Today’s lecture

• Remote procedure call
  – Encoding arguments and results
  – Fragmentation
  – Synchronization between client and server
  – Dispatching to the appropriate procedure
  – Concrete RPC protocols

• SunRPC
• DCE (CORBA)
What is RPC?

- Program running on the client makes call to a procedure that executes on the server
  - One of the most popular transport layer abstraction
  - Usually implemented on top of UDP
  - Used by many applications (NFS) often over LAN

RPC Components

- Protocol Stack – 3 microprotocols
  - BLAST: fragments and reassembles large messages
  - CHAN: synchronizes request and reply messages
  - SELECT: dispatches request to the correct process
- Compiler generated stubs

Presentation Formatting

- Marshalling (encoding) application data into messages
- Unmarshalling (decoding) messages into application data
- Data types we consider
  - integers
  - floats
  - strings
  - arrays
  - structs
- Types of data we do not consider
  - images
  - video
  - multimedia documents
Difficulties

- Representation of base types
  - floating point: IEEE 754 versus non-standard
  - integer: big-endian versus little-endian (e.g., 34,677,374)

```
Big-endian
(3) (2) (1) (0)
01111111 00000000 00101001 01101000

Little-endian
(0) (1) (2) (3)
00111111 00000000 00101001 01101000
```

- Compiler layout of structures

Taxonomy

- Data types
  - base types (e.g., ints, floats); must convert
  - flat types (e.g., structures, arrays); must pack
  - complex types (e.g., pointers); must linearize

- Conversion Strategy
  - canonical intermediate form
  - receiver-makes-right (an $N \times N$ solution)

Taxonomy (cont)

- Tagged versus untagged data

- Stubs
  - compiled
  - interpreted
eXternal Data Representation (XDR)

- Defined by Sun for use with SunRPC
- C type system (without function pointers)
- Canonical intermediate form
- Untagged (except array length)
- Compiled stubs

```c
#define MAXNAME 256;
#define MAXLIST 100;

struct item {
    int count;
    char name[MAXNAME];
    int list[MAXLIST];
};

bool_t
xdr_item(XDR *xdrs, struct item *ptr)
{
    return(xdr_int(xdrs, &ptr->count) &&
        xdr_string(xdrs, &ptr->name, MAXNAME) &&
        xdr_array(xdrs, &ptr->list, &ptr->count,
                  MAXLIST, sizeof(int), xdr_int));
}
```

Abstract Syntax Notation One (ASN-1)

- An ISO standard
- Essentially the C type system
- Canonical intermediate form
- Tagged
- Compiled or interpreted stubs
- BER: Basic Encoding Rules

```
tag, length, value

<table>
<thead>
<tr>
<th>INT</th>
<th>4 bytes length</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>length</td>
</tr>
<tr>
<td>value</td>
<td></td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>type</th>
<th>length</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>4 bytes length</td>
<td>value</td>
</tr>
</tbody>
</table>
Network Data Representation (NDR)

- Defined by DCE
- Essentially the C type system
- Receiver-makes-right (architecture tag)
- Individual data items untagged
- Compiled stubs from IDL
- 4-byte architecture tag

- IntegerRep
  - 0 = big-endian
  - 1 = little-endian

- CharRep
  - 0 = ASCII
  - 1 = EBCDIC

- FloatRep
  - 0 = IEEE 754
  - 1 = VAX
  - 2 = Cray
  - 3 = IBM

Today’s lecture

- Remote procedure call
  - Encoding arguments and results
  - Fragmentation
  - Synchronization between client and server
  - Dispatching to the appropriate procedure
  - Concrete RPC protocols
    - SunRPC
    - DCE (CORBA)

Bulk Transfer (BLAST)

- Unlike AAL and IP, tries to recover from lost fragments
- Strategy
  - selective retransmission
  - a.k.a. partial acknowledgements
BLAST Details

• Sender:
  – after sending all fragments, set timer DONE
  – if receive selective retransmission request, send missing fragments and reset DONE
  – if timer DONE expires, free fragments

BLAST Details (cont)

• Receiver:
  – when first fragments arrives, set timer LAST_FRAG
  – when all fragments present, reassemble and pass up
  – four exceptional conditions:
    • if last fragment arrives but message not complete
      – send SRR and set timer RETRY
    • if timer LAST_FRAG expires
      – send SRR and set timer RETRY
    • if timer RETRY expires for first or second time
      – send SRR and set timer RETRY
    • if timer RETRY expires a third time
      – give up and free partial message

BLAST Header Format

• MID must protect against wrap around
• TYPE = DATA or SRR
• NumFrags indicates number of fragments
• FragMask distinguishes among fragments
  – if Type=DATA, identifies this fragment
  – if Type=SRR, identifies missing fragments (bitmap)

• Compare to DCE solution
  – Selective ack made up of
    • Cumulative ack as an integer
    • Out of order fragments as a variable size bitmap
Request/Reply (CHAN)

- Guarantees message delivery
- Synchronizes client with server
- Supports *at-most-once* semantics

<table>
<thead>
<tr>
<th>Simple case</th>
<th>Implicit acks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td>Request</td>
<td>Request</td>
</tr>
<tr>
<td>ACK</td>
<td>1</td>
</tr>
<tr>
<td>Reply</td>
<td>2</td>
</tr>
<tr>
<td>ACK</td>
<td>3</td>
</tr>
</tbody>
</table>

CHAN Details

- Lost message (request, reply, or ACK)
  - set RETRANSMIT timer
  - use message id (MID) field to distinguish
- Slow (long running) server
  - client periodically sends “are you alive” probe
  - or server periodically sends “I’m alive” notice
- Want to support multiple outstanding calls
  - use channel id (CID) field to distinguish
- Machines crash and reboot
  - use boot id (BID) field to distinguish

Dispatcher (SELECT)

- Dispatch to appropriate procedure
- Synchronous counterpart to UDP
- Implement concurrency (open multiple CHANS)

- Address Space for Procedures
  - flat: unique id for each possible procedure
  - hierarchical: program + procedure number
Simple RPC Stack

- IP implements BLAST-equivalent
  - except no selective retransmit
- SunRPC implements CHAN-equivalent
  - except not at-most-once
- UDP + SunRPC implement SELECT-equivalent
  - Uses Port Mapper to map from programs to ports
  - UDP dispatches to program (ports bound to programs)
  - SunRPC dispatches to procedure within program

SunRPC

- IP implements BLAST-equivalent
- SunRPC implements CHAN-equivalent
- UDP + SunRPC implement SELECT-equivalent
  - Uses Port Mapper to map from programs to ports
  - UDP dispatches to program (ports bound to programs)
  - SunRPC dispatches to procedure within program

SunRPC Header Format

- XID (transaction id) is similar to CHAN’s MID
- Server does not remember last XID it serviced
- Problem if client retransmits request while reply is in transit