Мрежово програмиране

Разпределени системи
Отдалечено извикване на процедури

доц. д-р Йордан Денев
denev@fmi.uni-sofia.bg

Използван; http://www.linuxjournal.com/article/2204

Участници

- **Caller**: a program which calls a subroutine
- **Callee**: a subroutine or procedure which is called by the caller
- **Client**: a program which requests a connection to and service from a network server
- **Server**: a program which accepts connections from and provides services to a client

The caller always executes as a client process, and the callee always executes as a server process.

RPC механизъм

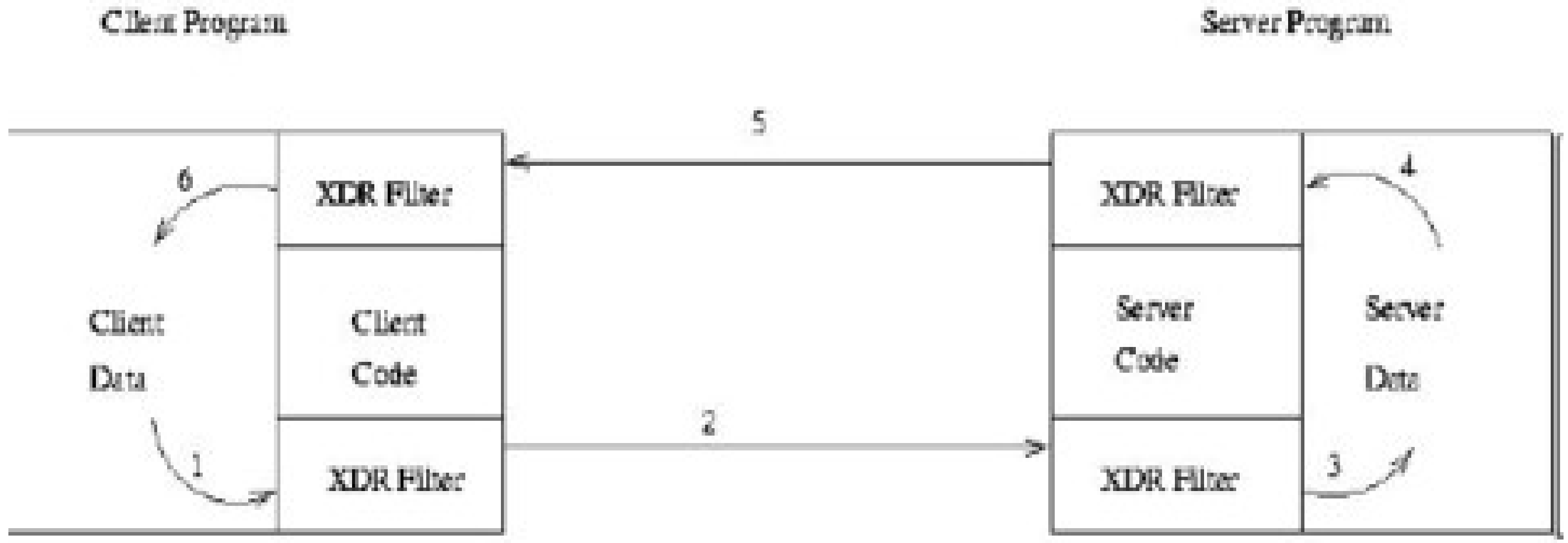
1. The caller program must prepare any input parameters to be passed to the RPC. Note that the caller and the callee may be running completely different hardware..
2. The calling program must somehow pass its data to the remote host which will execute the RPC. The RPC receives and operates on any input parameters and passes the result back to the caller.
3. The calling program receives the RPC result and continues execution.
4. The calling program receives the RPC result and continues execution.

Проблеми на представянето

The solution to this problem involves the adoption of a standard for data interchange.

One such standard is the ONC external data representation (XDR). XDR is essentially a collection of C functions and macros that enable conversion from machine specific data representations to the corresponding standard representations and vice versa. It contains primitives for simple data types such as int, float and string and provides the capability to define and transport more complex ones such as records, arrays of arbitrary element type and pointer bound structures such as linked lists.

RPC Dataflow



1. Client encodes data through XDR filter.
2. Client passes XDR encoded data accross network to remote host
3. Server decodes data through XDR filter.
4. Server encodes function call result through XDR filter
5. Server pass XDR encoded data accross network back to client
6. Client decodes RPC result through XDR filter and continues processing

Figure 1.

RPC Call Binding

An RPC application is formally packaged into a *program* with one or more *procedure* calls.

The RPC program is assigned an integer identifier known to the programs which will call its procedures. Each procedure is also assigned a number that is also known by its caller.

RPC uses a program called **portmap** to allocate port numbers for RPC programs. When an RPC **program** is started, it registers itself with the portmap process running on the same host. The portmap process then assigns the TCP and/or UDP port numbers to be used by that application.

Prior to calling the remote procedure, the *caller* also contacts portmap in order to obtain the corresponding port number being used by the application whose procedures it needs to call.

The correct procedure is reached through the use of a dispatch table in the RPC program. The same registration process that establishes the port number also creates the dispatch table. The dispatch table is indexed by procedure number and contains the addresses of all the XDR filter routines as well as the addresses of the actual procedures.

RPC програмиране

ONC RPC- facility developed by the Open Network Computing (ONC) group at Sun Microsystems.

The development of RPC applications can be greatly simplified through the use of **rpcgen**, the protocol compiler. rpcgen has its own input language which is used to declare programs, their procedures and the data types for the procedures' parameters and return values.

```
/*
 * The average procedure receives an array of real
 * numbers and returns the average of their
 * values. This toy service handles a maximum of
 * 200 numbers. */
const MAXAVGSIZE = 200;
struct input_data {
  double input_data<200>;
};
typedef struct input_data input_data;
program AVERAGEPROG {
  version AVERAGEVERS {
      double AVERAGE(input_data) = 1;
  } = 1;
} = 22855;
```

```
rpcgen  avg.x
```

produces:

1. **avg_clnt.c**: the stub program for our client (caller) process
2. **avg_svc.c**: the main program for our server (callee) process
3. **avg_xdr.c**: the XDR routines used by both the client and the server
4. **avg.h :** the header file