehrere Industrieprojekte motiviert, alle Systeme bestanden aus ARM- oder PowerPC-basierten Mikroprozessoren, 16-32 MB Ram und 16-256MB Nand-Flash. Bei allen Projekten stand im Mittelpunkt, die Funktionalität der Geräte ohne jegliche Einschränkungen zu erhalten und trotzdem die Aufstartzeit zu minimieren. Andere Betriebssysteme bedienen sich häufig dem Trick, gewisse Dienste erst nach Anzeigen der (grafischen) Oberfläche zu starten. Mit dieser Methode wird kaschiert, dass die Aufstartzeit wesentlich länger ist, als auf den ersten Blick sichtbar. Als Resultat ist das System für einige Minuten nicht sehr interaktiv und

das Arbeiten damit kaum möglich. Wir definieren hier die Aufstartzeit als die Dauer zwischen dem vollständigen Anliegen der Versorgungsspannung und dem Erscheinen eines Prompts, wenn alle Prozesse und Deamons vollständig gestartet sind.

2. Theorie

Um zu verstehen, wie die Aufstartzeit reduziert werden kann, muss der Prozess, den ein Linux-System bis zur vollen Funktionsfähigkeit durchlaufen muss, genau analysiert werden. Nach dem Anlegen der Versorgungsspannung muss der Prozessor bzw. das System als Erstes den Oszillator, d.h. den Taktgeber, für das System hochfahren. Dies geschieht meist in einer fest vorgegebenen, relativ niedrigen Frequenz. Zudem wird der Programmcounter (PC) auf eine feste Adresse (häufig 0x0000 0000) gesetzt und der Code (genannt Bootstrap-Code) an dieser Adresse wird ausgeführt. Bei PC-Systemen ist dieser BootstapCode das in einem ROM (heute meistens Flash-Rom) gespeicherten BIOS (Basic Input/Output System), welches nach Konfiguration einiger Prozessorregister dann den Masterbootrecord des ausgewählten Bootmediums (meist eine Festplatte) ließt und diesen ausführt. Von hieraus wird dann ein Bootprogramm geladen (in Linuxsystem häufig GRUB oder LILO). Dieses lädt und startet dann den Betriebssystemkern.

Bei eingebetteten Systemen funktioniert der Bootprozess recht ähnlich, obwohl solche Systeme meist keine Festplatte besitzen. Hier wird ebenfalls ein Bootprogramm verwendet, das sich im Flash-Rom (meist NAND- oder NOR-Flash) befindet, das zuerst ins Ram des Systems geladen und daraufhin ausgeführt wird. Dieses Bootprogramm (hier wurde U-Boot [1] oder Red-Boot [2] verwendet), startet nicht nur das Betriebssystem auf, sondern verfügt, ähnlich wie bei PCs, ebenfalls über eine interaktive Möglichkeit den Bootprozess zu konfigurieren. Während ein PC bereits ein menu-gesteuertes Interface zum BIOS verfügt, ist bei eingebetteten Systemen, die über keinen Bildschirm verfügen, dieses interaktive Interface meist über eine serielle Schnittstelle vorhanden. Über einfache Kommandos und Umgebungsvariablen kann der Aufstartprozess konfiguriert und optimiert werden. Hier werden unter Anderem die Partitionen des Speichers parametrisiert und die Boot-Parameter für den Linuxkernel (Bootargs) eingegeben und gespeichert. Des Weiteren bietet die Shell Möglichkeiten, Daten über TFTP zu laden und in das Flash-Rom zu speichern.

Um zu Entscheiden, ob die Shell gestartet werden soll, wird ein Mechanismus verwendet, der sehr einfach ist. Das U-Boot Programm wartet einige Sekunden (bootdelay), ob über die serielle Schnittstelle ein Zeichen (entweder ein bestimmter Charakter oder ein beliebiges Zeichen) ankommt. Ist dies der Fall, so wird der Bootprozess unterbrochen und die interaktive Shell aufgerufen. Hier können nun

diverse, meist sehr einfache Kommandos eingegeben und Umgebungsvariablen verändert werden. Diese Umgebungsvariablen sind ebenfalls im Flash oder in einem E^2Prom gespeichert. Nach Eingabe dieser Parameter, oder wenn die Shell nicht aufgerufen wurde, wird dann eine Sequenz durchlaufen, die das Betriebssystem vollständig hochfährt und die Applikationsprozesse startet.

In Bild 1 sind die einzelnen Schritte des Bootvorgangs gezeigt.

1.: Die CPU ruft eine durch Hardware vorgegebene Adresse im Flash aus, welches den Bootstrap Code enthält.

2.: Dieser Code initialisiert die CPU und das Ram des Systems.

3.: Der Bootstrap Code kopiert das Boot-Programm in das Ram und startet es.

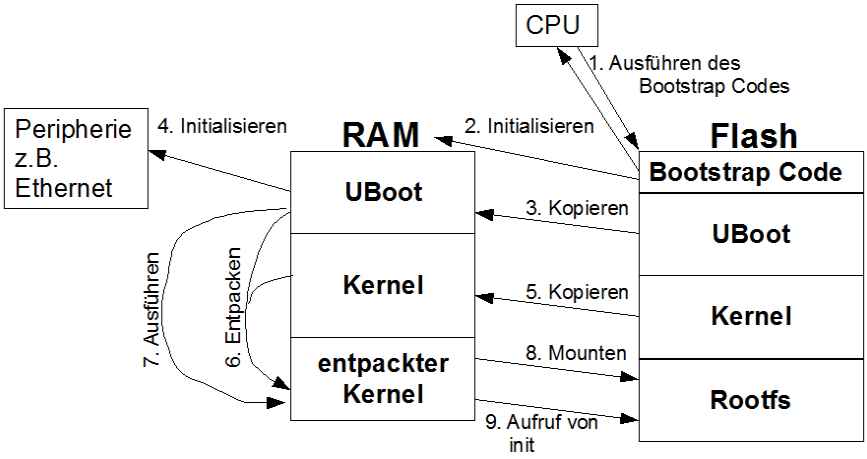


Bild 1: Ablauf des Aufstartmechanismus

4.: Das Programm initialisiert die Peripherie des System, die es benötigt, z.B. die Ethernet-Schnittstelle. Es wird eine vorgegebene Zeit gewartet, um zu prüfen, ob die Bootshell aufgerufen (s.o.) oder direkt der Kernel geladen werden soll.

5: Der Kernel wird vom Flash in das Ram kopiert.

6: Falls der Kernel komprimiert gespeichert war, so wird er nun entpackt.

7: Das Bootprogramm übergibt die Bootparameter an den Kernel und startet ihn.

8: Der Kernel hängt das Root-Filesystem ein.

9: Das init-Programm wird aufgerufen, welches dann die Dienste startet.

Aus Darstellungsgründen wurden einige Punkte weggelassen. Als Beispiel sei hier

das Löschen der nicht mehr benötigten Teile (Bootprogramm und gepackter Kernel) im Ram genannt.

3. Verschiedene Möglichkeiten zur Reduktion der Startzeit

Natürlich sind die Möglichkeiten, die Bootzeit des Systems zu reduzieren, extrem abhängig von der verwendeten Hardware. So ist z.B. die Zeit, die das System zum Kopieren des Bootprogramms oder des Kernels vom Flash in das RAM benötigt, extrem abhängig vom Bussystem und vom verwendeten Typ des Flash. Je nach Ausstattung einer Plattform können also bessere oder weniger gute Reduktionen der Startzeit erzielt werden.

Wir werden nun einige Vorschläge unterbreiten, wie die einzelnen Phasen des Aufstartens optimiert werden können. Dabei ist zu beachten, dass nicht in allen Systemen alle Punkte möglich sind bzw. Sinn machen.

a.) Unterdrücken des Scannen der Peripherie (Schritt 2 bzw.4):

Als eines der ersten Schritte des Boot-Prozesses werden die diverse Peripherielemente wie z.B. die PCI-Express-Schnittstellen gescannt. Dabei wird ein delay in der Größenordnung von Sekunden aufgerufen, damit sich die Peripherieelemente melden können. Weiß man, dass keine Geräte vorhanden sind, so kann das Scannen und das Delay eingespart werden. Ein einfacher Patch im U-Boot-Code z.B. in der Routine static void <arch>_pcie_init_bus ermöglicht eine Reduktion von einigen Sekunden.

b.) Reduktion des Bootdelay (Schritt 4):

Die einfachste Methode, die Startzeit zu reduzieren, ist eine Reduktion des Bootdelays. Dieses ist gewöhnlich auf eine Zeit zwischen 3 und 20 Sekunden gestellt und lässt sich einfach über der Umgebungsvariablen 'bootdelay' im U-Boot verändern. Aber Achtung: setzt man diese Größe auf 0, so kann der Bootprozess häufig überhaupt nicht mehr unterbrochen werden. Dies ist bei Geräten in der Produktionsphase teilweise sogar erwünscht, da damit vermieden werden kann, dass ein Kunde den Bootprozess verändert. In diesem Fall können meist die Bootparameter nur über die JTAG-Schnittstelle verändert werden.

c.) Abschalten der Checksummenverifikation des Boot-Loaders (Schritt 4):

Mithilfe der Umgebungsvariable verify kann die Checksummenkontrolle von U-Boot abgestellt werden, wenn sie auf den Wert 'n' gesetzt wird.

d.) Unterdrücken der Ausgaben des Bootoaders (Schritt 5-7):

Im U-Boot können die Ausgaben mittels der Umgebungsvariable silent unterdrückt werden. Diese Variable muss auf irgendeinen Wert gesetzt werden, um zu erreichen, dass U-Boot wesentlich weniger Meldungen ausgibt, als im Normalmodus. Messungen am System haben aber gezeigt, dass zwar die Meldungen des U-Boots nicht angezeigt werden, die Zeit aber nicht reduziert wird. Es liegt deshalb nahe, dass U-Boot die Meldungen zwar umleitet aber trotzdem noch ausgibt. Eine Analyse der entsprechenden Routinen hat gezeigt, dass in der Tat bei jedem auszugebenden Zeichen ein 'busy-waiting' aufgerufen wird. Durch ein einfaches Auskommentieren dieser Routine konnten die erwarteten Ergebnisse erreicht werden.

e.) Ausschalten von Ausgabemeldungen des Kernels (Schritt 7):

Ausgabemeldungen des Kernels auf die Konsole sind sinnvoll, wenn die einzelnen Schritte verifiziert werden müssen. Im Normalfall ist dieses aber nicht notwendig. Mit dem Kernel Bootparameter 'quiet' (bootargs = quiet), können die Meldungen sehr einfach ausgeschaltet werden.

f.) Optimierung der Kernels (Schritt 5-7):

Eine Optimierung des Kernels sollte eigentlich bei jedem System durchgeführt werden. Werden im Kernel nur diejenigen Funktionalitäten einkompiliert, die auch wirklich benötigt werden, so wird das System nicht nur schneller sondern auch sicherer. Ein kleinerer Kernel kann schneller vom Flash ins Ram kopiert, schneller entpackt und evt. auch schneller ausgeführt werden.

g.) Nicht komprimierter Kernel (Schritt 6):

Die Verwendung eines nicht komprimierten Kernels erspart den Schritt 6 im oben dargestellten Ablauf. Allerdings wird dadurch die Zeit zum Kopieren vom Flash ins Ram erhöht, da ja mehr Bytes kopiert werden müssen. Typische Kompressionsraten liegen bei >2. Je nachdem, ob das System I/O- oder CPU-optimiert ist, kann die Verwendung eines nicht komprimierten wesentliche Vorteile bringen aber eventuell auch Nachteile. Bei Systemen, die mit einigen hundert Megaherz getaktet sind, ist normalerweise das Entpacken eines komprimierten Kernels wesentlich schneller als das Laden eines größeren Kernelimages.

Es gibt natürlich noch die Möglichkeit, das Entpacken quasi 'on the fly'-vom Prozessor direkt während des Ladens der Daten durchzuführen. Damit kann die recht lange Zeit des Ladens bereits sinnvoll genutzt werden. Eine weitere Möglichkeit wäre auch, das Laden eines nicht komprimierten Kernels per Direct-Memory-Acces (DMA) anzustossen, der dann direkt ins RAM geladen wird und dem Prozessor die Möglichkeit gibt, zwischenzeitlich noch andere Aufgaben zu erledigen. Einige weitere Entwicklungen bei uns gehen in diese Richtung.

h.) Konfiguration des Ethernet-Interfaces im Userland (Schritt 7)

Meistens werden bei eingebetteten Systeme die Konfigurationsparameter für das Ethernet Interface bereits im Bootloader vorgegeben und dem Kernel als Bootparameter übergegeben. Dies dauert erfahrungsgemäß wesentlich länger als die Konfiguration des Interfaces als Userprozess. Die Konfiguration kann auch als Init-Script (/sbin/ifconfig, /sbin/route und /sbin/ifup) aufgerufen werden.

i.) Manuelle LPJ-Kalibrierung (Schritt 7):

Eine der ersten Aktivitäten des Kernels ist die LPJ-Kalibrierung zur Bestimmung der Prozessorgeschwindigkeit in BogoMips. Dies geschieht über eine Schleife, die den entsprechenden Wert in mehreren Versuchen ermittelt. Da dieser Schritt bei jedem Aufstarten des Systems erneut durchgeführt wird, obwohl er eigentlich immer denselben Wert ergibt, wäre es besser, bei bekanntem Wert diesen direkt dem Kernel beim Starten als Bootargument mitzugeben. Dieses ist mit dem Bootparameter 'lpj=wert' möglich. Der Wert wird bei einer automatischen Bestimmung normalerweise auf die Konsole ausgegeben oder in /var/log/dmesg eingetragen: z.B.:

Calibrating delay loop... 66.56 BogoMIPS (lpj=332800)

j.) Mounten des Rootfilesystems als read-only (Schritt 8):

Häufig müssen bei eingebetteten Systemen im Betrieb die meisten Daten (z.B. Programme) nicht mehr verändert werden. Dadurch bietet es sich an, einen Großteil des Root-Filesystem nur read-only einzuhängen. Dies geht in der Regel schneller als das Einbinden als read-write. Man kann also je nach Applikation neben einigen Filesystemen (wie z.B. /sys, /proc oder /tmp) im Ram auch eine kleine Partition mit Konfigurationsdaten als writeable-Filesystem mounten und den großen Teil als read-only.

k.) Verwenden eines anderen Filesystems (Schritt 8):

Inzwischen gibt es eine ganze Reihe von verschiedenen Filesystemen, die für die Verwendung in Flash-Roms geeignet sind [3]. Diese sind darauf optimiert, dass sie zum Einen den Zugriff auf die entsprechende Blockorganisation der Speicherchips optimieren, zum Anderen die Schreibbelastung der einzelnen Speicherzellen gleichmäßig verteilen um Zerstörungen bei zu häufigen Lösch-Schreib-Zyklen zu vermeiden. Flash-Speicher sind nämlich nicht beliebig häufig beschreibbar und können bei periodischer Löschung und Beschreibung bereits nach relativ kurzer Zeit zerstört werden. Unterschiedliche Filesysteme können sehr unterschiedliche Aufstartzeiten besitzen.

l) Alternative Initmethoden (Schritt 9):

Sehr häufig basiert das Starten der Userprozesse auf dem alten System-V Modell, das eine Parallelität weitgehend verunmöglicht. Wenn nach vollständigem Aufstarten eine Prozessliste (z.B. mittels ps ax) ausgegeben wird, so ist an der

höchsten PID ersichtlich, dass bereits einige hundert Prozesse gestartet wurden. Das Erzeugen so vieler Prozesse während des Aufstartens benötigt natürlich Zeit. Einige neuere Ansätze (das bereits häufig in Desktopsystemen eingesetzte upstart [4], das relativ neue Projekt systemd [5]) versuchen, diese Prozesse zu parallelisieren. Hiermit kann natürlich sehr viel eingespart werden, da in diesen Skripten sehr häufig auf andere Dienste gewartet werden muss.

Zuerst sollte allerdings immer eine genaue Analyse der Dienste stehen, die bei Systemstart wirklich benötigt werden. Sehr häufig sind die Dateien, die die Initialisierung konfigurieren, voll mit Diensten, die niemals verwendet werden.

m.) Initskripte vereinfachen (Schritt 9)

Initskripte können recht komplex werden, da sie häufug auch als Userinterface für die verschiedenen Dienste verwendet werden. In eingebetteten Systemen ist dies aber nur in den seltensten Fällen notwendig, so dass die Komplexität häufig reduziert werden kann, was die Skripte wesentlich schneller macht. Zu beachten ist auch, dass bei jedem Öffnen einer neuen shell zum Abarbeiten eines Skriptes, busybox, das die Standardfunktionen des Linuxsystems zur Verfügung stellt, neu geladen werden muss. Je nach Größe des busybox-Programms kann auch hier viel Zeit eingespart werden.

Natürlich gibt es noch viele weitere Maßnahmen zur Reduktion der Aufstartzeit, die hier aus Platzgründen nicht weiter erläutert werden können.

4. Ergebnisse

Um nun quantitative Aussagen darüber zu erhalten, in welchem Maße die oben angegebenen Schritte die Aufstartzeit reduzieren, ist es notwendig, eine Messmethode zu etablieren, die zuverlässig und reproduzierbar die Dauer der einzelnen Schritte des Aufstartmechanismus bestimmen können. Ideal wäre eine Lösung, die nach jedem Schritt auf einem digitalen Port des Systems den Zustand verändert und dies mit einem Logikanalysator mitgeschrieben wird. So eine Lösung ist ideal, da das Setzen oder Rücksetzen eines digitalen Ausgangs nur wenige Taktzyklen benötigt und dadurch die Messung die Aufstartzeit quasi nicht verändert. Außerdem ist sie extrem präzise. Einfacher ist es jedoch, einfach die Zeitstempel von Statusausgaben auf der seriellen Schnittstelle auszugeben. Dies kann z.B. sehr einfach mit dem Programm grabserial [6] durchgeführt werden, mit welchem solche Zeitausgaben über den Kommandozeilenparameter -t angefordert werden können. Grabsserial ist ein kleines Terminal-Programm, mit dem die Ausgaben einer seriellen Schnittstelle eingelesen werden können. Die Reproduzierbarkeit dieser Zeitangaben ist allerdings weniger gut, als bei der Methode mit den digitalen Ports und liegt bei einigen hundert Mikrosekunden.

Die hier gezeigten Ergebnisse wurden an einer Armadeus-Plattform APF27 [7] durchgeführt. Diese Plattform besteht aus einem Freescale i.mx27 Prozessor, der mit 400 MHz getaktet ist, aus 128 MB Ram und 256MB NAND-Flash. Des Weiteren verfügt sie über die gewohnten Schnittstellen seriell, USB, Ethernet sowie einigen Audio- und Video-Interfaces sowie über einen FPGA zur flexiblen Adaption von weiteren Peripherieelementen. Die Version von U-Boot war 1.3.4 und als Kernel wurde 2.6.29.6 verwendet. Die Aufstartzeit der unveränderten Armadeus Distribution liegt bei 15.6 Sekunden. In der folgenden Tabelle sind die Ergebnisse einiger der oben angegebenen Maßnahmen dargestellt, sortiert nach Größe der erzielten Reduktion:

Maßnahme	Reduktion um
j.) + k.) Verwendung eines read-only squash-Filesystems	4.65 s
b.) Reduktion des Bootdelays	3.00 s
h.) Initialisierung des Ethernet Interfaces als Userprozess	1.56 s
d.) und e.) Unterdrückung von Ausgaben	0.50 s
c.) Abschalten der Checksummenverifikation des Boot-	0.38 s
i.) Manuelle LPJ-Kalibrierung	0.19 s

Die Verwendung eines nicht komprimierten Kernels (Maßnahme g.)) führte bei dieser Plattform zu einer Verlängerung der Aufstartzeit um ca. 700 ms. Aufgrund der hohen Taktfrequenz des Prozessors ist das Laden und Entpacken eines 'kleinen' Kernels also wesentlich schneller als das Laden eines 'großen' (nicht komprimierten) Kernels.

Als Resultat konnte bei dieser Plattform die Aufstartzeit um ca. 10.3 Sekunden auf 5.3 Sekunden reduziert werden. Dies entspricht ungefähr einem Faktor 3. Durch eine genaue Analyse und Reduktion der einkompilierten Kernelfunktionen konnte die Startzeit um weitere 1.2 Sekunden auf 4.1 Sekunden reduziert werden. Eine Vereinfachung der Initskripte brachte weitere 400 ms ein. Natürlich sind die 2 letzten Maßnahmen extrem applikationsabhängig und nicht generalisierbar. Die Reduktionen der in der Tabelle dargestellten Maßnahmen sollten allerdings in den meisten eingebetteten Systeme angewendet werden können.

5. Ausblick

Kleinvieh macht auch Mist! Wenn auch jede einzelne der hier gezeigten Maßnahmen die Startzeit nicht um Faktoren reduzieren kann, so sind doch alle Maßnahmen zusammen in der Lage, die Aufstartzeit massiv zu reduzieren. So konnten wir neben den oben gezeigten Werten an der Armadeus-Plattform an

einem industriell eingesetzten Produkt mit GNU/Linux-System eine Reduktion von 14.6 auf 2.4 Sekunden (Faktor 6) erreichen und bei einem anderen, sich noch in der Entwicklung befindlichen System, von 63 Sekunden auf 7.5 Sekunden (Faktor 8!).

Natürlich ist bei der Diskussion der hier gezeigten Ergebnisse immer zu beachten, dass sie nicht in jedem System anwendbar oder sinnvoll sind. Es ist sicherlich kontraproduktiv, bei einem Desktop-PC ein read-only Filesystem zu verwenden oder das Scannen des PCI-Express auszuschalten. Dynamische Methoden sind ja gerade dafür da, dass sich der Benutzer nicht damit herumschlagen muss, alles selber sauber zu konfigurieren. Hier zeigen sich einmal mehr die großen Vorteile von GNU/Linux-Systemen, die so offen sind, dass sie optimal an die Gegebenheiten und Wünsche der Benutzer angepasst werden können.

Es gibt sicherlich noch viel mehr Methoden, die Bootzeit zu reduzieren, wir werden natürlich weiter daran arbeiten, die Startzeit weiter reduzieren zu können, andere Gruppen tun dies ebenfalls und wir können gespannt sein, was in den nächsten Jahren noch alles möglich wird.

6 Literatur

[1] http://www.denx.de/wiki/U-Boot

[2] http://www.cygwin.com/redboot/

[3] http://de.wikipedia.org/wiki/YAFFS#Dateisysteme_f.C3.BCr_Flash-Datentr.C3.A4ger

[4] http://elinux.org/Boot_Time

[5] http://0pointer.de/blog/projects/systemd.html

[6] http://elinux.org/Grabserial

[7] http://www.armadeus.org

Kryptographische Dateisysteme im Detail

Stefan Schumacher
Magdeburger Institut für Sicherheitsforschung
Stefan.Schumacher@Magdeburger-Institut-fuer-Sicherheitsforschung.de
http://www.Magdeburger-Institut-fuer-Sicherheitsforschung.de

Kryptographische Dateisysteme sind ein wirksames Mittel, um Dateien vor fremden Zugriffen zu schützen. Aber gerade in einem größeren Umfeld werfen diese Dateisysteme einige Probleme auf, z. B. bei der Datensicherung oder bezüglich Schlüsselhinterlegung und Zugriffe über mehrere Benutzer. Dieser Artikel stellt einige grundlegende Probleme sowie Details kryptographischer Dateisysteme im Vergleich vor.

Arten der kryptografischen Dateisysteme

Um Daten effektiv vor unerlaubten Zugriffen zu schützen, wurden kryptographische Dateisysteme entwickelt. Diese sollen idealerweise ohne großen Aufwand Dateien, die auf einem Datenträger liegen zuverlässig verschlüsseln und dem Anwender bzw. den Anwendungsprogrammen möglichst transparent unverschlüsselt zur Verfügung stellen. Diese Anforderungen können auf verschiedene Arten und Weisen umgesetzt werden, die jeweils ihre spezifischen Vor- und Nachteile haben. Diese möchte ich im folgenden Artikel vorstellen.

Eine einfache Möglichkeit Dateien zu verschlüsseln ist die einzelne Verschlüsselung durch ein Verschlüsselungsprogramm. Dazu muss jede Datei einzeln verschlüsselt werden und die unverschlüsselte Datei sicher gelöscht werden. Möchte man auf eine Datei zugreifen, muss sie entschlüsselt werden. Nach Beendigung des Zugriffs muss sie wieder verschlüsselt und die Klartextdatei sicher gelöscht werden. Zur Verschlüsselung können verbreitete und damit portable Verschlüsselungsprogramme wie GnuPG, Mcrypt oder OpenSSL eingesetzt werden, die auf unterschiedlichen Betriebssystemen laufen. Allerdings ist es sehr aufwändig die Dateien zu ver- und entschlüsseln, da der Anwender die Aufgaben alle von Hand erledigen muss. Es funktioniert daher nur sinnvoll wenn es sich um wenige Dateien handelt, die selten benutzt werden bspw. speziell zu schützende Dateien. Oder wenn es zwingend erforderlich ist, von unterschiedlichen Betriebssystemen auf die Dateien zuzugreifen.

Wesentlich komfortabler ist es, Dateien von einem Programm automatisch ver- und entschlüsseln zu lassen, wenn auf diese zugegriffen werden muss. Dies kann entweder durch einen Dæmon geschehen, der Zugriffe auf Dateien im Dateisystem abfängt und auf einen verschlüsselnden Layer umlenkt, oder in dem man die Festplatte bzw. eine Partition davon auf Blockebene verschlüsselt und die Verschlüsselung unterhalb des Dateisystems durchschleift. Abb. 1 zeigt 3 Varianten der Verschlüsselung mittels CFS und CGD, wobei CGD einmal auf Blockebene und einmal als Container arbeitet.

Beide Varianten haben ihre spezifischen Vor- und Nachteile. Eine Verschlüsselung auf Blockebene, wie sie CGD, GBDE, Geli oder loop-aes einsetzen, verhält sich nach dem Einbinden für den Benutzer transparent. Außer für eine Passworteingabe beim einmounten merkt der Benutzer in der Regel nicht, dass seine Partition verschlüsselt ist.

Nachteilig ist hierbei die Granularität der Verschlüsselung. Es kann immer nur ein Block-Device, also eine Partition verschlüsselt werden. Möchte man die Zugriffe auf Dateien feiner granuliert steuern, reichen Block-Devices nicht aus.

Es sei denn, man verwendet als Workaround Dateisystem-Container, die man mit einem Block-Device verschlüsselt. Dazu erstellt man eine Datei, die als Container dient und über ein Loopback-Device wie vnconfig(8) eingemountet wird. In diesem Container wird mit dem Verschlüsselungssystem die »Pseudo-Block-Ebene« verschlüsselt und entsprechend eingemountet, siehe Abb. 1(b)

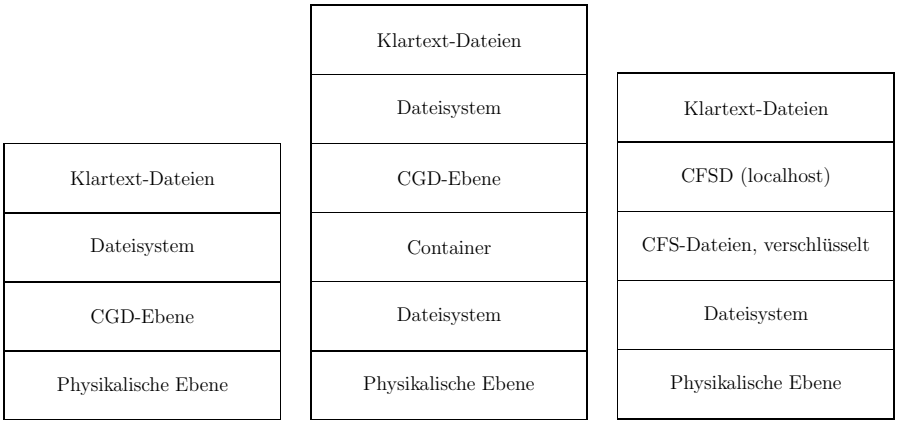
Vorteil dieser Variante ist die feinere Granularität: man kann bspw. jedem Benutzer einen eigenen Container in seinem Home-Verzeichnis zur Verfügung stellen oder auf einem Laptop verschiedene Container für verschiedene Daten einbinden. Beispielsweise einen Container für private E-Mails und den privaten GnuPG-Schlüsselring und einen Container für die beruflichen E-Mails und den beruflichen GnuPG-Schlüsselring. Je nach Situation bindet man nur den entsprechenden Container ein.

Ähnlich arbeiten Systeme, bei denen ein Dæmon die Zugriffe auf die Dateien im Dateisystem abfängt und die entschlüsselte Dateivariante dem Anwendungsprogramm zur Verfügung stellt. Dabei liegen die Quelldateien verschlüsselt im normalen Dateisystem und können theoretisch auch mit normalen Programmen wie vi bearbeitet werden. Dies ist aber nicht sinnvoll, da die Dateien verschlüsselt sind. Möchte der Anwender auf die Dateien unverschlüsselt zugreifen, fängt der entsprechende Dæmon den Zugriff ab und leitet ihn transparent auf das Entschlüsselungssystem um. Das Anwendungsprogramm erhält also die unverschlüsselten Daten ausgeliefert.

Vorteil dieser Variante ist die noch höhere Granularität gegenüber der Container-Variante, da hier im Prinzip jeder einzelnen Datei ein eigener Schlüssel zugeordnet werden kann. In der Praxis arbeiten die die meisten dieser Systeme aber auf Verzeichnisniveau, d.h. ein Verzeichnis wird rekursiv samt aller darin enthaltenen Dateien und Unterverzeichnisse ver- und entschlüsselt.

Alle drei Varianten haben verschiedene Vor- und Nachteile. Die Verschlüsselung auf Blockebene läuft weitestgehend transparent ab. Außer der Eingabe des Passwortes, die natürlich bei allen Verschlüsselungsprogrammen anfällt, bemerkt der Benutzer in der Regel nicht, dass die Partition mit seinen Daten verschlüsselt ist.

Umständlich kann es sein, die entsprechenden Partition im laufenden Rechner auszumounten, bspw. wenn man ihn per ACPI schlafen legen möchte. Es kann dann unter Umständen etwas umständlich sein, die Partition auszumounten, bspw. wenn Dienste wie PostgreSQL oder Apache auf sie zugreifen. Dies geht mit Containern



(a) CGD auf Blockebene einer Partition

(b) CGD auf Pseudo-Blockebene eines Container

(c) CFS im Dateisystem

Abbildung 1: Verschlüsselung mit CGD und CFS auf Block- bzw. Container-Ebene

oder Dæmonen einfacher, da man hier nur die entsprechenden Verzeichnisse bzw. Dateien schützen und ggf. ausmounten muss.

Problematisch ist es, verschlüsselnde Dateisysteme zu sichern. So ist es nicht möglich, verschlüsselte Partitionen auf Blockebene einfach mit dump(8) oder Bacula direkt zu sichern. Die verschlüsselte Partition muss immer von einem Anwender eingebunden werden, damit das eingebundene Pseudogerät gesichert werden kann. Es ist also immer der Eingriff eines Benutzers erforderlich, automatisierte Pull-Verfahren funktionieren hier nicht. Das Sicherungsarchiv ist dann gesondert zu schützen, also zu verschlüsseln.

Container bzw. Dæmonsysteme sind hier wesentlich einfacher zu handhaben, da die Container bzw. verschlüsselten Dateien direkt gesichert werden können. Sie sind dann bereits im Sicherungsarchiv verschlüsselt, eine gesonderte Verschlüsselung des Archivs ist daher nicht notwendig, aber empfohlen.

Nachteilig bei Containern ist hierbei die feste Größe, da der Container in der Regel weniger Nutzlast enthält, als er eigentlich groß ist. Wenn ein Container bspw. 700MB groß ist, aber nur 5MB Nutzdaten enthält, werden 695MB »Luft« gesichert. Bei einem System mag das tolerabel sein, in größeren Netzen mit hunderten von Benutzer in der Regel nicht mehr. Daher kann es auch hier erforderlich sein, die Container einzumounten und die unverschlüsselten Dateien direkt zu sichern. Praktischer sind in dieser Hinsicht Dæmonen-Systeme, da hier nur die Nutzdaten plus ein wenig Overhead anfallen.

Praktisch sind Container und Dæmonen auch bezüglich Zugriffe über Netz, so ist

es problemlos möglich, einen Container oder CFS-verschlüsselte Dateien von einem NFS-Server zu importieren und erst lokal zu entschlüsseln. Somit gehen die Daten verschlüsselt durchs Netz und sind auf dem Übertragungsweg geschützt. Außerdem lassen sich so Daten auf Wechseldatenträgern schützen.

Der Vorteil der Dæmonsysteme hinsichtlich Datenmüll ist aber auch gleichzeitig ein Nachteil bezüglich der Kryptoanalyse. Dæmonsysteme offenbaren durch die Verschlüsselung einzelner Dateien einige Metadaten, wie bspw. die Dateianzahl und -größe in einem Verzeichnis, sowie die mtime, ctime und atime. Dies ermöglicht eventuell Rückschlüsse auf den Dateiinhalt.

Betriebsarten des Kryptalgorithmus

Ein wichtiges Detail bei der Implementierung von Block-Geräten ist die Verwendung des Block-Modus als Betriebsart. Auf Block-Geräten werden Block-Chiffre eingesetzt, also Kryptoalgorithmen, die den Klartext in feste Blöcke zerlegen und diese Blöcke verschlüsseln. Alternativ gibt es noch sogenannte Stromchiffre, die den Klartext und den Schlüssel als Eingabestrom erwarten, da Festplatten etc. aber immer eine Blockstruktur aufweisen, wäre es unsinnig, einen Stromchiffre zu verwenden.

Zwei der bekanntesten Betriebsarten von Blockchiffren sind der Electronic Code Book Mode (ECB) und der Cipher Block Chaining Mode (CBC). Beim ECB wird jeder Block einzeln verschlüsselt, die Blöcke selbst werden nicht miteinander gekoppelt. Außerdem bildet ein Chiffrblock immer auf den selben Klartextblock ab. Dadurch ist es möglich, wahlfrei auf einen bestimmten Block zuzugreifen, in dem der gewünschte Block direkt ver- oder entschlüsselt wird.

Ein großer Nachteil ist aber, das im ECB statistische Kryptanalysen möglich sind, denn großflächig zusammenhängende Bereiche im Klartext bleiben auch im Chiffrat zusammenhängend. Außerdem ergeben identische Klartextblöcke immer identische Chiffrate. Daher ist der ECB definitiv als unsicher zu bezeichnen.

Ein weiteres Problem sind Watermarking- bzw. Wasserzeichen-Angriffe. Dabei sorgt ein Angreifer dafür, das eine speziell präparierte Datei bspw. als E-Mail auf dem System landet. Ziel des Wasserzeichen-Angriffs ist es, nachzuweisen, dass eben diese Datei auf dem Gerät liegt.

Der CBC verkettet die Blöcke hingegen, in dem er den Klartextblock vor der Verschlüsselung mit dem vorhergehenden Chiffreblock per XOR verknüpft. Dadurch werden statistische Zusammenhänge, die sich über mehrere Blöcke verteilen verwischt.

Mathematisch erfolgt die Verschlüsselung rekursiv im CBC wie folgt:

$$\forall \left(i \in \mathbb{N}^{+}\right) : \left(\begin{array}{rcl} C_0 & = & E_K(P_0 \oplus \mathrm{IV}) \\ C_i & = & E_K(P_i \oplus C_{i-1}) \end{array} \right)$$

Da die Verschlüsselung rekursiv definiert ist, erzeugt eine Änderung im n. Block einen Lawineneffekt auf alle Blöcke größer n, denn durch die Änderung des n. Blockes ändert sich auch das Chiffrat des n. Blockes und damit der Initialisierungsvektor für den Block $n-1$. Und damit das Chiffrat $n+1$, welches wiederum als Initialisierungsvektor für $n+2$ dient. Daher werden in der Praxis die Datenblöcke in logische »Partitiönchen« aufgeteilt, so dass sich der Lawineneffekt in Grenzen hält und nicht über die gesamte Platte erstreckt.

Da der 1. Block keinen Vorgängerblock hat, muss für ihn ein Initialisierungsvektor erst generiert werden. Im Prinzip muss dieser Initialisierungsvektor nicht geheimgehalten werden, allerdings ist bei schlechter Wahl des IV z.B. durch einen Zeitstempel oder die Wahl der Sektornummer, ein Wasserzeichen-Angriff gegen das System möglich. Abhilfe schafft das ESSIV-Verfahren, bei dem der IV aus der verschlüsselten Sektornummer und einem Hash aus dem Passwort generiert wird. Da das Passwort geheim bleibt, ist der IV nicht vorhersagbar.

Die Entschlüsselung von CBC ist nicht rekursiv folgendermaßen definiert:

$$\begin{aligned} P_0 &= D_K(C_0) \oplus \mathrm{IV} \\ \forall \left(i \in \mathbb{N}^+\right) : (\quad P_i &= D_K(C_i) \oplus C_{i-1} \quad) \end{aligned}$$

Das heißt für die Entschlüsselung wird der jeweilige Chiffrat-Block, das Vorgänger-Chiffrat und der Schlüssel benötigt. Ist der Chiffrat-Block n nicht mehr lesbar, kann daher auch der Block $n+1$ nicht mehr dechiffriert werden. Ist der Block $n+1$ aber noch lesbar, kann er wieder als IV für den Block $n+2$ dienen und damit der Block $n+2$ dechiffriert werden.

Schlüsselverwaltung

Bisher existieren noch keine kryptographischen Dateisysteme mit funktionierender Schlüsselverwaltung für mehrere Benutzer. Alle produktiv einsetzbaren Systeme setzen einfache symmetrische Verschlüsselungsverfahren meist mit einem Passwort ein. Das heißt in der Konsequenz, dass alle Benutzer, die Zugriff auf die verschlüsselten Dateien haben woll das Passwort kennen müssen. Wenn das Passwort geändert werden muss oder einem Benutzer die Zugriffsrechte entzogen werden sollen, müssen alle Benutzer ein neues Passwort bekommen – dass ihnen dann auf einem sicheren Kanal mitgeteilt werden muss. Bisher eignen sich kryptographische Dateisysteme daher nur für einige wenige Benutzer, wenn nicht nur für genau einen.

Es ist daher notwendig, komplexere Schlüsselverwaltungssysteme aufzubauen, die unter anderem Schlüsselhinterlegung und -widerruf sowie eine Einbindung von Datensicherungsverfahren ermöglichen.

Swap und Temporäre Verzeichnisse

Die Swappartition wird verwendet, um Daten aus dem Arbeitsspeicher auszulagern, bspw. wenn eine Anwendung mehr Speicher braucht als aktuell frei ist. Die kann natürlich die Sicherheit des Systems gefährden, wenn es sich dabei um eigentlich verschlüsselte Daten handelt, die unverschlüsselt im RAM liegen und ausgelagert werden. Bei einem Neustart des Systems wird zwar die /tmp-Partition bereinigt, die dort gelegenen Daten können aber problemlos rekonstruiert werden.

Damit dies nicht passiert, kann man auch die Swappartition entweder als RAM-Disk im Speicher anlegen oder mit auf Blockebene verschlüsseln. Im Prinzip geht man hier genauso vor wie bei einer normalen Datenpartition, lediglich die Schlüsselgenerierung wird abgeändert. CGD als Blocksystem unterstützt bspw. »urandomkey« als Schlüsselmethode. Hierbei werden einfach aus /dev/urandom, einem Pseudozufallsgenerator, Zeichen ausgelesen und als Schlüssel verwendet. Der Schlüssel wird dabei nicht gespeichert, so das beim Absturz des Systems oder nach dem Herunterfahren die Partition nicht mehr entschlüsselt werden kann. Dies ist aber bei einer Swappartition nicht notwendig, da sie bei jedem Systemstart neu initialisiert wird.

Das temporäre Verzeichnis /tmp legt man entweder in einer Ramdisk im Arbeitsspeicher an, oder verschlüsselt es ebenfalls auf Blockebene. Anders als bei der Swap-Partition muss in der Tmp-Partition aber ein Dateisystem angelegt werden. Dies kann man von einem Shellskript beim booten erledigen lassen. Da die Tmp-Partition in der Regel nicht besonders groß ist, dauert es auch nicht besonders lange.

Übersicht über verschlüsselnde Dateisysteme

CFS Cryptographic File System von Matt Blaze, veröffentlicht 1993. Der cfsd-Dæmon arbeitet ähnlich wie der NFS-Dæmon und entschlüsselt die verschlüsselt im Dateisystem liegenden Dateien. Ist für nahezu jedes Unix verfügbar und kann auch in Netzwerken bzw. auf Wechseldatenträgern eingesetzt werden. Mit Blowfish als Algorithmus hinreichend schnell und sicher.

CGD Cryptographic Disk Driver von Roland C. Dowdeswell. In NetBSD ab Version 2.0 verfügbar. Verschlüsselt ganze Partitionen auf Blockebene, kann Schlüssel aus /dev/urandom generieren und so Swap und Temp verschlüsseln. Unterstützt AES, 3DES und Blowfish jeweils in CBC. Schnell und transparent, nicht portabel. Kann via vnd auch in Containern eingesetzt werden.

Truecrypt Ursprünglich unter Windows entwickelt, arbeitet mit Containern oder auf Blockebene. Inzwischen auch auf Mac OS X und Linux lauffähig, daher zum Datenaustausch geeignet. Truecrypt arbeitet in der XEX-TCB-CTS-Betriebsart, welche in der NIST FIPS Pub 800-38E standardisiert ist.

GBDE steht für GEOM Based Disk Encryption, GEOM wiederum ist ein Plattenverwaltungsframework von FreeBSD. GBDE arbeitet auf Blockebene, verschlüsselt dabei aber jeden einzelnen Block mit einem *neuen* Zufallsschlüssel. Dadurch werden Watermarkingattacken erschwert, da Blöcke mit gleichem Inhalt durch den Zufallsschlüssel immer ein unterschiedliches Chiffrat ergeben.

GELI arbeitet wie GBDE im GEOM-Framework von FreeBSD auf Blockebene. Es bietet neben einem Zufallsschlüssel auch die Möglichkeit mehrere Schlüssel für eine Partition zu verwenden. Darüber hinaus verschlüsselt GELI Daten nicht nur, sondern kann auch mit verschiedenen Prüfsummen die Integrität der Dateien sicher stellen.

eCryptFS arbeitet ähnlich wie CFS im Dateisystem. Es ist ab Version 2.6.19. im Linux-Kernel enthalten. Enthält der Linux-Kernel Unterstützung für Public-Key Kryptographie, kann diese auch für eCryptFS eingesetzt werden. Das System legt alle Metadaten in der jeweiligen Datei ab, dadurch kann diese zwischen verschiedenen Dateisystemen ausgetauscht werden, außerdem unterstützt es mehrere Schlüssel für eine Datei bzw. Hierarchie. Damit können Daten bequem ausgetauscht werden.

EncFS basiert auf FUSE und verschlüsselt Dateien im Dateisystem. Es unterstützt die im Linux-Kernel vorhandenen Kryptalgorithmen und kann jede Datei mit einem eigenen Zufalls-IV verschlüsseln. Außerdem generiert EncFS eine 8-Bit-Prüfsumme für jede Datei, um Dateikorruptionen zu erkennen. EncFS ist auch für Mac OS X und NetBSD erhältlich, ließ sich aber zur Zeit auf einem NetBSD 5.1 nicht kompilieren.

CryptoFS basiert ebenfalls auf FUSE oder wahlweise LUFS und arbeitet ähnlich wie EncFS und CFS im Dateisystem, in dem es jede Datei einzeln verschlüsselt.

dm-crypt ist Teil des Linux-Kernels der 2.6er-Reihe. Es arbeitet im Device-Mapper-Framework und kann daher auf Blockebene einzelne Partitionen oder ganze Geräte, RAID-Volumes sowie einzelne Dateien verschlüsseln. Es ist eine weiterentwicklung von cryptoloop und unterstützt unter anderem XTS, LRW und ESSIV um Wasserzeichen-Angriffe zu vereiteln. Das System läuft neben Linux auch auf DragonFly BSD.

SD4L ScramDisk for Linux ist ein Linux-Programm, um ScramDisk-Container zu erzeugen und einzubinden. ScramDisk war ein Verschlüsselungsprogramm für Windows 9x. SD4L nutzt das Scramdisk-Containerformat und erzeugt bzw. mountet verschlüsselte Container fester Größe im Dateisystem.

BestCrypt ist ein kommerzielles Verschlüsselungsprogramm von Jetico für Windows und Linux. Es kann auf Blockebene komplette Datenträger oder Partitionen verschlüsseln oder Container im Dateisystem erzeugen.

Siehe auch:

- Stefan Schumacher (2006). „Verschlüsselte Dateisysteme für NetBSD“. Deutsch. In: *UpTimes* 4, S. 25 – 31. ISSN: 1860-7683. URL: http://kaishakunin.com/publ/guug-uptimes-cgd_cfs.pdf (besucht am 07. 11. 2009)
- Stefan Schumacher (2007). „Daten sicher löschen“. Deutsch. In: *UpTimes* 1, S. 7 – 16. ISSN: 1860-7683. URL: http://kaishakunin.com/publ/guug-uptimes-loeschen.pdf (besucht am 07. 11. 2009)
- Stefan Schumacher (2004). *Einführung in kryptographische Methoden*. URL: http://www.cryptomancer.de/21c3/21c3-kryptograhie-paper.pdf (besucht am 2005. 01. 06)

Über den Autor

Stefan Schumacher ist geschäftsführender Direktor des Magdeburger Instituts für Sicherheitsforschung und gibt zusammen mit Jan W. Meine das Magdeburger Journal zur Sicherheitsforschung heraus.

Er befasst sich seit über 15 Jahren mit Fragen der Informations- und Unternehmenssicherheit und erforscht Sicherheitsfragen aus pädagogisch/psychologischer Sicht. Seine Forschungsergebnisse stellt er regelmäßig auf internationalen Fachkongressen der Öffentlichkeit vor.

Zur Zeit organisiert er eine Ringvorlesung zur Informationstechnologie und Sicherheitspolitik an der Otto-von-Guericke-Universität Magdeburg und arbeitet an mehreren Publikationen zum Thema Cyber-War und Sicherheitsstrategien.

Darüber hinaus berät er Unternehmen bei der Umsetzung von Sicherheitsmaßnahmen und der Etablierung unternehmensweiter IT-Sicherheitsstrategien.

login correct. Benutzerauthentifizierung mit OTP-Tokens

Mario Lorenz
mario.lorenz@guug.de

Der Beitrag beschreibt die Vor- und Nachteile verschiedener Verfahren zur Benutzerauthentifizierung. Speziell wird dabei auf die Verwendung so genannter One-Time-Password-Tokens, insbesondere deren Verwendung unter Linux, eingegangen.

1 Passwörter - ein Problemfall

Für viele Transaktionen ist es wichtig, dass ein Computer feststellen kann, ob ein Benutzer wirklich der ist, für den er sich ausgibt. Der Dialog mit dem Computer beginnt daher meistens mit der Eingabe von Benutzername und Passwort. Ist keine hinreichend sichere Umgebung vorhanden, z.B. beim Abruf von E-Mail im Internet-Café, besteht die Gefahr, dass das Passwort abgegriffen und missbraucht werden könnte. Auch sind viele vom Nutzer beim Umgang mit Passwörtern eigentlich geforderten Verhaltensweisen mittlerweile nicht mehr praktikabel. Deshalb erfreuen sich Angriffe, bei denen versucht wird, das Passwort zu erraten, großer Beliebtheit.

Dies macht den (alleinigen) Einsatz von Passwörtern zunehmend fragwürdig. Für "wichtige" Fälle, wie z.B. Online-Banking, werden daher neuere Verfahren wie z.B. PIN/TAN schon viele Jahre lang verwendet. Die konsequente Weiterentwicklung dieses Verfahrens führte zur Entwicklung der Einmal-Passwort-Tokens (One-Time-Password-Token, OTP-Token). Diese Tokens sind mittlerweile auch für Privatanwender und kleine Firmen erschwinglich geworden.

Abbildung 1: OTP-Token. Quelle: Feitian

Dieser Beitrag gliedert sich in zwei Teile: Zunächst werden einige grundlegende Aspekte der Nutzerauthentifizierung dargestellt. Im zweiten Teil wird gezeigt, wie OTP-Authentifizierung funktioniert und wie man sie unter Linux mit Hilfe frei verfügbarer Software und preiswerter Tokens einrichten kann.

2 Grundlagen der Authentifizierung

Wenn ein Rechner die Aufgabe erhält, im Namen des Benutzers einen Auftrag zu erledigen, muss er in der Regel 3 Fragen beantworten:

1. Wer ist der Benutzer?
2. Ist er es wirklich?
3. Darf er die Operation ausführen?

Um die erste Frage zu beantworten, kann der Rechner einfach den Benutzer fragen. Leider besteht dabei die Gefahr, dass der Benutzer lügt. Deswegen wird die Aussage des Benutzers geprüft werden müssen. Die dritte Frage kann der Rechner dann anhand einer gespeicherten Sicherheits-Policy klären.

Zur Prüfung der Identität ("Authentifizierung") wird vom Benutzer ein Identitätsbeweis gefordert. Es gibt dafür mehrere Möglichkeiten. Alle diese Verfahren sind aus dem Alltag - auch unabhängig von Computer - gebräuchlich und wurden vermutlich schon seit Jahrhunderten, wenn nicht Jahrtausenden verwendet. Das Leisten einer Unterschrift - Beweis durch Können - dient der Authentisisierung genauso wie das Wiedererkennen einer Person anhand des Gesichtes - einem biometrischen Merkmal. Der Beweis durch Besitz - z.B. eines Siegelringes oder eines Schlüssels - ist ebenfalls allgemein bekannt. Ein indirekter Beweis ist ebenfalls möglich - man kann auf die Aussage einer dritten Partei, z.B. eines guten Freundes, vertrauen.

Für die Verwendung am Rechner - insbesonder über ein Netzwerk - gibt es jedoch eine wichtige Einschränkung: Die Führung des Beweises erfordert oft spezielle Eingabegeräte oder Schnittstellen. Zum Bestimmen eines Fingerabdrucks wird ein Fingerabdruckleser benötigt Aber auch eine Chip-Karte oder ein Datenträger mit einer Schlüsseldatei fällt darunter: Es werden spezielle Interfaces, oder mindestens spezielle Software benötigt, die mit Schlüsseldateien umgehen kann. In der Praxis hat sich daher zunächst der Beweis durch Wissen - eines geheimen Passworts - durchgesetzt, da eine entsprechende Eingabemöglichkeit in den meisten Fällen vorhanden ist.

Das Verfahren der Passwort-Authentifizierung ist allgemein bekannt. Eine geheime Zeichenfolge ("Geheimnis", "Secret") wird vom Benutzer eingegeben. Der Rechner hat die gleiche Zeichenfolge (oder eine kryptographische Transformation der Zeichenfolge) gespeichert und vergleicht sie mit der Nutzereingabe. Stimmen die Zeichenfolgen überein, gilt der Benutzer als authentifiziert. Eine Variante der Passwort-Authentifizierung ist ein Challenge-Response- Verfahren: Der Benutzer muss eine Frage beantworten, was er mit Hilfe des Geheimnisses kann.

Die Sicherheit des Verfahrens basiert dabei darauf, dass nur der echte Nutzer das korrekte Passwort eingeben kann, denn nur er kennt das Geheimnis. Ein ideales Passwort ist daher:

- einfach zu merken (bzw. nicht zu vergessen)
- nicht zu erraten, d.h. hinreichend komplex
- für jedes System anders (d.h. Passwort nicht zweimal verwenden)
- regelmäßig geändert
- geheim

Insbesondere der letzte Punkt ist wichtig. Er bedeutet unter anderem, dass das Passwort

- keinem Dritten mittgeteilt
- nicht aufgeschrieben
- nicht dupliziert (ausgespäht)

ist. Insbesondere erfordert das eine sichere Client-Umgebung und eine (abhör-)sichere Verbindung zwischen Client- und Server-System.

Da kein System sicher ist, o.g. Anforderungen daher manchmal - auch unbeabsichtigt - verletzt worden sein könnten, gibt es zwei wichtige Fallback-Kriterien. Zunächst sollte eine Kompromittierung bemerkt werden können. Das versehentliche Ausplaudern des Passworts wird die betreffende Person bemerken. Das Lesen eines Notizzettels am Monitor möglicherweise nicht. Nur wenn die Kompromittierung zeitnah bemerkt wird, lässt sich manchmal noch schlimmeres verhindern. Schließlich sollte das Passwort leicht änderbar sein.

3 Grenzen

Die den oben genannten Passwort-Richtlinien vollständig genügende Authentifizierung ist heutzutage unrealistisch geworden. Bereits die Anzahl an Web-Seiten, die eine Authentifizierung erfordern, ist unüberschaubar. Manche Passwörter verwendet man nur extrem selten, so dass man Gefahr läuft, sie zu vergessen. Oftmals werden darum gleiche Passwörter verwendet, oder die Passwörter werden aufgeschrieben (bzw. im Browser gespeichert). Das macht die Passwörter angreifbar.

Auch die geforderte sichere Client-Umgebung ist in vielen praktischen Fällen nicht gegeben.[1] Keylogger, Funktastaturen, Shoulder-Surfer und der leider immer noch oftmals fehlende Einsatz von SSL/TLS bei Webseiten sind nur die offensichtlichsten Beispiele.

Diese Grenzen wurden schon früh erkannt, und so wurde das Passwort weiterentwickelt.

[1] Ob man dieser Umgebung dann bezüglich der Sicherheit der restlichen Kommunikation, z.B. des Inhalts der gelesenen EMails, trauen sollte, wird hier nicht behandelt.

4 Einmal-Passwörter und Tokens

Viele Probleme lassen sich umgehen, wenn ein Passwort immer nur ein einziges Mal eingesetzt und dann sofort durch ein neues Passwort ersetzt wird. Ein Ausspähen des Passwortes wird so zwar nicht verhindert, jedoch ist das ausgespähte Passwort wertlos geworden. Das Verfahren wird zur Authorisierung von Transaktionen im Online-Banking in Deutschland schon seit vielen Jahren eingesetzt ("TAN"). Die Folge von Passwörtern wird dabei entweder nach einer Vorschrift generiert, oder die Passwörter werden zufällig bestimmt. In den meisten Fällen wird die Passwort-Folge den Nutzern dabei als Liste übergeben - ein Auswendiglernen ist wenig sinnvoll.

Die Authentifizierungs-Methode hat sich dadurch gewandelt - von "Beweis durch Wissen" in "Beweis durch Besitz", nämlich der Passwort-Liste. Um den Vorteil dieser Methode auszunutzen, muss die Liste als "Trusted Storage" betrachtet werden, d.h. man muss sicher gehen, dass die Liste geheim bleibt. Sicher aufbewahrtes Papier ist sehr vertrauenswürdig - vertrauenswürdiger jedenfalls als eine im Rechner gespeicherte Liste. Es besteht jedoch immer ein Risiko, dass diese Liste gestohlen oder unbemerkt kopiert wird. Es empfiehlt sich daher, das Einmal-Passwort um ein herkömmliches Passwort ("PIN") zu ergänzen. Man erreicht damit eine sogenannte Two-Factor-Authentifizierung, was einen deutlichen Sicherheitsgewinn darstellt.

Stand der Technik für Einmal-Passwörter ist die Generierung mit Hilfe von Tokens. Tokens sind kleine Geräte, meist in Scheckkarten- oder Schlüsselanhänger-Form, welche das derzeit gültige One-Time-Passwort berechnen und anzeigen. Am bekanntesten - weil schon seit langem verfügbar - sind hier die SecurID-Tokens der Firma RSA. Man kann das Token als kleines, vertrauenswürdiges Rechnersystem betrachten, in dem der Algorithmus und notwendiges Schlüsselmaterial ("Geheimnis") zur Erzeugung der Passwort-Sequenz gespeichert sind. Man geht dabei davon aus, dass es unmöglich ist, das Geheimnis aus dem Token auszulesen ("Tamper-Proof"). Die Tokens sind vollständig mit Plastik vergossen, so dass selbst ein (hoffentlich erfolgloser) Versuch, durch elektronische Manipulationen das Schlüsselmaterial auszulesen, deutliche Spuren hinterlässt ("Tamper-Evident").

Gibt ein Nutzer z.B. beim Ausscheiden aus der Firma das Token unbeschädigt zurück, kann daher davon ausgegangen werden, dass dieser Nutzer keine neuen gültigen Passwörter erzeugen und daher keinen Schaden mehr anrichten kann. Man vergleiche dies z.B. mit einem herkömmlichen mechanischen Schlüssel.

Deutlich billiger, aber entsprechend unsicherer, sind Software-Tokens. In diesem Fall wird die Token-Sequenz von einem Programm auf einem lokalen Rechner, z.B. einem SmartPhone ausgeführt. Die Sicherheit hängt hier wesentlich davon ab, ob das Schlüsselmaterial hinreichend gegen Auslesen geschützt werden kann.

5 Funktionsweise

Im wesentlichen die Funktionsweise des Tokens durch den Algorithmus der Passwort-Erzeugung definiert. Schon seit Anfang der 90er Jahre ist S/Key[3] bzw. dem dazu kompatiblen frei verfügbaren OPIE[2] ein in der Praxis eingesetztes System gewesen, allerdings meist mit Software-Client bzw. Papier-Listen. Es diente dazu, telnet-Netzwerkverbindungen wenigstens etwas zu schützen. "Echte" Tokens, wie das RSA SecurID-Token, verwendeten proprietäre Algorithmen.

Nachdem 2003 ein SourceForge-Projekt mit mOTP[1] eine frei verfügbare Implementierung für Software-Tokens gab, wurde ab dem Jahr 2004 durch die "Initiative for Open Authentication" [2] ein Algorithmus für Einmal- Passwörter spezifiziert und in RFC 4226 [5] formal publiziert. Diese Standardisierung ermöglichte die Herstellung von interoperablen Tokens verschiedener Hersteller, was letztlich zu sinkenden Preisen führte. Der RFC4226-Algorithmus wird HOTP, "HMAC-Based One Time Passwords" genannt. Er besteht aus drei Schritten:

1. Aus einem laufenden Zähler und dem Geheimnis wird eine HMAC-SHA1-Prüfsumme berechnet. HMAC (Hashed Message Authentication Codes) sind in RFC 2104 [4] definiert.

2. Diese Prüfsumme wird auf wenige Bits verkürzt

3. Aus diesen wenigen Bits wird eine 6...8-stellige Dezimal-Zahl berechnet.

Diese Zahl wird auf dem Display des Tokens angezeigt und kann nun vom Benutzer als Passwort verwendet werden. HOTP ist dabei ein sogenanntes Event-Based-System, d.h. der laufende Zähler wird durch ein Event, z.B. einen Knopfdruck, weitergeschaltet.

Eine Alternative sind Time-Based-Tokens. Hier wird als "Zähler" die aktuelle Uhrzeit, üblicherweise in 30-Sekunden oder 1-Minuten-Schritten, verwendet. Das Verfahren heisst TOTP[6] und ist derzeit ein Internet-Draft, die Standardisierung ist also noch nicht abgeschlossen. Time-Based-Tokens sind meist geringfügig teurer, da eine hinreichend ganggenaue Uhr eingebaut werden muss. Die Uhr im Token kann während der Token-Lebenszeit nicht gestellt werden.

6 Einsatz der Tokens unter Linux

Um ein OTP-System in Betrieb zu nehmen, sollte man zunächst eine Quelle für die Tokens suchen. Es sind dabei folgende Forderungen zu stellen:

[2] One-Time Passwords In Everything. Der Name "S/KEY" wurde als Markenzeichen von Bellcore geschützt.

1. Die Tokens sind OATH HOTP kompatibel. TOTP soll demnächst unterstützt werden.

2. Der Lieferant liefert die in die Tokens programmierten Schlüsseldaten.

3. Die Lieferantenkette bis zum Hersteller ist vertrauenswürdig.

Der erste Punkt ist selbsterklärend, die beiden anderen Punkte bedürfen näherer Erläuterung. Viele Hersteller/Lieferanten bieten gleichzeitig auch Token-Verwaltungs-Software für ihre Tokens an. Diese Software ist meistens verhältnismäßig teuer und wird per Token lizensiert. Im Internet wird berichtet, dass sich einige Hersteller weigern, die Schlüsseldaten herauszugeben, wenn nicht eine entsprechende Lizenz für die Token-Software mit erworben wird. Es empfiehlt sich daher, vorher nachzufragen. Der dritte Punkt liegt darin, dass sowohl Hersteller als auch die Lieferantenkette das Schlüsselmaterial kennen. Man sollte sich daher fragen, ob man den Herstellern und Händlern das nötige Vertrauen entgegenbringt, dieses Material nach Übergabe der Tokens zu löschen und keine Kopien angefertigt zu haben.

Der Autor verwendet Feitian C100 - Tokens, bezogen von Gooze [7]. Die Kosten für ein Token lagen zum Zeitpunkt der Bestellung unter 10 EUR. Andere Tokens, z.B. von Zyxel, sollen kompatibel sein. RSA SecurID-Tokens verwenden einen proprietären Algorithmus (Time-Based) und sind daher nicht geeignet.

Alternativ können die OTP-Sequenzen auch mit einem Diagnose-Tool, welches bei den meisten Software-Paketen enthalten ist, erzeugt werden. oder mit einem Smartphone-Software-Token berechnet werden. Software-Tokens sind für praktisch alle Mobiltelefone verfügbar, sowohl für iPhone[8], Android[9], oder herkömmliche J2ME-Geräte[10].

Die meisten Dienste unter Linux verwenden für die Passwort-Überprüfung PAM - Pluggable Authentication Modules. PAM ist ein modular aufgebautes System, bei dem für jeden Authentifizierungsvorgang definierte "Stapel" von Modulen abgearbeitet werden. So kann für jede Authentifizierungsmöglichkeit ein Modul definiert und in den entsprechenden Stapel eingebaut werden. Gesteuert wird PAM mit einer Konfigurationsdatei für jeden Dienst. Es gibt mit dem oath-toolkit[11] ein PAM-Modul zur Authentifizierung mittels OATH-Tokens. Zum Zeitpunkt der Manuskripterstellung war HOTP implementiert, an der Implementierung von TOTP wurde gearbeitet. Nach der Compilierung der Sourcen (configure –prefix=/ ; make; make install) kann das PAM-Modul eingesetzt werden.

Listing 1: /etc/pam.d/sshd

```
1 #%PAM-1.0
2 auth    sufficient      pam_oath.so usersfile=/etc/users.oath window=10
    digits=6
3 auth       include      system-auth
4 account    required     pam_nologin.so
5 account    include      system-auth
```

```
6 password   include      system-auth
7 session    optional     pam_keyinit.so force revoke
8 session    include      system-auth
9 session    required     pam_loginuid.so
```

Wir wollen als Beispiel den ssh-Server absichern. Wir ändern also die PAM-Konfigurationsdatei für den Dienst ssh - /etc/pam.d/sshd. Wie Listing 1 zeigt, wird eine zusätzliche Zeile - Zeile 2 - eingefügt, um das pam_oath-Modul als erstes Modul des "Auth"-Stapels auszuführen. Der Auth-Stapel wird bei jeder Passwort-Prüfung durch den ssh-Server abgearbeitet. In unserem Beispiel ist pam_oath als "sufficient", d.h. ausreichend, definiert. Ist die Authentifizierung mittels Tokens erfolgreich, so wird der Stapel nicht weiter abgearbeitet, sondern Erfolg gemeldet. Ansonsten wird die übliche Passwort-Authentifizierung über den system-auth-Stapel versucht. Durch geeignete Definition dieser Stapel kann der Login-Prozess beliebigen Vorgaben angepasst werden, insbesondere können verschiedene Authentisierungsmethoden zugelassen oder auch zwingend gefordert werden. Der Parameter "window" gibt an, wie genau der Zähler im Token mit dem Zähler im Rechner übereinstimmen muss. Setzt man das Fenster auf 0, fordert daher eine exakte Übereinstimmung, genügt ein versehentliches Betätigen des Tokens, um die Synchronisation zwischen Token und Rechner zu stören - der Rechner erwartet dann ein anderes als das gerade angezeigte Passwort. Setzt man den Wert zu groß, reduziert man die Sicherheit des Systems. Übliche Werte liegen zwischen 3 und 10. Die Anzahl der Stellen ist entsprechend des verwendeten Tokens zu wählen.

Listing 2: /etc/users.oath

```
1 HOTP    ml       test1    00000000000000000000000000000000000000000000          13
          632611   2011-01-17T00:06:13L
2 HOTP    ml1      test2    11111111111111111111111111111111111111111111          55
          552241   2011-01-16T03:36:12L
```

Schließlich müssen dem System noch die Geheimnisse der Tokens mitgeteilt werden. Dies geschieht über die Datei /etc/users.oath. Wichtig ist, dass diese Datei wie /etc/shadow behandelt wird, d.h. sie darf nur von root les- bzw. schreibbar sein. Der Aufbau dieser Datei ist einfach: Für jeden Nutzer gibt es eine Zeile. Nach der Kennung des Algorithmus - hier HOTP - steht der Nutzername, dann der feststehende Faktor - die PIN - im Klartext, und danach das Token-Geheimnis, welches man (s.o.) vom Hersteller erhalten hat. In den letzten Spalten wird der aktuelle Sequenzzähler, das letzte OTP-Passwort und die Zeit der Benutzung registriert. Zur Synchronisierung der Sequenzzähler von Rechner und Token lässt man sich vom Token ein Passwort ausgeben und verwendet oathtool wie folgt:

oathtool –hotp -w 1000 -d 6 33333333333333333333333333333333333333333 123456

Oathtool überprüft nun, ob es das 6-stellige Passwort 123456 in den ersten 1000 Positionen der Passwort-Serie aus dem Geheimnis

findet, und gibt die Nummer an. Diese Nummer kann dann in users.oath eingetragen werden.

In unserem Beispiel muss, damit das PAM-Modul mit dem Benutzer kommunizieren kann, in /etc/ssh/sshd_config der Parameter ChallengeResponseAuthentication auf "yes" gesetzt werden. Nach einem Neustart des ssh-Servers kann nun die Kombination aus PIN und dem nächsten angezeigten Einmal-Passwortes des Tokens zur Authentifizierung verwendet werden.

7 Zusammenfassung und Ausblick

Die Grundlagen der Authentifizierung wurden dargestellt und dabei die Schwächen herkömmlicher Passwörter herausgearbeitet. Diese Schwächen können durch die Verwendung von Einmalpasswörtern zumindest teilweise behoben werden. Die Erzeugung solcher Passwörter sowie die Einrichtung der serverseitigen Software "oath-toolkit" wurden erklärt. Oath-toolkit ist nur eine von mehreren verfügbaren Software-Lösungen zur Token-Anwendung; ist im Gegensatz zu "Enterprise-Class"-Lösungen jedoch frei erhältlich. Der Einsatz von OTP-Tokens und dieser Software ist daher für Privatanwender und kleine Unternehmen sehr attraktiv.

Literatur

[1] *Mobile One Time Passwords strong, two-factor authentication with mobile phones*. http://motp.sourceforge.net

[2] OATH Web site. http://www.openauthentication.org

[3] Haller: *The S/KEY One-Time Password System*. RFC 1760, February 1995. http://www.ietf.org/rfc/rfc1760.txt

[4] Krawczyk, Bellare, Canetti: *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104, February 1997. http://www.ietf.org/rfc/rfc2104.txt

[5] M'Raihi, Bellare, Hoornaert, Naccache, Ranen: *HOTP: An HMAC-Based One-Time Password Algorithm*. RFC 4226, December 2005. http://www.ietf.org/rfc/rfc4226.txt

[6] M'Raihi, Machani, Pei, Rydell: *TOTP: Time-based One-time Password Algorithm*. Internet Draft, Januar 2011. http://tools.ietf.org/html/draft-mraihi-totp-timebased-07

[7] Gooze Webshop.
http://www.gooze.eu

[8] *oathtoken HOTP/OATH one-time password token for iPhone*.
http://code.google.com/oathtoken/

[9] *androidtoken. Android OATH Token*.
http://code.google.com/p/androidtoken/

[10] DS3: *DS3 Oath Token Midlet Application*.
http://www.ds3global.com/index.php?option=com_content&task=view&id=71

[11] Josefson: *OATH Toolkit*.
http://www.nongnu.org/oath-toolkit/

Lokalisierung durch Messung von WLAN-Signallaufzeiten

Mario Haustein
mario.haustein@informatik.tu-chemnitz.de

1 Einleitung

Mobile Anwendungen erfreuen sich in jüngster Zeit starker Beliebtheit. Viele solcher Anwendungen benötigen Wissen über den Ort, an dem sie ausgeführt werden. Diese Information wird durch einen Lokalisierungsdienst bereitgestellt. Ein prominentes Beispiel hierfür ist GPS. Innerhalb von Gebäuden ist GPS mit herkömmlicher Hardware nicht nutzbar, und es müssen andere Methoden zum Einsatz kommen. Ein Ansatz besteht darin, existierende WLAN-Infrastrukturen für diesen Zweck zu nutzen. Zunächst – und zum Großteil auch noch heute – wurden dazu Informationen über die Feldstärke von empfangenen WLAN-Aussendungen herangezogen, wie u.a. in [BP00] beschrieben. Dieser als „Received Signal Strength Indication (RSSI)“ bezeichnete Wert ist leicht zu messen und wird von einem Großteil der verfügbaren WLAN-Adapter bereitgestellt. Es gibt aber auch ernsthafte Ansätze, die Signallaufzeit von WLAN-Übertragungen zu messen [GH05, HW08], welche über die Lichtgeschwindigkeit mit der Entfernung zusammenhängt.

Der Autor beschäftigt sich im Rahmen seiner Diplomarbeit damit, ein Framework für Linux bereitzustellen, das die Bestimmung von Signallaufzeiten ermöglicht. Diese Lösung soll dabei durch eine Reihe von Anforderungen möglichst universell einsetzbar sein.

- Die Lösung soll vollständig in Software umgesetzt werden. Die Modifikation von Firmware in WLAN-Adaptern ist zu vermeiden.
- Für das Endsystem soll keine besondere Hardware erforderlich sein, die spezielle Messeinrichtungen erfordert. Im günstigsten Fall kann jede „Off-the-Shelf-Hardware“ zum Einsatz kommen.
- Der normale Betriebsablauf eines WLAN-Netzes darf nicht tangiert werden. Es sind nur Methoden zugelassen, die der Standard IEEE 802.11 erlaubt.

Es bleibt jedoch zu bemerken, dass von einem WLAN-Lokalisierungsdienst nicht die selbe Genauigkeit erwartet werden kann wie z.B. von GPS. Wireless LAN war nie für diese Zwecke ausgelegt, und eine standardseitige Unterstützung von Lokalisierungsdiensten ist erst mit IEEE 802.11v zu erwarten.

Die Grundlagen von IEEE 802.11 sind für das Verständnis unablässig und werden zunächst in Abschnitt 2 angerissen. Es folgt anschließend ein kurzer Überblick über das MAC80211-Subsystem des Linux-Kerns. Die mathematischen Grundlagen der Laufzeitbestimmung werden in Abschnitt 4 erläutert und deren technische Umsetzung schließlich in Abschnitt 5 dargelegt.

2 IEEE 802.11

Maßgebend für die Wireless LAN Technologie ist der 11. Teil der Norm IEEE 802 [IEE07], die mehrere Darstellungsschichten (PHY-Layer) und eine Sicherungsschicht (MAC-Layer) definiert. Für eine vertiefte Einführung sei auf [Gas05, Rec08] verwiesen.

2.1 Der MAC-Layer

Die MAC-Schicht von Funknetzen muss eine Reihe von Problemen bewältigen, die in drahtgebundenen Netzen nicht auftreten

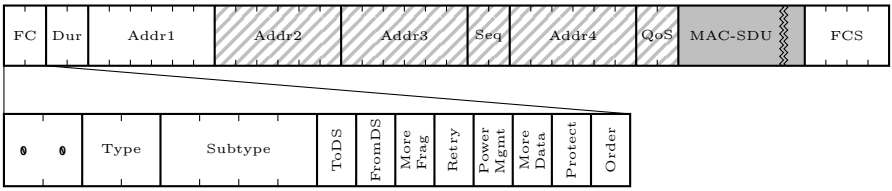


Abbildung 1: IEEE 802.11 Frame

können. Übertragungsfehler sind in Funknetzen keine Ausnahme, sondern zu erwarten. Die MAC-Schicht sieht deswegen einen Mechanismus vor, der dem ursprünglichen Sender die fehlerfreie Übertragung von Paketen bestätigt. Die Kommunikation läuft vollständig über ein Broadcast-Medium ab, wodurch es zu Kollisionen kommen kann. Bei klassischem Ethernet wird zunächst überprüft, ob das Medium belegt ist, bevor gesendet wird. Eine Station A, die zu B senden will, kann jedoch nicht wissen, ob C bereits das Medium bei B belegt. Das ist genau dann der Fall, wenn C in Reichweite von B, aber nicht von A liegt (siehe Abb. 2). Dieses Problem wird als „Hidden Station Problem“ bezeichnet.

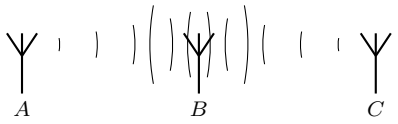


Abbildung 2: Das „Hidden Station“ Problem

Die Datenübertragung erfolgt nach einem festen Format, das in Kapitel 7 von IEEE 802.11 spezifiziert ist. Der IEEE 802.11 Frame ist in seiner allgemeinen Form in Abb. 1 dargestellt.

Der Nutzlast MAC-SDU wird ein Header vorangestellt. Zusätzlich wird der Frame durch ein CRC-Prüfzeichen (FCS) abgeschlossen. Die grau schattierten Elemente des Frames sind optional. Ob sie zum Frame gehören oder nicht, hängt vom Frame-Typ ab, der im Frame Control Feld (FC) kodiert wird. Einem optionalen Element des Frame-Kopfes kann nur die Nutzlast oder das rechte Nachbarelement des allgemeinen Frame-Formats folgen.

Es verwundert zunächst, dass bis zu vier Adressen Platz im Header finden. Aber in Funknetzen können sich Zieladresse (DA) und Empfängeradresse (RA) bzw. Quelladresse (SA) und Senderadresse (TA) unterscheiden. In Infrastrukturnetzwerken wird der gesamte Datenverkehr über Access Points abgewickelt. Bei der Übertragung eines Frames von einer Drahtlosstation in ein gewöhnliches LAN stellt er dabei den Empfänger des Frames fest. Das Ziel entspricht allerdings einer MAC-Adresse im LAN. Die Receiving Address (RA) ist immer in Addr1 und die Transmitting Address (TA) in Addr2 kodiert. Wie die Source Address und die Destination Address in den Adressfeldern kodiert sind, hängt von den Flags ToDS und FromDS ab und ist in Tabelle 1 aufgeschlüsselt. Die Flags sind gesetzt, wenn sich Ziel und Sender bzw. Quelle und Empfänger unterscheiden und somit das drahtgebundene Verbindungsnetz hinter dem AP in die Übertragung involviert ist. Access Points leiten Frames nur in dieses Distribution System (DS) weiter, wenn sich die Station vorher mit

To	From	Addr1	Addr2	Addr3	Addr4	Bedeutung
0	0	RA = DA	TA = SA	BSSID	-	Transfer im Ad-Hoc Modus
1	0	RA = BSSID	TA = SA	DA	-	Transfer Station → AP
0	1	RA = DA	TA = BSSID	SA	-	Transfer AP → Station
1	1	RA	TA	DA	SA	WLAN-Bridge

Tabelle 1: Bedeutung der Adressfelder im IEEE 802.11 Frame

dem AP assoziiert hat. Im nichtassoziierten Zustand werden nur einige spezielle Frames entgegengenommen, für die das „To DS“ und das „From DS“-Feld nicht gesetzt sind. In Ad-hoc-Netzen sind beide Flags immer gelöscht. Abschnitt 11.3 in IEEE 802.11 regelt alle Fälle detailliert.

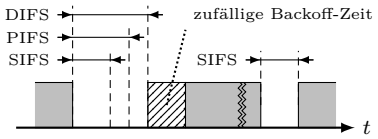


Abbildung 3: Frame-Abstände

Für einen möglichst kollisionsfreien Zugriff auf das Drahtlosmedium müssen sich die Stationen gegenseitig koordinieren. Zwischen der Übertragung zweier Frames muss für eine gewisse Zeit Ruhe auf dem Übertragungsmedium herrschen. Diese Frame-Abstände sind in Abb. 3 dargestellt. Nach Ende einer Übertragung ist ein Zugriff auf das Medium noch für den Zeitraum SIFS (short interframe space) verboten. Nach Ablauf des DIFS können sich prinzipiell mehrere Stationen um das Medium bewerben. Um die Kollisionswahrscheinlichkeit zu verringern, wird zur DIFS-Zeit zusätzlich eine zufällige Wartezeit nach dem Backoff-Verfahren addiert. Der PIFS-Abstand wird nur der Vollständigkeit halber erwähnt und ist im Rahmen dieser Arbeit nicht relevant. Eine Frame-Sequenz, bei der die Abstände zwischen allen konsekutiven Frames höchstens SIFS ist, nennt man unteilbar bzw. atomar. Sie kann nicht unterbrochen werden, da zu jedem Zeitpunkt geregelt ist, wer mit dem folgenden Frame das Medium belegen darf. Auf die konkreten Werte der symbolischen Platzhalter SIFS, PIFS und DIFS wird im folgenden Abschnitt eingegangen. Es gilt stets

$$PIFS = SIFS + s$$
$$DIFS = SIFS + 2s,$$

wobei s die sog. Slot-Zeit darstellt und ebenfalls von der PHY-Implementierung abhängt.

Laut Abschnitt 11.1 in IEEE 802.11 muss jede Implementierung eines WLAN-MAC-Layers eine sog. „timing synchronising function“ (TSF) implementieren. Basis für die TSF ist ein 64Bit-Zähler, der mit einer Rate von 1 MHz ± 0.01% inkrementiert wird. Der Standard legt zudem Verfahren fest, wie diese Zähler innerhalb einer Funkzelle synchronisiert werden. Dazu werden spezielle Beacon-Frames ausgetauscht. Die genaue Funktionsweise des Synchronisationsalgorithmus ist für den weiteren Verlauf allerdings nicht von Belang.

2.2 Die PHY-Layer

Die Darstellungsschichten bzw. „Physical Layer“ – kurz PHY – übernehmen die Aufgabe, die Frames der MAC-Schicht in einer für die Drahtloskommunikation geeigneten Form über einen Sender abzustrahlen und die an einem Empfänger abgegriffenen Signale wieder in 802.11-Frames umzuwandeln. IEEE 802.11 definiert eine ganze Reihe von PHY-Layern, z.B. ein auf Infrarotstrahlung basierendes PHY. Praktisch kommen jedoch nur „Direct sequence spread spectrum PHY (DSSS)“-PHYs und die aus ihnen hervorgegangenen Weiterentwicklungen nach 802.11a, 802.11b, 802.11g und 802.11n zum Einsatz.

- IEEE 802.11 spezifiziert Datenraten von 1 und 2 MBit/s im 2.4 GHz ISM-Band.
- IEEE 802.11a spezifiziert den Betrieb von Wireless LAN im 5 GHz ISM-Band. Es kommt orthogonales Frequenzmultiplexing (Orthogonal Frequency-Division Multiplexing, kurz OFDM) als Modulationsverfahren zum Einsatz, was Übertragungsraten von 6 MBit/s bis 54 MBit/s ermöglicht.
- IEEE 802.11b erweitert IEEE 802.11 durch Verwendung neuer Modulationsverfahren um die Datenraten 5.5 und 11 MBit/s.
- IEEE 802.11g setzt die Modulationsverfahren aus IEEE 802.11a im 2.4 GHz-Band um. Es werden entsprechende Vorkehrungen getroffen, damit Nicht-OFDM-Stationen keine Kollisionen auf dem Medium verursachen.

- IEEE 802.11n beschreibt die aktuellste Ausbaustufe der möglichen Übertragungsraten. Die Werte von bis zu 600 MBit/s werden allerdings nicht durch neue Modulationsverfahren erreicht, sondern durch die MIMO-Technik, bei der gleichzeitig mehrere Antennen am Sender und am Empfänger zum Einsatz kommen. Einerseits kann durch das Einschleifen von Verzögerungsgliedern in jedem Antennenpfad und anschließende Kombination der einzelnen Antennensignale ein besserer Signal-Rausch-Verhältnis erzielt werden. Andererseits ist es auch möglich, mehrere unterschiedliche Datenströme gleichzeitig zu übertragen. Eine knappe Einführung in die MIMO-Technik im Hinblick auf IEEE 802.11n kann in [HHSW09] nachgeschlagen werden.

Der grundsätzliche Aufbau einer WLAN-Aussendung ist unabhängig vom verwendeten Modulationsverfahren. Zunächst wird eine Präambel und ein Header mit einer geringen Datenrate ausgesandt. Die Präambel dient dazu, den Empfänger auf den folgenden Datenstrom zu synchronisieren. Der PLCP-Header kodiert die verwendete Datenrate und die Länge der folgenden MAC-Frames. Eine genauere Betrachtung der PHY-Layer würde den Rahmen dieses Artikels sprengen. Relevant für die hier thematisierten Zwecke ist nur, dass die Übertragungsdauer d eines Frames mit der Bitrate R und der Länge L berechnet werden kann. Hierzu noch einige ergänzende Anmerkungen.

- 802.11b erlaubt sog. „Short Frames“, deren Präambel und Header um 96 μs kürzer ist als bei konventionellen 802.11 „Long Frames“.
- OFDM-PHYs übertragen immer N_{DBPS} Bits in einem einzigen Symbol mit der Dauer von 4 μs. Der MAC-Frame muss ggf. mit Füllbits ergänzt werden, damit er ein Vielfaches von N_{DBPS} lang ist.

Die Längen der Aussendungen berechnen sich für IEEE 802.11(b) durch

$$t_{P\&H} = \begin{cases} 192\,\mu\text{s} & \text{802.11 o. „Long Frame“} \\ 96\,\mu\text{s} & \text{802.11b „Short Frame“} \end{cases}$$

$$d = t_{P\&H} + L/R.$$

und für IEEE 802.11{a,g} durch

$$N_{DBPS} = R \cdot 4\,\mu\text{s}$$
$$N_{SYM} = \lceil (22 + 8 \cdot L)/N_{DBPS} \rceil$$
$$d = 26\,\mu\text{s} + N_{SYM} \cdot 4\,\mu\text{s}.$$

Für die Frame-Abstände gilt bei IEEE 802.11(b)

$$SIFS = 10\,\mu\text{s}$$
$$s = 20\,\mu\text{s}$$

und für IEEE 802.11{a,g}[1]

$$SIFS = 10\,\mu\text{s}$$
$$s = \begin{cases} 20\,\mu\text{s} & \text{802.11b-Kompatibilität} \\ 9\,\mu\text{s} & \text{sonst.} \end{cases}$$

3 Wireless LAN unter Linux

Mit Kernelversion 2.6.22 wurde als zukünftige Alternative zur bestehenden Wireless Extension [Tou] das MAC80211-Subsystem [mac] zur Verwaltung von WLAN-Hardware eingeführt. MAC80211 übernimmt nicht nur die Verwaltung der Hardware, sondern implementiert in weiten Teilen die 802.11-MAC-Schicht, was eine Reihe von positiven Nebeneffekten hat.

Im Gegensatz zur Wireless Extension werden die Netzwerkschnittstellen nicht durch den WLAN-Treiber sondern durch MAC80211 bereitgestellt. Es ist sogar möglich, jedem WLAN-Gerät – im Sprachgebrauch von MAC80211 als „PHY“ bezeichnet – mehrere Netzwerk-Schnittstellen zuzuweisen. Diese Schnittstellen, werden auch „Virtual Interface (VIF)“ genannt. Das Erzeugen und Löschen von VIFs ist mit dem Kommandozeilenprogramm `iw` möglich.

[1] Streng genommen sind d und $SIFS$ für 802.11a und 802.11g unterschiedlich definiert. Diese Diskrepanz wird hier jedoch hingenommen und wirkt sich nicht auf die Laufzeitmessung aus.

```
$ iw phy ⟨PHY⟩ interface add
    ⟨Interface⟩ type ⟨Typ⟩
```

Der Typ bezeichnet dabei den Betriebsmodus der Schnittstelle und kann auch direkt mittels `iw` für bestehende VIFs gesetzt werden. Vorher muss das Interface ggf. deaktiviert werden.

```
$ ip link set ⟨Interface⟩ down
$ iw dev ⟨Interface⟩ set type ⟨Typ⟩
$ ip link set ⟨Interface⟩ up
```

Mögliche Betriebsmodi sind:

managed Das Interface repräsentiert eine Station in einem Infrastrukturnetzwerk. Eine Verbindung mit einer Zelle kann durch

```
$ iw dev ⟨Interface⟩ connect
    ⟨SSID⟩ [⟨Frequenz⟩] [⟨BSSID⟩]
$ iw dev ⟨Interface⟩ disconnect
```

hergestellt bzw. getrennt werden. Über das Netzwerkinterface werden Ethernet-Frames ausgetauscht.

ibss Über IBSS-Schnittstellen wird der Zugang zu Ad-hoc-Netzwerken ermöglicht. Das Beitreten und Verlassen einer Ad-hoc-Zelle erfolgt über:

```
$ iw dev ⟨Interface⟩ ibss join
    ⟨SSID⟩ ⟨Frequenz⟩
$ iw dev ⟨Interface⟩ ibss leave
```

Es werden ebenfalls Ethernet-Frames ausgetauscht.

monitor Im Monitor-Modus werden alle empfangenen 802.11-Frames, also insbesondere auch Steuer- und Management-Frames, an den Prozessbereich weitergeleitet. Dem eigentlichen Frame wird dabei ein sog. Radiotap-Header vorangestellt. Neben der Möglichkeit, den Datenverkehr über das Funkmedium zu überwachen, können auch mit einem Radiotap-Header versehene 802.11-Frames an das MAC80211-Subsystem übergeben werden. Die Frames werden anschließend ohne weitere Bearbeitung ausgesandt. Diese Technik wird als „Packet Injection“ bezeichnet.

Zur Steigerung der Nettodatenrate wählt ein Rate-Control-Algorithmus eine Datenrate für die Übertragung von Frames. Er versucht das Optimum zwischen Bruttodatenrate und der Wahrscheinlichkeit eines Übertragungsfehlers, die mit der Bruttodatenrate steigt, zu finden. Aktuell kommt hierzu der Minstrel-Algorithmus[2] zum Einsatz.

Der bereits erwähnte Radiotap-Header dient dazu, Informationen zwischen Anwendungsprogrammen und WLAN-Treiber zu transportieren, die über die im IEEE 802.11 MAC-Header kodierten Informationen hinausgehen. Im Wesentlichen handelt es sich dabei um Daten, die für die PHY-Schicht relevant sind, oder Statusinformationen die vom WLAN-Treiber gesammelt werden. Beispiele hierfür sind:

TSFT Im TSFT-Feld wird für einen empfangenen Frame der Wert des TSF-Zählers festgehalten, den dieser zu dem Zeitpunkt annimmt, zu dem die Firmware des Geräts das erste Bit des Frames an den MAC-Layer übergibt. Genau dieser Mechanismus wird später dazu dienen, Laufzeitinformationen zu ermitteln. Nicht jeder WLAN-Chipsatz stellt diesen Wert bereit.

Flags & Rate geben Auskunft über die Datenrate und die Modulation empfangener bzw. zu sendender Frames.

TX Flags legen das Verhalten des Treibers für Frames fest, die gesendet werden sollen. Es kann z.B. das RTS/CTS-Handshake angefordert werden, um festzustellen, dass das Drahtlosmedium am Empfängerort überhaupt frei ist. Dem Treiber kann ebenfalls mitgeteilt werden, ob er eine Empfangsbestätigung des Empfängers durch einen sog. ACK-Frame erwarten soll.

Das Radiotap-Protokoll erlaubt es, nur die benötigten Datenfelder im Protokollkopf zu kodieren. Alle definierten Datenfelder, Kandidaten für spätere Erweiterungen und ihre Bedeutung werden von der Entwicklergemeinde unter [rad] zusammengetragen.

[2] vgl. http://wireless.kernel.org/en/developers/Documentation/mac80211/RateControl/minstrel

4 Funktionsweise

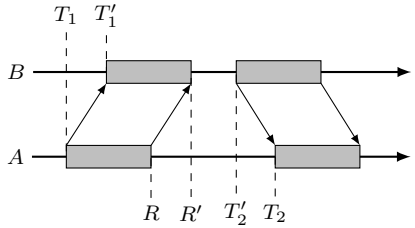


Abbildung 4: Time of Flight

Die Signallaufzeit einer WLAN-Aussendung lässt sich mit dem Time-of-Flight-Verfahren bestimmen, wie es schematisch in Abb. 4 dargestellt ist. Zum Zeitpunkt T_1 startet A eine Aussendung, die die von der Entfernung zu B abhängige Signallaufzeit benötigt, um dort registriert zu werden. Zum Zeitpunkt R' ist der Empfang der Nachricht bei B abgeschlossen. B verarbeitet nun den empfangenen Frame. Nach einer gewissen Zeit sendet er zum Zeitpunkt T_2' ein Antwort-Paket an A, was dort zu T_2 registriert wird. Die Zeitdifferenz zwischen T_1 und T_2 kann von A gemessen werden und setzt sich aus

$$\begin{aligned}\Delta = T_2 - T_1 = &\underbrace{(T_1' - T_1)}_{t} + \underbrace{(R' - T_1')}_{d} + \\ &\underbrace{(T_2' - R')}_{g} + \underbrace{(T_2 - T_2')}_{t} \\ &= 2t + d + g\end{aligned}$$

zusammen. Dabei bezeichnet t die Signallaufzeit, die für beide Übertragungsrichtungen als gleich angesetzt wird, d die Dauer der Aussendung von A und g die Zeit, die bei B zwischen Empfang des ersten Frames und Beginn der Aussendung des zweiten Frames vergeht. Der Wert von d kann über die Gleichungen in Abschnitt 2.2 berechnet werden. Je nachdem, wie der TOF-Ablauf realisiert wird, ist der Wert von g durch die Regelungen des Standards IEEE 802.11 bekannt. Damit kann schließlich t berechnet werden:

$$t = \frac{\Delta - d - g}{2}$$

Es bleibt hervorzuheben, dass die Differenz $T_2 - T_1$ an der Uhr bei A abgelesen wird und somit nur deren Drift die Genauigkeit der Messung beeinflusst. Zwischen A und B ist keine Uhrensynchronisation notwendig.

Am Ende von Abschnitt 3 wurde bereits angesprochen, dass der TSF-Zähler für diese Zwecke genutzt werden kann. Seine Zeitstempel liegen aber nur in einer Auflösung von einer Mikrosekunde vor. Dennoch ist es möglich, damit Zeiten im Nanosekundenbereich zu erfassen, wie man durch folgende statistische Rechnung einsieht.

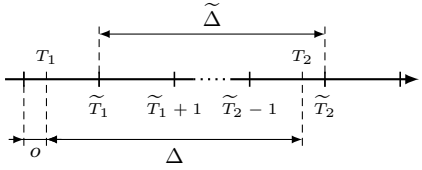


Abbildung 5: Bestimmung von Δ

In Abbildung 5 ist der Zeitstrahl der Uhr bei A abgebildet. Nach einer Periode von $1\,\mu$s schaltet die Uhr einen Schritt weiter. Wechselte der Zeitstempel $\widetilde{T}$ dieser Uhr genau zum Zeitpunkt $T = 0$ der kontinuierlichen Globalzeit von Null auf Eins, lesen wir zu jedem Zeitpunkt T der Globalzeit den Wert $\widetilde{T} = \left\lceil \frac{T}{P} \right\rceil$ ab. In den folgenden Rechnungen stellt $\langle X \rangle$ den Nachkommaanteil der Größe X dar. Für die Zeitstempeldifferenz $\widetilde{\Delta}$ gilt

$$\begin{aligned}\widetilde{\Delta} &= \left\lceil \frac{T_2}{P} \right\rceil - \left\lceil \frac{T_1}{P} \right\rceil \\ &= \left\lceil \frac{T_1 + \Delta}{P} \right\rceil - \left\lceil \frac{T_1}{P} \right\rceil \\ &= \left\lceil \left\langle \frac{T_1}{P} \right\rangle + \frac{\Delta}{P} \right\rceil - \left\lceil \left\langle \frac{T_1}{P} \right\rangle \right\rceil,\end{aligned}$$

wobei der Term $\left\langle \frac{T_1}{P} \right\rangle$ durch $\frac{o}{P}$ ersetzt werden kann:

$$\widetilde{\Delta} = \left\lceil \frac{o}{P} + \frac{\Delta}{P} \right\rceil - \left\lceil \frac{o}{P} \right\rceil$$

Bei einer großen Messreihe $\widetilde{\Delta}_i$ kann der Phasenversatz o als gleichverteilt im Intervall

$[\,0, P\,[$ angenommen werden, wenn das Ereignis T_1 die Uhr nicht beeinflusst. Durch Fallunterscheidung erhält man schließlich:

$$\widetilde{\Delta} = \begin{cases} \left\lceil \frac{\Delta}{P} \right\rceil & o = 0 \\ \left\lfloor \frac{\Delta}{P} \right\rfloor & 0 < \frac{o}{P} \leq 1 - \left\langle \frac{\Delta}{P} \right\rangle \\ \left\lfloor \frac{\Delta}{P} \right\rfloor + 1 & 1 > \frac{o}{P} > 1 - \left\langle \frac{\Delta}{P} \right\rangle \end{cases}$$

Bei zufällig verteiltem o ist $\widetilde{\Delta}$ selbst eine Zufallsgröße, die einer Zweipunktverteilung folgt. Ihr Erwartungswert berechnet sich zu:

$$\begin{aligned} E\left[\widetilde{\Delta}\right] &= \left(1 - \left\langle \frac{\Delta}{P} \right\rangle\right) \cdot \left\lfloor \frac{\Delta}{P} \right\rfloor + \\ &\quad \left\langle \frac{\Delta}{P} \right\rangle \cdot \left(\left\lfloor \frac{\Delta}{P} \right\rfloor + 1\right) \\ &= \left\lfloor \frac{\Delta}{P} \right\rfloor + \left\langle \frac{\Delta}{P} \right\rangle \\ &= \frac{\Delta}{P} \end{aligned}$$

Für n unabhängige, aber gleich verteilte Messwerte $\widetilde{\Delta}_i$ ist die Funktion

$$\widehat{\Delta} = \frac{P}{n} \sum_{i=1}^{n} \widetilde{\Delta}_i$$

ein erwartungstreuer Schätzer für Δ. Die Zweipunktverteilung von $\widetilde{\Delta}$ lässt sich auch als Summe der Konstanten $\left\lfloor \frac{\Delta}{P} \right\rfloor$ und einer Bernoulli-Verteilung mit dem Parameter $\left\langle \frac{\Delta}{P} \right\rangle$ darstellen. Die Konstante kann als Term $n \cdot \left\lfloor \frac{\Delta}{P} \right\rfloor$ aus der Summe herausgezogen werden, sodass n unabhängige Bernoulliverteilte Größen addiert werden. Die n-fache Faltung dieser Bernoulli-Verteilung ergibt eine $\mathrm{B}\left(n, \left\langle \frac{\Delta}{P} \right\rangle\right)$-Binomialverteilung. Die Verteilung von $\widehat{\Delta}$ lässt sich formal als

$$\widehat{\Delta} \sim P \cdot \left\lfloor \frac{\Delta}{P} \right\rfloor + \frac{P}{n} \cdot \mathrm{B}\left(n, \left\langle \frac{\Delta}{P} \right\rangle\right)$$

darstellen. Die Varianz von $\widehat{\Delta}$ ist ein Maß für die Genauigkeit des Schätzers und bestimmt sich zu

$$D^2\left[\widehat{\Delta}\right] = \frac{P^2}{n} \cdot \left\langle \frac{\Delta}{P} \right\rangle \cdot \left(1 - \left\langle \frac{\Delta}{P} \right\rangle\right) \leq \frac{P^2}{4n}.$$

Wie bereits intuitiv klar ist, wird der Schätzer um so genauer, je kleiner die Auflösung P ist oder je mehr Messwerte aufgenommen werden. Somit können mit hinreichend großem n prinzipiell Zeiten im ns-Bereich mit einer Uhr in μs-Auflösung bestimmt werden.

Für eine konkrete Realisierung eines TOF-Ablaufs lassen sich Mechanismen der IEEE-802.11-MAC-Schicht ausnutzen, die die in Abschnitt 2.1 angerissenen Effekte behandeln.

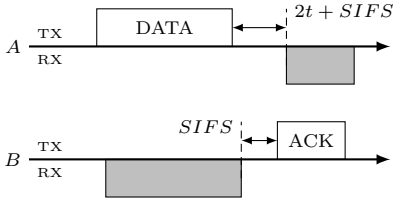


Abbildung 6: DATA-ACK-Sequenz

Ein Drahtlosmedium kann nicht mit hinreichend großer Sicherheit garantieren, dass ein ordnungsgemäß gesendeter Frame beim Empfänger auch korrekt empfangen wird. Daher ist bereits auf MAC-Ebene ein Bestätigungsmechanismus implementiert. Nach Verstreichen der Zeit $SIFS$ antwortet der Empfänger eines Frames dem ursprünglichen Sender mit einem speziellen ACK-Frame. Nicht alle Frames werden mit einem ACK bestätigt, auf alle Fälle jedoch Daten-Frames, die nicht für Broadcast- oder Multicast-Adressen bestimmt sind. Vom Anfang des DATA-Frames bis zum Anfang des ACK-Frames bei A vergeht die Zeit

$$\Delta = d_{\mathrm{DATA}} + 2t + SIFS.$$

Dem „Hidden Station"-Problem kann durch ein sog. RTS-CTS-Handshake[3] begegnet werden. Vor dem eigentlichen Frame-Austausch wird ein RTS-Frame ausgesandt, der nach Ablauf der Zeit $SIFS$ mit einem CTS-Frame quittiert wird. Bleibt dieser CTS-Frame bei A aus, war das Medium bei B vermutlich belegt. Der Frame-Austausch wird deshalb später

[3] Abk. für „Request to Send" und „Clear to Send"

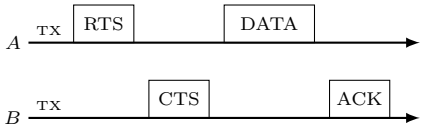


Abbildung 7: RTS-CTS-Handshake

wiederholt. Die RTS- und CTS-Frames enthalten in ihrem Duration-Feld (Dur) zusätzlich eine Zeitangabe, wie lange das Medium bis zum Abschluss des ACK-Frames belegt sein wird. Jede dritte Station wertet dieses Feld aus und nimmt bis zum Ende dieser Sperrzeit keine Aussendung vor, um zusätzliche Kollisionen zu vermeiden. Wird hingegen ein CTS-Frame empfangen, startet die eigentliche Datenübertragung nach einer weiteren $SIFS$-Zeit. Bei einem erfolgreichen Handshake beträgt die Zeit Δ zwischen Anfang des RTS-Frames und Anfang des ACK-Frames

$$\Delta = d_{\mathrm{RTS}} + d_{\mathrm{CTS}} + d_{\mathrm{DATA}} + 3t + 3 \cdot SIFS.$$

Die hier beschriebenen Ansätze wurden bereits in [GH05, HW08] verfolgt.

5 Umsetzung

Nachdem im zurückliegenden Abschnitt die prinzipielle Funktionsweise der Laufzeitmessung dargelegt wurde, soll in diesem Abschnitt auf deren technische Umsetzung eingegangen werden. Zur Realisierung möglichst nur einfache Linux-Bordmittel eingesetzt werden.

5.1 Messwerte erfassen

Für die Erfassung der Umlaufzeit Δ wurde in Abschnitt 2.2 bereits der TSF-Zähler eines WLAN-Adapters vorgeschlagen, da ihm durch Firmware oder Interrupt-Handler kein Jitter überlagert wird, der die Messwerte streut. Der Stand des TSF-Zählers wird über das Radiotap-Protokoll für alle empfangenen Frames an Userspace-Programme ausgeliefert.

Ein Problem stellt dabei dar, dass für den Initialframe, der zu T_1 versandt wird, kein TSF-Zeitstempel erfasst wird. Dieser Nachteil ist auch nicht durch Modifikation des Linux-Kerns behebbar, da in den meisten Fällen noch nicht einmal die Firmware der WLAN-Adapter den TSF gesendeter Frames zurückmeldet. Aus diesem Grund ist der Einsatz eines zweiten WLAN-Adapters M bei A unumgänglich. Für Mobilgeräte mit USB-Hosts stellt dies dank billiger USB-WLAN-Adapter keine all zu starke Einschränkung dar. Dieser zweite WLAN-Adapter ist vollkommen passiv und hat den Zweck, sowohl die Aussendungen von A als auch von B zu empfangen. Die Signallaufzeit von A zu M ist vernachlässigbar. An M liegen somit für die Ereignisse T_1 und T_2 gültige TSF-Zeitstempel vor. Weil M nicht Mitglied einer Funkzelle ist, wird sein TSF-Zähler auch nicht durch die Synchronisationsalgorithmen beeinflusst.

Die eigentliche Datenerfassung kann mit einem beliebigen Netzwerksniffer wie tcpdump [tcp], Wireshark [wir] oder auch direkt durch libpcap [pca] erfolgen. Vorher muss das Monitor-Interfaces konfiguriert werden.

```
$ ip link set ⟨Monitor-Device⟩ up
$ iw dev ⟨Monitor-Device⟩ set channel
  ⟨Kanal⟩
$ iw dev ⟨Minitor-Device⟩ interface add
  mon type monitor
$ ip link set mon up
```

Nun können am Interface `mon` alle empfangenen IEEE 802.11-Frames, mitsamt einem Radiotap-Header, aufgefangen werden. Wireshark kann die Frames zur späteren Auswertung direkt in eine Datei (z.B. „`mon.pcap`“) umleiten. Es ist zweckmäßig, mittels eines PCAP-Filters nur die Pakete aufzunehmen, die für die spätere Auswertung benötigt werden.

```
$ tshark -i mon -w mon.pcap -f \
  "(wlan addr1 ⟨B⟩ and wlan addr2 ⟨A⟩ \
     and (subtype rts or (type data \
     and wlan addr3 ⟨BSS⟩))) or (wlan \
     addr1 ⟨A⟩ and (subtype cts or \
     subtype ack))" \
  -c ⟨Anzahl⟩
```

Im Beispiel beendet sich Wireshark, nachdem ⟨*Anzahl*⟩ viele Pakete aufgenommen wurden. Im Filter ist ⟨*A*⟩ durch die MAC-Adresse von *A* zu ersetzen. Analoges gilt für *B*. Die BSSID ist ein Identifikator für die Funkzelle. In Ad-hoc-Netzen wird sie von der ersten Station dieser Zelle zufällig erzeugt, in Infrastrukturnetzen stimmt sie mit der MAC-Adresse des Access Points überein.

Tabelle 2 listet die vom Autor getestete Hardware auf. Damit auch an der IWL6000-Karte TSF-Zeitstempel abgegriffen werden konnten, war eine kleine Modifikation des Treibers notwendig. Hier musste nur ein Flag gesetzt werden, das die Übernahme des Zeitstempels aus der Firmware in den Radiotap-Header veranlasst.

5.2 Messwerte erzeugen

Nachdem nun klar ist, wie die TOF-Frames für die spätere Auswertung festgehalten werden, stellt sich die Frage wie, ein solcher TOF-Ablauf überhaupt ausgelöst werden kann. Die Autoren von [GH05] nutzen hierzu das Programm `ping`. Das ICMP-Echo-Request wird auf MAC-Ebene in einen DATA-Frame verpackt und durch ACK bestätigt. Das ICMP-Echo-Reply trägt nicht zur Laufzeitbestimmung bei. Hier soll eine alternative Lösungsmöglichkeit vorgeschlagen werden, die gegenüber dem Ping-Ansatz mehrere Vorteile verspricht.

Der IEEE 802.11 MAC-Layer definiert sog. NULL-Frames. Diese sind spezielle DATA-Frames, die keine Nutzlast enthalten. Ihr einziger Zweck ist die Übermittlung des PM-Bits im FC-Feld des 802.11-Headers. Dieses Bit zeigt einer anderen Station an, dass die übertragende Station anschließend in den Stromsparmodus wechselt und zukünftig nicht auf eingehende Frames reagieren wird. Die NULL-Frames werden direkt an die Station zugestellt, wodurch die „To DS“- und „From DS“-Flags immer gelöscht sind. Daraus ergeben sich eine Reihe von Vorteilen

- Ein Access Point beantwortet den Frame auch mit einem ACK, wenn die Station nicht mit ihm assoziiert ist.
- Der TOF-Ablauf belastet das Verbindungsnetzwerk hinter einem Access Point nicht. Eine Belastung des Drahtlosmediums durch überflüssige Ping-Replys findet nicht statt.
- Es ist kein zentraler Server zum Beantworten von Ping-Anfragen notwendig.

Der NULL-ACK-Sequenz kann natürlich ein RTS-CTS-Handshake vorausgehen. Die Erzeugung von NULL-Frames ist jedoch nicht so einfach wie das Erzeugen von Ping-Paketen. Da über Monitor-Schnittstellen allerdings beliebige Frames in den Kern eingeschleust werden können, ist das Auslösen einer NULL-ACK-Sequenz mit einem einfachen C-Programm in Verbindung mit libpcap möglich. Zunächst muss ein Monitor-VIF am Haupt-Interface erzeugt werden:

Gerät	Treiber	Bemerkung
IWL6000	`iwlagn`	Minimaler Patch in `iwlagn_rx_reply_tx()` erforderlich, um TSFT in Radiotap-Header auszuliefern. Nutzt eigenen Rate-Control-Algorithmus, der aber mittels Kernelpatch umgangen werden kann.
TL-WN422G (v2) TL-WN721N	`ath9k_htc`	USB. Gute Empfängerempfindlichkeit. TSF-Zähler liefert jedoch inkonsistente Werte, die aber u.U. korrigiert werden können. Der Rate-Control-Algorithmus ist in der Firmware implementiert. Es gibt keine Möglichkeit Frames mit einer bestimmten Datenrate einzuschleusen. Der TSF-Zeitstempel wird zum Ende der Aussendung erfasst.
AWUS036EH	`rtl8187`	USB. Schlechte Empfängerempfindlichkeit. Ad-hoc-Modus wird vom Treiber (noch) nicht unterstützt. Frames mit beliebiger Datenrate können ohne Treibermodifikation eingeschleust werden.

Tabelle 2: Getestete Hardware

```
# Haupt-Device
$ ip link set ⟨Haupt-Device⟩ up
$ iw dev ⟨Haupt-Device⟩ set channel
  ⟨Kanal⟩
$ iw dev ⟨Haupt-Device⟩ interface add inj
    type monitor
$ ip link set inj up
```

Anschließen können mit folgendem Code `cnt` Kopien eines NULL-Frames über das Interface `iface` abgesetzt werden. Der Frame ist dabei samt vorangestelltem Radiotap-Header im Puffer `frame` abgelegt und `len` Bytes lang. Zwischen zwei aufeinander folgenden Frames wird etwa `delay` Mikrosekunden gewartet.

```
void inject(char *iface, unsigned int
  cnt, uint8_t *frame, unsigned int
  len, unsigned long delay)
{
  char errbuf[PCAP_ERRBUF_SIZE];
  pcap_t *pcap;
  int result;
  unsigned int i;

  strcpy(errbuf, "");
  pcap = pcap_open_live(iface, 800, 1,
    20, errbuf);
  if(pcap == NULL)
    error(-1, 0, "ERROR:␣%s\n", errbuf)
      ;
  if(strlen(errbuf))
    fprintf(stderr, "WARNING:␣%s\n",
      errbuf);

  for(i = 0; i < cnt; i++)
  {
    result = pcap_inject(pcap, frame,
      len);
    if(result < 0)
    {
      fprintf(stderr, "ERROR:␣%s\n",
        pcap_geterr(pcap));
      break;
    }

    usleep(delay);
  }

  pcap_close(pcap);
}
```

Die Unterstützung der im Radiotap-Protokoll definierten Datenfelder für das Einschleusen von Frames ist im Linux-Kern nur unvollständig umgesetzt. Mit Version 2.6.37 ist es z.B. nicht möglich, die Bitrate für eine Aussendung festzulegen. Genau das ist aber notwendig, um Verfälschungen der Messwerte durch eventuelle Abhängigkeiten zur Datenrate ausschließen zu können. Es existierten für diesen Zweck aber bereits diverse Patches,[4] die im Falle eines vorhandenen Rate-Felds im Radiotap-Header den Rate-Control-Algorithmus des Linux-Kerns unterdrücken. Jedoch stellte sich heraus, dass einige Treiber weitere Modifikationen erfordern, um das Rate-Controling zu umgehen (vgl. Tabelle 2). Weiterhin wurden vom Autor u.a. noch folgende Patches entwickelt:

- Das TX-Flags Feld im Radiotap-Header wird ausgewertet, um ein RTS-CTS-Handshake erzwingen zu können.
- Es kann die Sendeantenne ausgewählt werden, sofern der Treiber diese Funktion umsetzen kann.
- Der Treiber kann angewiesen werden, nach der Übertragung eines Frames nicht auf den ACK-Frame zu warten. Ggf. könnte dann durch eine anschließende empirische Untersuchung festgestellt werden, ob Zeitstempel, die in den Interrupt-Handlern erfasst werden, eine Laufzeitmessung erlauben.

5.3 Messwerte auswerten

Die Auswertung der Messung kann getrennt von der Datenerfassung erfolgen. Wireshark kann festgehaltene Frames im CSV-Format exportieren.

```
$ tshark -r mon.pcap -T fields \
  -E header=n -E separator=, \
  -e frame.number          \
  -e frame.len             \
  -e radiotap.mactime      \
  -e radiotap.length       \
  -e radiotap.datarate     \
  -e radiotap.channel      \
  -e radiotap.flags        \
  -e wlan.fc.type_subtype  \
  -e wlan.sa               \
  -e wlan.da               \
  -e wlan.ta               \
  -e wlan.bssid ...
```

[4] http://thread.gmane.org/gmane.linux.kernel.wireless.general/47441 (keine Unterstützung für 802.11n) oder https://patchwork.kernel.org/patch/118564/ (unterstützt 802.11n, ist aber für Nicht-802.11n-Bitraten fehlerhaft)

Mit der `e`-Option sind alle zu exportierenden Wireshark-Felder anzugeben. Das hier gezeigte Beispiel ist aus Gründen der Übersichtlichkeit nicht vollständig. In Version 1.2.13 kann Wireshark noch nicht alle in [rad] definierten Datenfelder dekodieren. Daher muss der Decoder durch Patches erweitert werden.

Abschließend müssen nur noch alle gültigen NULL-ACK-Sequenzen (bzw. RTS-CTS-NULL-ACK-Sequenzen) aus dieser Folge von Frames ausgewertet werden. Eine Sequenz ist gültig, wenn ...

- die Frames unmittelbar aufeinander folgend aufgenommen wurden,
- die TA des NULL-Frames (und RTS-F.) mit der MAC-Adresse von A übereinstimmt,
- die RA des NULL-Frames (und RTS-F.) mit der MAC-Adresse von B übereinstimmt und
- die RA des ACK-Frames (und CTS-F.) mit der MAC-Adresse von A übereinstimmt.

Formal lässt sich die Datenauswertung wie in Algorithmus 1 beschreiben. Es ist dabei darauf zu achten, dass A und B für alle betrachtete Sequenzen den selben Stationen entsprechen und Modulation und Datenrate aller Frames identisch sind. Beim DATA-ACK-Verfahren müssen alle DATA-Frames die selbe Länge aufweisen.

Abbildung 8 zeigt ein Korrelogramm zwischen den ermittelten $\widehat{\Delta}$ und der Referenzentfernungen r, die jeweils in Schritten von 3 m festgelegt wurde. In der zugehörigen Tabelle 3 gibt die n-Spalte die Anzahl der verwertbaren NULL-ACK-Sequenzen an. Die Messungen fanden auf einem langen, geraden Flur statt, sodass von einer linearen Signalausbreitung ausgegangen werden kann. Für jede Entfernung wurden 5 Durchgänge zu je 10000 TOF-Abläufen, bei einer Datenrate von 54 MBit/s für die NULL-Frames, ausgeführt. Als Monitor-Gerät kam ein USB-WLAN-Adapter vom Typ TL-WN422G (vgl. Tabelle 2) zum Einsatz, der jeweils die Zeitdifferenz von Frame-Ende zu Frame-Ende bestimmt. Die Station B war mit einer Intel-Karte vom Typ IPW2200 ausgerüstet, die im Ad-hoc-Modus betrieben wurde.

6 Zusammenfassung/Ausblick

Im Rahmen dieser Arbeit wurde beispielhaft gezeigt, dass eine messbare Korrelation zwischen der Signallaufzeit von WLAN-Aussendungen und der Entfernung zwischen den WLAN-Stationen besteht. Die Messungen können mit konventionellen Linux-Bordmittel vorgenommen werden. Es wurden weiterhin Methoden vorgeschlagen, die sich im Rahmen eines einfachen Lokalisierungsframeworks implementieren lassen. Mit Hilfe von Kalibrierungsinformationen

Algorithmus 1 Auswertung der gemessenen $\widetilde{\Delta}_i$

$L \leftarrow ()$
Berechne Δ aus Angaben zu Modulation, Datenrate und Frame-Größe.
for $i = 1, \ldots, n-1$ **do**
  if (f_i, f_{i+1}) ist keine gültige NULL-ACK-Sequenz **then**
    continue
  end if
  Berechne $\widetilde{\Delta}_i$ aus Zeitstempeldifferenz.
  if Δ und $\widetilde{\Delta}$ nahe beieinander **then**
    Füge $\widetilde{\Delta}$ zu L hinzu.
  end if
end for
Bestimme $\widehat{\Delta}$ als arithmetisches Mittel über L.
Bestimme die Entfernung durch den Vergleich von $\widehat{\Delta}$ mit den Resultaten einer Referenzmessung.

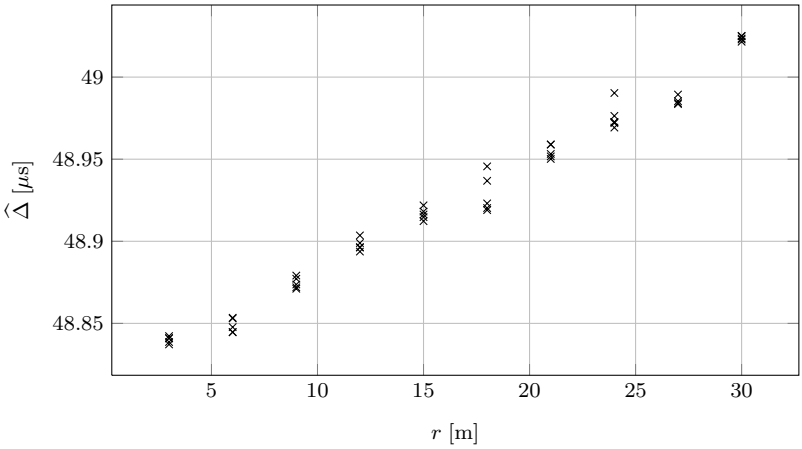


Abbildung 8: Korrelation r vs. $\widehat{\Delta}$

kann durch die Technik der Trilateration die Position eines Mobilgerätes bestimmt werden, wenn ausreichen Abstandsmessungen zu bekannten Referenzpunkten vorgenommen werden.

Folgende Ansätze könnten Ausgangspunkte für weitere Betrachtungen zur laufzeitbasierten WLAN-Lokalisierung sein.

- Welchen Einfluss haben Datenrate, WLAN-Chipsatz und Sichthindernisse auf die Laufzeitmessung?
- Können auch Zeitstempel aus Interrupt-Service-Routinen zur Laufzeitmessung herangezogen werden?
- Kann die quelloffene Firmware für den Treiber `carl9170`[5] so modifiziert werden, dass Zeitstempel für gesendete Pakete erfasst werden?
- Ist über besagte Firmware der Zugriff auf Hardware-Timer mit höherer Auflösung als der des TSF möglich?

[5] `http://wireless.kernel.org/en/users/Drivers/carl9170`

r [m]	n	$\widehat{\Delta} = \mu(\widetilde{\Delta}_i)$ [μs]	$\sigma^2(\widetilde{\Delta}_i)$ [$(\mu\text{s})^2$]	r [m]	n	$\widehat{\Delta} = \mu(\widetilde{\Delta}_i)$ [μs]	$\sigma^2(\widetilde{\Delta}_i)$ [$(\mu\text{s})^2$]
3	8330	48.841	0.134	18	9749	48.923	0.072
	8215	48.841	0.134		9856	48.946	0.052
	8178	48.842	0.133		9839	48.919	0.075
	7653	48.837	0.137		9800	48.937	0.060
	8751	48.839	0.135		9756	48.920	0.074
6	8238	48.845	0.132	21	9794	48.952	0.047
	7505	48.848	0.129		9677	48.959	0.041
	8061	48.845	0.132		9771	48.950	0.049
	7709	48.853	0.125		9784	48.953	0.046
	8192	48.853	0.125		9757	48.959	0.041
9	8256	48.872	0.112	24	9609	48.972	0.032
	8209	48.874	0.110		9592	48.969	0.035
	8339	48.879	0.107		9489	48.973	0.033
	8175	48.877	0.108		9535	48.973	0.035
	8285	48.871	0.112		9479	48.976	0.031
12	8914	48.899	0.091	27	8365	48.990	0.027
	8687	48.896	0.093		9313	48.985	0.027
	8577	48.896	0.093		9429	48.984	0.028
	8348	48.904	0.087		8536	48.984	0.029
	8235	48.894	0.095		9136	48.989	0.028
15	9409	48.922	0.073	30	8602	49.025	0.032
	8845	48.912	0.080		8679	49.022	0.030
	8688	48.915	0.078		8644	49.023	0.029
	8963	48.916	0.077		8651	49.025	0.031
	9193	48.918	0.075		8560	49.023	0.029

$\mu(\widetilde{\Delta}_i)$ ... Mittelwert der $\widetilde{\Delta}_i$
$\sigma^2(\widetilde{\Delta}_i)$ ... Varianz der $\widetilde{\Delta}_i$

Tabelle 3: Messwerte zu Abbildung 8

Literatur

[BP00] Bahl, P. ; Padmanabhan, V.N.: RADAR: an in-building RF-based user location and tracking system. In: *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* Bd. 2, 2000, S. 775–784

[Gas05] Gast, Matthew: *802.11 Wireless Networks: The Definitive Guide, Second Edition (Definitive Guide)*. O'Reilly Media, Inc., 2005 http://www.worldcat.org/isbn/0596100523. – ISBN 0596100523

[GH05] Günther, André ; Hoene, Christian: Measuring Round Trip Times to Determine the Distance Between WLAN Nodes. Version: 2005. http://dx.doi.org/10.1007/11422778_62. In: *NETWORKING 2005*. 2005. – DOI 10.1007/11422778_62, S. 768–779

[HHSW09] Halperin, Daniel ; Hu, Wenjun ; Sheth, Anmol ; Wetherall, David: *802.11 with Multiple Antennas for Dummies*. 2009

[HW08] Hoene, C. ; Willmann, J.: Four-way TOA and software-based trilateration of IEEE 802.11 devices. In: *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, 2008, S. 1–6

[IEE07] IEEE Computer Society (Hrsg.): *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*. IEEE, 3 Park Avenue, New York, NY 10016-5997, USA: IEEE Computer Society, Juni 2007

[mac] *The official Linux Wireless wiki*. http://wireless.kernel.org/

[pca] *libpcap*. http://www.tcpdump.org/

[rad] *Beschreibung des Radiotap-Protokolls*. http://www.radiotap.org/

[Rec08] Rech, Jörg: *Wireless LANs: 802.11-WLAN-Technologie und praktische Umsetzung im Detail*. Heise Zeitschriften Verlag GmbH & Co. KG, Hannover, 2008. – ISBN 9783936931518

[tcp] *tcpdump*. http://www.tcpdump.org/

[Tou] Tourrilhes, Jean: *Wireless Extensions for Linux*. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html

[wir] *Wireshark*. http://www.wireshark.org/

Meine Brücke zum Tor-Netzwerk

Jens Kubieziel
jens@kubieziel.de
http://kubieziel.de/blog/

Die Anonymisierungssoftware Tor hat im Laufe der Zeit einen zweiten Verwandungszweck entwickelt, nämlich die Umgehung von Zensurmaßnahmen. Dazu werden so genannte Brückenserver genutzt. Der Beitrag beschreibt die Funktionsweise der Brückenserver und die Benutzung sowie die Einrichtung solcher Dienste.

1 Das Tor-Projekt

1.1 Einleitung

Das Tor-Projekt war in den Jahren 2006[1] und 2010[2] Thema bei Vorträgen zu den Chemnitzer Linux-Tagen. Dieser Beitrag wird daher nicht detailliert auf die allgemeine Funktionsweise der Software eingehen. Stattdessen wird auf die vorhandenen Beiträge beziehungsweise auf die Erklärung auf der Webseite[3] verwiesen.

Das Tor-Projekt entstand Anfang dieses Jahrhunderts als Ergebnis diverser akademischer Forschungen. Nach einem Anschub durch das Office of Naval Research[4] (ONR) wurde das Projekt zunächst durch die amerikanische Bürgerrechtsorganisation Electronic Frontier Foundation[5] (EFF) finanziert. Seit dem Jahr 2006 steht das Projekt finanziell auf eigenen Beinen und lebt von Spenden diverser Organisationen oder Privatpersonen.[6] Ein Sponsor kann dabei Einfluss auf den Weg der Entwicklung nehmen. Einem der Sponsoren aus der Vergangenheit war es besonders wichtig, dass die Software nicht nur eine Lösung für die Anonymisierung bietet, sondern auch ein Werkzeug zur Umgehung von Zensurmaßnahmen wird.

1.2 Anpassungen für Zensurumgehung

In den ersten Versionen der Tor-Software war es so, dass das Clientprogramm Kontakt zu so genannten Verzeichnisservern aufnimmt. Dort lädt es die Liste aller verfügbaren Tor-Server herunter. Später ermittelt der Client aus dieser Liste drei Server und leitet die Verbindungen über diese Verbindungsstrecke.

[1] http://chemnitzer.linux-tage.de/2006/vortraege/detail.html?idx=353
[2] http://chemnitzer.linux-tage.de/2010/vortraege/detail.html?idx=398
[3] https://www.torproject.org/about/overview.html.en
[4] http://www.onr.navy.mil/
[5] https://www.eff.org/
[6] https://www.torproject.org/about/sponsors.html.en

Wenn nun jemand unterbinden möchte, dass Tor in seinem Netzwerk benutzt hat er verschiedene Möglichkeiten:

1. *Sperrung der Verzeichnisserver*: Die Liste der Verzeichnisserver ist öffentlich und findet sich im Quellcode der Software.[7] Damit kann jeder die IP-Adressen der Server ermitteln und beispielsweise in seinem Paketfilter (Firewall) sperren. Da sich die Teilnehmer im Netzwerk nun nicht mehr mit dem Verzeichnisserver verbinden können, fehlen die grundlegenden Informationen zum Tor-Netzwerk. Ohne diese Information kann der Tor-Client keine Verbindungsstrecken aufbauen.

   Diese Schwachstelle wurde seit längerem von den Entwicklern identifiziert und eine Gegenmaßnahme entwickelt. Aktuelle Versionen der Tor-Server spiegeln die Verzeichnisdaten und verteilen diese an Clienten. Sollte also der Zugriff auf den Verzeichnisserver gesperrt sein, kann ein Client diese Daten von einem anderen Server abrufen. Diese Daten werden von den Verzeichnisservern digital signiert. Damit wird sichergestellt, dass einzelne Server diese Daten nicht manipulieren.

2. *Fingerprinting des Datenverkehrs und Sperrung von Tor-Verkehr*: In früheren Versionen der Software lud der Client Daten von einem entfernten Unterverzeichnis `/tor` herunter. Das und andere Eigenschaften machen es einfach, Tor-Verbindungen zu identifizieren (Fingerprinting) und die betreffenden Verbindungen anschließend zu sperren.

   Eine wichtige Gegenmaßnahme war der Umstieg auf verschlüsselte Verbindungen zu der Verzeichnisservern. Damit ist der Angriff nicht mehr wirksam. Denn ein Angreifer kann die Verbindung nicht mehr abhören. Der englische Forscher Dr. Steven Murdoch verbesserte das Verhalten weiter. Nun sieht die Verbindung zwischen dem Tor-Client und dem Verzeichnisserver so aus, als würde ein Browser eine verschlüsselte Verbindung zu einem Apache Webserver aufnehmen. Dies erschwert das Fingerprinting wesentlich.

3. *Sperrung aller Tor-Server*: Ein Angreifer kann weiterhin auf die Idee kommen, den Zugriff auf alle Tor-Server zu sperren. Dazu muss nur die Liste von einem Verzeichnisserver heruntergeladen werden. Darin finden sich alle Server. Die Liste hat ein standardisiertes Format und die IP-Adressen lassen sich leicht ermitteln. Alle IP-Adressen werden gesperrt. Dann kann kein Client auf das Tor-Netzwerk zugreifen.

   Dieser Angriff ist sehr stark und es lassen sich keine einfachen Gegenmaßnahmen finden. Stattdessen wurde das Design der Software erweitert und so genannte Brückenserver eingeführt.[8]

[7] https://gitweb.torproject.org/tor.git/blob/HEAD:/src/or/config.c#l792
[8] https://gitweb.torproject.org/tor.git/blob/HEAD:/doc/spec/bridges-spec.txt

2 Brückenserver

Die Grundidee hinter den Brückenservern (Bridge server[9]) besteht darin, Tor-Server zur Verfügung zu haben, die nicht in offiziellen Listen zu finden sind. Diese Geheimhaltung erschwert einem Angreifer die Sperrung der betreffenden Server.

Ein Client, der Kenntnis von einer oder mehreren Brückenservern hat, nimmt zu diesem Server oder Servern eine Verbindung auf und baut eine Verbindungsstrecke auf. Diese Verbindung entspricht einer „normalen“ Strecke, wie sie andere Anwender verwenden. Das heißt, für den Benutzer macht es letztlich keinen Unterschied, ob er mit oder ohne Brückenserver unterwegs ist. Nur wenn er diversen Zensurmaßnahmen unterliegt, bringt die Benutzung Vorteile. Für die Benutzer stellt sich nun die Frage, woher sie von Brückenservern erfahren.

2.1 Brückenserver finden

Der Nutzer hat verschiedene Möglichkeiten, die Adresse eines oder mehrerer Brückenserver zu finden.

1. Freunde oder Organisationen betreiben Brückenserver. Sie verteilen die Adressen an Personen, die diese Server nutzen sollen. In dem Fall tauchen die Rechner nie in offiziellen Listen auf, sondern werden nur per Mundpropaganda weiter gegeben.

   Das hat auf der einen Seite den Nachteil, dass die betreffende Person Betreiber solcher Server kennen muss. Andererseits sollten derartige Server sicher vor Sperrungen sein. Denn für einen Angreifer ist nicht klar, dass es sich um Tor-Verbindungen handelt. Stattdessen sehen die Verbindungen nach außen wie „normale“ SSL-Verbindungen aus.

2. Die Betreiber von Brückenservern können ihre Server in öffentliche Listen beim Tor-Projekt eintragen lassen. Dann profitieren mehr Nutzer in der Welt von den Diensten. Von seiten des Tor-Projekts existieren drei Pools, in die neue Brückenserver eingeordnet werden.

   a) Abruf über die Seite https://bridges.torproject.org/. Ein Aufruf dieser Webseite zeigt drei Adressen von Brückenservern an. Der Anwender kann diese benutzen. Die dahinter liegende Anwendung versucht, die IP-Adresse des Aufrufers zu erkennen und gleichen Aufrufern immer wieder die gleichen Brückenserver anzuzeigen. Dadurch soll gewährleistet werden, dass ein Angreifer nicht alle Adressen der Server abgreift und diese dann sperrt. Jedoch haben die Administratoren der Great Firewall of China (GFW) genau das getan und tun es immer noch. Sie grasen die Seite regelmäßig ab und sperren die gefunden Adressen.

[9] https://www.torproject.org/docs/bridges

b) E-Mail an die Adresse bridges@torproject.org. Von einem Googlemail-Konto muss eine E-Mail mit dem Text „get bridges“ an die obige Adresse geschickt werden. Umgehend verschickt der Server eine Antwort mit drei Adressen. Das Tor-Projekt setzt auf Google Mail, da es dort recht schwer ist, automatisiert Konten anzulegen. Damit soll es Angreifern schwerer gemacht werden, schnell an eine große Zahl von Adressen zu gelangen. Jedoch ist ein Ansatz der GFW, viele Nutzer für kleine Aufgaben einzuspannen. So legten viele chinesische Nutzer GMail-Konten an und versuchten auf diesem Weg an Adressen von Brückenservern zu gelangen. In der Folge wurden auch diese in China gesperrt.

c) Gespeichert in einem internen Pool und nur bei Bedarf herausgegeben. Diese letzte Möglichkeit soll zusichern, dass es immer eine Grundmenge an Brückenservern gibt, die für Notfälle benutzbar und noch nicht gesperrt sind.

In der Vergangenheit waren in jedem Pool etwa ein Drittel aller Brückenserver enthalten. Durch die Angriffe aus China hat sich gezeigt, dass die Aufteilung zugunsten des internen Pools verschoben werden muss.

Eine große Anzahl von Anwendern setzen auf das Tor-Browser-Paket.[10] Das ist ein vorkonfiguriertes Paket mit der Tor-Software, Mozilla Firefox und weiteren Anwendungen. Das Archiv wird einfach entpackt und steht sofort für die Benutzung bereit. Das Paket enthält die grafische Oberfläche Vidalia.[11] Diese bietet einen einfachen Dialog für Brückenserver. In den Einstellungen findet sich ein Reiter „Netzwerk“. Dort wird der Menüpunkt „Mein Provider blockiert Verbindungen zum Tor-Netzwerk“ markiert. In dem nun folgenden Dialog kann man die Adresse einer Brücke eintragen oder man klickt auf „Jetzt eine Brücke finden“. Dann wird automatisch eine Adresse eingetragen.

Nutzer, die Tor über die Textdatei konfigurieren möchten, öffnen die Datei `torrc` im Verzeichnis `/etc/tor`[12] und fügen dort die Zeilen mit den Adressen der Server ein:

```
Bridge 69.70.147.42:444
Bridge 87.222.177.219:9001
Bridge 62.35.115.217:443
```

Nach einem Neustart der Software versucht Tor diese Adresse als Brücke in das Netzwerk zu verwenden.

[10] https://www.torproject.org/projects/torbrowser.html.en
[11] https://www.torproject.org/projects/vidalia.html.en
[12] Unter Umständen befindet sich die Datei in einem anderen Verzeichnis.

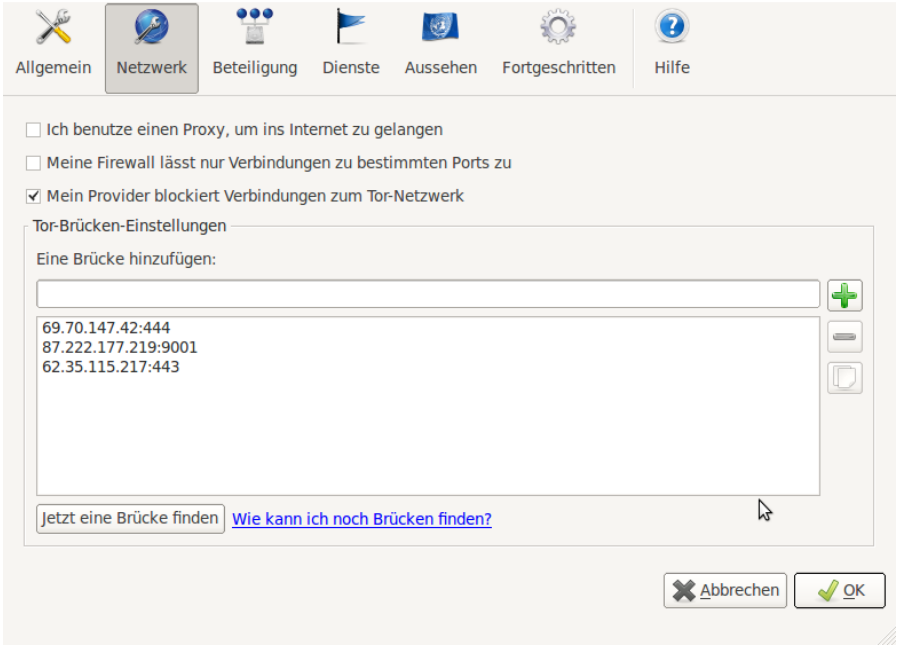


Abbildung 1: Brückenserver in Vidalia einstellen

2.2 Brückenserver betreiben

Wer einen DSL-Anschluss sein Eigen nennt, kann diesen als Brückenserver konfigurieren. Denn bereits Anschlüsse mit 1 Mbit/ sec sind ausreichend. Dies garantiert vielen Nutzern in Zensurländern ein Grundversorgung mit freien Informationen.

Auf der anderen Seite geht der Nutzer in Deutschland keine Risiken ein. Ein Brückenserver ist immer der erste in einer Verbindungsstrecke. Somit kann es nicht, wie im Fall von Exitservern passieren, dass Missbrauch betrieben wird und der Betreiber Kontakt zu Strafverfolgern erhält. Der Betrieb eines Anonymisierungsdienstes ist in Deutschland nach wie vor legal und wurde lange Zeit vom Wirtschaftsministerium gefördert (Siehe JAP der TU Dresden).

Nutzer von Vidalia wählen in den Einstellungen den Reiter „Beteiligung“. Dort findet sich ein Menüpunkt „Nutzern, die einer Internetzensur unterliegen, helfen das Tor-Netzwerk zu nutzen“. Im folgenden Menü sollte dem Server ein Spitzname[13] gegeben werden. Der Screenshot aus Abbildung 2 verwendet den Namen „CLT2011“. Die Kontaktinformation ist eine freiwillige Angabe. Es ist jedoch sinnvoll, einen gültigen

[13] Im englischen Original heißt der Konfigurationspunkt Nickname.

Wert einzutragen. Denn im Fall von Fehler in der Konfiguration, nehmen die Tor-Entwickler Kontakt mit dem Betreiber auf und weisen auf die Fehler hin. Bei fehlender Kontaktmöglichkeit wird der Server einfach geperrt und steht für Clienten nicht mehr zur Verfügung. Als Verteiler-Port empfiehlt es sich 443 einzustellen. Dies ist der Port über den normalerweise HTTPS-Verbindungen abgewickelt werden und der in Firewalls nicht gesperrt ist. Das erhöht die Chance, dass andere Nutzer auf den Server zugreifen können. Standardmäßig ist hier 9001 eingestellt.

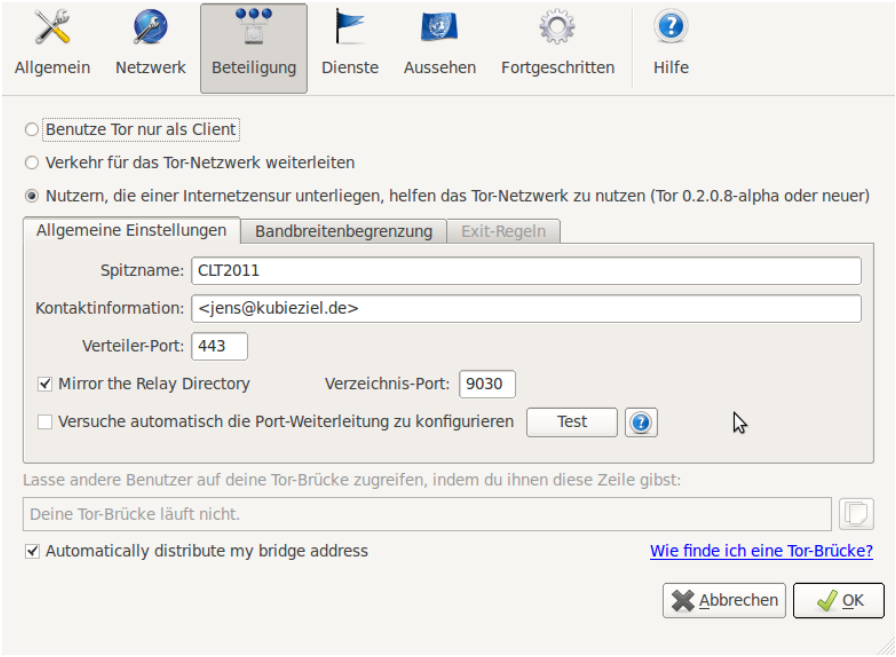


Abbildung 2: Brückenserver in Vidalia einrichten

Die Begrenzung der Bandbreite (zweiter Reiter) kann sinnvoll sein, um zu hohen Datendurchfluss zu vermeiden. Ohne weitere Einstellungen versucht Tor die komplett verfügbare Bandbreite zu beanspruchen. Das mag bei einer echten Flatrate keine Probleme hervorrufen. Wenn der Anschluss nur über ein begrenztes Datenvolumen pro Monat verfügt, könnten sich Probleme ergeben. Durch die Einschränkung der Bandbreite kann der Benutzer genau steuern, wieviel Daten pro Monat verbraucht werden und behält so die Kontrolle.

Für Freunde der textuellen Konfiguration sind entsprechende Einstellungen in der Datei `torrc` zu machen:

```
Nickname CLT2011
```

```
ContactInfo <jens@kubieziel.de>
ORPort 443
BridgeRelay 1
```

Diese Werte entsprechen den Einstellungen aus der Abbildung 2. Die letzte Zeile besagt, dass die Informationen über den Brückenserver an das Tor-Projekt verschickt werden sollen. Dort wird der Server dann in einen der Pools eingeordnet.

Nach einem Neustart versucht der Server mit dem Tor-Projekt Kontakt aufzunehmen und sich als Brückenserver zu registrieren. Hierbei sollte das Logfile im Auge behalten werden. Insbesondere sind folgende Einträge wichtig:

```
[notice] Your Tor server's identity key fingerprint is
         CLT2011 2D466A8E02AB9AB4CB8BE12E57C4C148D967E900'
[notice] Now checking whether ORPort 212.55.81.156:443 is
         reachable... (this may take up to 20 minutes --
         look for log messages indicating success)
[notice] Self-testing indicates your ORPort is reachable from
         the outside. Excellent. Publishing server descriptor.
```

Der erste Eintrag gibt eine Information zu dem Schlüssel des Servers. Danach versucht der Server festzustellen, ob er von der Außenwelt erreichbar ist und gibt in der dritten Zeile eine Erfolgsmeldung. Sollten diese Meldung nach längerer Zeit fehlen bzw. ein Negativeintrag kommen, so muss eventuell eine Weiterleitung am Router eingestellt oder andere Maßnahmen getroffen werden.

Der Brückenserver kann nun von anderen Anwendern gefunden und benutzt werden. Der Autor betreibt selbst mehrere solcher Server. An einem DSL-Anschluss fällt weder der entstehende Datenverkehr noch die Rechenleistung ins Gewicht. Laut den Statistiken gibt es um die zehn Benutzer, die auf den Dienst zugreifen. Rechner mit einem besseren Internetanschluss werden natürlich stärker frequentiert. Es ist mit vielen Benutzern und entsprechender Auslastung des Internetanschlusses zu rechnen.

Im Allgemeinen ist es sehr zu wünschen, dass viele Benutzer ihren Anschluss als Brückenserver zur Verfügung stellen. Denn das ermöglicht Nutzern rund um die Welt auf freie, nicht zensierte Informationen zuzugreifen. Diese Möglichkeiten sind essenziell, um demokratische Prozesse innerhalb der Länder in Gang zu bringen. Tor stellt ein gutes Mittel hierfür zur Verfügung.

Textbasierte Benutzeroberflächen mit STFL

Andreas Krennmair ak@synflood.at
http://synflood.at/

Dieser Beitrag beschreibt die Geschichte von textbasierten Benutzeroberflächen und die historische Entwicklung von heutzutage verbreiteten Bibliotheken. Weiters liefert es einen Einblick in die wichtigsten STFL-Konzepte, beschreibt ein paar "Best Practices", und vergleicht die STFL mit anderen, ähnlich gelagerten Bibliotheken und Frameworks.

1. Geschichte

Historisch gesehen ist die Verwendung von Terminals eng mit Unix verknüpft: während anfangs noch Fernschreiber ("TTYs") als Ein-/Ausgabegeräte Verwendung fanden, so etablierten sich Ende der 1970er Jahre Terminals wie das VT100 und Nachfolgermodelle, und neue Paradigmen von Benutzerschnittstellen etablierten sich: während die bewährte Kommandozeile beibehalten wurde, konnte diese um eine zweite Dimension erweitert werden. Text lief nicht mehr nur in Form einer Papierrolle vorbei, sondern wurde gezielt an verschiedenen Stellen des Bildschirms platziert. Diese neuartige Erfahrung macht sich beispielsweise im einleitenden Absatz der ersten Dokumentation des vi-Editors[2] bemerkbar:

> When using vi the screen of your terminal acts as a window into the file which you are editing. Changes which you make to the file are reflected in what you see.

Da Unix auf einer relativ großen Anzahl von verschiedenen Hardwareplattformen zum Einsatz kam, und so mehr als nur ein Typus von Terminal unterstützt werden musste, entstand schon bald eine gewisse Softwareinfrastruktur: zuerst termcap, eine Bibliothek, die es erlaubte, portable Textmodusapplikationen zu entwickeln, die also auf verschiedenen Terminals funktionierten (der Programmierer kann für das Terminal sogenannte "Capabilities" abfragen[7]), später curses, welche – basierend auf termcap – eine weitere Abstraktionsebene einführte, um die Programmentwicklung noch intuitiver zu gestalten, und zwar in Hinsicht auf optimiertes Bildschirmzeichnen (bei langsamen Terminals von Relevanz), Hardwareunabhängigkeit, die Einteilung des Bildschirms in Fenster, und Textattributen[6]. Curses ist heute meist in der Variante *ncurses* auf heutigen Unix-artigen Systemen zu finden. Im Rahmen von *ncurses* wurde auch das Widgetset *Cdk* entwickelt, das sich jedoch nie weiter durchsetzte.

Im Heim-PC-Bereich, insbesondere unter DOS, hatte sich im Gegensatz dazu kein wirklicher Standard durchgesetzt. Zwar war es möglich, einen Treiber zu laden, der die Verwendung von ANSI-Escape-Sequenzen ermöglichte, eine vereinheitlichte Bibliothek mit einer gewissen Abstraktion konnte sich jedoch nie durchsetzen, und so hing die eingesetzte Terminalbibliothek meist von der eingesetzten Programmiersprache bzw. vom verwendeten Compiler

ab. Die Firma Borland veröffentlichte mit *Turbo Vision* eine Widgetbibliothek auf einem hohen Abstraktionsniveau, die sich jedoch aufgrund des Einarbeitungsaufwands langfristig nicht durchsetzen konnte und mittlerweile als Open Source frei verfügbar ist.[9]

Insgesamt kann man als Programmierer nur ein eher geringes Abstraktionsniveau bei der Entwicklung von Textmodus-Anwendungen erwarten. So ist es dem Programmierer überantwortet, sich selbst um Layout Management und Resize Management (d.h. Veränderung der Terminalgröße) zu kümmern. Dies ist oftmals mit einem nicht geringen Aufwand verbunden.

2. Grundlagen

Die Structured Terminal Forms Language/Library[11] – kurz STFL – ist eine seit 2006 entwickelte Bibliothek, die es dem Programmierer ermöglicht, ohne großen Aufwand ansprechende ncurses-basierte Applikationen zu designen und zu entwickeln. Dies geschieht einerseits dadurch, dass sich STFL selbsttätig um Resize und Layout Management kümmert, andererseits dadurch, dass dem Programmierer eine textbasierte Layoutbeschreibungssprache zur Verfügung gestellt wird.

2.1. Layout

Ein Layout in STFL in eine Hierarchie von Widgets. Die Eigenschaften eines jeden Widgets werden durch sog. Variablen bestimmt, welche optional benannt werden können, um sie Laufzeit verändern zu können.

Die grundlegenden Widgets zur räumlichen Aufteilung heißen *vbox* und *hbox*. Innerhalb einer *vbox* bzw. *hbox* können sich weitere Widgets befinden. Die Widgets innerhalb einer *vbox* werden vertikal untereinander angeordnet, während die Widgets innerhalb einer *hbox* horizontal nebeneinander angeordnet werden. Innerhalb einer *vbox* oder *hbox* können wiederum andere *vbox*en oder *hbox*en eingefügt werden. Auch kann für jedes Widget definiert werden, wie es selbst angeordnet werden soll (`tie`), wie die Widgets innerhalb angeordnet werden soll (`.tie`), wie es sich im Fall einer Größenänderung verhalten soll (`.expand`), welche Größe es standardmäßig haben soll (`.height`, `.width`), und ob es überhaupt angezeigt werden soll (`.display`). All diese Eigenschaften eines Widgets sind auch zur Laufzeit veränderbar.

Dies gilt auch für die Layouthierarchie selbst: jedes Widget kann zur Laufzeit durch ein anderes Widget ersetzt werden. Eine Voraussetzung dafür ist die Benennung der Widgets analog zu den dazugehörigen Variablen.

Die STFL bietet zwei verschiedene Varianten, um Layouts textuell darzustellen, und zwar eine klammerfreie Syntax, bei der die Hierarchie durch Zeilenumbrüche und Einrückungen definiert wird, sowie eine klammerbehaftete Syntax, bei der die Hierarchie durch geschwungene Klammern festgelegt wird. Erstere Variante ist leichter lesbar, während zweitere Variante für die programmatische Generierung und Verwendung gedacht ist.

2.1.1. Beispiel: klammerfreie Syntax

```
vbox
  textview[text]
    .expand:vh
  label
    text[info]:""
    .expand:0
```

2.1.2. Beispiel: klammerbehaftete Syntax

```
{vbox {textview[text] .expand:vh} {label text[info]:"" .
  expand:0}}
```

2.2. Widgets

Neben den bereits oben erwähnten Widgets *vbox* und *hbox*, die primär zur Layoutgestaltung dienen, stehen noch weitere Widgets zur eigentlichen Darstellung von Inhalten zur Verfügung.

2.2.1. label

Das Widget `label` dient dazu, einen simplen, einzeiligen Text darzustellen. Optional kann dieser auch noch mit farblicher Gestaltung versehen werden.

2.2.2. input

Mit dem Widget `input` können einfache, einzeilige Eingaben eingelesen werden. Die farbliche Darstellung sowie verschiedene Tastenkombinationen können frei definiert werden, während der eingegebene Text, die Cursorposition innerhalb des Textfelds sowie innerhalb des eingebenen Texts über Variablen ausgelesen und gesetzt werden können.

2.2.3. table

Das `table`-Widget stellt das wohl mächtigste Widget innerhalb von STFL dar, da es eine Layoutgestaltung in tabellarischer Form ermöglicht.

Um Widgets in einer Tabelle anzuordnen, so sind diese von links nach rechts anzugeben. Das Ende der jeweiligen Tabellenzeile wird durch das spezielle Widgets `tablebr` markiert. Für

jedes Widget kann angegeben werden, wie groß es sein soll (`.height`, `.width`), wie es sich bei einer Fenstergrößenänderungen verhalten soll (`.expand`), wieviele Zeilen bzw. Spalten es belegen soll (`.rowspan`, `.colspan`), ob und an welchen Seiten der Zelle eine Trennlinie oder ein Trennabstand angezeigt werden soll (`.border`, `.spacer`) und an welchen Zellenseite das Widget gebunden sein soll.

2.2.4. list

Mit dem Widget `list` ist es möglich, eine List zu definieren. Eine Liste besteht dabei aus einer Menge von vertikal angeordneten `listitem`s, von denen jedes Listenelement einen Text enthält und einzeilig dargestellt wird. Eines dieser Elemente ist dabei markiert und besitzt den Fokus. Innerhalb dieser Liste kann gescrollt werden.

2.2.5. textview

Das Widget `textview` verhält sich ähnlich wie das `list`-Widget. Es besteht ebenso aus einer Menge von vertikal angeordneten `listitem`s, jedoch realisiert es eine mehrzeilige Textansicht, in der kein bestimmtes Element den Fokus hat. Zusätzlich besteht die Möglichkeit, Text innerhalb einer Zeile mit verschiedenen Farben und Attributen auszuzeichnen (“rich text”).

2.2.6. Tasteneingaben

Tasteneingaben, wie sie von STFL an das Programm weitergegeben werden, und auch an verschiedenen Stellen innerhalb der Layoutbeschreibung Verwendung finden können, unterliegen einer textuellen Repräsentation. Druckbare Zeichen werden als sich selbst zurückgegeben. Drückt der Benutzer die Taste A, so wird ein `a` an das Programm weitergegebren, bei Shift ⇧ + A ein `A`. Steuerzeichen werden in der Notation `^A` zurückgegeben. Drückt der Benutzer Ctrl + B auf seiner Tastatur, so gibt die STFL `^B` zurück. Für bestimmte Tasten existieren jedoch Ausnahmen:

- `ENTER`: wenn die Return- oder Enter-Taste gedrückt wurde.
- `SPACE`: bei Eingabe eines Leerzeichens.
- `TAB`: bei Drücken der Tab-Taste.
- `ESC`: bei Drücken der Esc-Taste.
- `F0`..`F63`: bei Drücken einer Funktionstaste.

2.3. Programmierschnittstelle

Die STFL kann von mehreren Programmiersprachen aus programmatisch genutzt werden. Die STFL selbst ist in C entwickelt, von daher stellt die C-API die primäre Schnittstelle dar. Weiters existieren *Bindings* für die Programmiersprachen Perl, Python, Ruby und SPL.[10] Mit der Ausnahme von SPL basieren alle Anbindungen an weitere Programmiersprachen auf Swig[8], einem Tool zur Anbindung von C- und C++-Code an andere High-Level-Programmiersprachen.

Die STFL verwendet intern zur Stringverwaltung den Datentyp `wchar_t`. Da auch noch auf der Unicode-Variante von ncurses aufgebaut wird, sind STFL-Anwendungen grundsätzlich Unicode-fähig. Zur Vereinfachung bietet die STFL auch noch eine Schnittstelle zur Konvertierung zwischen `char`-Strings und `wchar_t`-Strings.

Nachfolgend wird die grundlegende API von STFL beschrieben. Die Namen der API-Funktionen unterscheiden sich von Programmiersprache zu Programmiersprache, sodass hier die unter C zu findenden Namen aufgelistet sind.

2.3.1. stfl_create(text)

Parsed den angegebenen Text als Layoutbeschreibung und gibt ein Form-Handle zurück.

2.3.2. stfl_free(form)

Gibt das übergebene Form-Handle und alle damit verbundenen Ressourcen frei.

2.3.3. stfl_run(form, timeout)

Zeichnet das angegebene Form auf den Bildschirm, wartet den angegebenen Timeout und gibt den nächsten Event zurück.

Ist das Timeout 0, so wird bis zur nächsten Benutzereingabe gewartet, und diese dann zurückgegeben. Bei einem Timeout > 0 wird dieser Wert als Millisekunden interpretiert und solange gewartet, bis eine Benutzereingabe erfolgt oder aber ein Timeout geschieht. Ein Timeout wird als “TIMEOUT” zurückgegeben. Es gibt auch noch andere, “spezielle” Timeout-Werte: -1 zeichnet das Form neu, liest aber keine Benutzereingabe und gibt folglich auch keine zurück. -2 zeichnet das Form nicht neu, liest aber die nächste Benutzereingabe. -3 rendert das Form neu und bestimmt intern die Größe aller gerenderten Elemente, zeichnet das Form jedoch nicht und liest auch keine Benutzereingabe. Dies hat den Sinn, dass Teile des Renderprozesses in Applikation von bestimmten Größen von Widgets abhängen können, deren Größe aber vorher noch berechnet werden muss, beispielsweise die Breite einer `textview` für automatischen Zeilenumbruch.

Diese Funktion aktiviert und initialisiert bei Bedarf auch die ncurses-Bibliothek.

2.3.4. stfl_reset()

Diese Funktion dient dazu, wieder den Terminalmodus vor Initialisierung von ncurses herzustellen.

2.3.5. stfl_get(form, name)

Gibt den aktuellen Wert einer benannten Variable zurück.

2.3.6. stfl_set(form, name, value)

Setzt den Wert einer benannten Variable auf einen neuen Wert.

2.3.7. stfl_get_focus(form)

Gibt den Namen des Elements zurück, das aktuell den Fokus besitzt.

2.3.8. stfl_set_focus(form, name)

Setzt den Fokus auf das Element, das über den angegebenen Namen identifiziert wird.

2.3.9. stfl_quote(text)

Quotet den angegebenen Text, sodass er sicher in Layoutbeschreibungen verwendet werden kann.

2.3.10. stfl_modify(form, name, mode, text)

Diese API-Funktion ist ein möchtiges Werkzeug zur Modifikation der Layout-Hierarchie. Der übergebene Name gibt das zu bearbeitende Element innerhalb dieser Hierarchie an, der Modus die durchzuführende Operation, und der übergebene Text der möglicherweise notwendige neue Layout-Code.

Folgende Modi werden unterstützt:

- **delete**: löscht das angegebene Element und alle darunter befindlichen Elemente.
- **replace**: ersetzt das angegebene Element mit dem neuen, im Text angegebenen Element.

- **replace_inner**: ersetzt die unter dem angegebenen Element befindlichen Elemente mit den Elementen unter dem im Text angegebenen Element.
- **insert**: fügt ein neues Element am Anfang der unter dem angegebenen Element befindlichen Element ein.
- **insert_inner**: fügt die Elemente unter dem im Text angegebenen Element am Anfang der Liste der Elemente unter dem angegebenen Widget ein.
- **before**: fügt das neue Element vor dem angegebenen Element ein.
- **before_inner**: fügt die Elemente unter dem im Text angegebenen neuen Element vor dem angegebenen Element ein.
- **after**: fügt das neue Element nach dem angegebenen Element ein.
- **after_inner**: fügt die Elemente unter dem im Text angegebenen neuen Element nach dem angegebenen Element ein.

3. Verwendung

3.1. Pakete

Zum praktischen Einsatz von STFL empfiehlt es sich, die fertig gebauten Bibliotheken bzw. Module einer Distribution zu verwenden. Der Autor empfiehlt explizit die STFL-Pakete, die Debian[1] bzw. Ubuntu[2] beiliegen. Es existieren jedoch auch Pakete für andere Distributionen wie etwa Arch Linux, Gentoo, Slackware, OpenSUSE, Fedora, Mandriva sowie für FreeBSD.

3.2. iconv()-Wrapper

Wie bereits weiter oben erwähnt, bietet die STFL Utilityfunktionen, um in C `char`-Strings in `wchar_t`-Strings (und umgekehrt) umzuwandeln. Diese Utilityfunktionen basieren auf iconv(3) und sind als Pool realisiert. Ein Pool bietet die Möglichkeit, Ressourcen zu allokieren (in diesem Fall konvertierte C-Strings). Dabei wird aufgezeichnet, welche Ressourcen allokiert worden. Werden diese Ressourcen nicht mehr benötigt, so können sie allesamt aus dem Pool entfernt und freigegeben werden. Dies dient dazu, dem Programmierer die Aufgabe der manuellen Speicherverwaltung abzunehmen. Folgendes Beispiel zeigt die Verwendung dieser Utilityfunktionen:

[1] http://packages.qa.debian.org/s/stfl.html
[2] https://launchpad.net/ubuntu/+source/stfl

```
1 stfl_ipool * ipool = stfl_ipool_create("utf-8");
2
3 wchar_t * wstr = stfl_ipool_to_wc(ipool, "Some␣UTF-8␣text!")
    ;
4 char * str = stfl_ipool_from_wc(ipool, wstr);
5
6 stfl_ipool_flush(ipool); // str and wstr are now invalid!
```

3.3. Event Loop

Ein “Design Pattern”, das sich durch die meisten STFL-Programme zieht, ist eine Event-Loop als zentrale Stelle, an der der aktuelle Layoutzustand auf den Bildschirm gezeichnet wird, die Benutzereingabe gelesen wird, und dann je nach Eingabe darauf reagiert und evtl. die Inhalte der Layouthierarchie verändert werden. Ein großer Vorteil der STFL besteht jedoch darin, dass eine derartige Event-Loop nicht zum Einsatz kommen muss. So bietet es sich beispielsweise auch an, die STFL lediglich zum Zeichnen des Layouts ohne die Möglichkeit einer Interaktion des Benutzers zu verwenden.

In Newsbeuter[4], dem wohl ersten großen Open-Source-Projekt, bei dem STFL zum Einsatz kam, wurde als Abstraktion ein Dialog-Management eingeführt, bei dem Dialoge (u.a. ein STFL-Form-Handle, sowie Routinen zur Behandlung von Tastatureingaben) als ein Stack betrachtet wurde, wobei der jeweils oberste Dialog aktiv war. Jeder Dialog kann sich selbst vom Stack entfernen (wodurch der aufrufende Dialog wieder aktiv wird), oder selbst einen anderen Dialog erzeugen und auf den Stack legen kann (wodurch der neue Dialog aktiv wird). Dies erlaubt eine logische Trennung von Funktionalität je nach Dialog, ohne dass die Applikation eine große Menge an globalem Status halten und konsistent halten muss. Desweiteren wird die Event Loop selbst von der Reaktion auf Benutzereingaben und dem Update der Dialoginhalte logisch getrennt.[3]

4. Ähnliche Softwarebibliotheken

Im folgenden werden noch Softwarebibliotheken mit ähnlicher Funktionalität sowie deren Unterschied zu STFL beschrieben.

4.1. Newt

Newt[5] ist eine C-Library, die es ermöglicht, dialogorientierte Text User Interfaces mit Widgets zu entwickeln. Newt orientiert sich dabei an “klassischen” GUI-Paradigmen wie Fenstern und Buttons. Die Dialoge selbst werden durch Funktionsaufrufe zusammengesetzt. Das Fenstermanagement beschränkt sich auf einen Stack. Der Programmierer hat selbst Kontrolle über die Event-Loop.

4.2. Turbo Vision

Turbo Vision[9] wurde ursprünglich von Borland in den 1990er Jahren entwickelt und veröffentlicht, um die Entwicklung von intuitiven Text User Interfaces zu ermöglichen. Seit 1997 steht die C++-Version davon unter der GPL. Das bisher letzte Release erfolgte 2004.

Bei Turbo Vision wird das User Interface durch Methodenaufrufe zusammengesetzt. Die gesamte Ausführung wird über eine Turbo-Vision-eigene Event-Loop gesteuert. Auch Turbo Vision orientiert sich an "klassischen" GUI-Paradigmen wie Menüs, Buttons, Fenstern.

4.3. Curses::Forms

Das Perl-Modul Curses::Forms[1] stellt einen objektorientierten Framework für ncurses-basierten Anwendungen zur Verfügung. Das Layout sowie zusätzliche Information wird in Form eines Graphen, bestehend aus Hashes und Arrays (bzw. Referenzen darauf) formuliert. Die Ausführung wird über eine eigene Event-Loop gesteuert, die Steuerung funktioniert eventbasiert über Callbacks.

Literatur

[1] Arthur Corliss. Curses::Forms . *http://search.cpan.org/~corliss/CursesForms-1.997/Forms.pm*.

[2] William Joy and Mark Horton. An Introduction to Display Editing with Vi . *http://docs.freebsd.org/44doc/usd/12.vi/paper.html*.

[3] Andreas Krennmair. Erfahrungen in der Entwicklung großer STFL-Applikationen . *http://linuxwochenende2008.luga.at/docs/25_stfllwe08.pdf*.

[4] Andreas Krennmair. newsbeuter . *http://www.newsbeuter.org/*.

[5] Newt. *https://fedorahosted.org/newt/*.

[6] John Strang. *Programming with curses* . 1986.

[7] John Strang, Linda Mui, and Tim O'Reilly. *termcap & terminfo* . 1992.

[8] SWIG. *http://www.swig.org/*.

[9] Turbo Vision. *http://tvision.sourceforge.net/*.

[10] Clifford Wolf. SPL - The SPL Programming Language . *http://www.clifford.at/spl/*.

[11] Clifford Wolf. STFL - Structured Terminal Forms Language/Library . *http://www.clifford.at/stfl/*.

A. Beispiele

A.1. Ein einfacher Pager a la less(1)

```
#!/usr/bin/env ruby

require 'stfl'

if ARGV.size == 0
        print "usage: #{$0} <file> ...\n"
        Kernel.exit(1)
end

layout = <<EOT
vbox
  textview[text]
    .expand:vh
    bind_page_up:"** b"
    bind_page_down:"** SPACE"
  label
    text[filename]:""
    .expand:0
    style_normal:fg=black,bg=white,attr=bold
EOT

$stfl = Stfl.create(layout)

def load_file(file)
        text = "{list "
        text += IO.readlines(file).map { |l|
                "{listitem text:" + Stfl::quote(l) + "}" }.
                  join("")
        text += "}"
        $stfl.modify("text", "replace_inner", text)
        $stfl.set("filename", file)
end

idx = 0

load_file(ARGV[idx])

loop do
        event = $stfl.run(0)

        if event == "q"
```

```
41                  break
42          elsif event == "^N"
43                  idx += 1
44                  idx = 0 if idx >= ARGV.size
45                  load_file(ARGV[idx])
46          elsif event == "^P"
47                  idx -= 1
48                  idx = ARGV.size-1 if idx < 0
49                  load_file(ARGV[idx])
50          end
51  end
```

A.2. Ein Viewer für STFL-Layoutdateien

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use stfl;
6
7  if (scalar(@ARGV) < 1) {
8          print STDERR "usage: $0 <stfl-file>\n";
9          exit 1;
10 }
11
12 my $f = stfl::create("<$ARGV[0]>");
13
14 while (1) {
15         my $e = $f->run(0);
16         last if $e && $e eq "ESC";
17 }
```

A.3. Ein einfacher Rechner

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use stfl;
6
7  my $layout = <<'EOT';
8  vbox
```

```
  hbox
    @style_normal:fg=white,bg=blue,attr=bold
    .expand:0
    label
      text:"Next␣Operation:␣"
      .expand:0
    input[inputfield]
      text[nextop]:""
      .expand:h
      style_focus:fg=black,bg=green
  list[results]
    .expand:vh
    style_focus:fg=black,bg=white,attr=bold
    style_selected:fg=black,bg=white
EOT

my $f = stfl::create($layout);

my $old = 0;

while (1) {
        my $e = $f->run(0);
        next unless $e;

        if ($e eq "ENTER") {
                next if $f->get_focus ne "inputfield";

                my $input = $f->get("nextop");

                $input =~ /([*\/+-])? *([+-]?[0-9]+)/;

                if (defined($1) && defined($2)) {
                        my $new = eval("$old␣$1␣$2;");

                        my $line = "$old␣$1␣$2␣=␣$new";
                        $f->modify("results", "insert",
                                "{listitem␣text:" . stfl::
                                  quote($line) . "}");

                        $old = $new;
                } elsif (defined($2)) {
                        $old = $2;

                        my $line = "$old␣=␣$old";
```

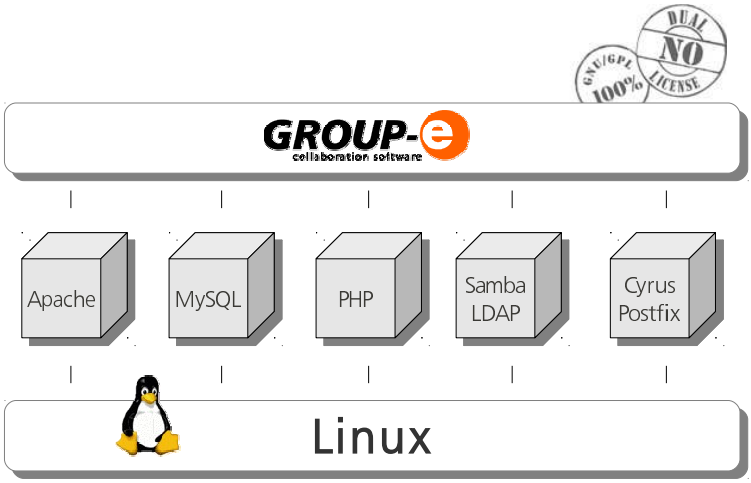
```
52                          $f->modify("results", "replace_inner
                              ",
53                                  "{list␣{listitem␣text:" .
                                      stfl::quote($line) . "}}")
                                      ;
54
55                  }
56                  $f->set("nextop", "");
57          } elsif ($e eq "ESC") {
58                  last;
59          } elsif ($e eq "^L") {
60                  stfl::reset;
61          }
62  }
```

Tiroler Taschenmesser meets Asterisk

Die Collaboration-Software GROUP-E mit CTI im Unternehmenseinsatz

Die IT-Agentur Endo7 aus dem schönen Bozen in Südtirol entwickelt wohl eine der interessantesten Groupware- und Collaboration-Lösungen im OpenSource-Umfeld. Auch wenn in Deutschland noch weithin unbekannt, setzen viele Behörden und Landesverbände in Südtirol GROUP-E seit Jahren als zentrales tägliches Werkzeug für E-Mails, gemeinsame Kalender, Aufgaben und Projektverwaltung ein. Abseits des Web-2.0-Hypes stellt die vollständig unter GPL lizenzierte Software eine skalierbare und sichere Businesslösung dar.

Vielleicht werden Sie sich jetzt fragen: „Noch eine von diesen zahlreichen OpenSource-Groupware-Lösungen?“ Nicht ganz. GROUP-E ist anders. Zum einen ist GROUP-E wirklich 100% OpenSource (GPL), ohne doppelten Boden und versteckte Lizenzkosten, und zum anderen ist es seit Jahren vielfach bewährt im professionellen Einsatz. Allein in Südtirol verwenden mehr als 125 Institutionen und regionale Unternehmen GROUP-E als tägliches Werkzeug, insbesondere



80 Prozent der Südtiroler Gemeinden. Auch in Deutschland etabliert sich die Collaboration-Lösung zusehends in Agenturen, Behörden und Unternehmen.

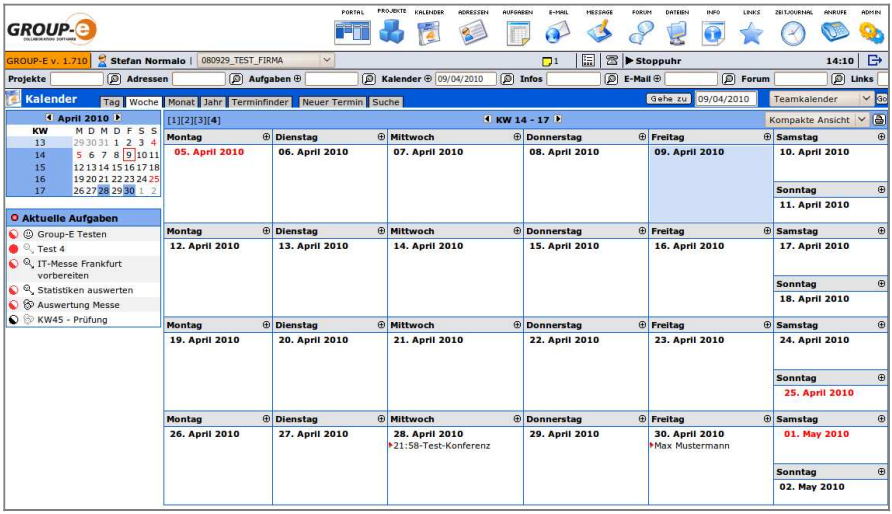
Die Südtiroler haben bei der Entwicklung von GROUP-E das Rad nicht neu erfunden, sondern bedienen sich des vielfach bewährten und robusten LAMP (Linux, Apache, MySQL und PHP)-Unterbaus in Verbindung mit zahlreichen Integrations- bzw. Erweiterungsmöglichkeiten wie z.B. Cyrus, Postfix, Samba, LDAP und SyncML. Somit lässt sich beispielsweise ein sogenanntes Single-Sign-On realisieren, welches für die zentrale Benutzer- und Rechteverwaltung in Unternehmen eine wichtige Rolle spielt.

Die *konsequente Nutzung* und Unterstützung **offener Standards** hat bei GROUP-E höchste Priorität. Das Herzstück von GROUP-E bildet die **integrierte Projektverwaltung**, welche Termine, Aufgaben, Adressen, Dateien, E-Mails und Zeiteinträge intelligent miteinander verknüpft. Die Applikationsoberfläche von GROUP-E ist 100 % webbasiert und bietet so eine effektive Lösung für ein standortunabhängiges Arbeiten in virtuellen Teams.

1. Leistungsübersicht:

- Organisation und Kommunikation im Team
- Projektmanagement mit integrierter Zeiterfassung
- Plattform für Unternehmenswissen
- Systemintegration & Single-Sign-On
- Synchronisation mobiler Endgeräte
- Webbasierter Zugriff auf Unternehmensdaten

1.1 Termin- und Aufgabenverwaltung



Das Kalendermodul bietet eine Reihe interessanter Features, wodurch GROUP-E sich deutlich von bestehenden Groupware Lösungen absetzt. Zu den „klassischen Funktionen" zählen z.B. Tag-, Wochen-, Monats- und Jahresanzeige, Terminfinder- und detaillierte Suchfunktionen. Zudem lassen sich verschiedene Arten von Terminen frei definieren, um beispielsweise interne Meetings von externen zu trennen. Teilnehmereinladungen können sowohl an einzelne Benutzer als auch an Gruppen- und Projektmitglieder sowie Externe versendet werden. Hierbei greift GROUP-E auf die interne Benutzerdatenbank sowie das integrierte CRM-Modul zurück. Eine intuitive Autovervollständigung zum einfachen Auffinden der gewünschten Teilnehmer bietet guten Komfort.

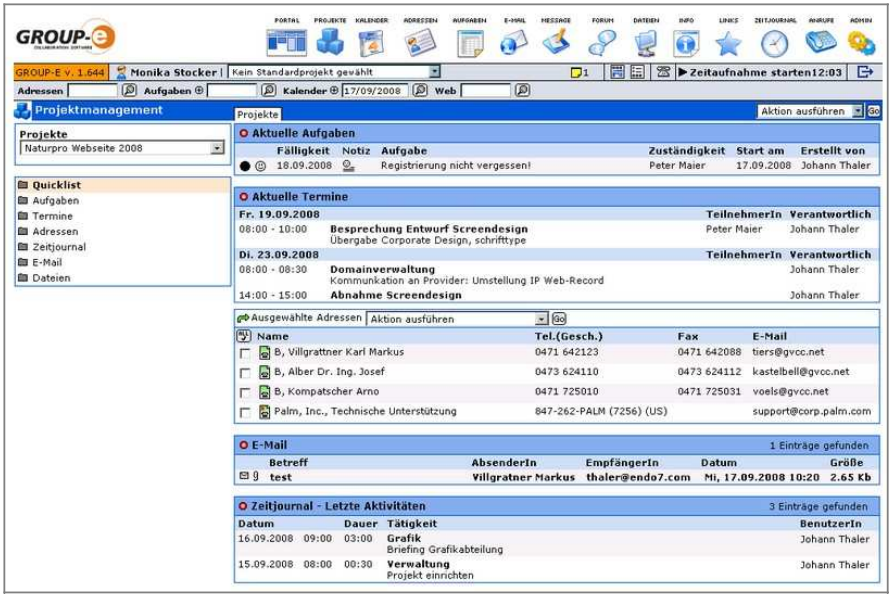
1.2 Integriertes Projektmanagement

Durch die intelligente Verknüpfung mit der Projektverwaltung lassen sich Termine auch für bestimmte Projekte definieren. Ein sehr sinnvolles Feature ist das **Setzen von Sichtbarkeiten**, um Projektmitglieder über anfallende Termine zu informieren, ohne sie dafür explizit einladen zu müssen. Somit hat ein Projektteam immer alle wichtigen Termine im Focus.

GROUP-E ist voll kompatibel mit anderen Kalenderprogrammen. Termineinladungen werden unter Verwendung des iCalendar-Standards nach **RFC 2445** als ICS verschickt, was eine hervorragende **Interoperabilität** z. B. zwischen **Outlook**, **Thunderbird Lightning** und **Entourage** bietet.

Termine können komfortabel mit einem Klick direkt in die Zeiterfassung übertragen werden, was Zeit und doppelte Eintragungen spart.

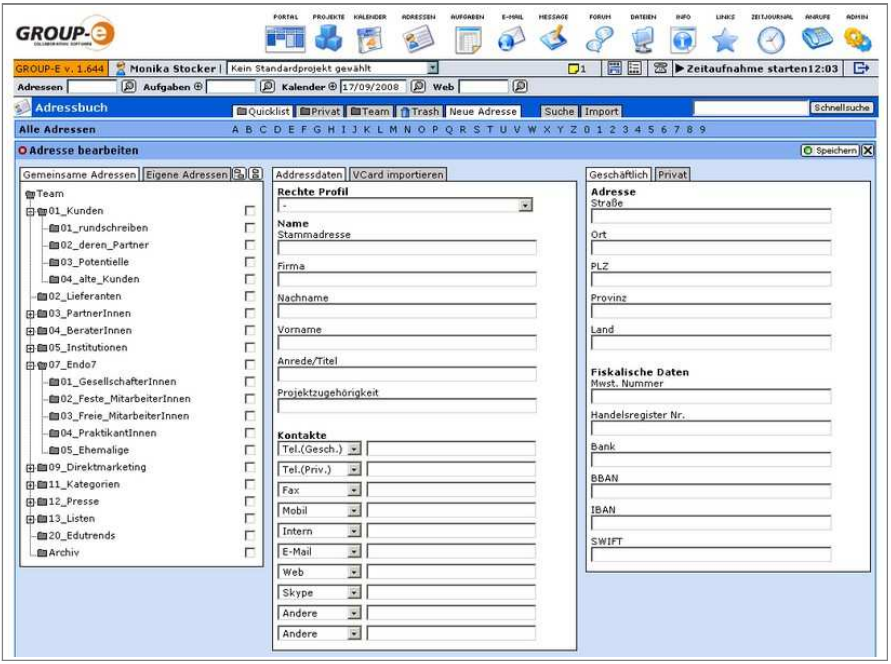
Das Modul Aufgaben ähnelt optisch sehr dem Kalender, erfüllt jedoch einen anderen Zweck. Fein granuliert lassen sich hier **Aufgaben mit Start- und Fälligkeitsdatum** versehen und einzelnen Benutzern, Gruppen oder Projektmitgliedern zuweisen. Eine sehr nützliche Funktion hierbei heißt: *„Aufgabe **delegieren** (Ich bin nicht dabei).“* Sobald diese Aufgabe vom Team oder von einem



Benutzer erledigt wurde, wird per E-Mail darüber informiert und sie wandert automatisch ins Archiv.

Anfallende, überfällige und delegierte Aufgaben werden übersichtlich als Liste und automatisch im Kalender grafisch dargestellt. Ferner sind im Aufgabenmodul folgende Features enthalten: **Logbuch**, Priorisierung, **Kategorisierung, Zuständigkeiten, Projektzuordnung,** Trennung privat und geschäftlich.

1.3 CRM - Adressverwaltung



Die Südtiroler Entwickler haben ihre Collaboration-Software natürlich auch mit einem professionellen Kontaktmanagement versehen. Dabei wird grundsätzlich zwischen privaten und gemeinsamen Adressen unterschieden. So können Benutzer ihr eigenes, persönliches Adressbuch führen und von dem des Unternehmens trennen. **Frei definierbare**, hierarchisch angeordnete **Kategorien** schaffen Ordnung im Adressbestand, welcher sich so hervorragend an die eigenen Bedürfnisse anpassen lässt. Zusätzlich können Adressen auch Projekten zugeordnet werden, um beispielsweise Ansprechpartner, Zulieferer oder Dienstleister in der Projektverwaltung zentral dem Team zugänglich zu machen.

Die integrierte Suche erlaubt schnellen Zugriff auf die gewünschten Adressen, sowie ausgefeilte Filtermöglichkeiten über den kompletten Datenbestand. Suchabfragen lassen sich als **dynamische Mailinglisten** abspeichern, um z.B. den monatlichen Newsletter über den **integrierten E-Mail-Client** zu versenden. Auf Wunsch können diese auch für Benutzer oder Gruppen freigegeben werden. Falls die vorgegebenen Adressfelder nicht ausreichen sollten, lassen sich mit den nötigen Rechten beliebige Felder bequem im Backend hinzufügen.

Eine integrierte, intuitiv zu bedienende **Importfunktion** erlaubt das Befüllen mit beliebigen Daten, was eine **Adressbuch-Migration** über das Austauschformat CSV

extrem erleichtert. Adressen, Verteiler, Kategorien und Suchergebnisse können per Mausklick als XLS- oder CSV-Dateien exportiert und so beliebig weiterverwendet werden. Gerade bei der Erstellung von **Serienbriefen** eine willkommene Funktion.

1.4 VoIP Integration & CTI

Aus Anwendersicht ist es für jedes CRM-System extrem komfortabel, daraus direkt Telefonnummern anwählen zu können. Hierfür wurde GROUP-E mit einem **URL-Handler** versehen, welcher **VoIP**-fähige Softphones oder **Skype** die gewünschte Rufnummer direkt aus dem Browser heraus anwählen lässt.

Seit der Version 1.721 stellt GROUP-E auch eine **direkte Anbindung** an den OpenSource PBX **Asterisk** zur Verfügung. Dies geschieht mittels der Manager Schnittstelle von Asterisk (manager.conf), der Editierung der Datei *callto.php* von GROUP-E, sowie der Anpassung des „callto: URL-Handlers" im Config-Bereich.

```
## AUSZUG callto.php ##
// Manager Zugangsdaten Asterisk
$strHost = "10.x.x.x";
$strUser = "USER";
$strSecret = "PASSWORD";
//Username => Telefonid,Context # Pro Benutzer SIP Telefon & Kontext
$accounts = array(
"amüller" => array("SIP/31","Büro"),
"bmeier" => array("SIP/32","Technik") );
```

Diese Zuordnung von Benutzer und Telefon erlaubt es dem Anwender mit einem Klick auf eine Rufnummer das Gespräch automatisch aufzubauen.

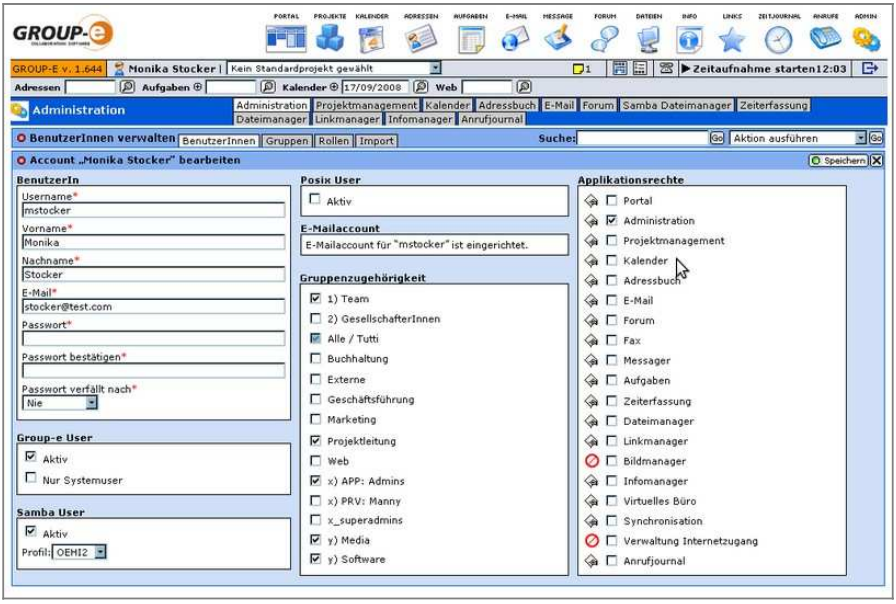
In Zukunft wird die Verwaltung komplett webbasiert erfolgen, momentan muss man dies noch händisch erledigen. Zudem soll über die zukünftige Browserintegration von *Websockets* eine bi-direktionale Kommunikation zwischen Asterisk und GROUP-E ermöglicht werden, so dass auch bei eingehenden Gesprächen automatisch der Kontakt des Anrufenden angezeigt werden kann, sofern er im CRM vorhanden ist.

1.5 iPhone, Android & Co.

Um Daten auch mobil verwenden zu können, wurde GROUP-E mit zwei verschiedenen Synchronisationsschnittstellen ausgestattet: **SyncML** und **Activesync (Mail for Exchange)**. Während SyncML etwas in Jahre gekommen ist, aber bei vielen Handys (vorallem Nokia) fest vorinstalliert war, findet sich Activesync als

moderne Variante des Datenabgleichs auf nahezu allen modernen Smartphones wieder. So lassen sich mit dementsprechend kompatiblen Handys und Smartphones **Emails**, **Adressen**, **Termine** und **Aufgaben** im *„Push-Verfahren“* **synchronisieren**. Leider bringen weder *iOS* noch *Android* eine native Aufgabenunterstützung mit, so dass die Auswahl des Telefons eng mit den eigenen Anforderungen abgestimmt werden sollte.

1.6 Zentrale Benutzerverwaltung



GROUP-E lässt sich entweder in einer reinen MySQL-Umgebung oder zusätzlich mit LDAP-Unterstützung installieren. Nur letztere Variante bietet den vollen Funktionsumfang der Collaboration-Software, da so Benutzernamen, Gruppen und Passwörter über das zentrale **LDAP**-Verzeichnis geführt werden. Die direkte **Anbindung** an den Dateiserver **Samba** ermöglicht auch den Betrieb eines **Domänencontrollers**, worüber sich besonders **Administratoren größerer Netzwerkumgebungen** freuen werden. Alle **Accounts** und Gruppenzugehörigkeiten werden zentral im GROUP-E Admin Interface **webbasiert verwaltet**. Vorteil für die Benutzer: Egal ob Windows-Workstation, Mail-Account, GROUP-E oder Mobiltelefon, sie können sich immer mit denselben Zugangsdaten in die verschiedenen Arbeitsbereiche einloggen. Echtes **Single-Sign-On**!

Auch die **interne Rechtevergabe** haben die Südtiroler Entwickler elegant gelöst.

Für jedes GROUP-E-Modul lassen sich beliebig viele „**Stellvertreter**“ definieren und so die **Rollenbesetzung** an die eigenen Anforderungen anpassen. Einer Sekretärin könnte zum Beispiel das Administrationsrecht für das Kalendermodul zugewiesen werden und Teamleiter und Geschäftsführer das der Zeiterfassung. So lassen sich auch unabhängig vom gemeinen Administrator **geschäftliche Prozesse** und **betriebliche Anforderungen** einstellen und ändern.

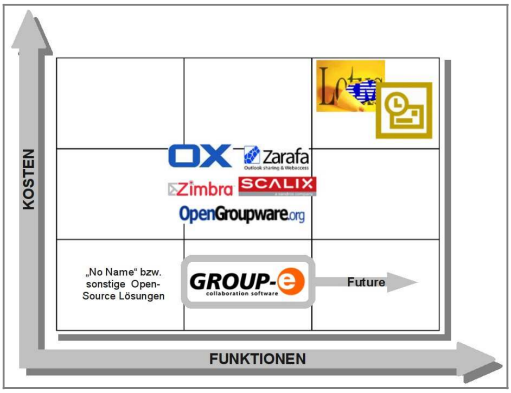
1.7 Zeiterfassung & Dateimanager

Dass GROUP-E weit umfangreicher als andere OpenSource-Groupware bzw. Collaboration-Lösungen ist, wurde bereits eingehend erwähnt. Zwei Highlights kommen aber noch.

Die integrierte, webbasierte Zeiterfassung erlaubt es, **Zeiteinträge auf Projekte** und Tätigkeiten zu buchen. Für jede **Tätigkeit** lassen sich im Backend Rechte und **Mitarbeiterkosten** hinterlegen. Die **Auswertung** der Zeitjournale kann nach Monat, Quartal oder frei definierbarem Datum erfolgen und bietet zahlreiche Filtermöglichkeiten für ein erfolgreiches **Controlling.** GROUP-E verfügt zudem über ein **Spesenmodul**, mit welchem sich zum Beispiel Hotel- und/oder Reisekosten projektbezogen erfassen lassen.

Als letztes **Highlight** sei noch der **Samba Dateimanager** erwähnt. Dadurch haben Anwender auch über den Browser direkten Zugriff auf den Dateiserver. Die Entwickler haben hier keine „zweite Dateiablage“ geschaffen, welche asynchron zum „normalen“ Netzwerk ist, sondern stellen eine **native, webbasierte Dateiserver-Anbindung** über das SMB-Protokoll zur Verfügung. Benutzer können je nach Rechtevergabe Dateien herunterladen, Verzeichnisse erstellen und neue Dateien hochladen. Wünschenswert wäre für zukünftige Releases die Möglichkeit, mehrere Dateien auf einmal gezippt zu laden sowie Dokumente direkt online editieren und abspeichern zu können.

2. Fazit



Im Comicgenre würde man **GROUP-E als *„Underdog"*** bezeichnen, in der IT-Welt ist es schlicht ein Geheimtipp. Auch wenn die schlichte grafische Oberfläche manchem (noch) etwas antiquiert vorkommen mag, steckt unter der Haube das Potenzial einer **professionellen, ganzheitlichen** und **integrativen Unternehmenslösung**, welche ohne versteckte, restriktive Lizenzpolitik wie bei vergleichbaren Open-Source-Lösungen auskommt. Die **Unterstützung offener Standards** und die **intensive Vernetzung der einzelnen Module** untereinander machen GROUP-E zu einem echten Teamplayer und einzigartig in der Angebotsvielfalt von Open-Source-Groupware-Lösungen.

Alle weitere Informationen finden Sie auf der GROUP-E Projektseite:

http://www.group-e.info

3. Der Autor

Harald Grzybowski ist Geschäftsführer von BYTESPRING E-Business Consulting & Solutions in Berlin, das sich auf Linux- und OpenSource-Anwendungen mit Schwerpunkt Groupware-, CRM- und Contentmanagementsysteme spezialisiert hat. Das Leistungsangebot umfasst strategische IT- und E-Business-Beratung sowie Produktion und Implementierung webbasierter Anwendungen.

[http://www.bytespring.de]

uBasic – eine kleine plattformunabängige Basic-Interpreter-Bibliothek

Uwe Berger
bergeruw@gmx.net

Wieso kann es sinnvoll sein einen eigenen Basic-Interpreter zu implementieren? An Hand einer kleinen C-Bibliothek, die ursprünglich für den Einsatz auf 8-Bit-Mikrocontrollern der Firma Atmel[1] konzipiert wurde, aber auch auf anderen Hardwareplattformen eingesetzt werden kann, sollen die Funktionsweise und Anwendungsmöglichkeiten eines solchen Interpreters erläutert werden.

1 Warum ein eigener Interpreter?

Oft steht man als Entwickler vor dem Punkt eine fertige und bereits installierte Anwendung um kundenspezifische Funktionalität erweitern zu müssen. Natürlich könnte man diese Funktionen irgendwie in sein Programm einbauen. Aber dazu ist eine Entwicklungsumgebung (Compiler, Linker usw.) und im Fall von Mikrocontrollern sogar spezielle Hardware zum Brennen des neu übersetzten Programms notwendig, die nicht immer zur Verfügung stehen. Viel einfacher wäre es doch diese fehlende Funktionalität auf einem beliebigen Speichermedium zur Verfügung stellen zu können, um sie einfach innerhalb der bestehenden Applikation auszuführen.

Dieses Ansinnen mit vorübersetzten Binär-Code-Fragmenten zu verwirklichen, dürfte allerdings ein schwieriges, wenn nicht sogar unmögliches Unterfangen sein. Zum einen wird hier wieder eine Entwicklungsumgebung benötigt, welche für die Zielplattform zugeschnitten sein muss. Der Code ist damit auch nicht ohne Weiteres portabel. Zum anderen dürfte es gerade auf Hardware ohne Betriebssystem fast unmöglich sein unabhängigen externen Binär-Code zu laden und auszuführen. Hier kommen nun Skript-Sprachen ins Spiel, die plattformunabhängig agieren

1 http://www.atmel.de

können. Einzige Bedingung ist, dass ein entsprechender Interpreter pro Hardwareplattform verfügbar ist. Das auszuführende Programm wird somit wieder portabel. Dass dieser Ansatz nicht neu ist, zeigen z.B. die Projekte rund um die Skript-Sprache Lua[2].

Es bleibt also eigentlich nur die Frage nach der Syntax der Sprache zu klären, wenn man nicht gerade Lua portieren möchte bzw. kann. Erster Ansatzpunkt könnte dabei sein, eine speziell für die Zielplattform konzipierte Sprache zu verwenden. Für resscourcenarme AVR-Mikrocontroller sind einige Beispiele, wie z.B. ECMDScript[3] oder avrsh[4] bekannt.

Möchte man aber plattformunabhängig bleiben, erscheint es sinnvoller eine universelle Interpreter-Sprache zu verwenden, welche sich durch eine einfache Sprachsyntax sowie einen hohen Bekanntheitsgrad auszeichnet. Die Wahl fiel auf das weit verbreitete und wohl jedem Programmierer bekannte Basic.

Der Zufall wollte es, dass Adam Dunkle im Jahr 2007 ein C-Gerüst für einen kleinen und ressourcensparenden Basic-Interpreter (uBasic)[5] veröffentlicht hat, dessen Sprachumfang eine Untermenge des weit verbreiteten TinyBasic[6] ist. Mit minimalen Modifikationen ist dieses Modul, welches als Bibliothek realisiert ist, auch auf Mikrocontrollern der Firma Atmel lauffähig und beeindruckt durch seinen extrem geringen Ressourcen-Verbrauch. Der Interpreter kommt in seiner Grundkonfiguration mit weniger als 8 kByte Programmspeicher und 1 kByte dynamischen Speicher aus.

Wegen des geringen Ressourcen-Verbrauchs, der durchgängigen Modularität, der leichten Erweiterbarkeit und der weitreichenden Konfigurierbarkeit bietet es sich an, diesen kleinen Interpreter in andere Anwendungen zu integrieren, um deren Funktionalität zu erweitern. Bei der Weiterentwicklung der Ursprungsversion wurden einige Mechanismen und Schnittstellen integriert, die eine Anpassung an die verschiedensten Hard- und Softwareumgebungen sehr einfach gestalten.

Die aktuellste Ausgabe des Interpreters, den es in zwei Versionen gibt, ist auf einer eigenen Projektseite[7] auf Mikrocontroller.net zu finden. Die folgenden Erläuterungen beziehen sich hauptsächlich auf den Entwicklungszweig von Uwe Berger. Einige Besonderheiten zur Version von René Böllhoff werden in einem abschließenden Kapitel beschrieben.

Es sei nochmals betont, dass die Basic-Interpreter-Bibliothek auf unterschiedlichen Plattformen einsetzbar ist. Im Programmarchiv sind Beispielversionen für AVR-Mikrocontroller und Linux zu finden, die mit der jeweiligen gcc-Version und der

2 http://www.lua.org
3 http://www.ethersex.de/index.php/ECMDScript
4 http://battledroids.net/downloads/avrsh.html
5 http://www.sics.se/~adam/ubasic/
6 http://www.ittybittycomputers.com/IttyBitty/TinyBasic/TBuserMan.htm
7 http://www.mikrocontroller.net/articles/AVR_BASIC

Standardbibliothek avr-libc[8] bzw. glibc[9] übersetzbar sind.

2 Wie funktioniert ein Interpreter?

Die Theorie über die Arbeitsweise von Compilern und Interpreter füllt diverse Standardwerke der Informatik. Hier sollen nur die Definitionen und Begrifflichkeiten aufgeführt werden, die zum Verständnis der Funktionsweise eines Interpreters notwendig sind.

Ein Interpreter ist ein Programm, das Anweisungen einer Programmiersprache versteht und unmittelbar ausführt. Im Gegensatz dazu überführt ein Compiler einen Programmquelltext in eine andere Sprache, die in der Regel maschinennah und deshalb prozessorspezifisch ist. Die Ausführung des übersetzten Programms erfolgt asynchron zur Interpretation.

Einer der Vorteile eines Compilers ist, dass die Ausführungsgeschwindigkeit des übersetzten Programms wesentlich höher ist als bei interpretierten Programmen, da der Code an den jeweiligen Prozessor angepasst ist. Daraus ergibt sich aber auch einer der Hauptnachteile: für jede Prozessorplattform wird jeweils ein eigener Compiler benötigt, der die Zielsprache erzeugen kann. Im Gegensatz dazu versteht ein Interpreter auf jeder Plattform das auszuführende Programm in gleicher Art und Weise, vorausgesetzt es existiert für die jeweilige Plattform ein entsprechender Interpreter.

Typischer Weise erfolgt der Interpretationsvorgang in nachstehenden Phasen:

- Analysephase, welche sich unterteilt in:
  - lexikalische Analyse: Erkennung von Schlüsselworten, Bezeichner, Operatoren, Klammern, Symbolen usw. und deren Klassifizierung in Token
  - syntaktische Analyse: Zusammenfassung von Tokens zu Satzbestandteilen, die einer festgelegten Grammatikbeschreibung (Syntax) folgen
  - semantische Analyse: Überprüfung der Satzbestandteile auf Konsistenz
- Ausführungsphase: es werden die Aktionen ausgeführt, die durch das Programm beschrieben wurden

In der Regel überschneiden sich Analyse- und Ausführungsphase bei einem Interpreter. Dies bedeutet, dass sobald eine in sich abgeschlossene Anweisung erkannt

8 http://www.nongnu.org/avr-libc/
9 http://www.gnu.org/software/libc/

worden ist, wird diese sofort ausgeführt.

Statt der Ausführungsphase folgt bei einem Compiler nach der Analyse des Quelltextes eine sogenannte Synthesephase, innerhalb der die erkannten Satzteile in die Zielsprache überführt werden.

Gerade zur Implementierung der Analyseschritte existieren abstrahierende Beschreibungsmethoden und Werkzeuge, die die teilweise automatisierte Generierung der entsprechenden Applikation ermöglichen. Hier ist wiederum auf die Standardliteratur zu verweisen. Bei der Entwicklung von uBasic wurde auf solche Hilfsmittel nicht zurückgegriffen. Lediglich für Dokumentationszwecke wurde der implementierte Basic-Sprachumfang in „Erweiterter Backus-Naur-Form“ (EBNF) beschrieben[10].

Die Funktionsweise von uBasic folgt dem oben beschriebenen Konzept eines klassischen Interpreters. Der Programmquelltext wird nach vordefinierten Tokens durchsucht (lexikalische Analyse). Diese werden dann dem eigentlichen Basic-Interpreter übergeben. Je nach Art des Token erfolgen dort entsprechende Verzweigungen bzw. Aktionen, welche teilweise auch rekursiv sein können. Die Überprüfung, ob die einzelnen Token in der kontextabhängigen korrekten Reihenfolge stehen (syntaktische und semantische Analyse), wird während der schrittweisen Abarbeitung der Basic-Anweisung durchgeführt. Ist eine korrekte Basic-Anweisung oder ein abgeschlossener Teil davon erkannt worden, erfolgt sofort die Ausführung der entsprechenden Aktion.

Ein konkretes Beispiel:

```
10 PRINT "Ergebnis a+1=", A+1
```

- Der Parser erkennt eine neue Befehlszeile (hier Basic-Zeile 10).
- Der Parser ermittelt das Schlüsselwort `PRINT` und übergibt das entsprechende Token an den Interpreter, der seinerseits in die entsprechende Routine verzweigt. Ab diesem Zeitpunkt sind bei der weiteren Abarbeitung nur noch die hier ausprogrammierten Tokens zulässig (beim `PRINT`-Befehl: Strings, Ausdrücke, einige weitere Steuerzeichen). Andernfalls wird das Basic-Programm mit einem Syntax-Fehler abgebrochen.
- Innerhalb der Print-Routine des Interpreters wird das nächste Token angefordert, hier ein String (der Text in Anführungszeichen). Dieser wird sofort ausgegeben.
- Es wird das „Komma-Token“ erkannt und ein Leerzeichen ausgegeben, wie es in der Sprachbeschreibung definiert ist.
- Es werden Elemente eines Ausdrucks (A+1) erkannt. Danach erfolgt eine Verzweigung in die entsprechende Routine zur Analyse/Berechnung des

10 innerhalb der Projektdokumentation

Ausdrucks. Das ermittelte Ergebnis wird wieder ausgegeben.

- Es wird das Befehlszeilenende und damit das Ende der `PRINT`-Anweisung erkannt. Da die Zeile ohne ein weiteres Steuerzeichen abgeschlossen wurde, erfolgt noch die Ausgabe eines Zeilenvorschubs.

3 uBasic-Sprachumfang

In der Folge wird eine kleine Übersicht des Sprachumfangs aufgeführt. Eine ausführliche Beschreibung ist in der Dokumentation zum Projekt zu finden.

Befehlszeile, Programmende

Eine Basic-Zeile beginnt immer mit einer eineindeutigen Zeilennummer, gefolgt von einer Basic-Anweisung, deren Schreibweise nicht case-sensitiv ist. Ein Programm muss mit dem `END`-Befehl abschließen.

Zahlenformate bzw. Darstellungsformen

Es werden nur vorzeichenbehaftete Integer-Zahlen unterstützt, deren Wertebereich sich nach den jeweiligen Plattformspezifika richten. Die Darstellung kann dezimal (Bsp. `1234`), hexadezimal (Bsp. `0X12AB`) oder binär (Bsp. `0B10101010`) erfolgen. Die Ausgabe (`PRINT`-Befehl) erfolgt grundsätzlich dezimal.

Variablennamen-Konvention

Es sind maximal 26 unterschiedliche Variablen möglich, deren Namen aus einem Buchstaben des Alphabets bestehen. Mehrstellige Variablennamen sind nicht zulässig.[11]

Operatoren

Im Vergleich zu zahlreichen anderen Basic-Dialekten sind hier einige Erweiterungen (z.B. Modulo, Bit-Operationen) eingeflossen, die gerade für Steuerungsaufgaben auf Mikrocontrollern interessant sind.

Operator	Bedeutung
ArithmetischeOperatoren	
+	Addition zweier Werte; z.B. *1+2*
–	Subtraktion zweier Werte; z.B. *3-2*
/	Division zweier Werte; z.B. *4/2*
*	Multiplikation zweier Werte; z.B. *3*5*

11 Auch das „Ur-Basic" von T.Kurtz und J.Kemeny sah nichts anderes vor (BIANCUZZI, WARDEN: *Visionäre der Programmierung*, O'Reilly, 2009; S. 81 ff.)!

Operator	Bedeutung
`%, mod`	Modulo (Divisionsrest); z.B. *5%4* bzw. *5 mod 4*
Bit-Operatoren	
`|,or`	bitweise OR-Verknüpfung; z.B. *2|3* bzw. *2 or 3*
`&, and`	bitweise AND-Verknüpfung; z.B. *2&4* bzw. *2 and 4*
`xor`	bitweise XOR-Verknüpfung; z.B. *3 xor 6*
`shr`	Rechtsverschiebung; z.B. *4 shr 1*
`shl`	Linksverschiebung; z.B. *4 shl 1*
Vergleichsoperatoren	
`<`	kleiner
`>`	größer
`=`	gleich
`>=`	größer gleich
`<=`	kleiner gleich
`<>`	ungleich

Basic-Befehle (Übersicht)

Befehl	Bedeutung
`FOR-NEXT`	Schleife von/bis, wobei Zählrichtung (`TO/DOWNTO`) und -weite (`STEP`) optional sind
`IF-THEN-ELSE`	Bedingte Anweisung; Besonderheiten siehe Dokumentation
`GOTO`	Sprungbefehl zur angegeben Zeilennummer
`GOSUB-RETURN`	Unterprogrammaufruf ; „externe“ Unterprogramme möglich (siehe weiter unten)
`SRAND/RAND`	Zufallsgenerator initialisieren; Zufallszahl generieren
`ABS(...)`	Absoluter Betrag des Ausdrucks
`NOT(...)`	Complement des Ausdrucks
`PRINT ...`	Ausgabe, Ausgabegerät plattformspezifisch konfigurierbar
`LET`	Variablenzuweisung (kann weggelassen werden)
`REM`	Kommentarzeile
`END`	Programmende

Folgende Befehle sind in vielen Basic-Dialekten vorhanden aber im Sprachumfang von uBasic nicht enthalten:

- `INPUT`: Dateneingabe (Anmerkung: gerade für kleine Programme auf Mikrocontroller auch nicht unbedingt erforderlich)

- `DIM`: Definition und Zugriff auf Feldvariablen
- `DATA, READ, REWIND, RESET`: Definition und Zugriff auf konstante Daten, welche im Basic-Quelltext definiert sind

4 uBasic-Einbettung und Aufruf

Wie eingangs beschrieben, ist uBasic dafür konzipiert in andere Programme eingebettet zu werden. Folgende Programmzeilen sind dazu so oder so ähnlich neben den dazugehörigen Header-Dateien an entsprechender Stelle einzufügen:

```
ubasic_init(program);
do {
      ubasic_run();
      /* hier koennte etwas anderes gemacht werden */
} while(!ubasic_finished());
```

Die Routine `ubasic_init()` setzt den Parser des Interpreters auf den Anfang des Basic-Programms, initialisiert einige interne Speicherbereiche und ermittelt das erste Token. Als Übergabeparameter ist der Zeiger auf die erste Stelle des Basic-Programms anzugeben, die sich in Typ und Art nach dem jeweiligen Speichermedium richtet (siehe Dokumentation und Beispiele im Projektarchiv).

Die folgende „do/while"-Schleife wird solange abgearbeitet, bis das Basic-Programm endet (Basic-Befehl `END`) oder dort ein Fehler aufgetreten ist. Die eigentliche Abarbeitung des Programms erfolgt innerhalb der Routine `ubasic_run()`. Dabei wird in der Regel bei jedem Funktionsaufruf nur eine Basic-Zeile abgearbeitet und danach die Kontrolle an die einbettende Applikation zurückgegeben. Es ist also ohne Weiteres möglich parallel zur Abarbeitung eines Basic-Programms noch andere Aktionen auszuführen.

5 uBasic Besonderheiten

In der Folge werden einige Besonderheiten des vorgestellten Basic-Interpreters aufgeführt.

Konfigurierbarkeit des Sprachumfangs und Ressourcenverbrauchs

Mit Hilfe von Defines und Makros können zahlreiche Basic-Befehle bei der Übersetzung des Interpreters ausgeklammert und der interne Ressourcenverbrauch (z.B.

Größe diverser Puffer- und Stack-Bereiche) gesteuert werden. Damit besteht die Möglichkeit den Interpreter an die jeweiligen Gegebenheiten und Erfordernisse weitestgehend anzupassen.

Plattform-spezifische Basic-Befehle

Da der Interpreter ursprünglich für den Einsatz auf AVR-Mikrocontroller gedacht war, wurden eine Reihe von Befehle zum Ansteuern der entsprechende I/O-Hardware implementiert:

- EEPROM lesen/schreiben: `EPOKE(), EPEEK()`
- digitale Ein-/Ausgänge festlegen, lesen, setzen: `DIR(), IN(), OUT()`
- Analog/Digital-Wandler auslesen: `ADC()`

Aufruf von externen Funktionen

Der Basic-Interpreter läuft eingebettet in einer Applikation, welche u.a. selbst Unterroutinen beinhaltet. Um deren u.U. sehr komplexe Funktionalitäten auch unter uBasic einfach nutzen zu können, wurde der Basic-Befehl `CALL()` implementiert. Innerhalb einer Konfigurationsdatei muss dazu lediglich ein Zeiger auf die Funktion, Typ und Anzahl der entsprechenden Ein- und Ausgabeparameter sowie ein Name, unter dem die jeweilige externe Funktion angesprochen werden soll, definiert werden.

Lesen/Schreiben von externen Variablen

Mit Hilfe der Basic-Befehle `VPEEK()` und `VPOKE()` ist ein lesender bzw. schreibender Zugriff auf Variablen der einbettenden Applikation möglich. Dazu sind lediglich einige einfache Definitionen in der dafür vorgesehenen Konfigurationsdatei vorzunehmen.

Wahlfreies Speichermedium der Basic-Programme

Mit Hilfe einer Reihe von Makros/Definitionen kann der Zugriff auf das jeweilige Speichermedium der Basic-Quelltexte (z.B. in einem Dateisystem auf einer SD-Karte oder Festplatte, auf einem EEPROM usw.) gesteuert werden. Die hardware- bzw. auch softwarespezifischen Zugriffsmethoden können mit wenigen Programmzeilen angepasst werden. Im aktuellen Quelltextarchiv findet man einige Beispiele für den Zugriff auf solche unterschiedlichen Speichermedien.

Erweiterung des GOSUB-Befehls

Normalerweise können mit dem Basic-Befehl `GOSUB` nur Unterprogramme aufgerufen werden, die sich in der selben Quelltextdatei befinden. uBasic wurde dahingehend erweitert, dass statt einer Zeilennummer auch ein Programmname (z.B. ein Dateiname) angegeben werden kann, um das dort befindlichen Basic-Programm nachladen und als Unterprogramm abarbeiten zu können. Bei Auftreten eines `RETURN` wird die Kontrolle an das aufrufende Programm zurückgegeben. Sämtliche

Basic-Variablen des Hauptprogramms verhalten sich wie global definierte Variablen. Innerhalb eines externen Unterprogramms können weitere aufgerufen werden.

GOTO-Zeilennummernpufferung

Zur Verbesserung der Abarbeitungsgeschwindigkeit von Basic-Programmen ist ein Zeilennummernpuffer zuschaltbar. Ist dieser aktiv, wird beim Auftreten eines `GOTO`-Befehls der Zeiger zur entsprechenden Programmzeile gepuffert. Somit muss ab dem zweiten Sprungbefehl zur selben Zeilennummer der Basic-Quelltext nicht mehr vollständig durchsucht werden, sondern es kann sofort an die entsprechende Stelle gesprungen werden.

6 Version von Renè Böllhoff

Nach der Veröffentlichung meiner ersten Versionen des Basic-Interpreters entfachte sich zwischen einigen Entwicklern eine kleine, aber intensive Diskussion[12] zu denkbaren Erweiterungs-, Konfigurationsmöglichkeiten und Performanceverbesserungen. Dabei stellte sich sehr schnell heraus, dass unterschiedliche, teilweise gegensätzliche Ziele verfolgt wurden, die nur schwer in einer gemeinsamen Version des Interpreters einpflegbar wären. Aus diesem Grund entschloss sich Renè eine eigene Version weiterzuentwickeln. Vor allem folgende Punkte zeichnen seine Version aus:

- ausgefeilte, teilweise komplexe Konfiguration über Präprozessor-Anweisungen, zur:
  - Definition des Basic-Syntax
  - Definition von Zugriffen auf externe C-Funktionen, Variablen und Felder
- Implementierung eines JIT[13]-Compilers, welcher das gesamte Basic-Programm vor der eigentlichen Ausführung in einen Byte-Code überführt und damit eine erhebliche Performance-Verbesserung darstellt.

Anzumerken ist, dass diese Punkte aber zu Lasten des Ressourcenverbrauches gehen. Das ist der Hauptgrund der Versionstrennung.

Andererseits fließen aber auch Ideen der einen Version teilweise in den jeweils anderen Entwicklungszweig ein, sodass man trotzdem von einem gemeinsamen Projekt sprechen kann und deshalb beide Interpreter im selben Projektverzeichnis gepflegt werden.

12 http://www.mikrocontroller.net/topic/177030
13 Just-in-time

Umgekehrte Vorratsdatenspeicherung

Urs Lerch mail@ulerch.net
Marcus Holthaus marcus.holthaus@imsec.ch
http://incubator.apache.org/alois/

Ob Vorratsdatenspeicherung zur Beweisführung für eine Straftat Sinn macht, ist umstritten. Könnte Sie allenfalls zur Entlastung einer davon betroffenen Person verwendet werden? Dies läge klar im Interesse jedes einzelnen Computernutzers. Diskutiert wird allerdings nur über eine zentrale vorrätliche Datenspeicherung durch Provider zur Ermittlung von Straftaten durch Polizeibehörden. Wieso also die Diskussion nicht umkehren, und über eine lokale, dezentrale, transparente Vorratsdatenspeicherung diskutieren, allenfalls mit Hinterlegung bei einem Anwalt? Apache ALOIS ist ein offenes Tool zur Sammlung und Analyse von Logdaten, das die technische Seite der Aufgabe übernehmen könnte.

1 Einleitung

Die Vorratsdatenspeicherung (VDS) ist gerade in Deutschland ein politisch "heisses Eisen". Es geht dabei nicht um eine technologische Debatte, sondern ganz grundsätzlich um das politische Abwiegen von (gemeinschaftlicher) Sicherheit gegenüber (individueller) Freiheit. Es überrascht deshalb auch nicht weiter, dass die Positionen weitgehend bezogen sind und es in der ausführlich, bisweilen hitzig geführte Diskussion kaum mehr inhaltliche Bewegung gibt. Sie gleicht mehr einem Stellungskrieg, in dem mitunter neue Akteure auftauchen, z.B. die EU-Kommission, und Begriffe inhaltlich umgedeutet werden wie z.B. "Quick Freeze".

Dieser Text hat nicht zum Ziel, die bisherige Diskussion zu protokollieren. Und wir möchten ganz bewusst auch nicht Stellung für oder gegen die VDS beziehen. Wir stellen jedoch nüchtern fest, dass jede Person bei der Benutzung eines Computers Spuren in dessen Log-Dateien hinterlässt - dazu sind Logs ja da, und das hat aus technischer Sicht auch Vorteile. Das Problem ist, dass die Spuren, die im Sinne einer Vorratsdatenspeicherung aufgezeichnet und aufbewahrt werden, den davon

Betroffenen nicht zur Verfügung stehen, oder allenfalls nur stark verzögert unter Wahrnehmung des Einsichtrechts des Datenschutzes verfügbar gemacht werden können. Wer kennt schon die Spuren, die er hinterlässt? In diesem Zusammenhang stellen wir die Software Apache ALOIS vor, die technologisch eine Basis darstellen könnte, um die eigenen Datenspuren aufzuzeichnen und auszuwerten.

2 Vorratsdatenspeicherung

Wie bereits erwähnt, wollen wir nicht die bisherige Diskussion zur Vorratsdatenspeicherung erneut aufrollen. Auch nehmen wir keinen Bezug auf den juristischen Bereich der Debatte. In der Folge wollen wir jedoch drei Aspekte aufgreifen, die aus unserer Sicht eine nähere Betrachtung verdienen.

2.1 Transparenz

Der einzelne Nutzer hat keine Ahnung, welche Daten konkret von ihm gespeichert werden. Wüsste er es, könnte das zweierlei bewirken:

- Erstens können allfällige Datenflüsse und Online-Aktivitäten, die vom Nutzer gar nicht gewollt sind, so eruiert wurden. Z.B. bietet das Firefox-Plugin NoScript durch das Blocken von nicht explizit bewilligten Javascripts nicht nur eine zusätzliche Sicherheit vor Viren etc., sondern es können so auch sehr einfach (unerwünschte) Datenflüsse wie etwa diejenigen von Google-Analytics eruiert und darauf blockiert werden. Dem Nutzer würde durch diese Identifikation also einen Anhaltspunkt zur Selbstkontrolle in die bekommen, den er nutzen oder ignorieren kann. Er kann zum Beispiel einen darin kompetenten Bekannten bitten, ein solches Plug-In oder eine lokale Firewall-Software so zu konfigurieren, dass solche Verbindungen verhindert werden.
- Zweitens würde der Aspekt der "Security by Obscurity" aufgelöst, welcher der Vorratsdatenspeicherung innewohnt: Da niemand weiss, was über konkret ihn aufgezeichnet wird oder wurde, kann auch niemand wissen, ob in diesen Daten nicht irgendwo etwas Verbotenes beschrieben wird oder sich daraus herleiten lässt. Auch wer immer seinen moralischen Grundsätzen folgt, ist angesichts der Flut bestehender Ge- und Verbote nicht immer unschuldig im juristischen Sinn. Wer sich mit VDS beschäftigt, muss also zurecht befürchten, dass aus Vorratsdaten Fälle konstruiert werden, die eine Selbstverteidigung erfordern - und das schreckt schon ab, unabhängig davon, ob diese Verteidigung erfolgreich

wäre, und erzeugt eine unangemessene Selbstzensur, die gesellschaftlich schädlich ist. Stünden dem Nutzer seine eigenen Log-Daten zur Verfügung und könnte er prüfen, ob sie etwas Fragwürdiges beschreiben - z.B. durch Vergleich mit einer Liste von URLs, die auf verbotene Inhalte verweisen (Verherrlichung von Nationalsozialismus etc.) - so könnte der Nutzer seine Aktivität überdenken und in Zukunft vermeiden. Zudem wäre er gewarnt, dass er sich unter Umständen verteidigen müsste - die eigentliche Verfolgung von potentiellen Straftaten wird also nicht behindert, sondern es wird mehr Fairness erzeugt. Um einen Vergleich eigener Online-Zugriffe mit einer Sperrliste anzustellen, müsste man solche Listen natürlich zur Verfügung haben... einige solcher ursprünglich geheimen Listen sind inzwischen öffentlich bekannt geworden und können verwendet werden.

Wir erachten es deshalb als sinnvoll, lokal eine Software zur Aufzeichnung des Datenverkehrs zu installieren. Wenn dies auf allen verwendeten elektronischen Geräten oder einer lokalen Proxy vorhanden ist und die Aufzeichnungskriterien denjenigen der Provider entsprechen, ist eine weitgehende Transparenz gewährt.

2.2 Dezentrale Speicherung

Es wäre auch möglich, die in eigener Regie gesammelten Daten nicht selbst zu bevorraten, sondern diese bei einem Treuhänder oder Notar individueller Wahl zu lagern - in diesem Fall betriebe der Notar das System zur Sammlung der Daten. Dies könnte mit einer client-seitigen Proxy verbunden werden, um Vollständigkeit zuzusichern und mit einer Client-seitigen Verschlüsselung kombiniert werden, so dass der Zugriff auf die Daten auch für den Treuhänder erst nach Öffnung eines versiegelten Umschlags möglich ist, welcher den Schlüssel enthält, oder überhaupt nur in Mitwirkung des Dateninhabers. Der Zugriff auf die Daten kann dann also zur Verteidigung desjenigen erfolgen, der sie hinterlegt hat. Damit ein solches Vorgehen hinsichtlich Manipulation nicht ähnliche Qualitäten wie eine zentrale VDS bekommt, müssen Schlüsselmanagement und notarielle Pflichten angemessen definiert werden.

2.3 Content Logging

VDS beschränkt sich explizit "nur" auf die Rahmendaten. Das ist aus Gründen des Datenschutzes und der benötigten Speichermenge nachvollziehbar, stellt aber auch aus kriminalistischer Sicht eine enorme Einschränkung dar. Werden die Logdaten lokal und geschützt aufgezeichnet, können diese Vorbehalte allerdings vernachlässigt werden. Die so gewonnenen Daten wären in einem allfälligen Fall

hingegen von grosser Bedeutung.

3 Apache ALOIS

Wie wir oben aufgeführt haben, erscheint es uns sinnvoll, die Vorratsdatenspeicherung dezentral zu organisieren. Dass ein solches Vorhaben technologisch nicht unmöglich ist, zeigt das aus ähnlichen Überlegungen entstandene Diaspora, ein Konkurrenzprojekt zu Facebook. Nicht zuletzt da auch immer noch das Schreckgespenst eines "Bundestrojaners" im Raume steht, muss ein solches Vorhaben zwingend mit Werkzeugen durchgeführt werden, die einer respektierten Open Source Umgebung entstammen, wie es z.B. die Apache Software Foundation (ASL) darstellt.

3.1 Wofür steht Apache ALOIS?

Apache ALOIS [1] ist eine Software für die Sammlung, Aufsplittung und Korrelation von (Log-)Meldungen mit einer integrierten Reporting- und Alarmierungsfunktionalität. Die Akronym ALOIS steht für "Advanced Log Data Insight System". Apache ALOIS ist ein „Security Information & Event Manager“ (SIEM). Darunter versteht man ein EDV-Toolset, das innerhalb eines (Unternehmens-)Netzwerks Meldungen von unterschiedlichsten Geräten und Software sammelt und zentral abspeichert, um darauf aufbauend diese Meldungen zu interpretieren, zu analysieren, zusammenzufassen, zu korrelieren und allenfalls zu alarmieren. Ein SIEM wird dabei nie als vollständig automatisiertes System verstanden, es benötigt immer Menschen, die mit dem System interagieren und entsprechendes Know-how haben.

Der Begriff SIEM ist eine Kombination aus SIM ("Security Information Management") und SEM ("Security Event Management"), was zwei unterschiedliche Produktkategorien sind. Während SIMs eine Langzeitspeicherung, Analyse und Reporting von Logdaten bieten, stehen SEMs in erster Linie für ein Monitoring in Echtzeit, der Korrelation von Ereignissen und der Notifikation von als kritisch erachteten Begebenheiten. Ein SIEM vereint die Langzeitarchivierung von Logdaten mit der Korrelation von Ereignissen in einem einzigen Tool.

Während sich die meisten anderen SIEMs, ob Open Source oder proprietär, primär um das technologische Sicherheits-Monitoring kümmern, konzentriert sich Apache ALOIS auf das Monitoring von sicherheitsrelevanten Inhalten. Es will dabei eine pro-aktive Rolle spielen in der Erkennung von potentiellem Datenverlust oder -Diebstahl, irrtümlichen Modifikationen von Dateien sowie unerlaubtem Zugriff.

Seit dem Herbst 2010 ist Apache ALOIS in Inkubation bei der Apache Software Foundation [2]. Die Inkubationsphase verschafft einem Softwareprojekt eine Stabilität, die mit derjenigen anderer erfolgreicher ASF-Projekten vergleichbar ist. Projekte der Apache Software Foundation definieren sich durch einen kollaborativen, konsens-orientierten Entwicklungsprozess, kombiniert mit einer pragmatischen, businessfreundlichen Lizenz. Apache-Projekte sind bestrebt, Software von höchster Qualität zu erstellen, die zu den Leadern im jeweiligen Feld gehören. Diese Eigenschaften sind auch bekannt als der "Apache way".

Die Aufnahme in den Inkubationsprozess setzt voraus, dass die Software einen stabilen Zustand erreicht und das Potential dazu hat, als vollwertiges Apache-Projekt aufgenommen zu werden. Apache ALOIS hat denn auch bereits über mehrere Jahre seine Stabilität im produktiven Einsatz bewiesen. Während die Software zwar schon immer Open Source war, wurde sie bisher in erster Linie durch eine einzelne Firma gewartet. Mit der Aufnahme in die Apache-Gemeinschaft soll der Entwicklungsprozess nun geöffnet werden. Dabei werden nicht nur neue Mitwirkende gesucht, das Projekt ist auch offen gegenüber Entwicklungen in Richtungen, die bisher nicht vorgesehen wurden.

3.2 Architektur von Apache ALOIS

Apache ALOIS besteht aus fünf interagierenden Modulen, die durch ihre weitgehende Unabhängigkeit die Skalierbarkeit eines SIEM sicherstellen:

- *Insink* ist das Sammelbecken, das all die unterschiedlichen einkommenden Meldungen in Apache ALOIS aufnimmt. Es basiert teilweise auf der weit verbreiteten Software `syslog-ng`. [5] Das Modul Insink nimmt Nachrichten entgegen (UDP), wartet auf Meldungen (TCP), kann komplexe Nachrichten (Dateien, Mails) entgegennehmen und führt eine vorgängige Filterung durch, um einen Überfluss an Nachrichten zu verhindern.
- *Pumpy* als teil von Insink stellt den FIFO-Buffer für eingehende Nachrichten dar. Implementiert als relationale Datenbank, welche die Originalmeldungen im RAW-Format enthält. In einer komplexen Umgebung kann es durchaus aus mehrere Instanzen von Insink und Punpy geben, z.B. für Gruppen von Hosts, für spezifische Meldungstypen oder zur Erhöhung der Ausfallsicherheit.
- *Prisma* enthält die Funktionalität und Logik zur Aufsplittung der Meldungen in separate Felder, basierend auf regulären Ausdrücken. Prisma besteht eigentlich aus einem Satz aus "Prismi", welches jedes ein Prisma für einen Nachrichtentyp (Cisco, Apache httpd, etc.) steht. Auf eine Nachricht können auch mehrere Prismi angewendet werden. Dies ermöglicht die Verarbeitung von gestapelten Nachrichten, also z.B.

weitergeleitete E-Mails, die komprimierte Dateien enthalten. Die so verarbeiteten Nachrichten werden in einer Datenbank namens Dobby abgespeichert. Aufgrund der Tatsache, dass Prisma in Ruby geschrieben ist, können Prismi interaktiv hinzugefügt und modifiziert werden (sofern die entsprechenden Berechtigungen vorhanden sind).

- *Dobby* ist die zentrale Datenbank und enthält die Meldungen aufgesplittet nach Typen (pro Typ eine Tabelle) und Feldern. In einer produktiven Umgebung ist Dobby gewöhnlich getrennt von Pumpy installiert, um eine optimale Performance und Verfügbarkeit zu gewährleisten. Die aktuelle Implementierung basiert auf MySQL.
- Der Analyzer enthält die beiden Subsysteme *Lizard* und *Reptor*. Lizard ist die Analyse-Engine inkl. Benutzeroberfläche von Apache ALOIS, in Ruby on Rails unter Verwendung von AJAX implementiert. Durch Lizard wird das interaktive Surfen auf den gesammelten Daten ermöglicht. Die Standard-Funktionalität beinhaltet Exklusion/Inklusion/Selektion, Sortieren, Filterung, das Erstellen von Ansichten sowie die ad-hoc Erstellung von textlichen und grafischen Reports. Reptor auf der anderen Seite ermöglicht die automatische Benachrichtigung bei Ereignissen aufgrund von vordefinierten Sichten, Filtern und Übereinstimmung von Pattern in diesen. Dies kann interaktiv im Browserinterface oder durch E-Mail geschehen.

Abbildung 1 zeigt eine Übersicht über den Datenfluss zwischen den Modulen:

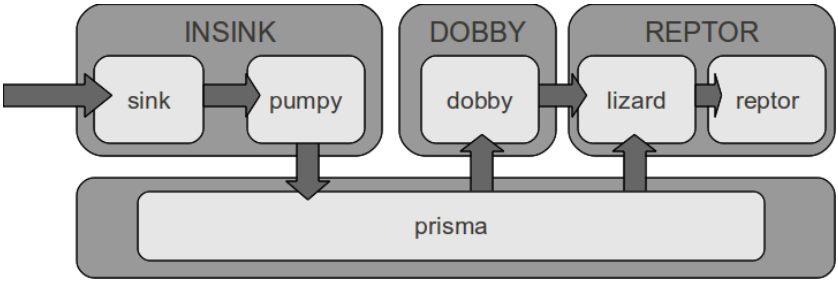


Abbildung 1: Komponenten und Datenfluss von Apache ALOIS

Die Installation von Apache ALOIS ist sehr einfach. Zur Integration in eine bestehende Umgebung müssen einerseits die vorhandenen Systeme, Geräte und Applikationen angschlossen werden. Das geschieht im einfachsten Fall dadurch, dass deren syslog-Ziel auf die ALOIS-Instanz zeigt. Für einige komplexere Applikationen gibt es bereits Connectoren, weitere können einfach in verschiedenen Programmiersprachen hinzugefügt werden. Andererseits sind Filter und Berichte zu erstellen. Apache ALOIS verwendet dazu SQL und reguläre Ausdrücke. Es gibt ebenfalls bereits zahlreiche vorderfinierte Filter und Berichte.

Dank seiner modularen Architektur kann Apache ALOIS auf einem oder mehreren Servern installiert werden. Skalierbarkeit ist denn auch integrativer Bestandteil der Software. Auch ein Dienst in der Cloud ist einfach zu realisieren. Aber auch eine Installation auf dem persönlichen Desktopcomputer oder Laptop ist out-of-the-box möglich.

Apache ALOIS ist grundsätzlich offen für jede Art von Input, es müssen also nicht zwingend Logdaten sein. Die Standardschnittstellen sind syslog, SMTP und ein Datei-Upload. Im SIEM-Umfeld spricht man dabei von "Agenten", welche als Schnittstelle für spezifische Datenformate dienen und auch wieder verwendet werden können. Durch das Hinzufügen von weiteren Agenten kann Apache ALOIS sehr einfach für beinahe beliebige Datenformate erweitert werden. Die Nutzung von Apache ALOIS ist deshalb nicht auf ein SIEM beschränkt, sondern kann für jede Anforderungen konfiguriert werden, bei der das Sammeln, Analysieren, Korrelieren und Auswerten von Daten wichtig ist.

Zur erleicherten Anbindung von Software von Dritten, sei sie proprietär oder Open Source, beabsichtigt das Apache ALOIS Projekt einen "Service-Bus" mit standardisierten Schnittstellen zu implementieren. Die vorgeschlagene Architektur ist in Abbildung 2 aufgezeigt.

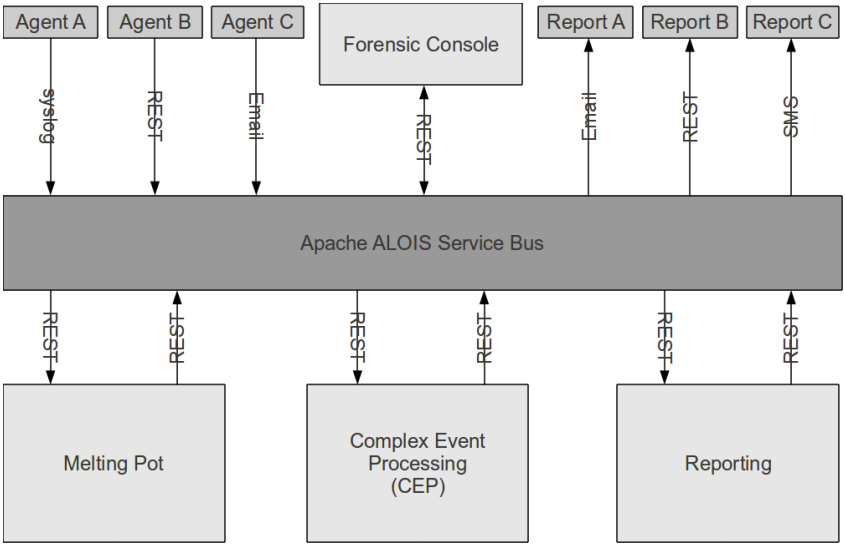


Abbildung 2: Apache ALOIS Service Bus

Somit wird es nicht nur einfach sein, ein weiteres System anzuhängen, sondern es können auch sehr einfach einzelne Bestandteile mit für eine bestimmte Anforderung besser geeigneten Tools zu ersetzen.

Als Open Source Software ist ALOIS selbstverständlich Work-In-Progress. Die

aktuelle Version verfügt über die Mehrzahl der oben erwähnten Eigenschaften, aber wir rufen hiermit auch zur Mitarbeit an Konzept, Technologie und Integration auf, ebenso wie zur Diskussion sinnvoller Einsatzszenarien - sei es im Bereich VDS, Data Leakage Detection oder andere.

4 Schlussfolgerungen

Zu Beginn haben wir aufgezeigt, dass sich die erhitzte Diskussion zur Vorratsdatenspeicherung durch den Aspekt der Denzentralisierung unter Umständen entpolarisieren liesse. Voraussetzung dafür ist allerdings eine transparente Software, die auf allen Geräten läuft und die Daten zu einem Treuhänder übermittelt. Technologisch ist dies realisierbar, wenn es sicherlich auch noch einige herausfordernde Probleme zu lösen gilt.

Als Grenzen der vorgeschlagenen "umgekehrten" Vorratsdatenspeicherung sehen wir insbesondere folgende Punkte:

- Alois löst das Problem nicht, dass immer der Missbrauchsverdacht durch Regierungsstellen oder Unternehmen besteht. Aus Sicht des Einzelnen müsste eine persönliche Log-Datensammlung natürlich in dem Moment vernichtet werden, in welchem es strafbar wird, sie einer strafrechtlichen Untersuchung vorzuenthalten - sie soll schliesslich nicht zusätzlich belastend wirken.
- Auch dass grosse zentrale Datensammlungen ein attraktives Ziel für Cyberkriminelle sind, kann Alois nicht lösen, sondern fügt dem zusätzliche dezentrale Datensammlungen hinzu. Alois muss daher selbstverständlich geschützt betrieben werden, stellt aber aufgrund der Verteilung ein weniger attraktives Ziel dar.
- Schliesslich liegt es im Interesse des Einzelnen, seine Daten vollständig oder eben nur teilweise selbst zu sammeln. Hier sind neben rechtlichen und praktischen Erwägungen auch technische Restriktionen relevant: Beispielsweise ist es heute noch kaum möglich, die Rahmendaten der eigenen UTMS- und GSM-Nutzung zu erhalten. Hier sind aber Werkzeuge denkbar, die auf den moderenen Smartphones implementiert werden können.

Dessen ungeachtet erfüllt das Projekt Apache ALOIS bereits viele der Anforderungen, die für ein solches Vorgehen notwendig wären. Da es sich um ein Open Source Projekt mit offener Community handelt, ist die Transparenz gewährleistet und die fehlende Funktionalität liesse sich hinzufügen. Dieser Aufwand kann nicht - und sollte auch gar nicht - durch eine einzelne Person oder Organisation getragen werden. Die Apache Software Foundation als Host von

Apache ALOIS bietet sowohl für Einzelpersonen, Regierungsstellen, Non-profit-Organisationen als auch Unternehmen gerade auch aus juristischer Sicht eine hervorragende Plattform dafür dar.

Wir verstehen diesen Text als Anreiz und nicht als Lösung. Einserseits sind viele politische und juristische Fragen noch nicht bis ins Detail durchdacht. Andererseits fehlt auch noch Funktionalität. Wir möchten deshalb alle Interessierten dazu einladen, an der Diskussion teilzunehmen sowie an Apache ALOIS mitzuwirken. Die Homepage des Projekts bietet dazu einige Einstiegspunkte.

Literatur

[1]http://incubator.apache.org/alois/

[2] http://www.apache.org

Wege aus der DOSBox

Henrik Kretzschmar

henrik@nachtwindheim.de

Dieser Beitrag beschreibt eine Erweiterung für den Emulator DOSBox, mit welcher Programmteile von DOS-Programmen durch eigenen Code ersetzt werden können. Anwendungsmöglichkeiten sind die Behebung von Fehlern, kleineren Anpassungen bis hin zur kompletten Rekonstruktion des zu emulierenden Programms.

1 DOS-Programme

MS-DOS, die bekannteste DOS-Variante, wurde für den i8086 Prozessor von Intel entwickelt. Dieser Prozessor, mit einer Wortbreite von 16bit und seinem 1MB Adressraum, war allerdings ein sehr kleiner Nenner, dessen Grenzen später mittels verschiedener Brückentechnologien (EMS, XMS) mit erheblichem Mehraufwand umgangen werden mussten.

Die Tatsache, das Anwendungsprogramme die Alleinherrschaft über den PC für sich beanspruchten, fiel nicht ins Gewicht, da DOS ursprünglich als „Single Task“ System entwickelt und oft nur als solches genutzt wurde. Somit warteten die Anwendungsprogramme z.B. aktiv auf Tastatureingaben, und verschwendeten somit Prozessorzeit, die außer ihnen sowieso niemand nutzte.

Nichtsdestotrotz werden auch heute noch DOS-Programme verwendet, sei es nativ in produktiv genutzten Kassenautomaten oder Computerkassen oder von Retro-Spielern. Letztere nutzen oft den Emulator DOSBox um die alte Software auf aktuellen Systemen zu betreiben.

2 DOSBox

Das Programm DOSBox ist ein Software Emulator, welcher Intel-Prozessoren bis einschließlich des „Pentium“, einen Großteil der damals üblichen Hardware und Technologien, sowie ein DOS inklusive Kommandointerpreter emuliert. Er ist portabel, mobil und für alle gängigen Betriebssysteme verfügbar. Die verwendeten Programmiersprachen sind C und C++. Als Softwarelizenz kommt die GPL zum Einsatz. Die Erstveröffentlichung des Quellcodes war am 27.07.2002 und auch jetzt 2011 wird das Projekt aktiv weiterentwickelt.

Leider ist dieser Emulator etwas langsam, was durch den in der DOS-Ära praktizierten Programmierstil noch verschlimmert wird. Das führt auf heutigen Systemen zu hoher Systemlast, da das aktive Warten auch emuliert werden muss. Die Entwickler von DOSBox arbeiten deshalb an allgemeinen Lösungen, von denen möglichst viele emulierte Programme profitieren. Wird jedoch ein DOS-Spiel in Kombination mit DOSBox als Produkt vertrieben, kann eine spezielle Anpassung des Emulators an das Spiel eine sinnvolle Alternative sein.

3 DOSBox custom

„DOSBox custom“ ist eine Erweiterung für DOSBox, die das Überspringen von Funktionsaufrufen ermöglicht. Mit ihr ist es möglich, Code, dessen Funktionsweise bekannt ist, auf dem Host auszuführen und die Emulation zu überspringen, was bei oft genutzten und aufwendigen Funktionen einen enormen Geschwindigkeitsvorteil bringen kann.

Getestet und angewandt wurde diese Erweiterung bis jetzt bei „Real Mode“ Programmen, welche mit dem Borland C++ 3.0 Compiler übersetzt wurden. Es ist davon auszugehen, dass es auch mit Kompilaten anderer Compiler, anderen Sprachen und im „Protected Mode“ funktioniert.

„DOSBox custom“ ist als Patch für DOSBox V0.74 verfügbar. Es kann beim Kompilieren abgeschaltet werden, was in einer unveränderten DOSBox resultiert. Wird es beim Kompilieren angeschaltet, hat es einen sehr geringen Overhead beim Starten und Beenden von Programmen, sowie bei “call“ Befehlen.

3.1 Initialisierung

Die Nutzung dieser Erweiterung ist nur sinnvoll, wenn das entsprechende Programm gestartet wurde. Ein neuer Prozess wird unter DOS mit dem execute Systemaufruf erzeugt. Vom Programmierer muss eine Funktion bereitgestellt werden, die Überprüft, ob es sich bei dem soeben gestarteten Programm wirklich um jenes handelt, für welches er seine Ersatzfunktionen geschrieben hat, da es sonst zu Fehlfunktionen kommen wird. Als einfache, aber sehr unsichere Variante sei die Prüfung des Dateinamens genannt. Besser ist es, zusätzlich noch nach bestimmten Zeichenketten, z.B. Versionsnummern, innerhalb des Programms zu suchen. Dem Sicherheitsempfinden des Programmierers sollen keine Grenzen gesetzt sein. Ist die Gültigkeit festgestellt und an „DOSBox custom“ rückgemeldet worden, wird die Erweiterung aktiviert.

3.2 Laufzeit

Nachdem „DOSBox custom“ zu Laufzeit aktiviert wurde, wird bei jedem „call“-Befehl eine Funktion aufgerufen, welche vom Programmierer bereitgestellt werden muss. Da es zwei unterschiedliche Arten von „call“-Befehlen gibt, die „nearcalls“, innerhalb eines maximal 64kb großen Speichersegments, und die „farcalls“, welcher sich über den gesamten Adressraum erstreckt, müssen vom Programmierer auch zwei Funktionen bereitgestellt werden. Diese beiden Funktionen werden als „nearcall“- und „farcall“-Handler bezeichnet. Die Aufgabe beider Funktionen ist es zu Überprüfen ob es sich bei der aufzurufenden emulierten Funktion um jene handelt, welche vom Programmierer gewünscht ist. Dies geschieht durch Vergleichen der Segment und Offsetadressen.

Eventuell nutzt das Programm auch die Overlay-Technik von Borland, mit welcher Programmcode-Segmente bei deren Aufruf dynamisch nachgeladen werden können. Das hat jedoch zu Folge, dass sich die Segmentadressen ändern können, was eine Prüfung bei „nearcalls“ erschwert. Als Brücke zwischen dem statischen und dem dynamischen Code dient ein kleines Codesegment mit fester Segmentadresse, welches „Stub“ genannt wird. Es dient als Einsprungspunkt für „farcalls“, hinter dem sich ein Sprungbefehl zu dem wirklichen Position befindet, falls das Overlaysegment schon im Speicher ist. Ist es nicht im Speicher, befindet sich an der Stelle ein Aufruf an den Interrupt 0x3f, hinter dem sich der Overlaymanager von Borland verbirgt. Dieser schafft, falls nötig, freien Platz, läd das angeforderte Codesegment nach, setzt die Adressen der Sprunganweisungen im Stub und führt anschließend die gewünschte Funktion aus. Da sich der „Stub“ immer an einer festen Position befindet, kann über diesen die momentane Adresse des Segments ermittelt werden. Stimmt diese mit der aktuellen Codesegmentadresse im Register CS überein, ist die gesuchte Funktion gefunden. Zum ermitteln dieser Adresse gibt es in „DOSBox custom“ ebenfalls ein Funktion, die dem Programmierer das Leben erleichtern kann.

Parameterübergabe

Die Aufrufparameter der gewünschten Funktion befinden sich auf dem Stack der DOSBox-CPU. Hält sich das DOS-Programm an die C-Konvention, so befindet sich der erste, links stehende Parameter als erstes auf dem Stack und kann mit dem DOSBox Befehl CPU_Pop16() oder CPU_Pop32() von der DOSBox-CPU ermittelt werden und sollte in lokalen Variablen gespeichert werden. Diese Befehle passen auch den Stackzeiger (Register SP) der DOSBox-CPU entsprechend an.

Laut C-Konvention räumt der Aufrufer den Stack wieder auf, weswegen der Programmierer die ursprüngliche Position des Stackzeigers wiederherstellen muss.

Eine Möglichkeit ist mit den Befehlen CPU_Push16() und CPU_Push32() in umgekehrter Reihenfolge die Werte der lokalen Variablen wieder auf den Stack zu

schreiben. Es spricht auch nichts dagegen die Position des Stackzeigers vor den CPU_Pop() Befehlen zu speichern und danach wieder ins SP Register der DOSBox CPU zu schreiben, da die Daten auf dem Stack nicht beschädigt werden.

Die Firma Borland hat allerdings einen Trick genutzt, welcher bei den „nearcalls" eine Sonderbehandlung erfordert. Das Gegenstück zu den „call"-Befehlen bilden die „ret"-Befehle, von denen es auch eine „near" und „far" Variante gibt. Um Funktionen sowohl innerhalb eines Segments und von Außerhalb aufrufen zu können, wurde nur der „farret"-Befehl benutzt. Wird diese Funktion von innerhalb des Segmentes aufgerufen, wird das aktuelle Codesegment auf den Stack gepusht und die Funktion mit einem „nearcall" aufgerufen. In diesem Fall befindet sich die Adresse des aktuellen Codesegmentes auf dem Stack und muss vom Programmierer manuell entfernt werden. Wenn er die gewünschte Funktion überspringen möchte, darf er diesen Wert nicht wieder auf den Stack zurücklegen, da Dieser, entgegen den C-Konventionen, vom Aufgerufenen entfernt würde. Soll die Funktion von der DOSBox-CPU emuliert werden, muss der Adresse des Codesegments wieder auf den Stack.

Eigener Code

Nun, da man die Werte der Parameter hat, kann man die Ersatzfunktion mit diesen Werten aufrufen. Möchte der Programmierer das DOS-Programm beobachten, kann er eine entsprechende Meldung, z.B. mit printf() an der Konsole ausgeben oder in eine Logdatei schreiben.

Rückgabewert

Wurde eine Funktion, die einen Wert zurück gibt, überbrückt, muss dieser Wert noch an die DOSBox-CPU übermittelt werden. Dazu dienen für 8- und 16-bit Werte das Register AX, welches über die Variable reg_ax zu erreichen ist. Bei 32bit-Werten, wozu auch Zeiger zählen, muss der niederwertige Teil oder der Offset ins Register AX, der höherwertige Teil ins Register DX. Dieses ist über die Variable reg_dx erreichbar.

Überspringen oder Emulieren

Als letztes muss „DOSBox custom" informiert werden, ob die Funktion emuliert oder der Funktionsaufruf übersprungen werden soll. Der Rückgabewert 0 steht für Emulieren, der Rückgabewert 1 steht für das Überspringen der Funktion des DOS-Programms.

3.3 Beenden

Das Beenden von „DOSBox custom“ erfolgt über den exit() DOS-Systemaufruf, innerhalb von DOSBox. Wird dieser ausgeführt muss vom Programmierer eine Aufräumfunktion bereitgestellt werden, die den Zustand vor dem Initialisieren wieder herstellt.

4 Zusammenfassung

Die Lösung „DOSBox custom“ ist sehr speziell, da sie Kenntnisse der Programmiersprache C, der Funktionsweise von Intelprozessoren der x86 Reihe, sowie eine eingehende, statische Untersuchung des DOS-Programms voraussetzt. Die ersten beiden Fähigkeiten können universal vorausgesetzt und anderweitig genutzt werden. Das statische Untersuchen muss für jedes einzelne Programm durchgeführt werden. Gibt es verschieden Versionen dieser Programme welche unterstützt werden sollen, muss jede Version untersucht und für jede Version je ein „near“- und ein „farcall“-Handler bereitgestellt werden.

Als Einschränkung sind zu nennen, dass die Einsprungspunkte der aufzurufenden Funktionen konstant sein müssen. Es darf sich somit nicht um selbst-modifizierende Programme handeln.

Als weiterer Nachteil kann sich herausstellen, dass die ersetzten Funktionen keine zu emulierenden Funktionen aufrufen dürfen. Softwareinterrupts, sowie andere ersetzte Funktionen sind erlaubt. Somit kann es sein, dass die zu ersetzende Funktion des DOS-Programms viele andere Funktionen aufruft, welche ebenfalls erst ersetzt werden müssen, bevor man die eigentliche Funktionen ersetzen kann. Dieser Aufwand muss im Vorfeld bei der statischen Analyse des Programms ermittelt werden.

Die Lösung eignet sich am Besten für eine komplette, schrittweise Rückentwicklung von DOS-Software. Der große Vorteil daran ist, dass während des gesamten Rückentwicklungsprozesses immer eine funktionierende Version vorliegt, da nach jedem ersetzten Programmteil die Funktionalität überprüft werden kann.

Literatur

- [1] Website von DOSBox http://www.dosbox.com

3 Zusammenfassungen der weiteren Vorträge

3.1 2 Become 1 Again – Reflections on Merging KVM into QEMU

Jan Kiszka, Siemens AG, Corporate Technology
jan.kiszka@siemens.com
When the Linux hypervisor extension KVM was born, it required a user space mate: qemu-kvm. It started as a fork of the QEMU emulator. But on its way into upstream, KVM was actually merged as a reimplementation. These versions were slowly converging over the past years, but they are still not the same.
The talk will work out the motivation for this approach. It will provide an overview on the current status, the yet unmerged bits, and how they affect both KVM users and developers. Reflecting this way of merging back a fork into upstream, the talk will name both benefits and problems seen so far.

WWW-Seite zum Vortrag: http://www.linux-kvm.org

3.2 3D-Internet mit XML3D

Rainer Jochem, Universität des Saarlandes, DFKI GmbH
rainer.jochem@dfki.de, http://www.dfki.de
Mit dem heutigen Trend ist es an der Zeit, auch das Internet um interaktive 3D-Inhalte zu erweitern. Mit XML3D, welches ausschließlich existierende Techniken wie HTML5, CSS, DOM und AJAX für dynamische Inhalte nutzt, ist es Webdesignern möglich, ihr Wissen auch auf 3D-Inhalte anzuwenden. XML3D ist in die Browserframeworks von Mozilla und WebKit/Chromium integriert, eine portable Implementierung mittels JavaScript und WebGL steht ebenfalls zur Verfügung. Inhalte für eine 3D-Webseite können entweder mittels der XML3D-Spezifikation oder über den XML3D-Blender-Exporter erzeugt werden.

WWW-Seite zum Vortrag: http://www.xml3d.org/

3.3 AHCI and KVM – how to do storage access right

Alexander Graf, SUSE
Virtualization usually looks at making CPU and memory accesses fast, maybe even networking. But what about disk access? Some workloads require you to access a storage device fast and not every guest operating system can run paravirtualized drivers. This talk will give an overview on why IDE is slow for virtualization and KVM in particular, what AHCI actually is and how it helps virtualizing disk workloads like running Windows guests.

WWW-Seite zum Vortrag: http://www.qemu.org

3.4 Aktiver Datenschutz mit dem BDSG

Stephan Kambor, BlinkenArea
st@blinkenarea.org, http://www.blinkenarea.org/

Das Bundesdatenschutzgesetz (BDSG) bietet besonders durch die Änderungen im Jahr 2009 Bürgern die Möglichkeit zum Schutz von persönlichen Daten. Jeder hat das Recht zu wissen, was über ihn im einzelnen gespeichert ist und woher die Daten stammen. Ferner muss er darüber informiert werden, sobald es zu einem Datenschutzverstoß gekommen ist. Laut BDSG sind Geldbußen bis zu 300.000 € möglich. ABER: Wo kein Kläger, da kein Richter.

Der Vortrag beinhaltet eine kurze Einführung und vor allem praktische Beispiele: Wie bekomme ich Auskunft? Was mache ich als Betroffener? Brauche ich einen Rechtsanwalt?

WWW-Seite zum Vortrag: http://st.blinkenarea.org/datenschutz/

3.5 Aktuelle Entwicklungen beim Linux-Kernel

Thorsten Leemhuis, Heise Zeitschriften Verlag GmbH & Co. KG
thl@ct.de, http://www.heise.de/

Alle drei Monate erscheinen neue Linux-Kernel mit erweitertem Funktionsumfang sowie besserer Hardware-Unterstützung durch neue und überarbeitete Treiber. Der Vortrag gibt einen Überblick über die Neuerungen der Ende März oder Anfang April erwarteten Linux-Version 2.6.38 und ihrer Vorgänger.

Außerdem kommen Erweiterungen zur Sprache, die Linux-Distributoren einsetzen oder Entwickler zur Aufnahme in künftige Kernel-Versionen vorbereiten. In diesem Zusammenhang kommen Entwicklungsprozess, Versionsschema sowie einige andere für die kurz- und langfristige Entwicklung von Linux wichtige Themen im Kernel-Umfeld zur Sprache.

WWW-Seite zum Vortrag: http://www.heise.de/open/kernel-log-3007.html

3.6 Barrierefreier Videorecorder mit Linux

Sebastian Andres
sebastian@sebastianandres.de

Die Flexibilität von Linux und besonders die Veränderbarkeit von allen Anwendungen eröffnen unerschöpfliche Möglichkeiten. Sind DVD-Recorder, welche im Handel erhältlich sind, nur auf die breite Masse zugeschnitten, bleiben Personen, welche aufgrund von unterschiedlichsten Einschränkungen oder Behinderungen nur mit größten Bemühungen diese Geräte verwenden können, im Regen stehen. Dieser Vortrag zeigt auf, wie ein fast vollständig barrierefreier Videorecorder auf der Basis von Debian für weniger als 170 Euro «zusammengebastelt» werden kann.

3.7 Binäremulation – Linux unter *BSD

Karl Uwe Lockhoff

Heutzutage hört man häufig von Virtualisierung. Es gibt jedoch auch andere Methoden, Fremdsoftware zur Ausführung zu bringen, z.B. die Binäremulation. Was ist dies genau? Binäremulation ermöglicht das Ausführen von native binaries, also ausführbaren Programmen, in Form von Maschinencode, die eigentlich für ein anderes Betriebssystem geschrieben wurden.

3.8 Community erleben

Torsten Franz, ubuntuusers
http://ubuntuusers.de

Im Fokus des Vortrags steht ubuntuusers.de. Es wird auf die Entwicklung des Portals in den letzten Jahren eingegangen und erklärt, wie sich das Team zusammensetzt, wie die Kommunikation in dieser Gemeinschaft aussieht und wie die Nutzer einbezogen werden. Weitere Schwerpunkte sind Softwarezusammenstellung und Problembehandlung. Es können Fragen zum Portal und zur Gemeinschaft gestellt werden, getreu nach dem ubuntuusers.de-Motto: «Fragen ist menschlich».

WWW-Seite zum Vortrag: http://ubuntuusers.de

3.9 Communtu – Erstelle deine individuelle Live-DVD

Timo Denissen, Univention GmbH
info@communtu.de, http://www.univention.de

Communtu ist ein Web-2.0-Projekt, welches sich als Ziel gesetzt hat, die Installation von Ubuntu und die nachträgliche Software-Installation zu vereinfachen. Der Benutzer wählt nach Themen sortierte Programmzusammenstellungen (sogenannte Bündel) aus und kann diese entweder direkt installieren oder sich hieraus ein ISO-Image erstellen lassen, welches dann auf eine CD/DVD gebrannt oder auf einen USB-Stick kopiert werden kann. Der Benutzer kann auch eigene Bündel erstellen und diese anderen Benutzern zur Verfügung stellen.

WWW-Seite zum Vortrag: http://de.communtu.org

3.10 Cross Compiler – Windows-Anwendungen unter Linux entwickeln

Wolfgang Dautermann, FH Joanneum
http://www.fh-joanneum.at/fzt/

Unter Linux Anwendungen für Windows oder andere Betriebssysteme entwickeln? Mit einem Cross-Compiler ist das durchaus möglich. Wie funktioniert das Ganze? Wozu braucht man das überhaupt? Wie kann man das selbst einsetzen? Der Vortrag beantwortet diese Fragen.

3.11 Customer Relationship Management – Sugar CRM meets SAP

Christoph Steinhauer, it-novum GmbH
ruth.heidingsfelder@it-novum.com, http://www.it-novum.com

Im Customer Relationship Management hat sich im Open-Source-Umfeld SugarCRM als Lösung etabliert. Durch die Parallelwelten ERP und CRM entstehen oft doppelte Arbeit, inhomogene Daten und inkonsistente Informationen über die Kunden. Die richtige Verbindung von ERP und CRM verhindert diese Probleme, indem z.B. Produktdaten aus SAP in SugarCRM genutzt oder Angebote durch den Vertrieb in SugarCRM erstellt und nach Auftragserteilung nahtlos zur Weiterverarbeitung nach SAP transferiert werden. Damit sind auch AdHoc-Abfragen aus SugarCRM z.B. über den aktuellen Umsatz möglich.

WWW-Seite zum Vortrag: http://www.it-novum.com/portfolio/sugarcrm.html

3.12 Das Cmake Buildsystem – Programmentwicklung leichtgemacht

Wolfgang Dautermann, FH Joanneum
http://www.fh-joanneum.at/fzt/

Cmake ist ein Buildsystem, das in der Open-Source-Welt das bisherige Autoconf-/Automake-System langsam ablöst (z.B. im KDE-Projekt). Wie funktioniert das Compilieren mit Cmake? Was kann man damit sonst noch machen? Wie kann man Cmake und das dazugehörige Cpack im eigenen Projekt einsetzen?

WWW-Seite zum Vortrag: http://www.cmake.org

3.13 Datenbanken von MySQL zu PostgreSQL portieren

Andreas Scherbaum
ads@pgug.de

Seit dem Kauf von MySQL durch Oracle sind viele Anwender verunsichert und schauen sich nach Alternativen um. PostgreSQL ist eine dieser Alternativen, sie verzeichnete speziell im letzten Jahr regen Zulauf. Der Vortrag zeigt Wege zum Portieren einer Datenbank von MySQL nach PostgreSQL, zusätzlich wird auf häufige Stolpersteine und Probleme eingegangen.

3.14 Dem Hack keine Chance: LAMP sicher betreiben

Holger Uhlig, Heinlein Prof. Linux Support
h.uhlig@heinlein-support.de, http://www.heinlein-support.de

Backdoors und Manipulation auf dem Webserver? Wir demonstrieren, wie Sie Einbrüche durch eine sichere PHP-Konfiguration verhindern. Die einzelnen Sicherungsschichten werden durch einfache praktische Einbruchsversuche veranschaulicht.

3.15 Die Helfer der Kommandozeile

Axel Beckert, ETH Zürich / Debian
abe@deuxchevaux.org
Der Vortrag ist ein Crashkurs für Einsteiger mit Schwerpunkt auf den Standardprogrammen der Kommandozeile. Zuerst werden für viele Kommandozeilenprogramme gültige Grundlagen, wie typische, häufig vorkommende Optionen, erklärt, und es wird gezeigt, wie man für konkrete Programme Hilfe und Informationen über Optionen, Parameter und Nutzung bekommt. Der zweite Teil zeigt anhand von Beispielen und den wichtigsten Optionen und Parametern, wie man häufig anfallende Arbeiten, z.B. Kopieren, Verschieben, Umbenennen und Suchen, mit den Standard-Programmen auf der Kommandozeile bewerkstelligt.

WWW-Seite zum Vortrag: http://noone.org/talks/commandline-helpers/

3.16 Die Open-Source-Infrastruktur-Cloud

Peter Ganten, Univention GmbH
info@univention.de, http://www.univention.de
Open Source hat für das Cloud Computing eine besondere Bedeutung, denn nur sie kann technikbedingte Abhängigkeiten zu Anbietern nachhaltig vermeiden. Die Präsentation gibt einen Überblick, welche Open-Source-Technologien heute bereits für die Realisierung von Clouds existieren, wie diese miteinander verbunden werden können und welche Vorteile hinsichtlich Flexibilität und Nachhaltigkeit sich für Anwender daraus ergeben. Anhand eines praktischen Beispiels aus der Finanzbranche wird gezeigt, wie Open-Source-Infrastruktur-Clouds mit Open-Source-Werkzeugen auf einfache Weise verwaltet werden können.

3.17 Dokumentation von Softwareprojekten mit Sphinx

Markus Zapke-Gründemann, Deutscher Django-Verein e.V.
info@keimlink.de, http://www.django-de.org/
Mit Hilfe von Sphinx lassen sich Softwareprojekte einfach und trotzdem visuell ansprechend dokumentieren. Ursprünglich zur Dokumentation der Programmiersprache Python geschrieben, benutzen inzwischen auch viele andere Projekte Sphinx. Mit Hilfe einer einfachen Markup-Sprache werden Sphinx-Dokumente formatiert und in einer Baumstruktur abgelegt. Es werden diverse Ausgabeformate unterstützt: HTML, LaTeX zur Erzeugung von PDF-Dateien, Manual Pages und Plain Text. Sphinx ist Open-Source-Software (BSD Lizenz).

WWW-Seite zum Vortrag: http://sphinx.pocoo.org/

3.18 Dokumentenmanagement im Intranet mit Plone

Maik Derstappen, Inqbus Hosting – Zope & Plone Hosting
maik.derstappen@inqbus.de, http://inqbus-hosting.de

Plone ist ein leistungsfähiges und flexibles Content-Management-System für den professionellen Einsatz. Insbesondere das ausgefeilte und sichere Zugangsmanagement für Benutzer und Gruppen sowie die workflow-gesteuerte, kollaborative Verwaltung der Inhalte heben Plone aus der Vielzahl anderer CMS heraus. Der Vortrag möchte anhand der aktuellen Version zeigen, wie Plone im Intra-/Extranet zur Dokumentenverwaltung verwendet werden kann. Themen sind unter anderem Einbindung ins Dateisystem von Linux, Windows und MacOS, die leistungsstarke Suchmaschine für PDF und ODT sowie die Arbeitsabläufe bei der Verwaltung.

WWW-Seite zum Vortrag: http://plone.org

3.19 Dokumentenmanagement mit Open Source und SAP

Andreas Böhm, it-novum GmbH
ruth.heidingsfelder@it-novum.com, http://www.it-novum.com

Viele Unternehmen setzen SAP ein, möchten aber SAP-Belege und -Dokumente zentral verwalten. Der ITN Connector DMS ist eine bidirektionale Schnittstelle zwischen SAP und dem Open-Source-Dokumentenmanagementsystem Alfresco. Durch diese kann man SAP-Belege wie Rechnungen direkt in Alfresco ablegen. Dadurch können Mitarbeiter, die keinen eigenen SAP-Zugang besitzen, webbasiert auf SAP-Belege zugreifen. Gleichzeitig ist es möglich, aus Alfresco heraus Daten in den SAP-Beleg zu schreiben. Der Vortrag zeigt die Funktionsweise der Schnittstelle und mögliche Einsatzszenarien.

WWW-Seite zum Vortrag: http://www.it-novum.com/sap-open-source/

3.20 Ein Jahr OpenStreetMap

Andreas Tille, Debian
tille@debian.org, http://www.debian.org

In exakt einem Jahr praktischer Erfahrung mit OpenStreetMap sind einige Anfangshürden zu nehmen gewesen. Der Vortrag soll Hilfe und praktische Anleitung geben, wie diese Hürden genommen werden können, wie das Kartenmaterial von OpenStreetMap genutzt und insbesondere auch verbessert werden kann und wie weitere Interessenten für OpenStreetMap gewonnen werden können. Der Vortrag legt Wert auf praktische Aspekte von der kurzen Einführung in typische Programme bis zu Tips, wie man den Rest der Familie überzeugt, dass Urlaub nicht nur Erholung, sondern auch Sammeln von Geodaten bedeutet.

3.21 E-Mail-Verschlüsselung mit GPG. Von der Key-Erzeugung zur verschlüsselten E-Mail.

Birgit Hüsken, Hochschule Niederrhein, Abt. KIS - IT Service Management
birgit.huesken@hs-niederrhein.de, http://www.hs-niederrhein.de
«E-Mails soll man verschlüsseln», hört man als Anwender immer wieder; dass es so etwas wie GnuPG gibt, auch – aber wie geht das Ganze? In meinem Vortrag möchte ich möglichst viele Fragen von der Key-Erzeugung über das Web of Trust bis hin zur praktischen Anwendung klären. Er richtet sich vor allem an Anfänger.

3.22 Enterprise-Cloud-Lösung – OpenStack in der Praxis

Christian Baumann, B1 Systems GmbH
info@b1-systems.de, http://www.b1-systems.de
Cloud Computing ist in aller Munde! Linux ist hier die dominante Plattform, und mit OpenStack haben sich viele IT-Größen zusammengeschlossen, um eine offene Cloud-Lösung zu entwickeln. Die B1 Systems GmbH ist auf dem Markt vor allem für ihr Know-how im Bereich Virtualisierung und Hochverfügbarkeit bekannt und hat bei ersten Kunden bereits mit der Evaluation von OpenStack begonnen. In diesem Vortrag wird aus der Praxis berichtet: Warum entscheiden sich große Firmen für OpenStack, wie wird evaluiert, paketiert und getestet? Antworten auf diese und weitere spannende Fragen erhalten Sie im Vortrag OpenStack in der Praxis.

3.23 Erstellung großer und größter Dokumente mit dem Satzsystem TEX

Herbert Voß, DANTE e.V.
Herbert.Voss@FU-Berlin.de
Die Erstellung großer und vor allen Dingen druckreifer Dokumente erfordert besondere Voraussetzungen. Traditionelle Textverarbeitungen wie Word oder OpenOffice kommen sehr schnell an ihre Grenzen, wenn es gilt, Dokumente mit mehr als 500 Seiten zu handhaben. Besondere Schwierigkeiten treten zudem auf, wenn viele Abbildungen und/oder Gleichungen Teil des Dokuments sind. Das Satzprogramm TEX mit dem Makropaket LATEX bietet hier sehr viele Vorteile, da es erlaubt, die reine Texterstellung unabhängig von der eigentlichen Formatierung zu halten.

3.24 Fakten statt Bauchgefühl: RAID-Mathematik für Admins

Holger Uhlig, Heinlein Prof. Linux Support
h.uhlig@heinlein-support.de, http://www.heinlein-support.de
Ein RAID ist ein RAID ist kein RAID. Mythen und Legenden, welches RAID für welche Applikation denn nun die beste Wahl ist, gibt es viele. Doch welche Grundlagen und Faktoren sind bei der Einrichtung zu beachten, und wo sind Fallstricke versteckt? Diese und weitere Fragen werden durch eine sachliche Betrachtung der Grundlagen beleuchtet und nehmen einem das bekannte schlechte Bauchgefühl.

3.25 Fehler finden – ein strategischer Guide

Ralph Angenendt, CentOS
ralph+clt@strg-alt-entf.org, http://www.centos.org/
«Und dann sagt er immer '403', obwohl ich /var/www/html/ schon auf '777' habe.» Fehler unter Linux treiben Einsteigern und auch vielen gestandenen Linux-Nutzern Tränen in die Augen. Dabei macht Linux nicht nur das Leben, sondern auch die Fehlersuche einfach, wenn man sich nur an ein paar Regeln hält. Dieser Vortrag will eben jene Regeln aufzeigen und dem Nutzer ein grundlegendes strategisches Rüstzeug in die Hand geben, mit dem sich Fehler finden und eventuell auch vermeiden lassen. Also keine Panik und tief durchatmen.

3.26 Firefox und Thunderbird plattformübergreifend nutzen

Jana Wisniowska
Nach einem Umstieg auf ein neues Betriebssystem dauert es üblicherweise einige Zeit, bis man sich in der neuen Arbeitsumgebung zu Hause fühlt. Bei dem E-Mail-Client Thunderbird und dem Browser Firefox besteht die Möglichkeit, diese Anwendungen plattformübergreifend zu nutzen. In diesem Vortrag wird erklärt, wie man in wenigen Schritten seine Firefox- und Thunderbird-Einstellungen inklusive Bookmarks und E-Mails auf eine eigene Partition verschiebt, so dass man sie auch auf dem neu installierten Betriebssystem nutzen kann.

3.27 freedroidz – Mit Robotern Programmieren lehren

Joscha Häring, Torsten Wylegala
j.haering@tarent.de
Mit freedroidz haben wir uns das Ziel gesetzt, Kindern und Jugendlichen Programmieren beizubringen. Anhand von Lego-Mindstorm-Robotern haben wir verschiedene Methoden entwickelt, wie man Kindern ohne technische Vorkenntnisse Java beibringen kann. Wir haben schon Workshops an Schulen durchgeführt, möchten aber, dass freedroidz in den normalen Informatikunterricht einfließt. Freedroidz ist Open Source und läuft auf Linux-Systemen, deshalb enstehen keine Lizenzkosten.

WWW-Seite zum Vortrag: http://freedroidz.org

3.28 Gemeinsam stärker: Eine integrierte Open-Source-BI-Plattform mit Palo und Pentaho

Stefan Müller, it-novum GmbH
http://www.it-novum.com

Die Pentaho-Suite und Jedox Palo stellen Business-Intelligence-Lösungen auf professionellem Niveau dar, unterscheiden sich aber funktional und technisch stark voneinander. Pentaho bietet leistungsstarke Features bei Datenintegration und Standardreporting, während Palo mit seinem OLAP- und Excel-Integrationsansatz überzeugen kann. Der Vortrag stellt die Unterschiede der beiden Anbieter dar und zeigt danach auf, wie sich die jeweiligen Vorzüge in einer gemeinsamen BI-Architektur kombinieren lassen, die eine breite Palette von Anforderungen abdeckt.

3.29 Grafiktreiber im Linuxkernel – die Außenseiter

Lucas Stach

Moderne Grafikkarten integrieren vielfältige Funktionen. Entsprechend komplex sind auch die zugehörigen Treiber. Essentielle Teile dieser Treiber residieren gar im Userspace und haben mit dem Kernel nur über verschiedene Interfaces zu tun. Wie ist der Linux-Grafikstack aufgebaut, und wieso ist er so komplex? Wie sehen die aktuellen Entwicklungen aus? Wo wird es in Zukunft hin gehen? Ist die Entwicklung dieser Treiber nur etwas für Gurus, oder kann auch ich etwas beitragen? Auf all diese Fragen versucht der Vortrag eine umfangreiche Antwort zu geben.

3.30 Hochverfügbarkeit von SAP auf SLES 11 SP1

Ralph Dehner, B1 Systems GmbH
info@b1-systems.de, http://www.b1-systems.de

Dieser Vortrag stellt verschiedene von Novell empfohlene Lösungen für die Hochverfügbarkeit von SAP Netweaver unter SUSE Linux Enterprise Server 11 SP1 mit der HA-Extension vor. Detailliert werden der HA- sowie der I/O-Stack vorgestellt. Anhand unterschiedlicher Anwendungsbeispiele wird gezeigt, wie SAP auf SLES 11 SP1 hochverfügbar gemacht werden kann.

3.31 Höher, Schneller, Weiter – openSUSE 11.4

Sirko Kemter, openSUSE
gnokii@opensuse.org, http://www.opensuse.org

Mitte März wird die neueste Version von openSUSE, Version 11.4, erscheinen. Sie bietet viel Neues: von Systemd als Bootmechanismus über Ayatana für Gnome bis hin zu Tumbleweed und Evergreen.

3.32 Icinga – Open Source Monitoring More Powerful Than Before

Bernd Erk, Icinga – Open Source Monitoring
info@icinga.org, http://www.icinga.org
Icinga is an enterprise grade open source monitoring system. With scalable and extensible monitoring, notification, and reporting capabilities, it is ideal for distributed environments. Beyond a Nagios fork, Icinga features PostgreSQL and Oracle support, an API based modular architecture, and a highly dynamic UI. This presentation will introduce Icinga's technical foundations, explain how it's designed to enable greater redundancy in complex networks, and discuss REST API development alongside useful addons. We'll demo the new web interface with integrated reports and business process views, and give insight into future plans for the project.

WWW-Seite zum Vortrag: http://www.icinga.org

3.33 Integration und Anwendung von BIRT im PLM

Chris Hübsch, ARC Solutions GmbH
chris.huebsch@arcsolutions.de, http://www.arcsolutions.de
In Unternehmen der diskreten Fertigung ist die Verwendung von Software-Systemen für das Product Lifecycle Management (PLM) üblich. Zur Unterstützung der Entscheidungsfindung ist es notwendig, die in diesen Systemen gespeicherten Informationen aufzubereiten. Mit dem BIRT-Projekt existiert ein Open-Source-Projekt der Eclipse Foundation, welches umfangreiche Funktionen der Datenbeschaffung, -verarbeitung und -aufbereitung bietet. Wir zeigen am PLM-System Siemens Teamcenter, wie man weitere Datenquellen für BIRT erschließt, z.B. mittels SOA. Des Weiteren wird auf die Aufbereitung der gewonnenen Daten in Reports eingegangen.

3.34 Integration von Open-Source-Software in proprietäre Softwareprodukte – Lizenzverletzungen vermeiden

Hendrik Schöttle, Osborne Clarke
http://www.osborneclarke.de
Open-Source-Software findet immer größere Verbreitung – auch als Teil kommerzieller Software. Viele Entwickler setzen inzwischen existierende Open-Source-Lösungen als einzelne Komponenten oder auch als zentrales Herzstück in ihren Produkten ein. Gleichzeitig häufen sich die Streitfälle infolge von Lizenzverletzungen beim Einsatz von Open-Source-Software. Im Vortrag soll dargestellt werden, welche Herausforderungen bei der Integration von Open-Source-Komponenten in kommerzielle Software zu beachten sind.

3.35 IPv6 Intrusion Detection mit Snort

Martin Schütte, Uni Potsdam
info@mschuette.name
IPv6 kommt – und damit kommen auch Angriffe auf seine Design- und Implementationsfehler. Hier wird ein neues Snort-Plugin vorgestellt, um solche Angriffe im eigenen Netz zu erkennen.

3.36 IT-Service-Optimierung mit OSS

Frederik Kramer, Otto von Guericke Universität Magdeburg
http://www.wif.ovgu.de/
Mit der Zahl von Anwendungssystemen nimmt die Komplexität der Steuerung von Dienstleistungsprozessen mit verschiedenen Anbietern zu. Die Medien Telefon und E-Mail genügen weder der Forderung, Leistungsprozesse nachvollziehbar zu dokumentieren, noch ermöglichen sie gezieltes Monitoring und Steuerung. Die Folge sind Effizienzverluste. Der Open-Source-Tasktracker Mantis ermöglicht eine Vielzahl von Konfigurationen, um die Prozesse nachhaltig zu dokumentieren und zu steuern. Anhand von Praxisbeispielen wird aufgezeigt, wie das Informationssystem Dienstleistungsprozesse verbessern kann.

3.37 Kalender im privaten und beruflichen Alltag – Synchronisierung mit mobilen Geräten *ohne* Outlook

Sebastian Andres
sebastian@sebastianandres.de
Jedes noch so einfache Handy bringt einen Kalender von Haus aus mit. Alle Linux-Distributionen enthalten ebenfalls Kalenderanwendungen. Die Probleme fangen aber dann an, wenn die Kalender vom Mobiltelefon mit dem vom Computer synchronisiert werden sollen. Dieser Vortrag erklärt zunächst, was Caldav ist, um im zweiten Schritt die Installation und Konfiguration eines eigenen Calendarservers ausführlich zu erklären. Der Vortrag richtet sich ausdrücklich an Ein- und Umsteiger von Windows nach Linux!

WWW-Seite zum Vortrag: http://calendarserver.org

3.38 Kerberos – Was ist das und wie funktioniert das?

Mathias Feiler, Rechenzentrum Universität Hohenheim
feiler@uni-hohenheim.de, http://www.uni-hohenheim.de
In diesem Theaterstück zum Mitmachen wird die grundlegende Idee der Kerberos-Authentifizierung durchgespielt. Es geht nicht um reale Implementierungen, vielmehr soll ein allgemeines Verständnis für die Arbeitsweise, die Einsetzbarkeit und die sensiblen Punkte dieses Protokolls vermittelt werden.

3.39 KMUX – Ein ganzes Unternehmen in einer Box

Georg Schütz, KMUX-Projekt, KaMUX GmbH & Co. KG
georg.schuetz@kamux.de, http://kmux.de
Das KMUX-Projekt hat eine integrierte DV-Umgebung für KMU geschaffen. Die miteinander integrierten Anwendungen bieten gerade kleinen Unternehmen Lösungen an, die ihre IT sonst eher als Qual und Problem von und für Spezialisten sehen. Wir haben «Business Intelligence» nicht durch umfassende neue Entwicklungen erreicht, sondern durch die clevere Kombination vieler verfügbarer spezialisierter F/OSS-Komponenten. Der Vortrag schildert die Erfahrungen der letzten 3 Jahre, zeigt den aktuellen Stand der Möglichkeiten und blickt ein bisschen in die Zukunft.

3.40 Konfigurationsdateien mit Git verwalten

Julius Plenz
julius@plenz.com
Sie haben Nutzeraccounts auf mehreren Rechnern, wollen aber überall eine weitgehend gleich aussehende Umgebung vorfinden? Sie haben ein neues Shell-Alias erstellt – dann sollte das auch ohne großen Aufwand auf allen anderen Rechnern verfügbar sein!
Wenn Sie Ihre Konfigurationsdateien (wie z.B. .Xmodmap oder .vimrc) mit Git verwalten, dann können Sie eine Stammkonfiguration sowie eine Liste von rechnerspezifischen Anpassungen automatisch vorhalten.

WWW-Seite zum Vortrag: http://git-scm.org

3.41 LibreOffice in der Praxis

Jacqueline Rahemipour, LibreOffice / The Document Foundation
jrahemipour@documentfoundation.org, http://de.libreoffice.org
LibreOffice ist neu und doch schon altbekannt. Die freie Office-Suite basiert auf OpenOffice.org, bringt aber bereits jetzt viele eigenständige Neuerungen mit. Der Vortrag zeigt LibreOffice im Praxiseinsatz: Was gibt es Neues in LibreOffice bzw. OpenOffice.org 3.3? Worin unterscheiden sich die beiden Office-Suiten tatsächlich? Der Vortrag lässt darüber hinaus viel Raum für konkrete Fragen zu Anwendungsproblemen oder speziellen Anwendungsfällen.

3.42 Linux im Büro von Kleinunternehmen

Klaas Freitag, SUSE, openSUSE.org
freitag@kde.org, http://www.opensuse.org
Kein Kleinunternehmen kann heute ohne Computer auskommen, insbesondere bei Angebots- und Rechnungserstellung. Dass sich Linux als Desktop-System im Kleinunternehmen eignet, zeigt dieser Vortrag: Standardwerkzeuge wie Office und moderne Desktops sind vorhanden. Es gibt aber auch Software zur Verwaltung von Fi-

nanzen, zur Rechnungs- und Angebotserstellung: Kraft ist ein KDE-Programm, mit dem einfach, effizient und flexibel Geschäftsdokumente erstellt werden können. Dieser Vortrag berichtet aus der Praxis, gibt Lösungsempfehlungen und stellt das Kraft-Projekt im Detail vor.

WWW-Seite zum Vortrag: http://www.volle-kraft-voraus.de

3.43 Low-Level Memory Management in the Linux Kernel

Jörg Rödel, AMD
http://www.amd64.org/
The main purpose of the Linux kernel is to abstract the hardware for the applications. An important part of the hardware is the computer's main memory. This talk gives an introduction to the algorithms and techniques the Linux kernel uses to manage the memory for itself and the application. It will concentrate on the allocators implemented for the available pages and the different SLAB allocators. The talk will also explain why the allocators perform well down from embedded devices up to large scale NUMA machines.

3.44 LXC: Des Vanilla Kernels Container

Erkan Yanar
erkan.yanar@linsenraum.de
OpenVZ ist noch immer die Containerlösung für Linux, welche es bisher nicht in den Vanilla Kernel geschafft hat. LXC ist dagegen im Vanilla Kernel. Es baut auf den Control Groups auf, welche auch eigenständig genutzt werden können. So hat der Vortrag den Anspruch, in die schöne Implementierung von LXC einzuführen. Dies vertieft das Verständnis seiner Funktionsweise. Die Konfiguration und Administration sowie Use Cases von LXC werden besprochen. Außerdem werden wir diskutieren, warum der Autor in Produktivsystemen trotzdem OpenVZ nutzt.

WWW-Seite zum Vortrag: http://lxc.sourceforge.net/

3.45 Mailtrace: E-Mail-Recherche für Helpdesks

Stefan Neben, Heinlein Prof. Linux Support
s.neben@heinlein-support.de, http://www.heinlein-support.de
Heinlein Mailtrace bringt Licht ins Dunkel Ihrer Logfiles. Lassen Sie sich die Logdaten Ihres Postfix-Mailservers in Echtzeit auswerten. So lässt sich sofort nachvollziehen, was aus einer versandten E-Mail geworden ist. Erfolgreiche Zustellungen vermeintlich verlorengegangener E-Mails lassen sich dem Endanwender schnell nachweisen. Auch die Ursachen für Zustellfehler und Mailverzögerungen analysiert Heinlein Mailtrace sofort. Es bietet für Endanwender verständliche Erklärungen und gibt Hinweise zur Abhilfe.

WWW-Seite zum Vortrag: http://www.heinlein-support.de/mailtrace

3.46 Mein eigener Webserver – das (un-)bekannte Wesen

Sandro Grundmann, The unbelievable Machine Company
clt@unbelievable-machine.com
Fast jeder hat sich wahrscheinlich schon mit dem Gedanken getragen, einen Webserver zu betreiben, sei es privat, sei es beruflich, für den Verein oder sein Hobby. Was gibt es zu beachten? Wie findet man sich in der breiten Auswahl der Angebote zurecht? Der Vortrag versucht einen Bogen zu schlagen, Themen wie Performance und Skalierbarkeit (auch ohne Cloud) werden hierbei ebenso angerissen wie Netzwerkfragen und -sicherheit.

3.47 Memory Control Groups

Johannes Weiner
hannes@cmpxchg.org
In einem Linux-System haben standardmäßig alle Prozesse ein gleiches Anrecht auf den verfügbaren Hauptspeicher, und der Kernel vergibt diesen solange, bis aller Speicher in Verwendung ist. Memory Control Groups hingegen erlauben es dem Administrator, Prozesse in Gruppen zu organisieren und diesen Gruppen physikalische Speicherlimits zu setzen. Der Vortrag wird erklären, wann und wie der Kernel die Speichernutzung einer Memory Control Group verfolgt und wie er ihr Limit forciert. Im Anschluss wird demonstriert, wie man Memory Control Groups im Userspace anlegt und diese konfiguriert.

3.48 Mit dem Midnight Commander Freiheit leben

Thomas Winde, Thomas Winde Ausflugsfahrten
ausflug@web.de, http://www.ausflug-web.de
In meinem Einsteigervortrag möchte ich zeigen, wie man das Programm Midnight Commander (mc) effizient benutzen kann. Es wird auf die Tastaturbedienung, die vielfältigen Konfigurationsmöglichkeiten und die zahlreichen eingebundenen Programme eingegangen.

3.49 Mit OTRS und opsi IT-Serviceprozesse optimieren

Rico Barth, c.a.p.e. IT GmbH
info@cape-it.de, http://www.cape-it.de
Der Vortrag betrachtet die Vorfallsbearbeitung im Service Management sowie die Unterstüzung derartiger Prozesse durch OTRS mit der ITIL-konformen Erweiterung KIX4OTRS. Die Schaffung einer einheitlichen Konfigurationsdatenbasis, die das Hardware- und Software-Management integriert, steht dabei im Fokus. Durch die Kopplung des Software-Management-Systems opsi, welches Inventarisierungsdaten liefert, wird der manuelle Pflegeaufwand der Konfigurationsdaten stark reduziert.

WWW-Seite zum Vortrag: http://www.cape-it.de

3.50 Modern Perl

Maik Hentsche, Florian Ragwitz
Perl hat sich gegenüber den Versionen, die jeder aus den 90-ern/frühen 2000er Jahren kennt, deutlich weiterentwickelt. Viele der Vorurteile über Perl gelten für modernes Perl nicht mehr. Wir wollen einige moderne Möglichkeiten vorstellen. Dazu gehören Moose für OO, DBIx::Class für Datenbankhandling und Catalyst als Webframework.

WWW-Seite zum Vortrag: http://www.perl.org

3.51 Neues in Kolab 3

Christoph Wickert, Kolab Systems
wickert@kolabsys.com, http://kolabsys.com
Kolab ist eine freie Groupware-Lösung und seit 2003 auf dem Markt. In dieser Zeit hat sich viel getan: Schnelle Internetverbindungen gehören fast überall zum Standard, Smartphones und Tablets werden immer wichtiger und leistungsfähiger, und soziale Netzwerke schicken sich an, die klassische Groupware mit E-Mail, Kalender und Aufgabenliste abzulösen. Was das für Kolab bedeutet und wie die Zukunft einer Groupware aussehen kann, wird Christoph Wickert, Release Manager des Kolab-Servers, in seinem Vortrag erläutern und dabei einen Ausblick auf die kommende Version geben.

WWW-Seite zum Vortrag: http://kolab.org

3.52 Open Source Hardware als Innovation für wandlungsfähige Fabriken

Andreas Merkel, Hendrik Hopf, Egon Müller, Technische Universität Chemnitz
Andreas.Merkel@mb.tu-chemnitz.de, http://www.tu-chemnitz.de/mb/FabrPlan/
Fabriksysteme befinden sich in einem stetigen Wandel. Die Frage ist, ob geschlossen entwickelte Maschinen und Anlagen hierfür noch geeignet sind. Die Schnittstellendokumentation zur Einbindung der Fabrikkomponenten wird beim Kauf mitgeliefert. Die Funktionsweise der Komponenten bleibt hingegen ein Geheimnis des Herstellers. Stellen Sie sich vor, dass auch dieses offenbart wird. Im Beitrag erfahren Sie, wie durch die Anwendung des universal einsetzbaren Open-Source-Mikrocontrollers Arduino der Beginn von offenen Entwicklungen im industriellen Umfeld aussehen kann.

WWW-Seite zum Vortrag: http://arduino.cc/

3.53 Open Source Software Usage and Benchmarking in HPC

Christian Kuelker, ETH Lab
c.kuelker@ethlab.com, http://www.eurotech.com
Parallelism and new ways of programming are at the forefront of new computer architectures these days since achievement through frequency increase seems a dead end. Even laptops have now many cores! But parallelism comes with the price of complexity.
This lecture addresses from a practical perspective achievements and problems in Free Open Source Software usage and benchmarking on HPC clusters and supercomputers. New solutions in HPC today are likely to become standards in everyday usage and programming in the future.

3.54 Open Source und Google

Alexander Schreiber, Google Switzerland GmbH
als@google.com, http://www.google.ch/
Der Vortrag gibt einen Überblick über die Beziehung zwischen Google und Open Source: warum und wie Google Open-Source-Software verwendet und wie Google sich am Open-Source-Prozess beteiligt. Mit dem *Google Summer of Code* unterstützt das Unternehmen Open-Source-Entwicklungen. Am Beispiel von Ganeti, einem Cluster Manager für hochverfügbare virtualisierte Systeme, wird ein Open-Source-Projekt kurz vorgestellt. Ein Überblick über interne Google-Technologie zur Verarbeitung großer Datenmengen schließt den Vortrag ab.

WWW-Seite zum Vortrag: http://code.google.com/opensource/

3.55 Open-Source-Unternehmen führen: Geschäftsmodell, Erfolg, Wachstum

Elmar Geese
info@tarent.de
Aus der Führungssicht eines über 10 Jahre alten, stetig wachsenden Open-Source-Unternehmens berichtet Elmar Geese seine Erfahrungen und die des Unternehmens. Dabei spielen sowohl wirtschaftliche als auch strategische Aspekte, wie die Umsetzung einer Open-Source-Strategie oder auch der Erhalt der Werte im Wachstum, eine Rolle. Dass man mit Open Source Geld verdient, ist inzwischen durch viele Beispiele belegt. Wie das konkret im Unternehmen und im Markt umgesetzt wird, erfährt man jedoch deutlich weniger. Der Vortrag erläutert, welche Ressourcen, Werte, Prozesse und Ziele zu einem erfolgreichen Open-Source-Business gehören.

3.56 openITCOCKPIT: Umbrella Management und Monitoring

Steffen Rieger, it-novum GmbH
ruth.heidingsfelder@it-novum.com, http://www.it-novum.com

openITCOCKPIT ist ein Open-Source-Projekt, das im Juni 2010 gestartet wurde. Als Weiterentwicklung von Nagios ist es das Ergebnis mehrerer Infrastrukturprojekte bei Puma, Seat, dem Bundesversicherungsamt, Deutsche BKK, Fraunhofer Institut und anderen Unternehmen. Der Vortrag stellt openITCOCKPIT vor und hebt die Vorteile gegenüber einer reinen Nagios-Lösung hervor, darunter proaktives Systemmonitoring zur Überwachung komplexer System- und Anwendungslandschaften, SLAs, Eventkorrelation, End-2-End-Messungen, SAP, Business-Process-Monitoring, grafische und webbasierte Oberflächen.

WWW-Seite zum Vortrag: http://www.open-itcockpit.com

3.57 OTRS 3.0: Was bringt mir ein Upgrade?

Shawn Beasley, OTRS AG
shawn.beasley@otrs.com, http://www.otrs.de

Vor kurzem ist OTRS 3.0 auf den Markt gekommen. Dieser Vortrag soll einen Einblick in den Entwicklungsprozess geben, Neuigkeiten der Version 3.0 vorstellen und deren Vorteile beleuchten. Ein Ausblick auf neue Features zukünftiger Versionen schließt den Vortrag ab.

3.58 Part-Time-Scientists – Entwicklung und Bau eines Mondrovers und eines Lunarlanders

Michael Mußler, Part-Time-Scientists Gmbh
mm@part-time-scientists.com, http://www.part-time-scientists.com

Die Part-Time-Scientists sind Teilnehmer am Google Lunar X PRIZE. Wir haben in den letzten zwei Jahren einen Prototyp für einen Mondrover und einen Lander entwickelt. Der Vortrag soll Einblicke in unsere Arbeit und den Stand der Entwicklung geben.

3.59 Perl::Critic – Perl-Code wartbar halten

Renée Bäcker
module@renee-baecker.de, http://www.perl-magazin.de

Perl hat den Ruf, unwartbar zu sein. Einen Ausweg zeigt dieser Vortrag zu Perl::Critic. Gerade in Projektteams sollte man darauf achten, wartbaren Code zu schreiben, so dass ihn auch andere Teammitglieder verstehen. Perl::Critic hilft dabei, gewisse Regeln durchzusetzen. Es wird erläutert, was Perl::Critic ist, wie man damit arbeitet und wie man eigene Regeln erstellt.

3.60 POWERLINK und der RT-Preempt Patch – ein Erfahrungsbericht

Daniel Krüger, SYS TEC electronic GmbH
info@systec-electronic.com, http://www.systec-electronic.com
POWERLINK ist ein Kommunikationssystem zur Echtzeit-Datenübertragung über Ethernet. Der RT-Preempt Patch erlaubt harte Echtzeit-Anwendungen mit einem normalen Linux-Kernel. Unter Nutzung der POWERLINK-Erweiterung «PollResponse Chaining» wurde das Echtzeitverhalten (insbesondere der Zykluszeit-Jitter) des openPOWERLINK Protokollstacks auf einem Linux-System mit RT-Preempt Patch untersucht. Der Vortrag erläutert die notwendigen Anpassungen am Stack und die Untersuchungsergebnisse. Dabei wird kurz auf die Funktionsweise von POWERLINK und der Erweiterung «PollResponse Chaining» eingegangen.

WWW-Seite zum Vortrag: http://openpowerlink.sourceforge.net/

3.61 Professionelle IT-Dokumentation – Anforderungen aus rechtlicher Sicht

Rechtsanwalt Dr. Christian Klostermann
kanzlei@drklostermann.de, http://www.drklostermann.de
Verständliche und nachvollziehbare Dokumentation ist zentraler Bestandteil des IT-Vertrages. Die Rechtsprechung ist hier sehr streng – unzureichende oder fehlerhafte Dokumentation führen direkt in die Haftung oder können den Kunden berechtigen, den Vertrag zu kündigen.
Der Vortrag gewährt einen Überblick über die rechtlichen Aspekte der Dokumentationspflichten in der IT und die Gefahren, die aus unzureichender Dokumentation drohen.

3.62 RAS and Linux

Borislav Petkov, AMD
http://www.amd64.org/
RAS is a collection of hardware features and pertains mostly to server systems. Originally coined by IBM, this term is used to cover all mechanisms which ensure correct program execution, failover behavior and graceful handling of system errors without possibly affecting system uptime. Although Linux has support for a couple of those features, there is still a lot missing for a generic RAS infrastructure in the kernel. This talk describes what Linux has to offer today and also gives a possible future direction into which we want to go for a reasonably transparent system error handling.

3.63 Ressourcenverwaltung in Linux mit Control Groups

Daniel Gollub, B1 Systems GmbH
info@b1-systems.de, http://www.b1-systems.de
Control Groups sind seit 2.6.24 im Linux-Kernel und Bestandteil von aktuellen Linux-Distributionen. Control Groups erlauben es, Prozesse in Gruppen zu fassen und deren Eigenschaften mit Hilfe von Control-Groups-Subsystemen zu manipulieren. Sie stellen eine neue einheitliche Möglichkeit zur Verfügung, Systemressourcen gezielt an Prozesse zu verteilen. Der Vortrag gibt eine Kurzeinführung und erläutert das Buzzword cgroups. Ressourcen-Verwaltung in GNU/Linux wird bei weiter wachsenden Systemressourcen und neuen Funktionen wie NUMA immer wichtiger.

3.64 SAP goes Open Source – Schnittstelle zu Pentaho

Stefan Müller, it-novum GmbH
http://www.it-novum.com
Viele Unternehmen setzen SAP ein, leiden aber unter dem «Blockcharakter» des Systems. Der ITN Connector ERP ist eine Schnittstelle zwischen SAP und der Open Source Business Intelligence Suite Pentaho, der die Kluft zwischen der SAP- und der Open-Source-Welt schließt. Er ist unter der GPLv2 veröffentlicht. Mit dem Konnektor können Daten aus SAP gezogen und in Pentaho Data Integration (früher Kettle) zu Berichten, Analysen etc. weiterverarbeitet werden. Der Vortrag stellt den Konnektor vor, zeigt, wie Daten aus SAP geladen werden können, und wie sich der Konnektor im SAP-Umfeld einsetzen lässt.

WWW-Seite zum Vortrag: http://www.it-novum.com/sap-open-source/

3.65 SPF, DKIM und Greylisting – Was bringen Authentifizierung und Spam-Schutz?

Peer Heinlein, Heinlein Professional Linux Support GmbH
p.heinlein@heinlein-support.de, http://www.heinlein-support.de
Auf den ersten Blick scheinen Verfahren wie SPF oder DKIM zur Verifizierung von E-Mail-Absendern eine gute Sache zu sein, scheint hier doch ein Mittel gegen Phishing- und Spam-Mails zur Verfügung zu stehen. Doch der Teufel steckt im Detail: Weiterleitungen, Webportale und Mailinglisten lösen diverse Schwierigkeiten aus, gleichzeitig können falsche Konfigurationen deutlichen Schaden anrichten.
Was es dabei zu beachten gibt, wo sich SPF und DKIM unterscheiden, welches Verfahren sinnvoll ist oder ob am Ende alles überhaupt funktioniert oder bloß Quatsch ist – das sind Themen dieses Vortrages.

3.66 StegFSTest: Testen von Datenträgern auf versteckte Informationen

Thomas Pucklitzsch, DIAKOMED
http://www.diakomed.de
StegFS ist ein Kernelmodul, welches es ermöglicht, Daten auf einem Datenträger zu verstecken. Es basiert auf EXT2 und benutzt die leeren Bereiche der Festplatte, um darin Informationen einzubetten. Bei StegFSTest handelt es sich um ein Tool, welches mit KDE entwickelt wurde und vermeintliche EXT2-Partitionen analysiert. Im Vortrag wird erläutert, was Steganographie ist, wie StegFS funktioniert und mit welchen Methoden man herausfinden kann, ob Daten versteckt wurden.

3.67 Storage – aber richtig: DRBD als SAN-Ersatz

Martin Gerhard Loschwitz, LINBIT Information Technologies GmbH
martin.loschwitz@linbit.com, http://www.linbit.com/
Kaum eine andere Komponente moderner IT-Systeme lässt Unternehmen die Klauen der großen Anbieter so sehr spüren wie Storages bzw. SANs – der Vendor-Lock-In sorgt für lange Bindung. Dieser Vortrag zeigt, wie Sie der Hersteller-Bindung entgehen und ein offenes und leistungsfähiges SAN schaffen – ausschließlich mit Open-Source-Komponenten. In Kombination mit iSCSI ergeben DRBD und der Clustermanager Pacemaker ein komplettes OSS-SAN. Es genügt Enterprise-Ansprüchen und lässt sich bei Bedarf sogar auf mehrere Standorte verteilen.

3.68 Streifzug durch die Welt der Kommandozeile

Holger Trapp, TU Chemnitz, URZ
hot@hrz.tu-chemnitz.de, http://www.tu-chemnitz.de/urz
Neben modernen Desktop-Umgebungen bieten Linux-Systeme auch weiterhin die klassische Shell mit dem Unix-Werkzeugkasten, die wegen ihrer Mächtigkeit von erfahrenen Anwendern oft verwendet wird, häufig unter der grafischen Oberfläche.
Der Vortrag will mit den Zuhörern einen Streifzug durch diese andere Welt der Computerbenutzung unternehmen und ihnen anhand einiger praktischer Beispiele der täglichen Arbeit ein Gefühl für die Grundprinzipien und sinnvollen Anwendungsgebiete dieser zunächst wohl «kryptisch» anmutenden Befehlsschnittstelle vermitteln, die interaktiv und in Skripten nutzbar ist.

WWW-Seite zum Vortrag: http://www-user.tu-chemnitz.de/~hot/unix_linux_werkzeugkasten/

3.69 Technischer Service mit Open Source bei der Roth & Rau AG

Rico Barth, c.a.p.e. IT GmbH
info@cape-it.de, http://www.cape-it.de
Roth & Rau AG, Zulieferer der Photovoltaik-Industrie, entwickelt und fertigt Produktionsanlagen für die Herstellung von Solarzellen. Der Vortrag betrachtet die Vorfallsbearbeitung im technischen Service der Roth & Rau AG sowie die Unterstützung der Serviceprozesse durch das Service-Management-System OTRS. Neben der Integration des Systems in die Abläufe des Customer Support spricht Stephan Kambor über die Erfahrungen, die er und seine Kollegen mit der spezifischen IT-Unterstützung des technischen Kundenservice gemacht haben. Ergänzend werden die genutzten Optimierungspotentiale erläutert.

3.70 Umdenken! Provokante Thesen zur IT-Administration

Peer Heinlein, Heinlein Professional Linux Support GmbH
p.heinlein@heinlein-support.de, http://www.heinlein-support.de
Höher, schneller, weiter: Wenn es um die Umsetzung von IT-Projekten geht, findet der Wunschzettel oft kein Ende. Doch oft machen IT-Projekte mehr Frust als Lust. Dieser Vortrag bringt Altbekanntes, aber oft Verdrängtes auf den Punkt. Er gibt aber auch neue Ideen und Anregungen, was wirklich wichtig und sinnvoll ist. Er gibt Anstöße zum Nach- und Umdenken.
Vorsicht: Diese Thesen könnten anders lauten als das, was Vertriebsmannschaften etablierter Hersteller normalerweise erzählen.

3.71 Unbekannte, aber nützliche Kommandozeilen-Tools

Axel Beckert, ETH Zürich / Debian
abe@deuxchevaux.org
Es gibt Arbeiten, die immer wieder auftauchen und für die es auch Tools gibt, nur sind diese oft unbekannt. Dieser Vortrag will einige davon vorstellen, die für die Administration von Linux- und BSD-Systemen besonders nützlich sind, wie z.B. Werkzeuge zum automatisierten Umbenennen von Dateien.

WWW-Seite zum Vortrag: http://noone.org/talks/useful-tools/

3.72 Virtualisierung mit KVM

Michel Rode, B1 Systems GmbH
info@b1-systems.de, http://www.b1-systems.de
Dieser Vortrag behandelt die Kernel-based Virtual Machine (KVM), welche seit der Version 2.6.20 Bestandteil des Linux-Kernels ist und für Virtualisierungstechniken eingesetzt wird. Es werden ein Überblick über die Virtualisierungsthematik gegeben und notwendige Tools vorgestellt, die für das Aufsetzen und Verwalten von Gästen notwendig sind. Weiterhin werden Themen wie Migration von virtuellen Maschinen und Hochverfügbarkeit angeschnitten. Grafische Verwaltungstools wie virt-manager erleichtern die Administration.

3.73 VoiP Session Border Controller mit FreeSWITCH

Karsten Horsmann, Seventhsignal Ltd. & Co. KG
karsten.horsmann@seventhsignal.de, http://www.seventhsignal.de/
FreeSWITCH ist eine Open-Source-Alternative zu Asterisk. Mit der industriefreundlichen Mozilla Public License und dem modularen Aufbau erfreut sich das Projekt einer wachsenden Beliebtheit. Es sind alle wichtigen Bestandteile für ein enterprisefähiges SIP-Switch enthalten. Angefangen bei der IPv6-Unterstützung bis hin zu dem mod_sofia genannten SIP Stack von Nokia bietet es zahlreiche Möglichkeiten.

WWW-Seite zum Vortrag: http://www.freeswitch.de

3.74 Vorzüge freier und offener Standards

Michael Stehmann
Eigentlich ist der Begriff «Standard» eindeutig. Dennoch müssen heute die Begriffe «offen» und «frei» hinzugefügt werden, um einer Verwässerung vorzubeugen. Im Vortrag werden nicht nur die Begriffe erläutert, sondern auch die aktuelle «Kampflinie» aufgezeigt. Am Ende werden die Vorzüge freier und offener Standards für den Anwender dargestellt.

3.75 Was Linus kann. . . : Einfache Kernel selbstgemacht

André Przywara, AMD OSRC
http://www.amd64.org
Wer schon immer mal auf den Spuren von Linus Torvalds wandeln wollte, der bekommt hier gezeigt, wie man einen einfachen Kernel «from scratch» schreiben kann. Mit dem normalen gcc und dem Linker kann man einfach und schnell Mini-Kernel bauen, die dank Multiboot direkt von Grub gebootet werden können. Mit Hilfe von QEMU geht das sogar noch komfortabler. Eine simple Bildschirmausgabe und ein printf gibt es auch schon, man kann also direkt mit «Hello, World» loslegen. Von da an gibt es kaum Grenzen, einige Erweiterungen wie Paging werden kurz angerissen.

3.76 Wordpress: Websites für jedermann

Steffen Brose, TU Chemnitz
Steffen.Brose@hrz.tu-chemnitz.de
Wer heute Websites ohne große Spezialkenntnisse erstellen möchte, wird bei der Suche nach einer geeigneten Open-Source-Software irgendwann auf «Wordpress» stoßen. Mit Version 3.0 wurde der Schritt vom reinen Weblog- zum anwenderfreundlichen Content Management System vollzogen. Gerade für Einsteiger ist Wordpress aufgrund der Vielfalt an Vorlagen und Erweiterungen, der einfachen Administration und der großen Verbreitung bestens geeignet. Der Vortrag führt in das Thema ein und zeigt unter anderem eine Anwendung aus der IT-Dokumentation.

3.77 Zarafa Collaboration Platform – Von A wie Archiv bis Z wie Z-Push

Andreas Rösler, Zarafa Deutschland GmbH
info@zarafaserver.de, http://www.zarafaserver.de
In den letzten Jahren hat sich die Groupware Zarafa zu einer Collaboration Platform entwickelt. In diesem Vortrag wird ein Überblick gegeben, wie sich Zarafa in seinem Umfang entwickelt hat, was heute Bestandteil der Zarafa Collaboration Platform ist und wie wir von Zarafa die Zukunft der Collaboration sehen.

WWW-Seite zum Vortrag: http://www.zarafaserver.de

3.78 Zentrales Infrastrukturmanagement in heterogenen Umgebungen mit GOsa²

Christian Patsch, GONICUS GmbH
Christian.Patsch@gonicus.de, http://www.gonicus.de
GOsa² wird erfolgreich in Organisationen eingesetzt, um IT-Infrastruktur effizient zu verwalten. Das Leistungsspektrum erstreckt sich dabei von einer verzeichnisdienstbasierten Administration typischer Benutzer- und Gruppenkonten bis zur Steuerung von System-Rollouts und komplexer Dienste. Durch die Plugin-Architektur ist es möglich, auch speziellen Anforderungen gerecht zu werden. Ausgehend von den Grundlagen eines Verzeichnisdienstes wird die Funktionalität von GOsa² erläutert, unterstützt von Vorführungen an einem Demo-System. Ebenso werden Szenarien aus der Praxis aufgezeigt.

WWW-Seite zum Vortrag: http://www.gosa-project.org

Personen