

Rechnernetze

Versuch 3: Versionsneutrale Socket-Programmierung

Ziele:

- Praktisches Erarbeiten der Netzwerkprogrammierung in C
- Ein dialogorientiertes Protokoll ("Request-Response") realisieren

Durchführung:

Implementieren Sie in der Sprache C eine Client-/Server-Anwendung unter Verwendung von TCP, welche folgendes vereinfachte Anwendungsprotokoll im http-Stil interaktiv realisiert (benutzen Sie eine einfache Textdatei als Beispieldaten):

Client Kommando

Server Antwort

„**List**“

<Serverhostname>
<Benutzte Server-IP-Adresse>
<Datum + Uhrzeit>
Datei Listing des Server-Verzeichnis‘

„**Get <dateiname>**“

<Datei-Attribute: last modified, size>
Inhalt von <dateiname>

„**Put <dateiname>**“

„OK <clienthostname>:<clientport>“
nach vollständigem Empfang und Speicherung der Datei

„**Quit**“:

Clientseitige Beendigung der Verbindung.

Hierbei soll dem Client sowohl der **DNS-name**, als auch die **IPv4/6 Adressen** sowie der Port des Servers beim Kommandozeilenaufruf mitgegeben werden können. Die Anwendung soll darüber hinaus **protokollunabhängig** (sowohl für IPv4 als auch IPv6) lauffähig sein.¹

¹ Als praktisches Handbuch sei [Beej's Guide to Network Programming](#) empfohlen

Testen Sie Ihre Anwendung, indem Sie

- (a) Ihre Laborrechner für das Interface `/dev/eth1`, über den Router `RNS1` und für die Protokoll IPv4 und IPv6 kommunikationsfähig machen.
- (b) Einsatztests sowohl über die vorhandenen DNS-Namen, als auch über den unter (a) konfigurierten laborinternen Weg durchführen.

Dokumentieren und erläutern Sie Ihre Tests in einem Protokoll

Hinweise:

Sie können dabei folgendermaßen vorgehen:

1. Starten Sie zunächst mit den "Simple"-Socket-Programmen: [client](#) und [server](#) aus der Vorlesung.
2. Erweitern Sie diese um eine DNS-Namensabfrage mithilfe des Aufrufs `getaddrinfo` (s. MANPAGE):
Dieser Call erzeugt die für den `socket`-Aufruf notwendigen Adressstrukturen in folgender Gestalt: `result` liefert einen Pointer auf eine verkettete Listen von `addrinfo` Adressstrukturen (s. MANPAGE). Mithilfe einer Indirektion liefert `addrinfo` transparenten Zugriff auf die protokollabhängigen `sockaddr*`-Strukturen.
Achtung: Bei der Erkennung der genutzten IP-Version gem. Aufgabenstellung denken Sie bitte daran, dass IPv4-Adressen in IPv6 eingebettet werden. Es genügt deshalb nicht, die Adress-Familie auszulesen. Zur Untersuchung der Einbettung stehen in `in.h` Macros zur Verfügung.
3. Für die Socketerzeugung iterieren Sie über die `result`-Liste und benutzen diejenige Adressstruktur, welche als erste funktioniert.
Achtung: Da es an der HAW offiziell kein IPv6 gibt, enthält der DNS auch keine IPv6-Adressen. Die Laborrechner haben (per Autokonfiguration) auf `/dev/eth1` unique local IPv6 Adressen.
4. Ergänzen Sie nun Ihr Programm für den Zugriff mithilfe von Namen und IP-Adressen aus der Kommandozeile. IP-Adressen können Sie mit `inet_pton` (s. MANPAGE) in die Netzwerkdarstellung, `getaddrinfo` kann mit der Eingabe einer IP-Adresse ebenfalls benutzt werden.
5. Ihre Kommunikationsschnittstelle sollte nun auf der Socket-Ebene fertig sein. Programmieren Sie nun den weiteren Protokollinhalt.

Abgabe:

Wohldokumentierter C-Code und Protokoll der Praxistests.