


---

---

---

---

---

---

---

---

- ## XDR as a case study
- Sun RPC uses XDR.
  - A good example of a “layer”.
  - Interesting API.
  - Powerful paradigm for creation and transfer of complex data structures.
- Netprog: XDR 2

---

---

---

---

---

---

---

---

- ## XDR
- XDR provides a service associated with the OSI Presentation Layer.
    - Common data representation
    - Library (not part of the O.S.).
    - Easy to port to new architectures.
    - Independence from transport layer.
- Netprog: XDR 3

---

---

---

---

---

---

---

---

## Data Conversion

- Asymmetric Data Conversion
  - client always converts to the server's data representation.
- Symmetric Data Conversion
  - both client and server convert to/from some standard representation.

XDR is Symmetric Data Conversion

Netprog: XDR

4

---

---

---

---

---

---

---

---

## XDR Data Types

- boolean
- char
- short
- int
- long
- float
- double
- enumeration
- structure
- string
- fixed length array (1-D)
- variable length array (1-D)
- union
- opaque data

Netprog: XDR

5

---

---

---

---

---

---

---

---

## Implicit vs. Explicit Typing

- *Explicit typing*: each piece of data includes information about the type.
- *Implicit typing*: the sender and receiver must agree on the order and type of all data.

XDR uses Implicit Typing

Netprog: XDR

6

---

---

---

---

---

---

---

---

## XDR Programming

- XDR libraries are based on a *stream* paradigm.
- The process of converting local (host) data to XDR also puts the data in the XDR stream.
- When extracting an item from an XDR stream, conversion is done to the local (host) representation.

Netprog: XDR

7

---

---

---

---

---

---

---

---

## XDR streams

- XDR Streams can be attached to a file, pipe, socket or memory.
- Individual data items are added to the stream one at a time.
- Individual data items are removed from the stream one at a time.

Netprog: XDR

8

---

---

---

---

---

---

---

---

## Conversion Terminology

- Converting from local representation to XDR representation is called *Encoding*.
- Converting from XDR representation to local representation is called *Decoding*.



Netprog: XDR

9

---

---

---

---

---

---

---

---

## XDR Stream Creation

- There are a number of stream creation functions in the XDR library.
  - `xdrmem_create`
    - » destination for encoding -or- source for decoding is a chunk of memory.
  - `xdrstdio_create`
    - » destination for encoding -or- source for decoding is a file descriptor.

Netprog: XDR

10

---

---

---

---

---

---

---

---

## XDR filters

- The XDR library includes an extensive set of *filters* that perform encoding/decoding operations.
- Each XDR stream includes an attribute that determines the specific operation that will be performed by a filter (encoding or decoding).

Netprog: XDR

11

---

---

---

---

---

---

---

---

## XDR Filters

- The filter to encode/decode an integer is called `xdr_int`:

```
bool_t xdr_int(XDR *xdrs, int *ip);
```

  - the return value indicates success.
  - `xdrs` is a pointer to an XDR stream
  - `ip` is a pointer to an integer.

Netprog: XDR

12

---

---

---

---

---

---

---

---

## xdr\_int()

If the XDR stream operation is `XDR_ENCODE`:

```
int count;
XDR *xstream;
xdr_int(xstream, &count);
```

will convert (encode) the value of `count` to the integer representation used by XDR (big-endian) and put the result on the XDR stream.

Netprog: XDR

13

---

---

---

---

---

---

---

---

## xdr\_int() again

If the XDR stream operation is `XDR_DECODE`:

```
int count;
XDR *xstream;
xdr_int(xstream, &count);
```

will get an XDR integer from the stream, convert (decode) the value to local integer representation and put the result in `count`.

Netprog: XDR

14

---

---

---

---

---

---

---

---

## xdr\_int() encode vs. decode

### Source Encodes

```
int count;
xdr_int(xstream, &count);
```



### Destination decodes

```
int count;
xdr_int(xstream, &count);
```

Netprog: XDR

15

---

---

---

---

---

---

---

---

## Complex Data Filters

- The XDR library includes a set of filters designed to translate complex C data structures to and from XDR representation.
- Many of these filters make use of the simpler filters to convert individual components.

Netprog: XDR

16

---

---

---

---

---

---

---

---

## xdr\_array()

- The `xdr_array()` filter provides support for encoding/decoding a *variable* length array.
- As with all XDR filters, the same function works for both encoding and decoding.

Netprog: XDR

17

---

---

---

---

---

---

---

---

## xdr\_array prototype

```
bool_t xdr_array(  
    XDR *xdrs,  
    char *arrp,  
    u_int *sizep,  
    u_int maxsize,  
    u_int elsize,  
    xdrproc_t elproc);
```

a pointer to the size of the array

pointer to a function that can encode/decode individual array elements

Netprog: XDR

18

---

---

---

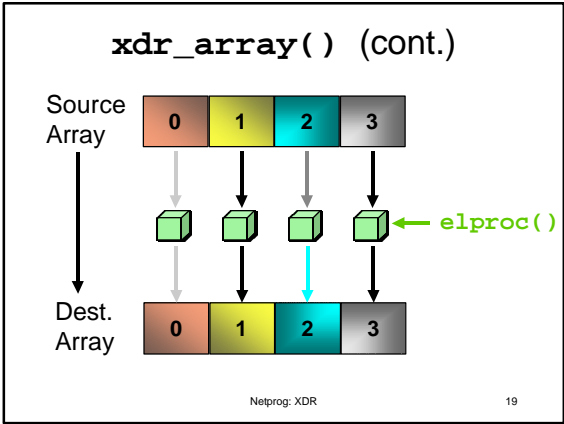
---

---

---

---

---




---

---

---

---

---

---

---

---

### Inside xdr\_array()

```

xdr_int(xdrs, &sizep);

for (i=0; i<sizep; i++)
    elproc(xdrs, arrp+elsize*i);

```

encode/decode the number of elements in the array

encode/decode each array element.

Netprog: XDR 20

---

---

---

---

---

---

---

---

### xdr\_string()

The string conversion filter looks a little different since XDR strings have a maximum size.

```

bool_t xdr_string(
    XDR *xdrs,
    char *string,
    u_int maxsize);

```

Netprog: XDR 21

---

---

---

---

---

---

---

---

## Problem!!

- We want to send an array of strings between processes.
- What is the problem (using `xdr_array`)?
- What is a possible solution?

Netprog: XDR

22

---

---

---

---

---

---

---

## XDR and RPC

- RPC uses XDR, but the XDR filters are built for you automatically!
  - you define a complex data type and a code generator writes an XDR filter that can encode/decode your data type.
- Sometimes you need to understand XDR anyway...

Netprog: XDR

23

---

---

---

---

---

---

---