

RPC2

A Snapshot of Understanding

Peter J. Braam

Robert Watson

Aims

- RPC2 manual describes API, we need
 - Protocol description
 - More details about internals
- On implementation side
 - Bug fixes: security, laptop networking, stability
 - Considerably more clarity in implementation
 - Changes in protocols will accompany some bug fixes
- Ignore all MultiRPC and Failure package
 - final decisions will have to take these into acc.

Layers in RPC2

- Layers are not clearly separated
- Underlying design does allow good layering
- Compare with ISO stack and relate to it where appropriate, ie at network and transport layer

Keep in mind !

Client Send	Client Receive
Server Send	Server Receive

All these types of code are mixed

Principal Abstractions

- PacketBuffers: header, body; sit in a pool
- Centry: RPC2 logical connection data
- Handles: point to Centry's
- SLEntry's: handles for packets, with retry info, destination info
- HostEntry: Encode remote host info

RPC2 Transport Layer

RPC2 transport

Reliable transmission

Timeouts

Bandwidth adaptation

Binding

Encryption

RPC2 Connections with handles

Packet format, header XDR

UDP Transport

Sockets

Ports

Network

IP Addresses

Software Protocol Layer

RPC2 subsystems

Vice.rpc2, callback.rpc2 etc.
currently the “Coda
protocols”

RPC2 remote procedure calls
and marshalling

An RPC is specified by an
integer and takes arguments
which are of such and so a
type.

RPC2 XDR

The arguments for RPC's
are to be packed and
unpacked in the following
way. (Language
Independent)

Sending Protocol Layer

RPC level (requests)

- allocate buffers
- marshalling of args
- invoke SideEffects
- wait

top and bottom of
RPC2_MakeRPC

RPC (replies)

- adds local IP&port
 - builds packet
- coding
RPC2_SendResponse
violates layering

Packetbuffer handles

- name is SL_Entry
 - arrange reliable transmission
 - yield to SocketListener
 - decide on Sle->return_code
- rpc2_SendReliably & snippets

Logical connections

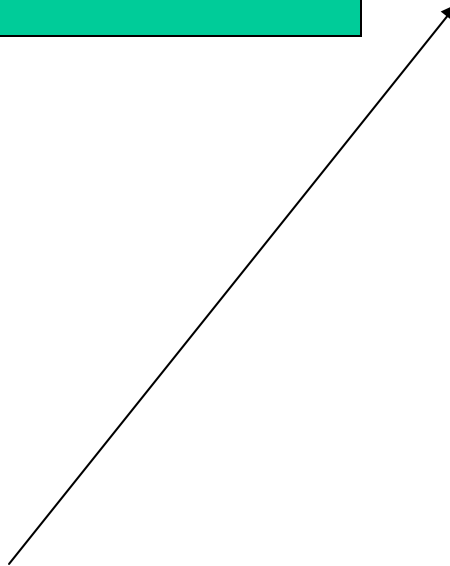
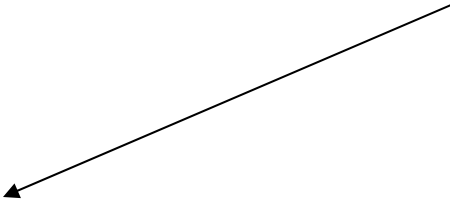
- name is CEntry
- map Handle to CEntry
- await correct state of connections
- encryption of data
- snippets of code all over
(e.g. in RPC2_MakeRPC)

UDP layer:

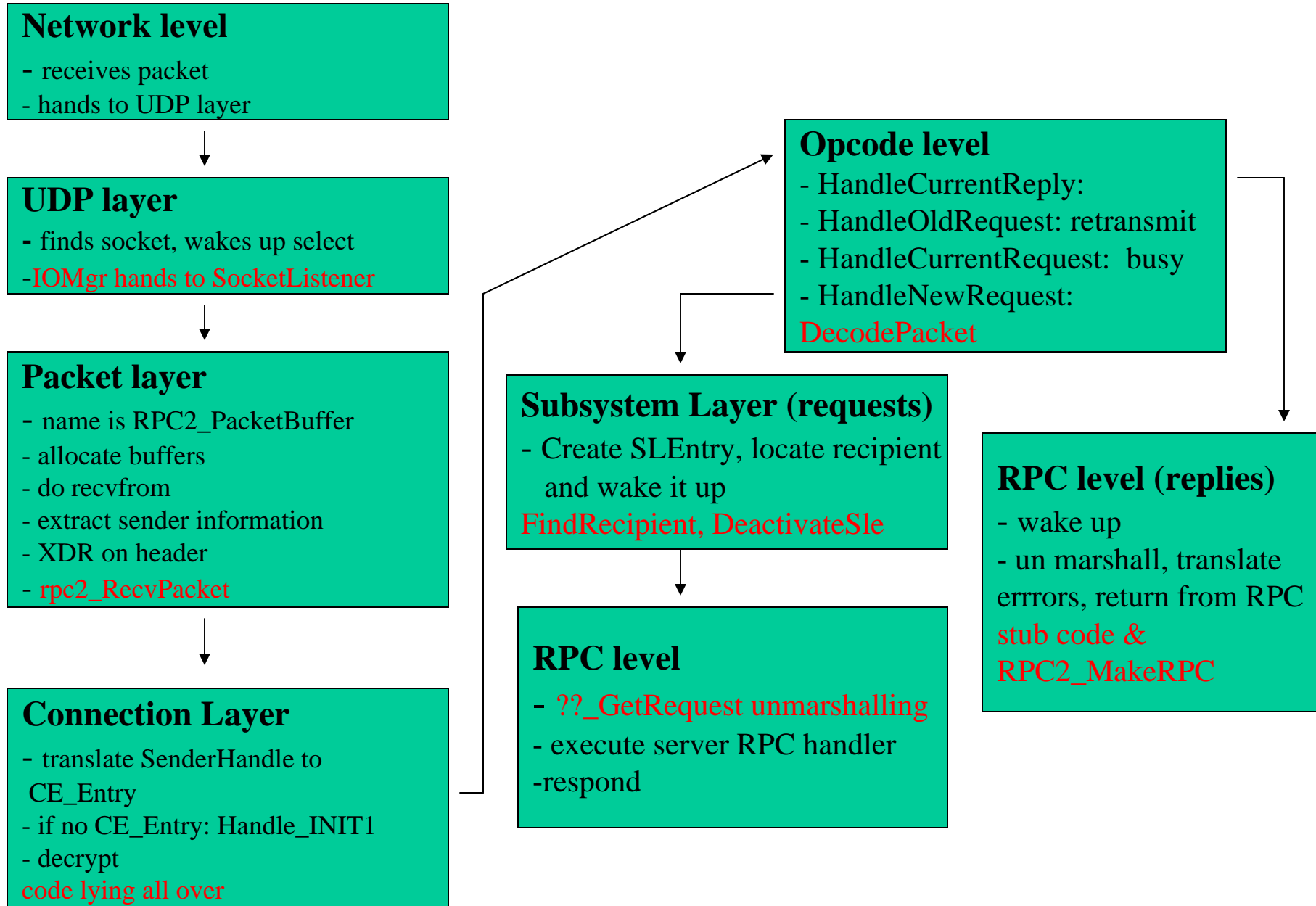
- sendto()
 - use socket fd
 - add remote port/ip
- rpc2_XmitPacket

Network layer

- adds local IP&port
- builds packet



Receiving Protocol Layer



Packet Headers (60 octets)

- ProtoVersion
 - identify protocol
- RemoteHandle
 - SenderHandle
- LocalHandle
 - ReceiverHandle (not needed)
- Flags
 - 4 octets: SE, Color, 2 for RPC
- BodyLength
 - ? Checksum?
- SeqNumber
 - goes up by one
- OpCode
 - to decipher request
- SEFlags
- SEDataOffset
- SubsysId
 - to hand to subsystem
- ReturnCode
 - result
- Lamport
 - distributed time
- Uniquefier
- TimeStamp
- BindTime

What is a side effect?

- It is a collection of hooks
- Primary purpose is to do a “pre” and “post” RPC processing, e.g.:
- Fetch(0x21.0x3.0x6) initiates and completes a side effect in RPC2 stub code
- Most hooks are functions: at setup, binding, before and after RPC's
- Also hook in the connection area.
- SideEffect are not part of RPC2 protocol but are **active parameters** (bind and RPC)

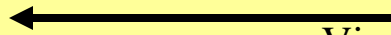
Vice & Callback Protocol

client opcode

opcode server

INIT1

Sent as result of RPC2_NewBinding
Client sends SenderHandle

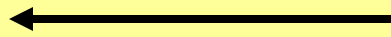
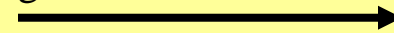


After sending this Vice runs
ViceNewConnection to register client handle

INIT2

NewConnectfs

Sent as result of RPC2_NewBinding
Client sends SenderHandle

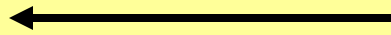


Server sets up callback connection
if there isn't one yet for client

INIT1

INIT2

After sending this Vice runs
ViceNewConnection to register client handle

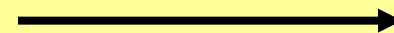


Verify Client is working, make
a gratuitous callback

Callback

REPLY

To callback request



NewConnectFS now complete

REPLY