21

Infrastructure: The Mailslot

> Nothing lasts longer than
> a provisional arrangement.
>
> — Unknown
> (thanks to Olaf Barthel)

We touched on the Mailslots and Named Pipes back in Section 16.4.1 on page 321, and then we pulled our collective hand away really fast as if those subjects were much too hot to handle. We will need to be brave and give them another go, though, because the Browse Service relies on them. Sorry 'bout that, folks.

Mailslots and Named Pipes are like the wiring and plumbing in an old house.[1] It probably all made sense when it was installed, but over the years new construction has built upon the old. Some parts have been reused, some replaced, and other bits and pieces recycled in ways that make it seem as though no one remembers their original purpose. As long as it looks good on the surface (and isn't a fire hazard), it's okay.

And it really is okay. The old stuff has held up remarkably well. So well, in fact, that it is sometimes forgotten — which is exactly why we need to take a closer look at it.

---

1. "Old" is a relative term. In Minnesota, a 100-year-old house is considered old. In cities like Charleston, SC, the houses go back 300 years or so... and that's nothing compared to what they've got in places like Japan, Europe, the Middle East, etc.

The goal here is to provide a basic understanding of the Named Pipe and Mailslot concepts. We won't be going in too deep. If you want, you can find more detail in the X/Open book *IPC Mechanisms for SMB*. Named Pipes and Mailslots are nifty constructs, and are worthy of further study when you have the time.

21.1 Meet the Plumbing: Named Pipes

As you are by now well aware, SMB is a protocol that implements a network filesystem and, of course, a network filesystem is the result of extrapolating the general concepts that lie behind a disk-based filesystem. The difference is that the network variety uses higher level protocols to stretch things out across a network.

Some disk-based filesystems (such as those used in Unix and its kin) can handle the inclusion of objects that aren't really files at all, but which — through the use of some clever abstraction layers — can be made to look and work like files. For those familiar with such things, common examples include device nodes, the contents of `/proc`, and Named Pipes.

We are interested in the latter.

A Named Pipe is, at its heart, an interprocess communications channel. It allows two programs running independently to exchange messages. The SMB protocol, as you have already guessed, provides support for Named Pipes, but it can stretch them out over the network so that programs on different machines can talk to one another.

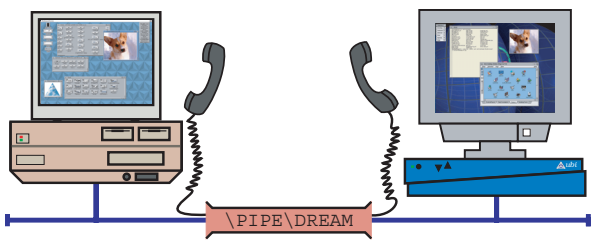

Figure 21.1: *Named Pipes*

An SMB named pipe is an abstraction that provides two-way communication between processes on remote nodes. The pipe is given a name ("`\PIPE\DREAM`", in this example) so that it can be easily identified by programs that wish to use it.

A Named Pipe is "named" so that it can be identified by the programs that want to use it. It is a "pipe" because data is shoved in at one end and then falls gracefully out the other. CIFS Named Pipes have some additional qualities:

They are transported over TCP.
The use of SMB (over NBT) over TCP means that Named Pipe transactions are reliable.

They are built on SMBtrans transactions.
SMBtrans allows for data transfers up to 64K in size, per transaction.

They are bi-directional.
As with other protocols that we have studied, data may be sent and received over the same connection.

They are filesystem abstractions.
CIFS Named Pipes can be opened, closed, read from, and written to.

These features make CIFS Named Pipes ideal for transporting network function calls, which is one of the key ways (but not the only way) they are used. The **R**emote **A**dministration **P**rotocol (RAP) and Microsoft's **R**emote **P**rocedure **C**all implementation (MS-RPC) are both built on top of Named Pipes.

Although they are filesystem abstractions, CIFS Named Pipes are kept separate from the real files and directories made available by the SMB Server Service. They are placed in a special share — the `IPC$` share — which is "hidden." You won't be able to browse to it using the Windows Network Neighborhood tool. If you know it's there, however, you can access it just as you would any other SMB share — specifically, by sending a `SESSION SETUP` followed by a `TREE CONNECT`.

Hidden Expense Alert

Share names that end with a dollar sign ('$') are considered "hidden" shares. It is expected that client software will not display hidden share names unless specifically asked to do so. Note that it is the client, not the server, that takes care of hiding the hidden shares. Samba's `smbclient` *tool and the jCIFS* `List.java` *utility will both happily display hidden share names for you.*

Named Pipes within the IPC$ share have names that match the following format:

\PIPE**pipename**

where *pipename* is determined by the service that created the pipe. Since they are filesystem abstractions, it would be logical to assume that the full name of a Named Pipe (in UNC format) would look something like this:

server\IPC$\PIPE**pipename**

As it turns out, however, the DOS, OS/2, and Windows functions that manipulate Named Pipes abbreviate the name by removing "\IPC$" from the string, which gives:

server\PIPE**pipename**

Named Pipes are created on the SMB server side by applications and tools that are willing to provide specialized services. The architecture is quite analogous to services that register NetBIOS names to make themselves available, it's just that there are more intervening protocol layers which provide additional features. For example, Named Pipes can take advantage of the SMB authentication and MAC signing mechanisms.

Microsoft has created several services that use Named Pipes, but the set of services that are actually available will vary depending upon the host OS and/or the CIFS implementation. Luke K. C. Leighton's book *DCE/RPC over SMB: Samba and Windows NT Domain Internals* (which we have referenced often) lists several known pipes that offer services based on MS-RPC.

Our particular interest, however, is with the specific Named Pipe that will connect us to the **R**emote **A**dministration **P**rotocol service. That pipe is:

\PIPE\LANMAN

We will be using it a little later on, when we dig into the one RAP function that is commonly used by the Browser Service: the NetServerEnum2 function.

...and that is really all we have to say about CIFS Named Pipes.

There is, of course, a lot more that *could* be said. Named Pipes can be used in a variety of ways to support a variety of different kinds of operations. Our goal, however, is to explore the Browse Service, so the scope of this discussion is purposfully limited.

21.2 The Mailslot Metaphor

CIFS supports two kinds of Mailslots :

Class 1: Reliable

Class 1 Mailslots are a whole heck of a lot like Named Pipes. Class 1 messages are packed into an SMBtrans SMB, and then sent across the network using TCP. The only real difference is that Class 1 Mailslot calls indicate only success or failure. They do not return any other results.

We won't be paying any attention to Class 1 Mailslots, because they are not used by the Browse Service.

Class 2: Unreliable and Broadcast

Class 2 Mailslots are a whole 'nother kettle of bananafish. On the surface, they look like Named Pipes and Class 1 Mailslots, but that's just because the interface was designed to be somewhat consistent.

Below the surface, Class 2 Mailslot messages are sent using the NBT Datagram Service. That means that the underlying transport is UDP, not TCP. It also means that Class 2 Mailslot messages can be broadcast or multicast to the local LAN. Another difference is that Class 2 Mailslot messages are simply sent. No result (success, failure, or otherwise) is returned.

Mailslots, like Named Pipes, exist within the `IPC$` share. Mailslot names have a familiar format, as shown below.

`\MAILSLOT\`*mailslotname*

The general form of a Mailslot name. Similar to the convention used for Named Pipes.

`\MAILSLOT\BROWSE`

The Mailslot name used by the Windows NT Browse Service.

`\MAILSLOT\LANMAN`

The Mailslot name used by the older LAN Manager Browse Service.

The Browse Service uses Class 2 Mailslots extensively. With the exception of the `NetServerEnum2` RAP call, all Browse Service announcements and requests are sent using Class 2 Mailslots.

In general, if the destination is local, a Browse Service Mailslot message will be sent as a broadcast at the IP level. Unicast UDP datagrams are used if the destination is on a remote subnet (e.g., a remote DMB). Broadcast messages that contain an NBT group name in the `DESTINATION_NAME` field are considered Multicast at the NBT level. If the `DESTINATION_NAME` is unique, the message may still be broadcast to avoid the need to resolve the name using the NBT Name Service. As suggested in Figure 21.2, the receiver of a broadcast datagram should discard the message if it is not listening on the `DESTINATION_NAME` or the given Mailslot name.

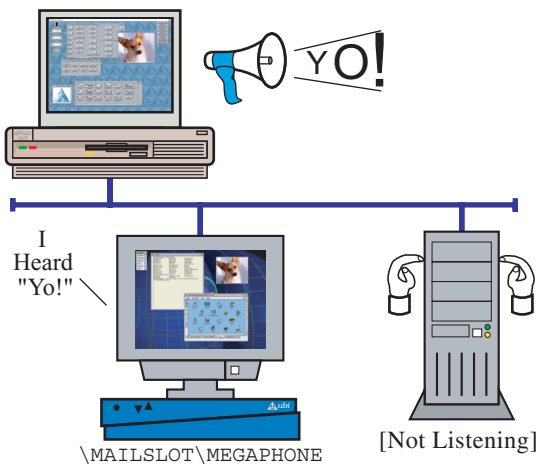

Figure 21.2: *Class 2 Mailslots*

Class 2 Mailslot messages may be broadcast, multicast (NBT group), or unicast. If a message is sent as a broadcast or multicast datagram, all of the NBT nodes on the LAN will receive it. Those that are not listening on the specified Mailslot (`\MAILSLOT\MEGAPHONE`, in this case) or on the specified NBT name should discard it.

Class 2 Mailslot messages are kind of quirky. Even though they use the NBT Datagram Service, they are formatted as `SMB_COM_TRANSACTION` (that is, SMBtrans) messages. That's right: SMB over UDP. Go figure. This may be the result of someone being overly enthusiastic about reusing existing code. In any case, it means that we will be learning how to format an SMBtrans message.

It's Not a Bug It's a Feature Alert

The one-way nature of Mailslot messages has an interesting side-effect, which is this: All *Browse Service Mailslot messages are sent to UDP port 138.*

The reason for this seemingly incorrect behavior is that the receiver's response to a Mailslot message is not really a reply. It is a "response" in the sense of "stimulus-response." When a browser node receives one of the Browse Service Mailslot Messages, it may react by sending messages of its own. In many cases, those messages must be multicast to a NetBIOS group name, so sending them to the source port of the original "stimulus" datagram would simply not work.

(You're getting to love this stuff, aren't you...)

The upshot is that the only way to properly implement the Browse Service (and Class 2 Mailslots in general) is to run a daemon that listens on UDP/138.