Auszug aus der Benutzerinformation des Rechenzentrums der Universität Stuttgart, März 1993

DFN-RPC für verteilte Anwendungen

Das DFN Remote Procedure Call Tool (DFN-RPC) ermöglicht es Ingenieuren und Naturwissenschaftlern, eigene Anwendungsprogramme zwischen ihrer Workstation und entfernten Höchstleistungsrechnern unter User-Account verteilt auszuführen. Der DFN-RPC wurde im Auftrag des Deutschen Forschungsnetzes (DFN-Verein) vom Rechenzentrum der Universität Stuttgart entwickelt.

Schnelle Simulationsberechnungen auf dem Compute Server, graphische Visualisierung und gleichzeitige Interaktionen auf der Workstation vor Ort ermöglichen einen neuen Arbeitsstil und können die Entwicklungs- und Forschungszyklen verkürzen.
Das Projekt wurde in enger Kooperation mit vier Pilotanwendern, dem Alfred-Wegener-Institut, Bremerhaven (AWI), dem Institut für Kernenergetik und Energiesysteme, Stuttgart (IKE), dem Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Stuttgart (ISD) und dem Lehr- und Forschungsbereich Theoretische Astrophysik, Tübingen (TAT) durchgeführt und vom BMFT unter TK 558 VA 005.1 gefördert.

Das RPC-Prinzip

Remote Procedure Calls sind ein mächtiges, aber dennoch einfaches Mittel, um Programme zwischen Rechnern zu verteilen.

Der Anwender teilt sein Anwendungsprogramm in zwei oder mehr Teile, das Hauptprogramm (Client Part) und ein oder mehrere Moduln aus Unterprogrammen (Server Part). Der Client ruft Unterprogramme im Server mittels Remote Procedure Call (RPC). Dabei sind Client und Server verschiedene Prozesse auf verschiedenen Rechnern und Rechnertypen.

Client / Server

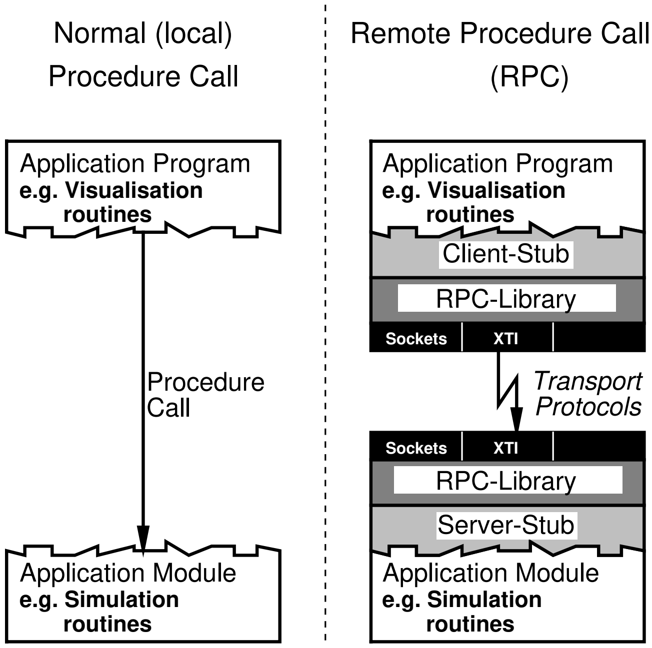

Abb.1: Das RPC-Prinzip

Der DFN-RPC realisiert die entfernten Unterprogramm-Aufrufe (Remote Procedure Calls) so, daß sie für das Anwendungsprogramm genau wie bei lokalen Aufrufen aussehen. Wenn der Client Part via RPC eine Routine xyz im Server Part rufen möchte, dann ruft er in Wirklichkeit eine gleichnamige Routine im Client Stub. Der Stub (dt. Stummel) schickt ein Datenpaket, bestehend aus einer Kennung für die Routine xyz und den Input-Parametern zum Server.

Der *Server Stub* empfängt dieses Datenpaket, allokiert Memory für die Parameterliste und schreibt die Input-Parameter dort hinein. Dann ruft er die echte Routine xyz im Server Part auf. Gegebenenfalls werden Output-Parameter auf die gleiche Weise anschließend zurückübertragen.

Datenkonvertierung

In die Stubs ist auch die Datenkonvertierung eingebettet. Die vom DFN-RPC unterstützten Rechner haben zwei Integer-Datenformate (big- und little endian, jeweils 2er Komplement) und 5 Real-Datenformate (IEEE big endian, DEC (=IEEE little endian), VAX, CRAY, CONVEX). Beim DFN-RPC werden die Daten beim Transfer (quasi on-the-fly) automatisch konvertiert. Die Zahlendarstellung, die für die Übertragung zwischen Client und Server benutzt wird, die sog. Transferkodierung, bestimmt, in welchem der Stubs die Zahlen umgewandelt werden. Sie kann vom Anwender als Compile-Option frei gewählt werden. Auf Vektorrechnern wird die Konversion weitgehend vektorisiert durchgeführt.

Die Übertragung der Datenpakete übernimmt die *RPC-Library*. Diese enthält eine verallgemeinerte Schnittstelle zu den verschiedenen Transport-Dienst-Schnittstellen und Protokollen.

Stub-Generator

Interface Definition

Die Stubs sind zwar anwendungsspezifisch, können aber mit dem Stub-Generator automatisch generiert werden. Die Ausgangsbasis hierzu ist eine Remote Procedure Interface Definition (RPI). Darin wird im wesentlichen die Parameterliste der mittels RPC zu exportierenden Routinen des Server Parts beschrieben. Durch ergänzende optionale Angaben kann eine beträchtliche Effizienzsteigerung im Sinne der Anwendung erreicht werden.

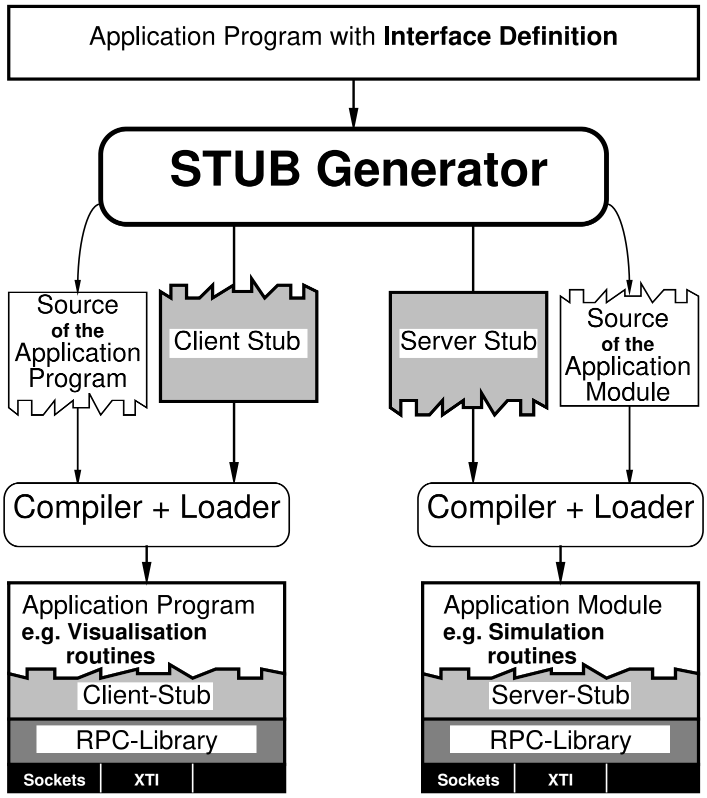

Abb. 2: Stub-Generierung

```
C*[RPI_BEG]
       SUBROUTINE example (n,a,sq)
       INTEGER n [IN]
       REAL a(*) [IN, MAXINDEX=n]
       REAL sq(*) [OUT,MAXINDEX=n]
C*[RPI_NON]
       DO 1, i=1,n
1      sq(i) = SQRT(a(i))
       END
C*[RPI_END]
```

Abb. 3: Beispiel einer RPI

Task-Start

Da verteilte Anwendungen aus mindestens zwei Prozessen bestehen, die normalerweise auf unterschiedlichen Rechnern laufen sollen, ist der Start der Anwendung natürlich komplizierter als bei einer unverteilten Anwendung. Da der DFN-RPC für den Einsatz auf zentralen Compute-Servern entwickelt wurde, ermöglicht er den Start der Server-Prozesse unter User-Account. Er setzt hierzu auf vorhandenen Diensten (z.B. nqs, rsh, telnet) auf.

Beispiel-Anwendungen

Um die Einarbeitung in das RPC-Tool zu erleichtern und um Verteilungsmöglichkeiten aufzuzeigen, wurden beispielhafte Anwendungen entwickelt. Ein verteiltes Benchmarkprogramm und eine Testsuite werden ebenfalls mitgeliefert.

Szenarien

Abb. 4 zeigt typische Szenarien, wie Anwendungen im Bereich der wissenschaftlich-technischen Simulation und Visualisierung zwischen Workstation und Compute Server verteilt werden können.
Das Programm ist in einen Compute-Teil, einen Visualisierungsteil und eine Graphics Library aufgeteilt. Bei verteilter Simulation und Visualisierung, wie in Abb. 4, enthält der Compute-Teil die Simulation. Bei verteilter Visualisierung enthält der Compute-Teil rechenintensive Routinen der Visualisierung.

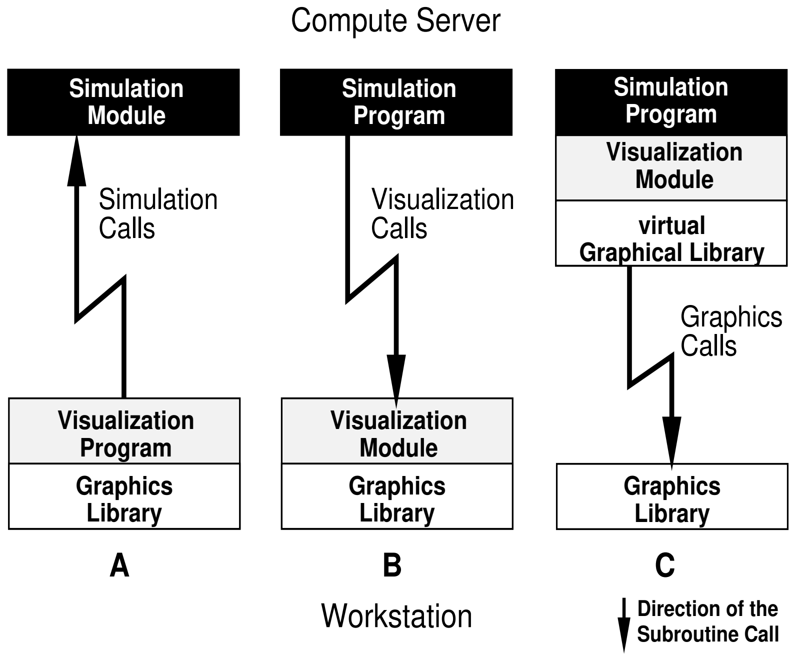

Abb. 4: Szenarien

Verteilte Simulation

Im Fall A ist das Hauptprogramm auf der Workstation, d.h. sie ist der Client, und macht Remote Procedure Calls zum Compute-Teil auf dem Compute Server.
Abb. 5 zeigt hierzu als Beispiel eine verteilte Visualisierung. Die Integration der Strom- und Streichlinien zur Darstellung einer Hohlraum-Überströmung wird auf dem Compute Server auf der Basis eines 30 Mbyte großen zeitabhängigen Strömungsfeldes

berechnet. Diese Integrations-Rechnungen wurden auf den Compute Server ausgelagert, da die Visualisierung animiert stattfinden muß und daher trotz der 30 MByte kein Paging stattfinden darf.

Interaktionen

Der Wissenschaftler an der Workstation kann nun interaktiv Farbsonden setzen und die Strömungssimulation mit den von ihnen ausgehenden Streichlinien untersuchen. Außerdem kann er über ein Menü verschiedene Darstellungsformen wählen, z.B. kann er ein „Sägemehl“ einstreuen und die resultierenden Bahnlinien sofort verfolgen. Er kann also graphisch-interaktiv, unter Einbeziehung eines Supercomputers, die Ergebnisse der Strömungsberechnung untersuchen.

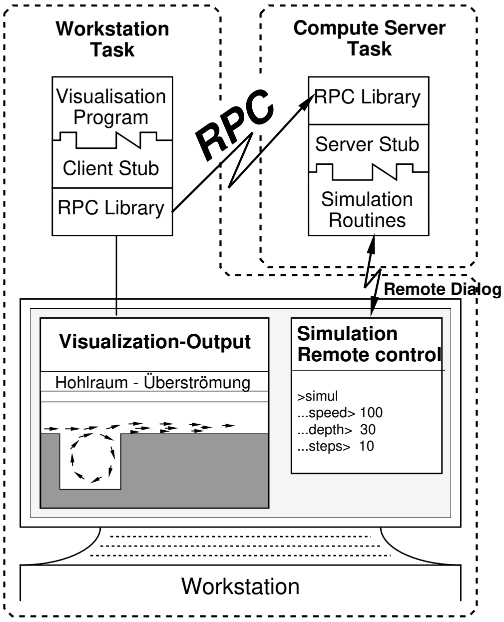

Abb. 5: Beispiel zu Szenarium A

Verteilte Visualisierung

Im Fall B liegt das Hauptprogramm auf dem Compute Server, d.h. er ist nun der Client, und die Workstation übernimmt die Rolle des Servers (bzgl. der RPCs).

Mit dem DFN-RPC können derartige Anwendungen mit geringem Aufwand verteilt werden. Beim Programm-Entwurf sollte man darauf achten, daß das Interface, an dem das Programm verteilt wird, so gewählt wird, daß ein Minimum an Daten und RPCs übers Netz geht.

Verteilte Graphik

Im Fall C wird das komplette Anwendungsprogramm auf dem Compute Server ausgeführt. Nur die Graphics Library befindet sich auf der Workstation. In den meisten Anwendungsfällen wird man hier X Window, PEX (PHIGS in X Window) oder DGL (Distributed (Iris) Graphics Language) verwenden. Wenn man aber eine beliebige Graphics Library einsetzen will, kann man dies mit dem DFN-RPC realisieren. Der DFN-RPC enthält hierfür spezielle Pufferungsmechanismen. Der Einsatz einer verteilten Graphics Library ist vor allem für den Anwendungsprogrammierer sehr komfortabel, da er quasi nur die Graphics Library durch die Remote Graphics Library ersetzen muß.

Beim Einsatz des DFN-RPCs kann der Anwender frei entscheiden, an welchen Schnittstellen er seine Anwendung auf zwei oder mehrere Prozesse aufteilen möchte.

Pilot-Anwendungen

Darüber hinaus werden mit dem DFN-RPC verteilte Anwendungen für aktuelle Forschungsvorhaben eingesetzt.

Die vier Pilotpartner stellten auf dem 10. HDN-Arbeitskreis (Hochgeschwindigkeitsdatenkommunikation im DFN) Forschungsprojekte vor, bei denen sie den DFN-RPC einsetzten.

Das AWI entwickelte ein Remote File Access Tool (RFA) zum netzwerkweiten Zugriff auf Binärdateien. Es wird zum Datentransport inklusive Konversion zwischen Supercomputern des DKRZ (Hamburg) und des AWI (Bremerhaven) eingesetzt.

Das IKE benutzt zur Vorhersage von Schadstoffausbreitungen ein online auf drei Rechner verteiltes Szenarium. Auf einer Workstation läuft der Kontrollprogrammteil und der Visualisierungs- und Animationsteil SCAN2D-T. Auf einem Vektorrechner wird aus topologischen und meteorologischen Daten ein Windfeld berechnet. Ein Parallelrechner simuliert die Schadstoffausbreitung und berechnet die Prognose von Depositionen.
Das ISD verteilte das Finite-Elemente System EPDAN. Ein nichtlineares Berechnungsmodul und ein Kontrollmodul laufen auf einem Hochleistungsrechner des RUS. Das zentrale Datenverwaltungsmodul bleibt auf einer Workstation des ISD. Entkoppeltes Pre- und Postprozessing läuft lokal auf der Workstation.

Das Programm ROD des TAT lagerte eine kinematische Simulation mit dem DFN-RPC auf einen Server aus. Die graphische Oberfläche greift gemeinsam mit dem RPC-Client auf ein Shared Memory zu. Über Drehknöpfe lassen sich von der Rechnung entkoppelt physikalische Parameter einstellen, die direkt in der laufenden Simulation berücksichtigt werden.

Verfügbarkeit

Der DFN-RPC wurde für FORTRAN-Anwendungsprogramme entwickelt. Er kann auch für C-Programme oder gemischte Programme eingesetzt werden. Hierbei muß die Schnittstelle, an der das Programm verteilt werden soll, FORTRAN-kompatibel sein.

Transportprotokolle

Als Transportprotokolle werden TCP, ISO-OSI TP4 und UltraNet unterstützt. TCP steht auf der Basis von BSD Sockets auf mehreren UNIX-Systemen (z.B. Alliant, Apollo, CONVEX, CRAY, DecStation, IBM RS/6000, Silicon Graphics, Sun, Vax) zur Verfügung (DFN-RPC Rel. 1.0). Eine Testversion gibt es außerdem für die Intel Paragon, für HP und für VMS.

UltraNet auf der Basis der Kompatibility-Library gibt es für CONVEX, CRAY, Silicon Graphics und Sun.

ISO-OSI TP4 gibt es auf der Basis von TLI für UNICOS 6.0 mit CRAY OSI 1.0, mit dem OSI-Socket-Interface (ein POSIX-Vorschlag) für UNICOS 7.0 mit CRAY OSI 2.0 und mit SunLinkOSI und SunNetOSI für Sun.

Für diese Systeme gibt es Referenzmaschinen beim Entwickler oder bei einem Pilotanwender. Weitere Portierungen sind vorgesehen bzw. schon realisiert (Fujitsu, Siemens VPx00 u. Sx00). Hier werden die Referenzmaschinen von interessierten Anwendern bereitgestellt.

Zukünftige Vorhaben

Entwicklungen

Zusätzlich zur bisherigen Theorie der Remote Procedure Calls muß die durch die beteiligten Prozesse inhärent vorhandene Parallelität explizit für parallele Anwendungsteile nutzbar gemacht werden. Die im DFN-RPC bisher erfolgreich eingesetzte Parallelisierung durch asynchrone Remote Procedure Calls wird dazu um weitere Synchronisationskonstrukte ergänzt.

Der DFN-RPC enthält Konvertierungsmacros um Real- und Integerzahlen binär übertragen zu können. Diese Makros werden den Anwendern als Library und als Konvertierungstool verfügbar gemacht, damit sie Anwendungsdaten mittels Filetransfer oder NFS binär übertragen können.

Für die verteilten Anwendungen wird ein komfortables Diagnosewerkzeug entwickelt, das bei Laufzeit Informationen und Zustände sowohl auf Anwendungsebene sowie auf der Ebene der RPCs als auch auf der Ebene der netzwerkweiten Interprozesskommunikation veranschaulicht. Dieses Monitortool kann sowohl in der Entwicklungsphase als auch beim Einsatz der verteilten Anwendung Aufschluß über den erwünschten oder fehlerhaften Programmablauf geben.

Das OSI-Protokoll wird in Zukunft auf der Basis eines einheitlichen POSIX-Standard-Interfaces realisiert (XTI oder Socket).

Zugang

Der DFN-RPC kann allen Interessenten im Rahmen des Deutschen Forschungsnetzes kostenlos zur Verfügung gestellt werden. Er wird als komprimierter Tar-File mittels ftp ausgeliefert, dieser enthält die Dokumentation als Postscriptfile, das RPC-Tool und Beispielprogramme.
Innerhalb des Netzes der Universität Stuttgart kann über das Network File System (NFS) direkt auf fertige Installationen auf dem Fileserver ruscy zugegriffen werden.
DFN-RPC Installationen für die Rechnertypen CONVEX, CRAY-2, DECStation, VAX mit Ultrix, IBM RS/6000, Silicon Graphics, Sun 3 und Sun 4 liegen unter dem Pfad:

`/ruscy/sw/dfnrpc/1.0/sys.`

Für CONVEX, CRAY-2, Silicon Graphics und Sun 4 gibt es zusätzlich eine Version für UltraNet.

Allerdings müssen für den erfolgreichen Zugriff auf den Fileserver einige Voraussetzungen erfüllt sein.

Der jeweilige Benutzer benötigt einen Login auf dem Rechner ruscy, seine Benutzer-Id muß im uid-mapping der ruscy eingetragen sein und die ruscy muß das z-Filesystem an den Rechner des Benutzers exportieren. Die dazu notwendigen Anträge für Benutzernummern und für NFS-Zugang sind bei Frau Feucht im RUS zu stellen.

Außerdem muß der Rechner des Benutzers dieses Filesystem mounten (NFS).

Beispiel

Die einzelnen Schritte zur Entwicklung einer verteilten Anwendung werden nun für ein möglichst einfaches Beispiel gezeigt. Als Rechnertyp wird die Workstation IBM RS/6000 angenommen.

Umgebung

Zuerst werden die Pfade zur installierten DFN-RPC-Software der Umgebung des Benutzers bekanntgemacht, bei Verwendung einer C-Shell jeweils mit dem Kommando setenv, mit einer Korn-Shell:

```
DFNRPCCONF=/ruscy/sw/dfnrpc/1.0/sys/ibmrs6
export DFNRPCCONF
DFNRPCLIB=$DFNRPCCONF/lib
export DFNRPCLIB
DFNRPCINCLUDE=$DFNRPCCONF/include
export DFNRPCINCLUDE
DFNRPCBIN=$DFNRPCCONF/bin
export DFNRPCBIN
```

RPI-Definition

Ausgehend von der Programmquelldatei wird die Remote Procedure Interface Definition spezifiziert, siehe Beispieldatei `$DFNRPCINCLUDE/simple.rpi` :

```
C*[BLK_BEG] cb, NAME='simple client block', DEPTH=0, CLIENT_SITE
C*[MOD_USE]     cm, LOCAL
C*[MOD_USE]     sm, CLIENT
C*[BLK_END]

C*[BLK_BEG] sb, NAME='simple server block', DEPTH=0, SERVER_SITE
C*[MOD_USE]     sm, SERVER
C*[BLK_END]

C*[MOD_BEG] cm, NAME='simple client module - main program'
      PROGRAM SIMPLE
      INTEGER IIN,IOUT
C*RPC CALL RPCINI
      IIN = 1
      IOUT = 0
      CALL RADD1(IIN,IOUT)
      PRINT*,'Simple-Result: ',IIN,' + 1 = ',IOUT
C*RPC CALL RPCFIN
      END
C*RPC SUBROUTINE RPCINI
C*RPC   INCLUDE 'rpcini.inc'
C*RPC END
C*RPC SUBROUTINE RPCFIN
C*RPC   INCLUDE 'rpcfin.inc'
C*RPC END
C*[MOD_END]
```

```
C*[MOD_BEG] sm, NAME='simple server routines'
C*RPC PROGRAM RPCSRV
C*RPC   INCLUDE 'rpcsrv.inc'
C*RPC END
C*[RPI_BEG]
      SUBROUTINE RADD1(JIN,JOUT)
      INTEGER JIN
C*RPC+             [IN]
      INTEGER JOUT
C*RPC+             [OUT]
C*[RPI_NON]
      JOUT = JIN + 1
      PRINT*,'RADD1: Computing ',JIN,' + 1'
      RETURN
      END
C*[RPI_END]
C*[MOD_END]
```

Die für die verteilte Version der Anwendung benötigten Stubgenerator-Anweisungen sowie der zusätzliche Programmcode für RPC-Initialisierungen und Server-Hauptprogramm werden von FORTRAN-Compilern als Kommentar interpretiert.

Programmentwicklung

Das Beispielprogramm kann man bei sich testen mit:

```
cd <eigene Arbeitsdirectory>
cp $DFNRPCINCLUDE/simple.rpi .
```

Aufruf des Stubgenerators:

```
$DFNRPCBIN/dfnrpcgen  -T ieee simple
```

Kompilation der Programmteile:

```
xlf -c -qcharlen=8192 -I$DFNRPCINCLUDE simple_cb.f
xlf -c -qcharlen=8192 -I$DFNRPCINCLUDE simple_cb_stb.f
xlf -c -qcharlen=8192 -I$DFNRPCINCLUDE simple_sb.f
xlf -c -qcharlen=8192 -I$DFNRPCINCLUDE simple_sb_stb.f
```

Wenn ein Fortrancompiler die Option -I für Fortran-INCLUDE-Statements (z. B. im Gegensatz zum Preprozessor `#include)` nicht unterstützt, kann man für die benötigten Dateien `rpcini.inc, rpcfin.inc, rpcsrv.inc, dfnrpc.inc` und `dfnrpc_stub.inc` Symbolic Links anlegen (vgl. [2], S. 15).

Client- und Serverprogramm binden:

```
xlf -o simple_cb simple_cb.o simple_cb_stb.o
$DFNRPCLIB/libdfnrpc.a
xlf -o simple_sb simple_sb.o simple_sb_stb.o
$DFNRPCLIB/libdfnrpc.a
```

Programmlauf

Für Client- und Serverteil dieses Beispiels wird je ein eigenes Fenster benötigt. Zunächst wird der Client simple_cb z.B. auf dem Rechner servus10 gestartet. Der Server `simple_sb` soll z.B. auf dem Rechner servus09 laufen. Die Sitzungen verlaufen wie folgt:

```
servus10{zrfs0376} [2]$ simple_cb
  rpcini - please type Local Hostname
servus10
  rpcini - please type Remote Hostname
servus09

 rpo1cs - Information for the Slave:
          Start_Flags     = 00008224
          Master_Host     = servus10
          Master_Tsap     = 2932

 rpo1cs - Object_Start_String =
          Please start simple*_sb on servus09

 rpo1cs - Please START the slave process characterized
          by the Object_Start_String
```

Auf servus09

```
servus09{zrfs0376} [126]$ simple_sb
rpo1ss - Please type the Start_Flags ?
8224

rpo1ss - Please type the Master_Host ?
servus10

rpo1ss - Please type the Master_Tsap ?
2932

simplesrv - rpc correctly initialized
 simplesrv - starting rpc main dispatch
     RADD1: Computing     1  + 1
simplesrv - rpc main dispatcher correct
  simplesrv - rpc correctly finished
STOP
servus09{zrfs0376} [127]$
```

Hier gibt man die vom simple_cb auf servus10 bereitgestellte Information ein.

Dadurch koennen Client- und Server-Teil eine Verbindung aufbauen, über die dann der (die) RPC(s) ihre Parameterlisten austauschen.

(wieder auf servus10:)

```
rpcini - rpc correctly initialized
  Simple-Result:      1  + 1 =            2
  rpcfin - rpc correctly finished
STOP
servus10{zrfs0376} [3]$
```

Literatur

[1] Rolf Rabenseifner, Hans Dieter Reimann: Verteilte Anwendungen über Hochgeschwindigkeitsdatenkommunikation, der DFN Remote Procedure Call. Benutzerhandbuch, Rel. 1.0, Rechenzentrum Universität Stuttgart, 23.11.92.

Das Handbuch steht unter `/ruscy/sw/dfnrpc/1.0/dfnrpc.ps` als Postscriptfile zur Verfügung. Diese Datei enthält am Anfang Macros zum doppelseitigen Ausdrucken auf einem hp LaserJet IIISi. Bei Problemen mit anderen Druckern können diese Zeilen zwischen `statusdict begin` und `end` gelöscht werden.

[2] Rolf Rabenseifner: Distributed Applications between Workstation and Supercomputer using ISO/OSI Protocols. Proceedings, Twenty-Seventh Semi-Annual CRAY User Group Meeting, p. 80-84. London. April 22-26, 1991.

[3] Ralf Allrutz, Rolf Rabenseifner: Der DFN-RPC, ein Remote Procedure Call Tool. Vortragsband des 15. DECUS München Symposiums, S. 523-532, 1992.

Wenn Sie einen Ansprechpartner suchen, können Sie sich an folgende RUS-Mitarbeiter wenden:

Rolf Rabenseifner, NA -5530	`rabenseifner@rus.uni-stuttgart.de`
Hans Dieter Reimann, NA -5790	`reimann@rus.uni-stuttgart.de`
Werner Kollak, NA -5982	`kollak@rus.uni-stuttgart.de`