Fuse

Filesystem in Userspace

PRÄSENTATION VON TIM WELGE

INHALTSVERZEICHNIS

EINFÜHRUNG

Was ist Fuse?

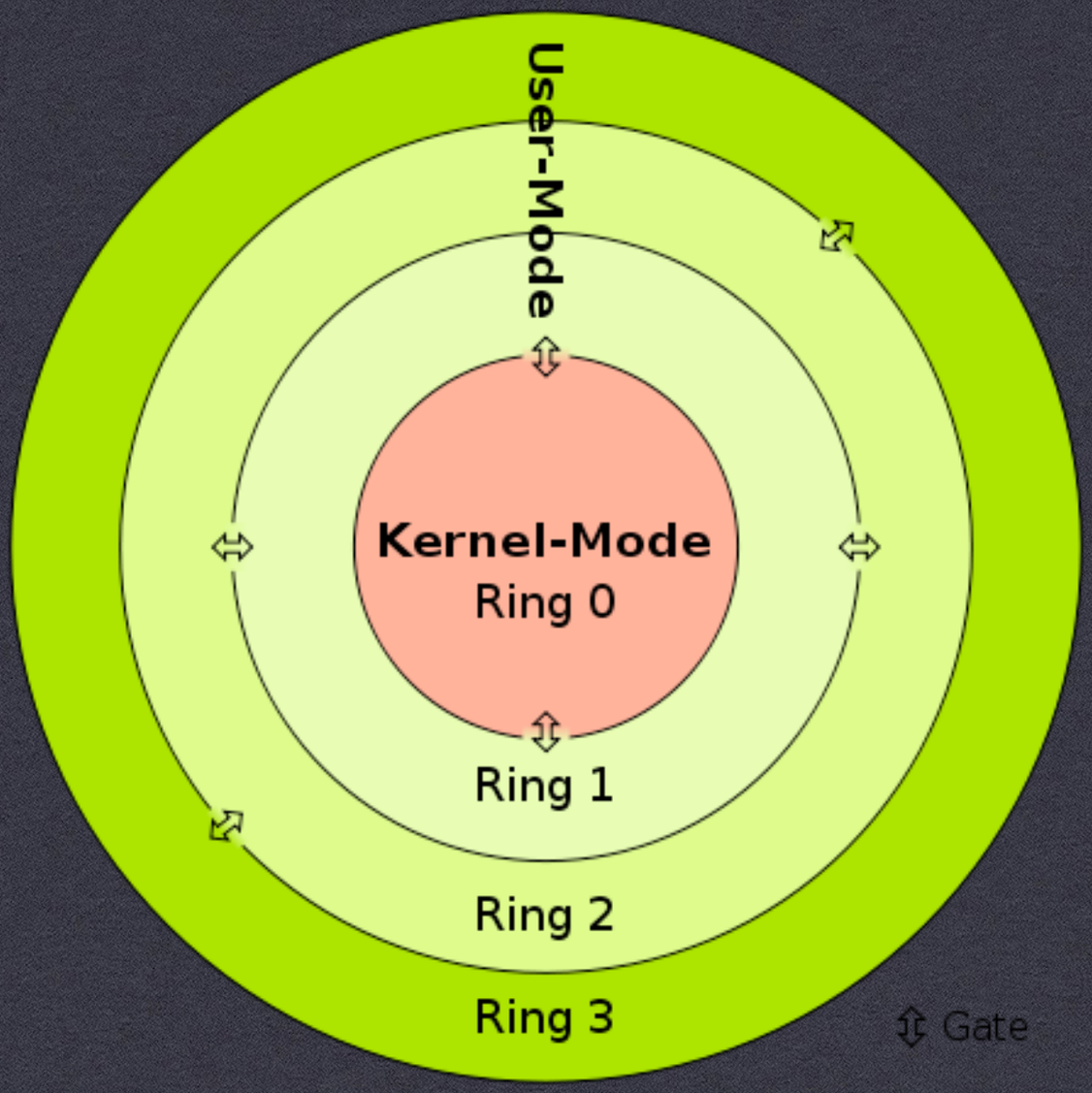
- ist ein Kernel-Modul für UNIX-Systeme(GNU/ Linux, FreeBSD, OpenSolaris, Mac OSX ,Android)
- ermöglicht Dateisystem-Treiber aus dem Kernel-Mode in den User-Mode zu verlagern

WAS IST EIN DATEISYSTEM?

- Das Dateisystem (filesystem) ist eine Ablageorganisation auf einem Datenträger eines Computers
- Dateien können gespeichert, gelesen, verändert oder gelöscht werden
- Dateien haben in einem Dateisystem fast immer mindestens einen Dateinamen sowie Attribute
- Die Dateinamen sind in Verzeichnissen abgelegt; Verzeichnisse sind üblicherweise spezielle Dateien
- Ein Dateisystem bildet somit einen Namensraum
- Alle Dateien über eine eindeutige Adresse (Dateiname inkl. Pfad oder URI) aufrufbar

WAS IST DER USERSPACE?

- Das OS hat zwei Modi: Kernel-und User-Modus
- Kernel: Direkter Resourcenverwaltung
- User: User applications(Firefox,Open Office,…)
- Dateisysteme sind normalerweise Kernel Module



FUSE

- Macht es einfach neue Dateisysteme zu schreiben
- ohne wissen zu müssen wie der Kernel arbeitet
- schneller und einfacher zu implementieren als traditionelle Dateisysteme, welche als Kernel Modul gebaut werden

ANDERE SCHLÜSSEL FUNKTIONEN

- Funktioniert auf allen Unix-Systemen
- Breiter Programmiersprachen support:
  ursprünglich in C,mit Bindungen in C++, Java, C#, Haskell, TCL, Python, Perl, Shell Script, SWIG, OCaml, Pliant, Ruby, Lua, Erlang, PHP

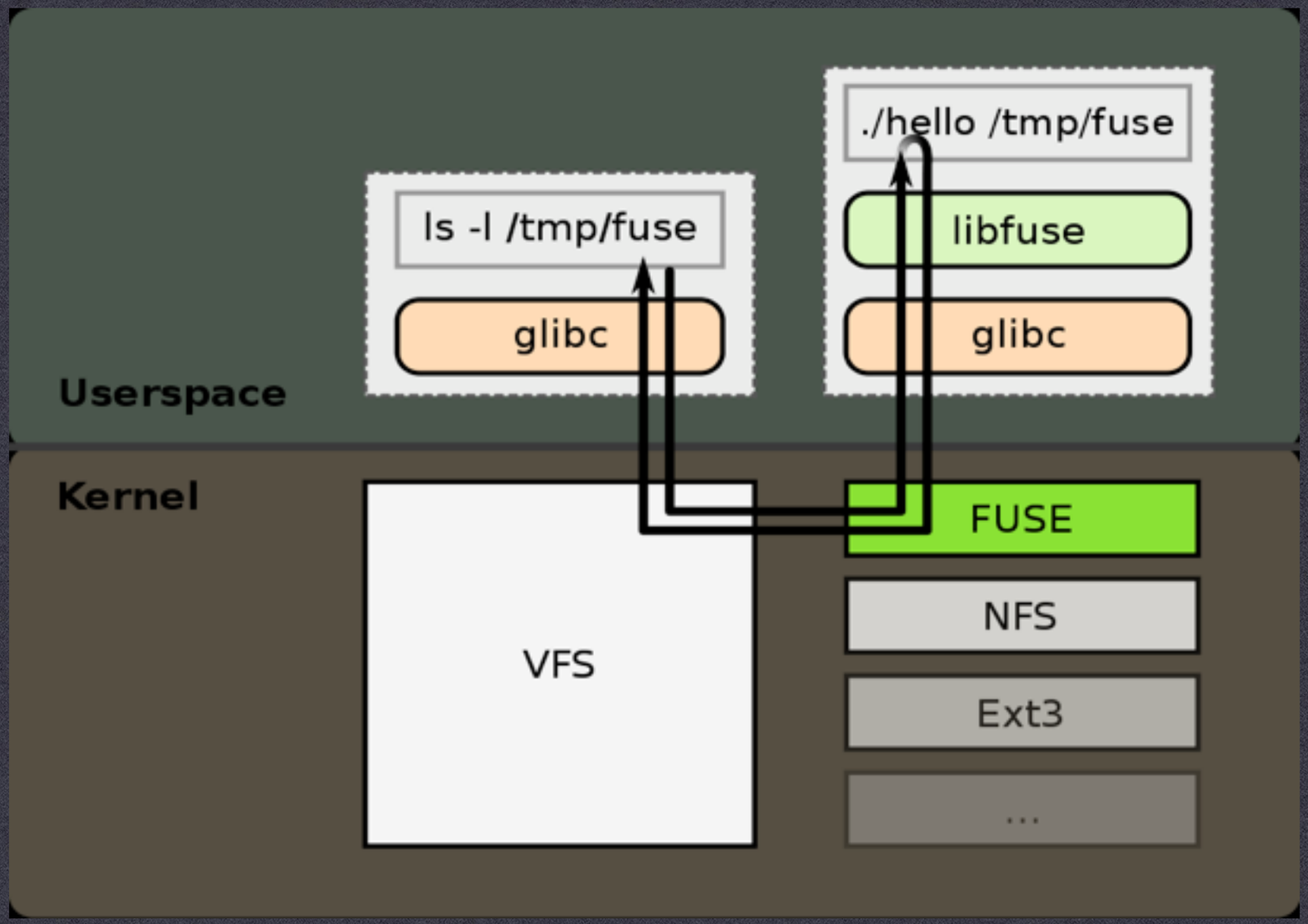
BEISPIELE

- Hardware basiert:ext2, iso, ZFS
- Netzwerk basiert: NFS, smb, SSH
- Andere:Gmail, MySQL

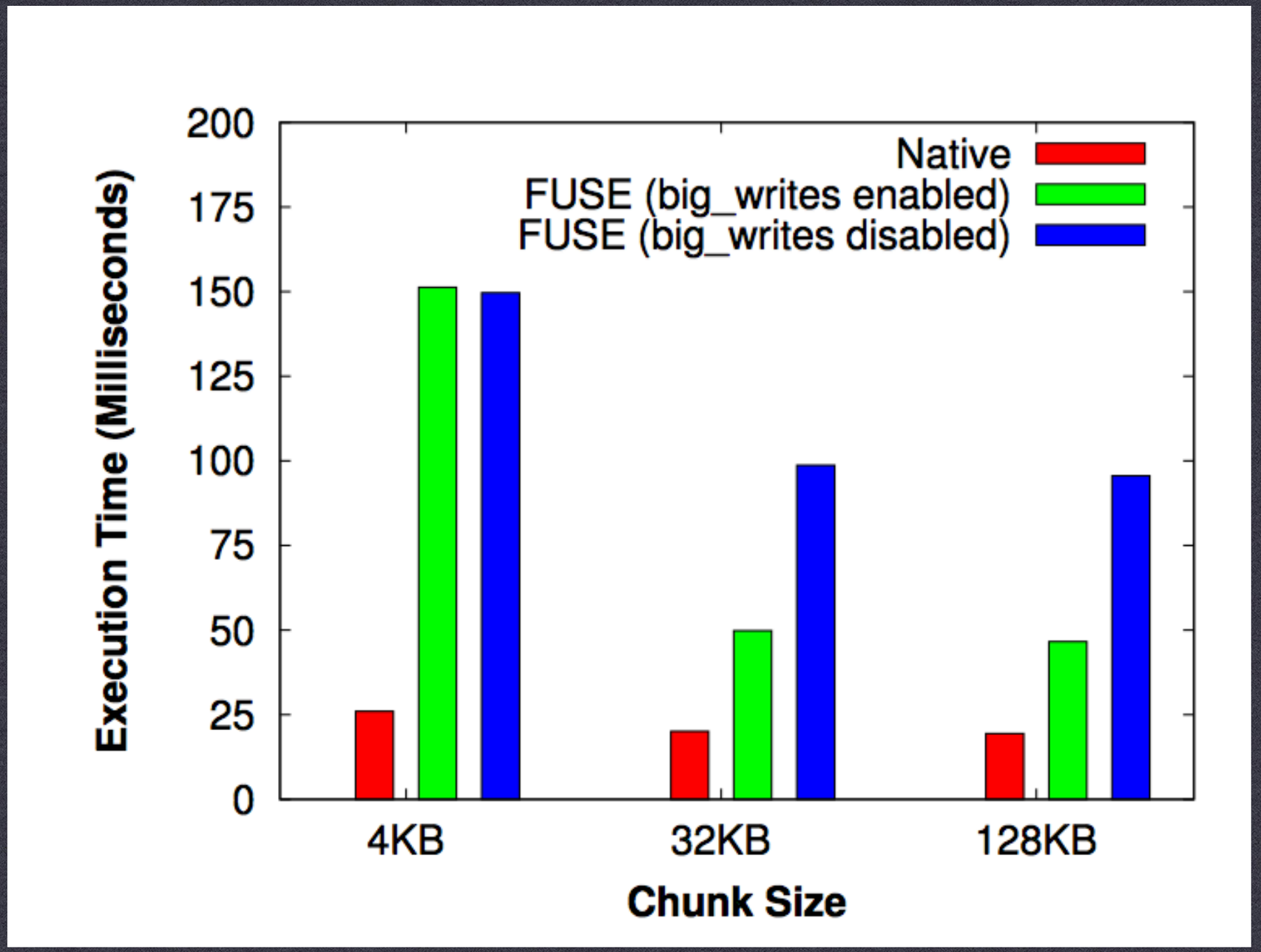
WIE FUNKTIONIERT FUSE

- Eine Anwendung macht einen dateiabhängigen Systemaufruf
- Der Kernel findet heraus, dass die Datei in dem gemounteten FUSE Dateisystem ist
- Das FUSE Kernel Modul leitet den Aufruf weiter in die selbst geschriebene FUSE App im Userspace
- Die FUSE App sagt wie mit der Datei zu verfahren ist

WIE FUNKTIONIERT FUSE



PERFOMANCE VON FUSE



[12]

SCHREIBEN EINES FUSE DATEISYSTEM

- Programmieren einer Anwendung in einer der unterstützen Sprachen
- definieren von Funktionen/Methoden die FUSE aufrufen wird ,um Operationen behandeln
- ca 35 mögliche Operationen
- Für eine Dateisysteme reichen 4 Funktionen/ Methoden,da der Rest gut Defaultwerte besitzt

FUSE OPERATIONEN

- Es werden folgende Operationstypen unterschieden:
  - Verzeichnis-Operationen
  - Datei-Operationen
  - Metadateien-Operationen

VERZEICHNIS-OPERATIONEN

- readdir(path): yield directory entries for each file in the directory
- mkdir(path, mode): create a directory
- rmdir(path): delete an empty directory

GRUNDLEGENDE DATEI-OPERATIONEN

- mknod(path, mode, dev): create a file (or device)
- unlink(path): delete a file
- rename(old, new): move and/or rename a file

GRUNDLEGENDE DATEI-OPERATIONEN

- open(path, flags): open a file
- read(path, length, offset, fh)
- write(path, buf, offset, fh)
- truncate(path, len, fh): cut off at length
- flush(path, fh): one handle is closed
- release(path, fh): file handle is completely closed (no errors)

METADATEI-OPERATIONEN

- getattr(path): read metadata
- chmod(path, mode): alter permissions
- chown(path, uid, gid): alter ownership
- fsinit(self): initialize filesystem state after being mounted start threads, for example

FEHLER IN FUSE

- errno.ENOSYS: Function not implemented errno.EROFS: Read-only file system
- errno.EPERM: Operation not permitted
- errno.EACCES: Permission denied
- errno.ENOENT: No such file or directory
- errno.EIO: I/O error
- errno.EEXIST: File exists
- errno.ENOTDIR: Not a directory
- errno.EISDIR: Is a directory
- errno.ENOTEMPTY: Directory not empty

BEISPIELE

DATEISYSTEM MIT 4 OPERATIONEN

- getAttr()
- readdir()
- open()
- read()

```
static const char *hello_str = "Hello World!\n";
static const char *hello_path = "/hello";
```

GETATTR()

```c
static int hello_getattr(const char *path, struct stat *stbuf)
{
        int res = 0;

        memset(stbuf, 0, sizeof(struct stat));
        if (strcmp(path, "/") == 0) {
                stbuf->st_mode = S_IFDIR | 0755;
                stbuf->st_nlink = 2;
        } else if (strcmp(path, hello_path) == 0) {
                stbuf->st_mode = S_IFREG | 0444;
                stbuf->st_nlink = 1;
                stbuf->st_size = strlen(hello_str);
        } else
                res = -ENOENT;

        return res;
}
```

READDIR()

```c
static int hello_readdir(const char *path, void *buf, fuse_fill_dir_t filler,
                         off_t offset, struct fuse_file_info *fi)
{
        (void) offset;
        (void) fi;

        if (strcmp(path, "/") != 0)
                return -ENOENT;

        filler(buf, ".", NULL, 0);
        filler(buf, "..", NULL, 0);
        filler(buf, hello_path + 1, NULL, 0);

        return 0;
}
```

OPEN()

```c
static int hello_open(const char *path, struct fuse_file_info *fi)
{
        if (strcmp(path, hello_path) != 0)
                return -ENOENT;

        if ((fi->flags & 3) != O_RDONLY)
                return -EACCES;

        return 0;
}
```

READ()

```
static int hello_read(const char *path, char *buf, size_t size, off_t offset,
                      struct fuse_file_info *fi)
{
        size_t len;
        (void) fi;
        if(strcmp(path, hello_path) != 0)
                return -ENOENT;

        len = strlen(hello_str);
        if (offset < len) {
                if (offset + size > len)
                        size = len - offset;
                memcpy(buf, hello_str + offset, size);
        } else
                size = 0;

        return size;
}
```

STRUCT

```
static struct fuse_operations hello_oper = {
        .getattr        = hello_getattr,
        .readdir        = hello_readdir,
        .open           = hello_open,
        .read           = hello_read,
};
```

FAZIT

- Einfach und schnell zu implementieren
- Universell einsetzbar
- Auch aus sicherheitstechnischen Gründen gut, da die User nicht direkt im Kernel arbeiten
- Vielseitig durch Bindings in anderen Sprachen(Python, Java, C#)

QUELLENANGABEN:

QUELLEN

[1] HTTPS://DE.WIKIPEDIA.ORG/WIKI/FILESYSTEM_IN_USERSPACE (AUFGERUFEN AM 28.08.2014)

[2]HTTP://WIKI.UBUNTUUSERS.DE/DATEISYSTEM (AUFGERUFEN AM 28.08.2014)

[3]HTTPS://DE.WIKIPEDIA.ORG/WIKI/DATEISYSTEM(AUFGERUFEN AM 28.08.2014)

[4]HTTPS://DE.WIKIPEDIA.ORG/WIKI/RING_%28CPU%29

[5]NETWORK DISTRIBUTED FILE SYSTEM IN USER SPACE IVAN VORAS, MARIO ŽAGAR FACULTY OF ELECTRICAL ENGINEERING & COMPUTING, UNIVERSITY OF ZAGREB,

[6]FUSE – FILESYSTEMS IN USERSPACE LONDON OPEN SOLARIS USERGROUP, NOV'08 FRANK HOFMANN, OPEN SOLARIS FUSE, PRESENTED BY FRANK HOFMANN.

[7]HTTPS://DE.WIKIPEDIA.ORG/WIKI/ERLANG_%28PROGRAMMIERSPRACHE%29(AUFGERUFEN 03.09.2015)

[8]HTTPS://HACKAGE.HASKELL.ORG/PACKAGE/HALFS (AUFGERUFEN 03.09.2015)

[9]HTTPS://WWW.RUBY-LANG.ORG/DE/ (AUFGERUFEN 03.09.2015)

[10]HTTP://WWW.NETZMAFIA.DE/SKRIPTEN/UNIX/UNIX1.HTML (AUFGERUFEN 03.09.2015)

[11]HTTP://FUSE.SOURCEFORGE.NET/ (AUFGERUFEN 11.09.2015)

[12] HTTP://WWW.CSL.SRI.COM/USERS/GEHANI/PAPERS/SAC-2010.FUSE.PDF(AUFGERUFEN 23.09.2015)