Computernetze 1

Übung 1

Johannes Morgenroth
morgenro@ibr.cs.tu-bs.de

Institut für Betriebssysteme und Rechnerverbund
Technische Universität Braunschweig

16. April 2009

Überblick

Überblick

Allgemeines

Datenverkehr im Internet

Datenraten

Kapazität und Datenrate

Netztopologien

Socket-Programmierung

Allgemeines

- Kontakt:
  - Johannes Morgenroth
  - morgenro@ibr.cs.tu-bs.de
  - "Sprechstunde": nach Vereinbarung (EMail)
    $\Longrightarrow$unmittelbar vor der Klausur wird es eng...
- Übung ca. 14-tägig
- Neue Aufgabenblätter i.d.R. eine Woche vorher
- Termin: auf Aufgabenblatt und LV-Homepage
- Klausur am 03.08.09, 08:30 Uhr, SN 19.1 und PK 2.1
  - Keine Unterlagen
  - Kein Taschenrechner
  - Den Stoff bitte **nicht unterschätzen**
  - Anmeldung im Fachbereich **und** im Onlinesystem

Material

- Material, aktuelle Informationen und Aufzeichnungen auf
  http://www.ibr.cs.tu-bs.de/courses/ss09/cn1/
  (Login erforderlich)
- Folien der VL+Übung
- Videos/Podcasts der VL, auch als Newsfeed
  - Quasi ein "außergewöhnlicher Service"
  - Sollten die Teilnahme ergänzen, nicht ersetzen!
  - Aufwand für Erstellung nicht unerheblich, Feedback erwünscht
- Aufgabenblätter
- Organisatorisches
  - Termine (auch webcal/ical!)
  - Ausfälle/Änderungen
  - **Anmeldung** und Infos zur Klausur
  - ...

Grundsätzliches zur Übung

- Übung ist eine **interaktive** Veranstaltung!
  - Selber denken macht schlau ;)
  - Wer nicht fragt, bleibt dumm ;)
  - Aus Fehlern lernt man...
- Verschiedene Studiengänge, unterschiedliche Vorkenntnisse
  $\Longrightarrow$**Es gibt keine dummen Fragen!!!**

Datenverkehr im Internet

a) Wie groß ist der Anteil von TCP?

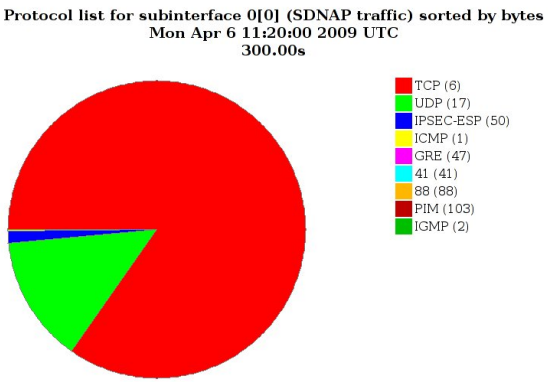
b) Welchen Anteil hat der WWW-Verkehr am Gesamtaufkommen?

c) Welchen haben Peer-to-Peer Filesharing Dienste?

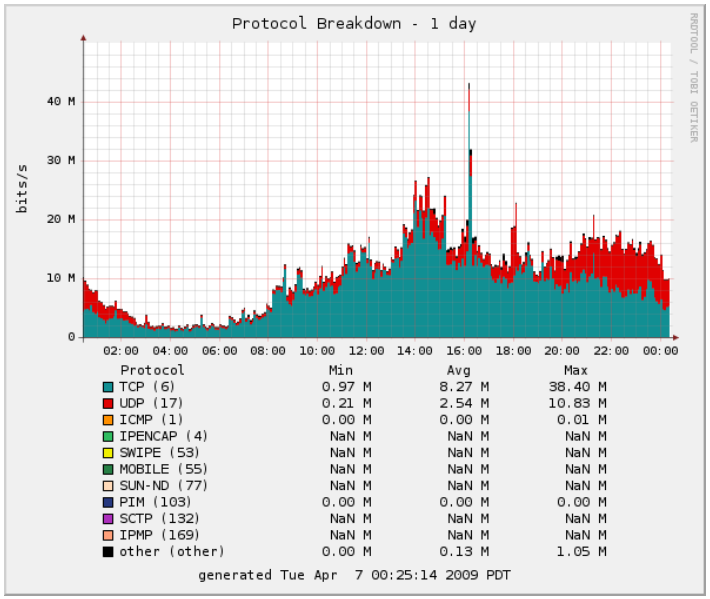
d) Wie hat sich die Menge des Datenverkehrs in den letzten zwei Jahren entwickelt?

Anteil von TCP

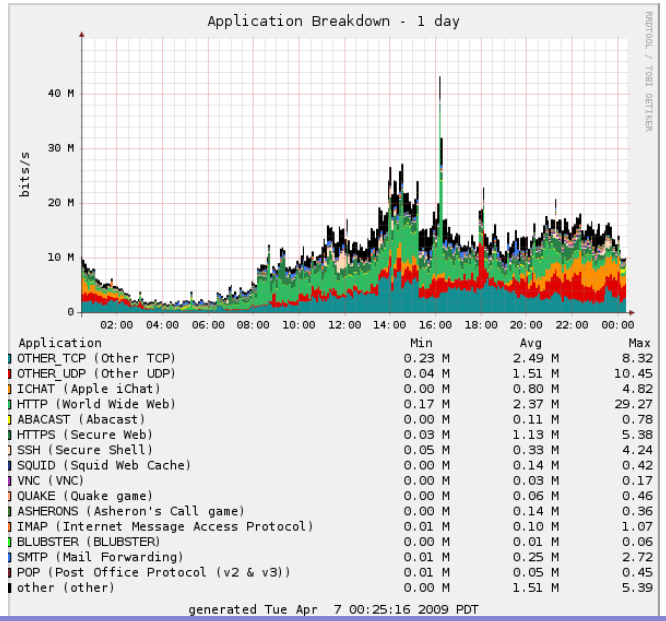
Online Measurement at San Diego Network Access Point:
`http://www.caida.org/dynamic/analysis/workload/sdnap/0_0_/proto_bytes.html`



Anteil von TCP



Anteil von WWW und P2P



Überblick Allgemeines 1)Datenverkehr 2)Datenraten 3)Kapazität 4)Topologien 5)Sockets

Anteil von WWW und P2P

- Ipoque Internet Study 2008/2009:
  - P2P: 52,79%
  - WWW: 25,78%

`http://portal.ipoque.com/downloads/index/get/?id=265`

Überblick Allgemeines 1)Datenverkehr 2)Datenraten 3)Kapazität 4)Topologien 5)Sockets

P2P: Protokolle

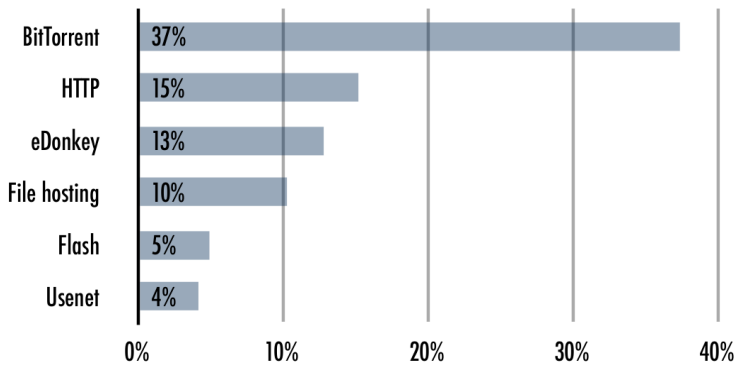
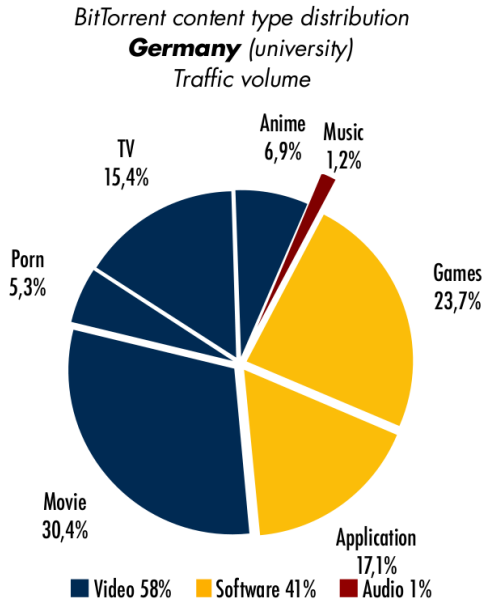


Abbildung: Verwendete P2P Protokolle in Deutschland

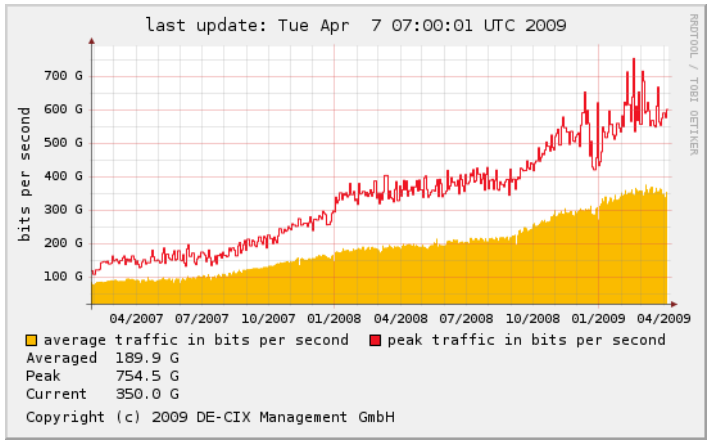
Überblick Allgemeines 1)Datenverkehr 2)Datenraten 3)Kapazität 4)Topologien 5)Sockets

P2P: Content



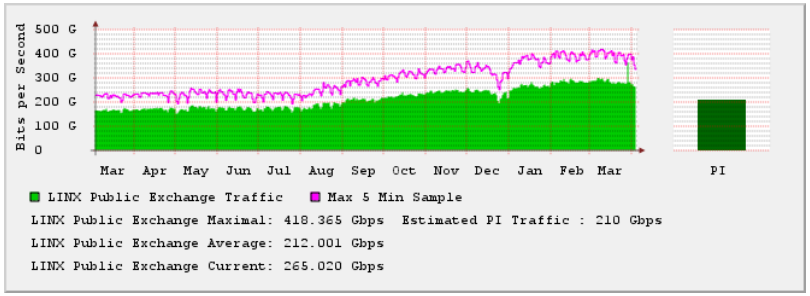
Entwicklung des Datenverkehrs: DE-CIX

Deutscher Commercial Internet Exchange (DE-CIX)
`http://www.decix.de`

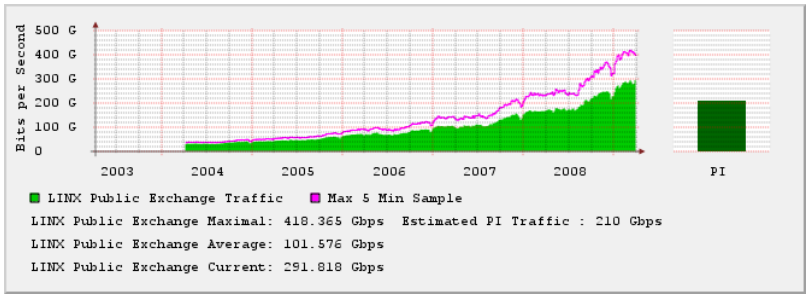


Entwicklung des Datenverkehrs: LINX

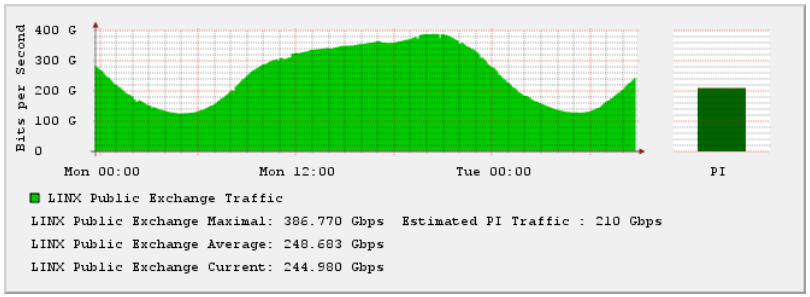
Zum Vergleich: London Internet Exchange (LINX)
`http://www.linx.net`



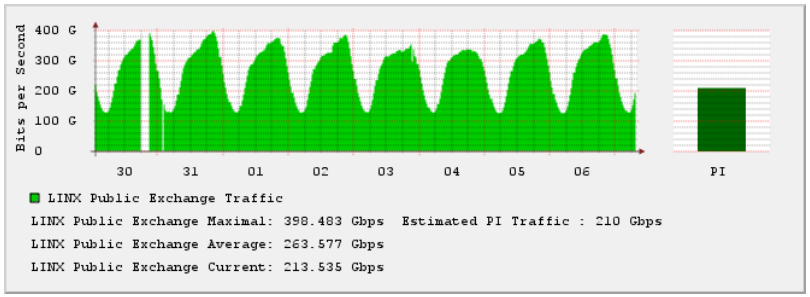
Entwicklung des Datenverkehrs: LINX über Jahre



Entwicklung des Datenverkehrs: LINX über Tage



Entwicklung des Datenverkehrs: LINX über Wochen



Aufgabe 2: Datenraten

Ein LKW transportiert 200 Magnetbänder à 250 GByte ($1 GByte = 10^9 Byte$) mit 80 km/h zwischen Braunschweig und Wolfsburg (Entfernung: 25 km).

a) Welche Datenrate erreicht der LKW bei einer Fahrt von Braunschweig nach Wolfsburg?

b) Wie lange dauert die Übertragung derselben Datenmenge über ein Glasfasernetz mit 622 Mbit/s?

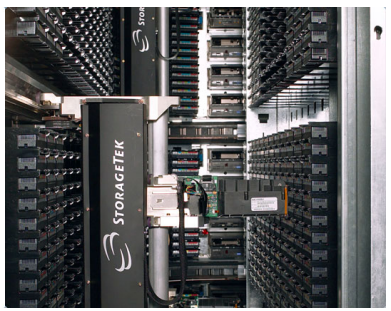
c) Welche Vor- bzw. Nachteile haben diese beiden Übertragungsarten jeweils?

Aufgabe 2: Ziele und Anmerkungen

- Erste Reaktion ist oft: "Was für eine alberne Aufgabe!"
- Ziel:
  - Anschauliche Darstellung von Datenmengen und Datenraten
  - Unterschied Übertragungs- und Ausbreitungsverzögerung verdeutlichen
- Übrigens: Tapes sind auch heute noch ein gängiges Speichermedium für große Datenmengen...

Beispiel "Magnetbänder"

- Sun "StorageTek SL8500"
  - Bis zu 56 Petabytes (entpr. 112.000 500GB-Festplatten) auf 70.000 Tapes
  - Eindrucksvolles Video: `http://www.sun.com/storagetek/tape_storage/tape_libraries/sl8500/gallery/index.xml?p=1&s=3`



Aufgabe 2

a) Welche Datenrate erreicht der LKW bei einer Fahrt von Braunschweig nach Wolfsburg?

Datenmenge:
$200 \cdot 250 \cdot 10^9 Byte = 50000 \cdot 10^9 Byte = 400000 \cdot 10^9 bit = 400 \cdot 10^{12} bit = 400 Terabit$

Fahrzeit:
$v = \frac{s}{t} \Leftrightarrow t = \frac{s}{v} = \frac{25km}{80\frac{km}{h}} = 1125s$

Datenrate:
$\frac{400 \cdot 10^{12} bit}{1125s} = 355 \cdot 10^9 \frac{bit}{s} = 355 Gbps$

Aufgabe 2

b) Wie lange dauert die Übertragung derselben Datenmenge über ein Glasfasernetz mit 622 Mbit/s?

$\frac{400 Terabit}{622\frac{Mbit}{s}} = \frac{400 \cdot 10^6}{622} s = 643000s = 7,442 Tage \approx 1 Woche$

Hinweis:
622 Mbit/s = "OC-12-Link" =
Optical Carrier Level 12 = 12facher OC-1 Link (ca. 52Mbit/s)
Bezeichnung stammt aus dem Synchronous Optical Network (SONET) Standard

Aufgabe 2

c) Welche Vor- bzw. Nachteile haben diese beiden Übertragungsarten jeweils?

Aufgabe 3

Ein Transatlantikkabel verbindet Europa mit den USA. Es ist 6000 km lang und hat eine Kapazität von 9.6 Gbit/s (1 Gbit=$10^9 bit$).

a) Wieviele Daten sind im Glasfasernetz gespeichert? Die Signalausbreitungsgeschwindigkeit sei $2 \cdot 10^8$ m/s.

b) Wieviele DVDs pro Stunde bräuchte man, um alle gesendeten Daten zu protokollieren?
Die Kapazität einer DVD sei 4 GByte.

Ausbreitungs- und Übertragungsverzögerung

- Demo-Applet "Transmission vs. Propagation Delay":
  `http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/transmission/delay.html`

- Wichtig zum Verständnis:
  In High-Speed-Datennetzen sind die Bits nicht schneller, sondern kürzer!

Aufgabe 3

a) Wieviele Daten sind im Glasfasernetz gespeichert? Die Signalausbreitungsgeschwindigkeit sei $2 \cdot 10^8$ m/s.

s = 6000 km, v = $2 \cdot 10^8$ m/s

Signalausbreitungsverzögerung:
$t = \frac{s}{v} = \frac{6 \cdot 10^6 m}{200 \cdot 10^6 \frac{m}{s}} = 30 ms$

Speicherkapazität der Leitung:
$9,6 Gbit/s \Rightarrow 9,6 \cdot 10^9 \frac{bit}{s} \cdot 30 \cdot 10^{-3} s = 9,6 \cdot 10^6 \cdot 30\, bit = 288\, Mbit$

Aufgabe 3

b) Wieviele DVDs pro Stunde bräuchte man, um alle gesendeten Daten zu protokollieren?
Die Kapazität einer DVD sei 4 GByte.

$9,6 Gbit/s \Rightarrow \frac{9,6}{8} \cdot 10^9 \frac{Byte}{s} \Rightarrow \frac{9,6}{8} \cdot 10^9 \cdot 3600 \frac{Byte}{h}$

$\frac{9,6}{8} \cdot 10^9 \cdot 3600 \frac{Byte}{h} \cdot 1h = 4320 \cdot 10^9 Byte = 4320\, GByte$

$\Rightarrow 1080 DVDs$

Aufgabe 4: Netztopologien

Nennen Sie die Vor- und Nachteile der folgenden Netztopologien:

- Bus
- Ring
- Vollständige Vermaschung
- Stern

Aufgabe 4

	Ausfallsicherheit	Kabellänge	Kapazität	Wegewahl
Bus				
Ring				
Vermaschung				
Stern				

Aufgabe 4 a)

	Ausfallsicherheit	Kabellänge	Kapazität	Wegewahl
Bus	o	++	- -	+ (nicht nötig)
Ring				
Vermaschung				
Stern				

Aufgabe 4 b)

	Ausfallsicherheit	Kabellänge	Kapazität	Wegewahl
Bus	o	++	- -	+ (nicht nötig)
Ring	-	+	-	+ (nicht nötig)
Vermaschung				
Stern				

Aufgabe 4 c)

	Ausfallsicherheit	Kabellänge	Kapazität	Wegewahl
Bus	o	++	- -	+ (nicht nötig)
Ring	-	+	-	+ (nicht nötig)
Vermaschung	++	- -	++	+ (direkt)
Stern				

Aufgabe 4 d)

	Ausfallsicherheit	Kabellänge	Kapazität	Wegewahl
Bus	o	++	- -	+ (nicht nötig)
Ring	-	+	-	+ (nicht nötig)
Vermaschung	++	- -	++	+ (direkt)
Stern	o	-	+	+ (im Switch)

Aufgabe 7: Sender, verbindungslos (1/2)

```
/**
* hello_udp_sender.c (for IPv4)
* 04/11/2004, marc
* syntax:  hello_udp_sender <dest_address> <dest_port>
* NO ERROR HANDLING HERE !!!
*/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <net/if.h>
#include <string.h>

/// The Main Function
int main (int argc, char **argv)
{
   int sock, len;
   struct addrinfo *remote_ai;
   struct addrinfo hints; // specifies the necessary parameters
   char *message="Hello World!";
   char toHost[2048], toPort[2048];

   /// some basic definitions, might be overwritten later on
   if (argc != 3) {
      printf("Call Me \"hello_udp_sender <dest_address> <dest_port>\"\n");
      exit(0);
   }
```

Sender, verbindungslos (2/2)

```
   /// Initialization
   memset(&hints, 0, sizeof(struct addrinfo));
   hints.ai_socktype = SOCK_DGRAM; // specify UDP socket
   hints.ai_family = PF_INET; // Use IPv4
   strcpy(toHost, argv[1]); // Remote Host
   strcpy(toPort, argv[2]); // Remote Port
   getaddrinfo(toHost, toPort, &hints, &remote_ai); // initialize remote host

   /// Open the socket
   sock = socket(remote_ai->ai_family, remote_ai->ai_socktype, remote_ai->ai_protocol);

   /// Send the Message
   len = sendto(sock, message, strlen(message), 0, remote_ai->ai_addr, remote_ai->ai_addrlen);
   printf ("Sent \"%s\" of length %i, %i transmitted bytes\n", message, strlen(message), len);

   /// Free the resources
   close(sock);
   freeaddrinfo(remote_ai);
  exit(0);
}
```

Empfänger, verbindungslos (1/2)

```
/**
* hello_udp_receiver.c
* 04/11/2004, marc
* syntax:  hello_udp_receiver <ip address> <port>
* NO ERROR HANDLING HERE !!!
*/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <asm/msr.h>
#include <net/if.h>
#include <string.h>

/// The Main Function
int main (int argc, char **argv)
{
   struct addrinfo *local_ai;
   struct addrinfo hints; // specifies the necessary parameters
   int sock, len;
   char buf[2048]; // MTU for UDP
   char myPort[2048], myHost[2048];

   if (argc != 3) {
      printf("Call Me \"hello_udp_receiver <ip address> <port>\"\n");
      exit(0);
   }
```

Empfänger, verbindungslos (2/2)

```
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_socktype = SOCK_DGRAM; // specify UDP socket
    hints.ai_family = PF_INET; // specify IPv4 only
    strcpy(myHost, argv[1]);
    strcpy(myPort, argv[2]);
    getaddrinfo(myHost, myPort, &hints, &local_ai); // initialize local interface

    /// Open the socket
    sock = socket(local_ai->ai_family, local_ai->ai_socktype, local_ai->ai_protocol);

    /// Bind to the socket -> Enables transmission to specified port
    bind(sock, local_ai->ai_addr, local_ai->ai_addrlen);

    /// Receive the message
    len = recvfrom(sock, (void *)buf, 2048, 0, NULL, NULL);
    buf[len]=?\0?; // End of string is not transmitted!
    printf("Received:  %s\nwith a length of %i bytes\n", buf, len);

    /// Clean up
    close(sock);
    freeaddrinfo(local_ai);
    exit(0);
}
```

Sender, verbindungsorientiert (1/2)

```
/**
* hello_tcp_sender.c (for IPv4)
* 04/11/2004, marc
* syntax:  hello_tcp_sender <dest_address> <dest_port>
* NO ERROR HANDLING HERE !!!
*/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <net/if.h>
#include <string.h>

/// The Main Function
int main (int argc, char **argv)
{
   int sock, len;
   struct addrinfo *remote_ai;
   struct addrinfo hints; // specifies the necessary parameters
   char *message="Hello World!";
   char toHost[2048], toPort[2048];

   /// some basic definitions, might be overwritten later on
   if (argc != 3) {
      printf("Call Me \"hello_tcp_sender <dest_address> <dest_port>\"\n");
      exit(0);
   }
```

Sender, verbindungsorientiert (2/2)

```
    /// Initialization
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_socktype = SOCK_STREAM; // specify TCP socket
    hints.ai_family = PF_INET; // Use IPv4
    strcpy(toHost, argv[1]); // Remote Host
    strcpy(toPort, argv[2]); // Remote Port
    getaddrinfo(toHost, toPort, &hints, &remote_ai); // initialize remote host

    /// Open the socket
    sock = socket(remote_ai->ai_family, remote_ai->ai_socktype, remote_ai->ai_protocol);

    /// Connect...  does the binding for us, too :-)
    connect(sock, remote_ai->ai_addr, remote_ai->ai_addrlen);

    /// Send the Message
    len = send(sock, message, strlen(message), 0);
    printf ("Sent \"%s\" of length %i, %i transmitted bytes\n", message, strlen(message), len);

    /// Free the resources
    close(sock);
    freeaddrinfo(remote_ai);
    exit(0);
}
```

Empfänger, verbindungsorientiert (1/3)

```
/**
* hello_tcp_receiver.c
* 04/11/2004, marc
* syntax:  hello_tcp_receiver <ip address> <port>
* NO ERROR HANDLING HERE !!!
*/

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <asm/msr.h>
#include <net/if.h>
#include <string.h>

/// The Main Function
int main (int argc, char **argv)
{
   struct addrinfo *local_ai;
   struct addrinfo hints; // specifies the necessary parameters
   int sock, sock2, len;
   char buf[2048]; // MTU for UDP
   char myPort[2048], myHost[2048];
   if (argc != 3) {
      printf("Call Me \"hello_tcp_receiver <ip address> <port>\"\n");
      exit(0);
   }
```

Empfänger, verbindungsorientiert (2/3)

```
memset(&hints, 0, sizeof(struct addrinfo));
hints.ai_socktype = SOCK_STREAM; // specify TCP socket
hints.ai_family = PF_INET; // specify IPv4 only
strcpy(myHost, argv[1]);
strcpy(myPort, argv[2]);
getaddrinfo(myHost, myPort, &hints, &local_ai); // initialize local interface

/// Open the socket
sock = socket(local_ai->ai_family, local_ai->ai_socktype, local_ai->ai_protocol);

/// Bind to the socket -> Enables transmission to specified port
bind(sock, local_ai->ai_addr, local_ai->ai_addrlen);
printf("Binding to the specified socket...\n");

/// Listen for up to 1 incoming connection call...
listen(sock, 1);
printf("Waiting for incoming connections...\n");

/// Accept an incoming connection
sock2 = accept(sock, NULL, NULL);
printf("Accepted incoming connection...\n");

/// Receive the message
len = recv(sock2, (void *)buf, 2048, 0);
buf[len]='\0'; // End of string is not transmitted!
printf("Received:  %s\nwith a length of %i bytes\n", buf, len);
```

Empfänger, verbindungsorientiert (3/3)

```
    /// Clean up
    close(sock2);
    close(sock);
    freeaddrinfo(local_ai);
    exit(0);
}
```