Data Representation and Remote Procedure Calls

Srinidhi Varadarajan

Topics

- External data representation
  - Motivation
  - Approaches
  - NDR, ASN.1, and XDR
- Remote procedure calls
  - Concepts
  - ONC RPC
    - General operation
    - Code example

Need for Data Representation (1)

- Network applications pass many types of data
  - Characters and character strings
  - Integers (of different lengths)
  - Floats (of different lengths)
  - Arrays and structures (flat types)
  - Complex types (using pointers)
- Different host architectures may use different internal representations
  - Networked environments are often heterogeneous

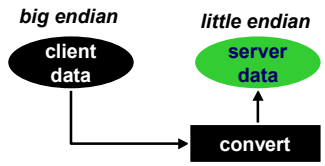
Need for Data Representation (2)

- Example: $(300)_{10} = (13C)_{16}$
  - Stored as a long integer: 00 00 01 3C
  - "Big endian" versus "little endian"

	big endian	*little endian*
byte i:	00	3C
byte i+1:	00	01
byte i+2:	01	00
byte i+3:	3C	00

Potential Solutions (1)

- Asymmetric conversion
  - Convert at one end (client or server)
  - Must know the host type of destination or source
  - With N types of hosts, need N(N-1) converters total.
  - Sometimes known as "receiver-makes-right"
  - Basis for NDR

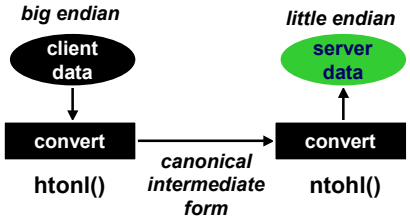


Potential Solutions (2)

- Symmetric conversion
  - Convert to and from a *canonical intermediate form* -- an external data representation
  - Flexible and portable, but at a cost in computation
    - Conversion required even if client and server use the same internal representation
  - With N types of hosts, requires 2N converters
    - Fewer converters than for asymmetric conversion
    - But, N is usually small
  - Basis for XDR and ASN.1

Potential Solutions (3)

- **Symmetric conversion (continued)**



4/4/2001 7

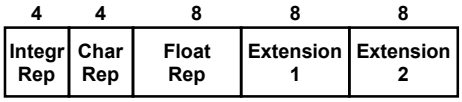
Network Data Representation (1)

- **NDR is used in the Distributed Computing Environment (DCE)**
- **Uses asymmetric "receiver-makes-right" approach**
- **Format**
  - **Architecture tag at the front of each message**
    - **"Big endian" or "little endian"**
    - **ASCII or EBCDIC**
    - **IEEE 754 or other floating point representation**

4/4/2001 8

Network Data Representation (2)

- **Architecture tag**



4/4/2001 9

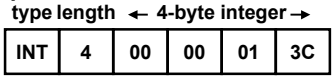
Abstract Syntax Notation One (1)

- **ASN.1 is an ISO standard**
  - **Scope is broader than network data representation**
  - **Basic Encoding Rules (BER) defines representation**
- **Uses a canonical intermediate form (symmetrical)**
- **Uses a triple to represent each data item**
  - **< tag, length, value >**
  - **Tag defines type (usually 8 bits)**
  - **Length is number of bytes in value field**
  - **Value is in canonical intermediate form**

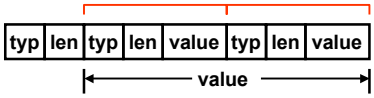
4/4/2001 10

Abstract Syntax Notation One (2)

- **Example**



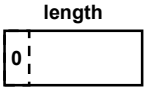
- **Compound data types can be represented by nesting primitive types**



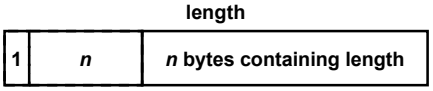
4/4/2001 11

Abstract Syntax Notation One (3)

- **Length field can be made arbitrarily large**
  - **1- to 127-byte value**



Greater than a 127-byte value



4/4/2001 12

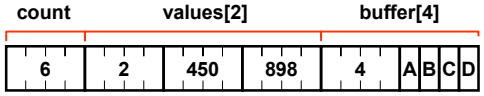
External Data Representation (1)

- XDR is used with SunRPC (Open Network Computing RPC)
  - Defined in RFC 1014
- Uses a canonical intermediate form (symmetrical)
- Types are implicit
  - XDR codes data, but not the type of data
  - Type of data must be determined by application protocol
- Tags are not used except to indicate array lengths

External Data Representation (2)

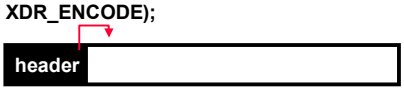
- Example XDR encoding of a structure

```
struct example {
     int  count;
     int  values[2];
     char buffer[4];
}
```



Creating an XDR Data Stream (1)

1) Create buffer
   - xdrmem_create(xdrs, buf, BUFSIZE, XDR_ENCODE);



2) Make calls to build buffer
   - int i = 300;
     xdr_int(xdrs, &i);

Creating an XDR Data Stream (2)

- Sample routines (see fig 20.4 in text)
  - xdr_bool()
  - xcr_bytes()
  - xdr_enum()
  - xdr_float()
  - xdr_vector()
  - xdr_string()
  - xdr_opaque()
- Same calls are used to encode and decode
- Stream header specifies direction
  - For decode: xdrmem_create(xdrs, buf, BUFSIZE, XDR_DECODE);

Comparing XDR, ASN.1, and NDR

- Symmetric versus asymmetric trade-off for comparing ASN.1 and XDR to NDR
  - Potentially more converters needed for NDR, but number of different host types is small
  - Overhead of type fields
  - Conversion can often be avoided
- Comparing ASN.1 and XDR
  - XDR has less overhead than ASN.1 since it does not use tags
  - XDR adheres to natural byte boundaries
  - Expressiveness of ASN.1 is very rich, more flexible than XDR

Remote Procedure Calls

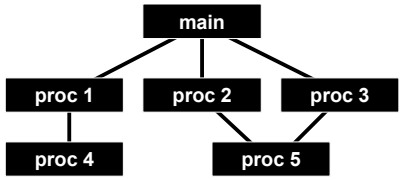
- Remote Procedure Call (RPC) is an alternate model for networked applications
- Used for many standard applications
  - NFS
  - NIS, NIS+
  - Microsoft Exchange Server
  - and others …
- Closely associated with data representation
  - Function parameters must pass over the network

Models for Distributed Applications

- **Communication-oriented design**
  - Focus on protocol and communications
  - Our approach to date
- **Application-oriented design**
  - Focus on application program structure and make communications "transparent"
  - RPC approach

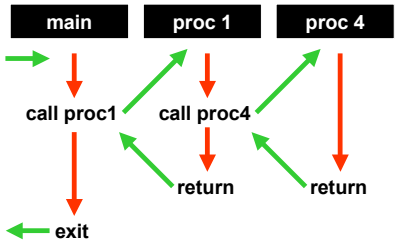
4/4/2001 19

A Traditional Program (1)



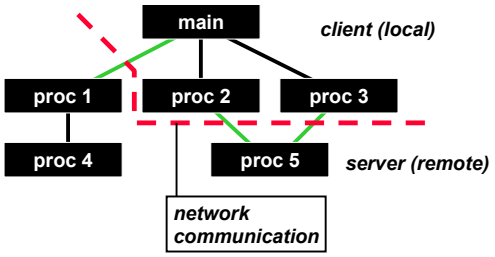
4/4/2001 20

A Traditional Program (2)



4/4/2001 21

Make the Program Distributed (1)

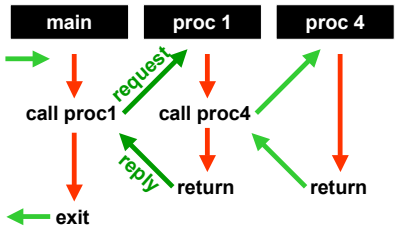


- **proc1, proc4, and proc5 are *remote procedures***

4/4/2001 22

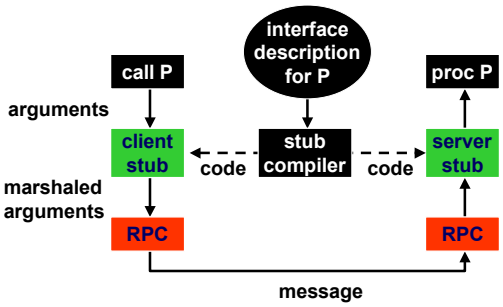
Make the Program Distributed (2)

- **Call -- send message to invoke remote procedure**
- **Return -- send reply back to client**



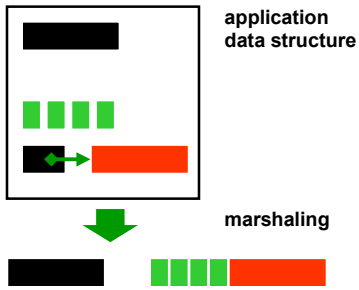
4/4/2001 23

RPC Components



4/4/2001 24

Marshaling Arguments



RPC Design Issues

- Control is multithreaded
  - Procedures executed on different hosts
  - Different threads for each call
- No shared memory
- No shared resources, e.g. files
- More arguments
  - Since no shared memory or other resources
- Server must be active or can be invoked
- Message interface

ONC RPC

- Open Network Computing (ONC) RPC
  - Developed by Sun Microsystems
- "Remote programs"
  - Remote procedures plus shared global data
  - Not just remote procedure
- Functionality
  - Message formats -- carried by TCP or UDP
    - Pass arguments, results, other information
  - Naming scheme for remote programs and procedures
    - Program, version, procedure
  - Authentication scheme

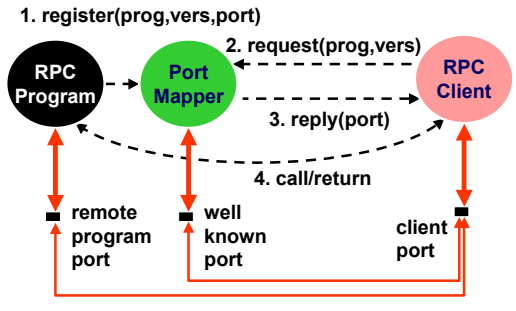
ONC RPC Communications

- Can use TCP or UDP
  - RPC does nothing itself to provide reliability
- With UDP …
  - If client receives a reply, then "at least once" semantics apply
  - If client does not receive a reply, then "zero or more" semantics apply
  - Must be considered in design
    - "read 20 bytes starting at 100", not
    - "read the next 20 bytes"
- With TCP …
  - Reliable due to use of TCP

Port Mapper (1)

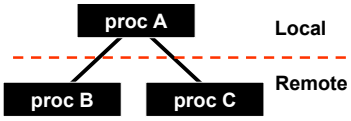
- "Port mapper" allows dynamic maping between protocol port numbers and remote programs
- Remote programs (servers) register with the port mapper on their local host
- Clients query port mapper at well-known port number (111) to get port for remote program

Port Mapper (2)



Stub Routines (1)

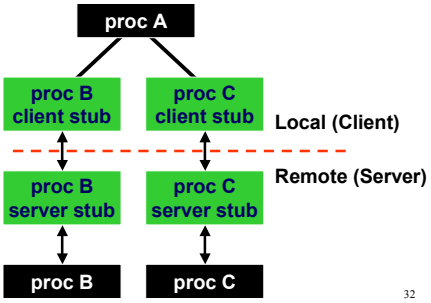
- Traditional program to be partitioned



4/4/2001 31

Stub Routines (2)

- After partitioning with stub routines



4/4/2001 32

Client Stub

- Is called by client program
- "Marshals" arguments
  - XDR used to encode (with ONC RPC)
- Sends CALL to server
- Waits for reply
- "De-marshals" arguments
  - XDR used to decode
- Returns to client program
  - Client just makes a call that then returns

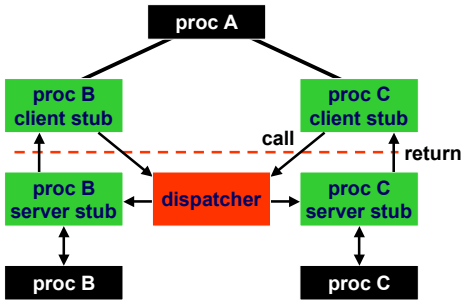
4/4/2001 33

Server Stub

- Is dispatched
- Accepts arguments, de-marshals and decodes with XDR
- Calls server program procedure
- Procedure returns to stub
  - Server procedure is just called and later returns
- Marshals results and encodes with XDR
- Sends results back to client
- Exits

4/4/2001 34

Dispatcher



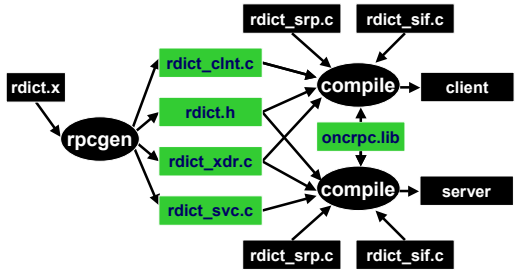
4/4/2001 35

RPCGEN

- RPCGEN is the RPC program "generator"
- Simplifies the creation of a distributed application using RPC
- Input descriptions of …
  - Remote procedures and interfaces
  - User-defined data types, e.g. structures
- Output files …
  - Client and server stub files
  - Conversion routines for user-defined data types
  - Common header file

4/4/2001 36

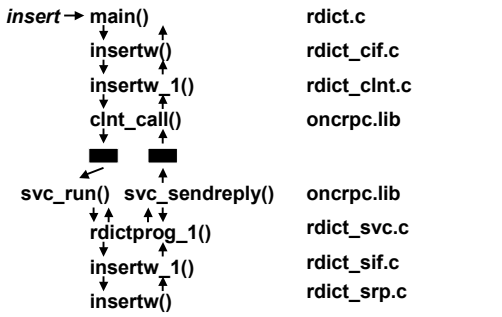
Code Generation using RPCGEN



ONC RPC Code Example Files

- rdict.x: interfaces, common values, data structures
- rdict.h: common header file
- rdict_xdr.c: XDR translations
- rdict_clnt.c: sends calls from client to server
- rdict_svc.c: dispatcher, sends calls from server to client
- rdict.c: main client
- rdict_cli.c: client stub procedures
- rdict_srp: main server routines
- rdict_sif.c: server stub procedures

ONC RPC Code Example Call Sequence



You should now be able to … (1)

- Describe different schemes for data representation and identify strengths and weaknesses
  - Generic models
  - Specific schemes (NDR, ASN.1, XDR)
- Show how simple data types would be represented using NDR, ASN.1, and XDR
- Describe the structure of an RPC application including role of stub procedures
- Describe the need for marshaling and wher marshaling is implemented

You should now be able to … (2)

- Describe the structure and operation of …
  - ONC RPC
- Define the role of …
  - RPCGEN
- Design and analyze simple applications using ONC RPC