Likewise Software

Let's go to the library - advantages and difficulties of creating componentised DCE/RPC environment

Brian Koropoff
Rafal Szczesniak
Small journey across the layers:
• SMB transport
• Named pipes
• DCE/RPC runtime
• MS-RPC libraries
• DCE/RPC environment released by Novell/PADL, originally based on OSF libraries
• Samba 3 libsmbclient providing support for named pipes read/write
• Samba 4 idl files providing details of interface
<table>
<thead>
<tr>
<th>libsmbclient</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS-RPC API</td>
</tr>
<tr>
<td>DCE/RPC runtime</td>
</tr>
<tr>
<td>named pipes muxer</td>
</tr>
</tbody>
</table>
• Originally not designed to carry named pipe traffic, but to provide POSIX compatibility layer
• Pretty well established API though not very extendible
Changes made:

• Extended to provide NT-style open (NTcreate)
• Kerberos credentials cache supported in authentication callback
• Access to SMB session key necessary for cryptography in MS-RPC layer
• ABI remains the same
Possible improvement:
• Small part of the code could turn into named pipes library
<table>
<thead>
<tr>
<th>MS-RPC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCE/RPC runtime</td>
</tr>
<tr>
<td>libsmbclient</td>
</tr>
</tbody>
</table>

- **named pipes muxer**
Named pipes muxer

• Thin layer providing access to SMB named pipes via unix domain sockets
• A connection between DCE/RPC libraries and SMB client library, thus allowing to call remote procedures over named pipes/SMB transport
The muxer works in two modes:

• Daemon listening for commands on unix domain socket
• Transparent proxy forwarding the traffic between the client socket and named pipe
libnpc library allows sending messages to the muxer

- Credentials handling
- Impersonation
- Transport layer's session key handling
- Opening connection and entering proxy mode
NP muxer – daemon

Messages:
NpcSetAuthInfo
NpcClearAuthInfo
NpcImpersonate
NpcConnect

Client app ↔ Npc muxer
unix socket
Connecting remote named pipe enters the proxy mode:

- Remote host's IPC$ is connected
- Named pipe is opened
- Established session key is returned along with status code
If NpcConnect is successful the muxer becomes transparent proxy forwarding RPC traffic back and forth as long as the connection is active.
NP muxer – future

Possible changes/extensions:
• Separate functions for sending/receiving PDU's (instead of proxy mode)
• Access to all transport layer's data for MS-RPC layer
<table>
<thead>
<tr>
<th>MS-RPC API</th>
</tr>
</thead>
<tbody>
<tr>
<td>named pipes muxer</td>
</tr>
<tr>
<td>libsmbclient</td>
</tr>
</tbody>
</table>

DCE/RPC runtime
OSF heritage:
• Machinery that enables a client to call remote function on a server
• Includes IDL compiler
• Heavily based on threads
• Did not include ncacn_np endpoint support (named pipes)
• Used by Microsoft ages ago for their own DCE/RPC environment
What had to be done to make it useful:

• IDL compiler cleanup to make use of certain MS-specifics in function calls
• Missing piece – ncacn_np endpoint support (named pipes)
• Threads – a big cleanup
• Impersonation token
### Existing DCE/RPC implementations

<table>
<thead>
<tr>
<th></th>
<th>Samba 3</th>
<th>Samba 4</th>
<th>Likewise DCE/RPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IDL compiler</strong></td>
<td>PIDL (Perl script)</td>
<td>PIDL (Perl script)</td>
<td>C code + Flex/Bison + C preprocessor</td>
</tr>
<tr>
<td><strong>generated code</strong></td>
<td>Human readable</td>
<td>Human readable</td>
<td>Definitely machine readable</td>
</tr>
<tr>
<td><strong>Multitasking</strong></td>
<td>Processes</td>
<td>Processes + state machines</td>
<td>Threads</td>
</tr>
</tbody>
</table>
Next steps to be done:
• Support for GSS authenticated binds
• Even more severe threads overhaul
• Further IDL compiler extensions
• Provide interfaces for well known named pipes – Lsa, Samr, Netlogon, etc.
• Idl files are based on Samba 4 IDLs
• Different calling convention than what Samba 4 uses – the same result
• All functions are synchronous
• 2-byte unicode strings are used, just like on the wire
• A separate libunistr library has been developed to make our life easier
• Originally the code (and most importantly idl's) have been tested on Windows
### Existing MS-RPC implementations

<table>
<thead>
<tr>
<th></th>
<th>Samba 3</th>
<th>Samba 4</th>
<th>Likewise DCE/RPC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function execution</strong></td>
<td>Synchronous</td>
<td>Asynchronous</td>
<td>Synchronous</td>
</tr>
<tr>
<td><strong>RPC layer status</strong></td>
<td>Status code</td>
<td>Status code returned by <code>_recv</code> function</td>
<td>Exception code</td>
</tr>
<tr>
<td><strong>String encoding</strong></td>
<td>Single byte</td>
<td>Single byte</td>
<td>2-byte</td>
</tr>
<tr>
<td><strong>RPC function result</strong></td>
<td>Status code</td>
<td>Stored in RPC function's I/O struct</td>
<td>Status code returned by rpc function</td>
</tr>
</tbody>
</table>
Changes coming soon:
• More complete set of MS-RPC functions in existing pipes
• Another MS-RPC pipe supported (spoolss)

Changes coming (not necessarily soon):
• Server side
Architecture Overview

MS-RPC client app

- MS-RPC calls
- RPC binding
- Connect
- Credentials

"net use"

libnpc

DCE/RPC library

- Metadata & arguments
- send/revc RPC packets

npcmuxd

libsmbcclient

RPC server

CIFS

Pipe reads & writes

RPC packets

MS-RPC calls

client stub

RPC binding
<table>
<thead>
<tr>
<th>Extra library</th>
<th>NetAPI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MS-RPC</strong></td>
<td></td>
</tr>
<tr>
<td><strong>API</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DCE/RPC</strong></td>
<td></td>
</tr>
<tr>
<td><strong>runtime</strong></td>
<td></td>
</tr>
<tr>
<td><strong>named pipes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>muxer</strong></td>
<td></td>
</tr>
<tr>
<td><strong>libsmbclient</strong></td>
<td></td>
</tr>
</tbody>
</table>
• Designed to be the exact replacement of windows network management library (NetApi)
• Provides the same API and the same header file (LM.h)
• Allows to use the same source code with no changes on both Windows and Linux
• There's more APIs than “official” ones :)
Let's make our life easier...

<table>
<thead>
<tr>
<th></th>
<th>Samr</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arguments</strong></td>
<td>server/domain name username</td>
<td>server/domain name username</td>
</tr>
<tr>
<td><strong>Number of function calls involved</strong></td>
<td>~5</td>
<td>1</td>
</tr>
<tr>
<td><strong>Intermediary arguments</strong></td>
<td>~6 (handles, ids, access rights)</td>
<td>none!</td>
</tr>
</tbody>
</table>
Changes/extensions:
• 100% of original NetAPIs supported
• Server side (\pipe\wkssvc)
• Functions taking 1-byte strings (just for convenience)