Tubes in LING

Maxim Kharchenko, Cloudozer LLP

07/07/2014

1 Overview

Tube is a high-speed data link between two LING domains residing on the same box. The traditional way to communicate between two local domains is to use virtual network interfaces bridged in Dom0. This is undesirable for a few reasons:

1. Virtual interfaces must be configured before the domain starts; no option to add an interface after launch.
2. Linux bridges are slow (400–500Kpps max).
3. It is nearly impossible to isolate traffic from the networking stack in Dom0.

The slugishness of Linux bridges can be aleviated by using netmap-enabled drivers that have their own non-standard bridges. At the same time, netmap imposes rigid constraints on the Linux kernel in Dom0.

Tubes addresses all three issues in a clear and simple way.

There is an early effort to implement an inter-domain transport for Xen similar to a tube (XenSocket). [1]. Tubes in LING are more advanced and are not based on XenSockets.

Tubes use the Xenstore API explained elsewhere [2].

2 Tube API

The Tube API is similar to BSD sockets.

```
tube:open(Domid)        -> {ok,Tube} | {error,_}
tube:open(Domid, Tid)   -> {ok,Tube} | {error,_}
tube:accept()           -> {ok,Tube} | {error,_}
tube:close(Tube)        -> ok | {error,_}
```

[1] http://xen.xensource.com/files/xensummit_4/SuzanneMcIntosh_XenSummit_2007.pdf

[2] xenstore.pdf

The `open()` call takes the domain id of the destination domain. The easiest way to obtain domain id from Erlang is to use `xenstore:domid()` helper function. It is possible to open many tubes between two domain. This is what the second argument of `open()` call is for. It is a unique (numeric) id of the tube.

To open a tube between two domain, one of them should run `open()`, while the other should run `accept()`. The ordering of the calls does not matter.

A tube is a special type of an Erlang port with associated data kept in Xenstore. One can examine the tube configuration and their statuses using external tools, such as `xenstore-ls`.

Sending data to a tube is the same as with any other port.

```
port_command(Tube, Data)
-or-
Tube ! {self(),{command,Data}}
```

Incoming packets are delivered as messages to the process that opened (or accepted) the tube in the following format:

```
{Tube,{data,Data}}
```

Tubes are packet-oriented. The size of packets may vary from 1 to 4096 bytes.

In the current implementation the delivery of packets is not guaranteed. The sender never blocks and if it is too fast the tube will drop packets.

The memory footprint of a tube is 256K.

3 Tearing a tube down

Each side of a tube sense the state of the other side and close automatically when the opposite side closes or the peer domain disappears.

The clean way to close the tube is to call `tube:close(Tube)`. Its effects are mostly the same as simply calling `erlang:port_close(Tube)`.

4 How fast are the tubes?

The preliminary measurements show that tubes can send/receive up to 2Mpps. This is in line with performance of netmap-enabled virtual network interface.

XenSocket project claim that their transport is 100x faster than TCP/IP and is roughly the same as the Linux local domain sockets.