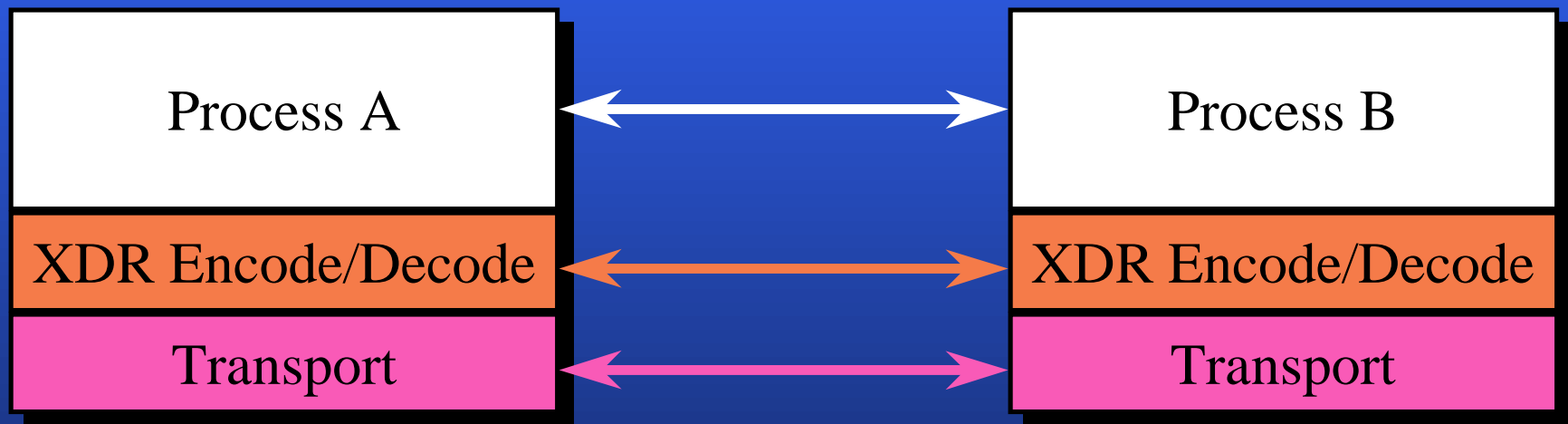


# XDR

## External Data Representation

---



# XDR as a case study

- Sun RPC uses XDR.
- A good example of a “layer”.
- Interesting API.
- Powerful paradigm for creation of complex data structures.

# 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.

# Data Conversion

## ■ Asymmetric Data Conversion

- client always converts to the server's data representation.

## ■ Symmetric Data Conversion

- both client and server convert to some standard representation.

# 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

# Implicit vs. Explicit Typing

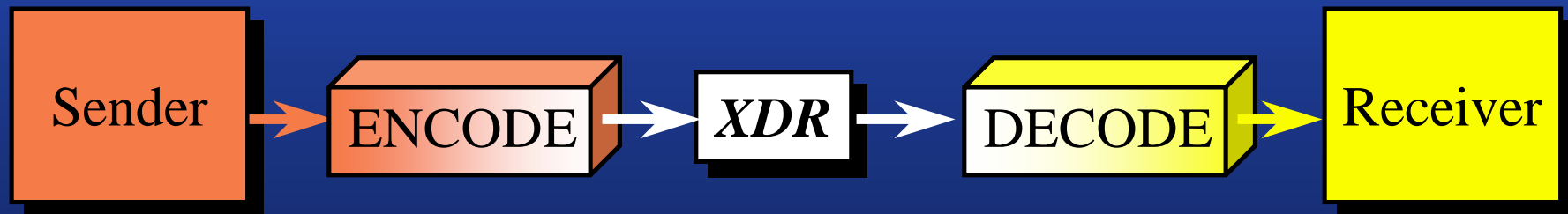
- Explicit Typing means that each piece of data includes information about the type.
- Implicit typing means that the sender and receiver must agree on the order and type of all data.
- XDR uses Implicit Typing

# XDR Programming

- Most XDR implementations are based on a *buffer* paradigm.
- A buffer is allocated that is large enough to hold an entire message.
- Individual data items are added to the buffer one at a time.
- XDR buffers can be attached to a file, pipe, socket or memory.

# Conversion Terminology

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



# XDR Buffer Creation

- There are a number of buffer 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.

# 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).

# 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 or failure.
- `xdrs` is a pointer to an XDR stream
- `ip` is a pointer to an integer.

## 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.

## xdr\_int()

- 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.

# xdr\_int()

## Source Encodes

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

## Destination decodes

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

# 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.

# xdr\_array()

- The `xdr_array()` filter provides support for encoding/decoding a variable length array.

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

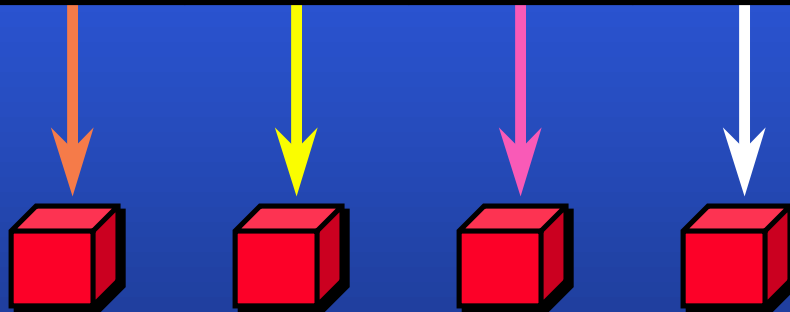
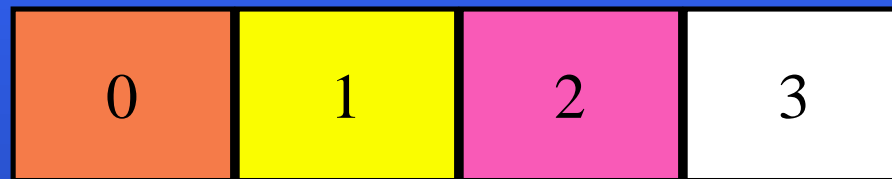
`sizep` is a pointer to the size of the array.

`elsize` is the size of each element (in bytes).

`elproc` is a pointer to a function that can encode/decode individual array elements.

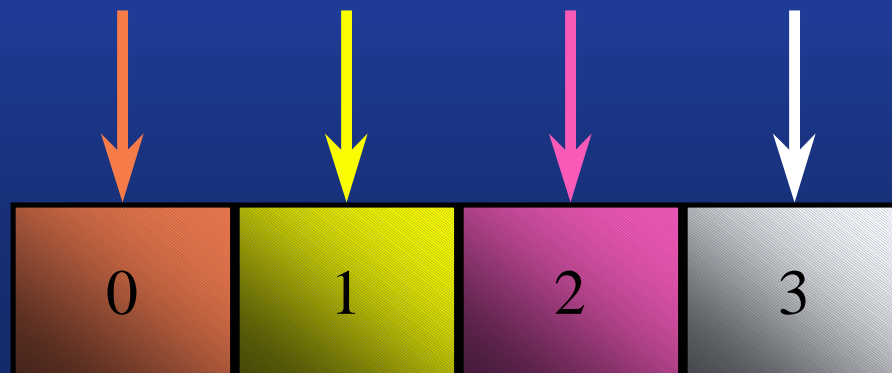
# xdr\_array()

Source  
Array ->



← elproc()

Destination  
Array ->



# Inside xdr\_array()

```
xdr_int(xdrs, &sizep);
```

encode/decode the  
number of elements  
in the array

```
for (i=0; i<sizep; i++)
```

```
    elproc(xdrs, arrp+elsize*i);
```

encode/decode each array element.

## xdr\_string()

- the string conversion filter is a little different since XDR strings have a maximum size.

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

# Problem!!

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