Vios – A Network Proxy Tunnel Between QEMU Host and Guest Systems Using Virtioserial Ports

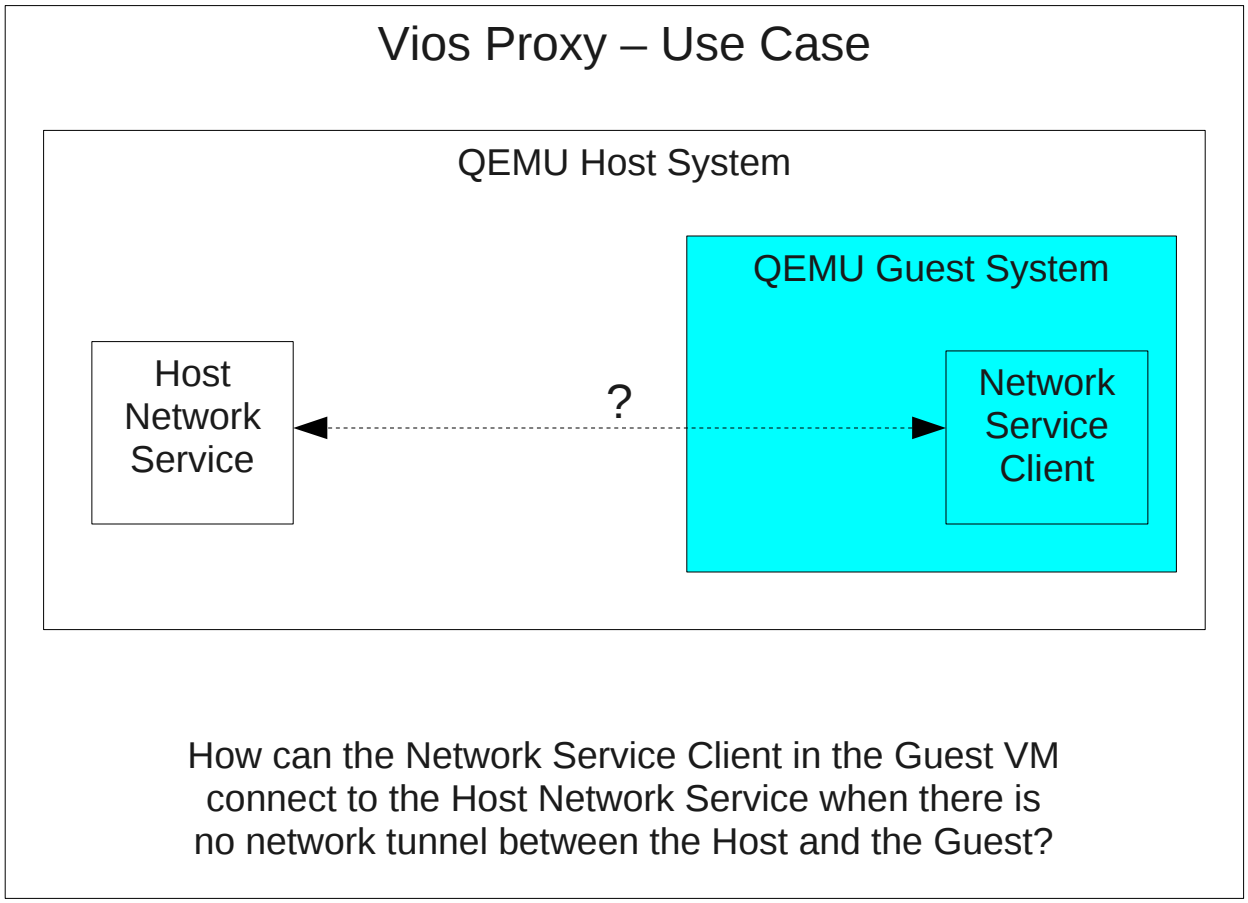
This document describes the motivation and design of a network tunneling mechanism between a virtual machine host and guest. The tunneling is accomplished using virtioserial ports in the absence of any network connections.

Author: Chuck Rolke
Date: June 24, 2011
Version: 1.0

Table of Contents

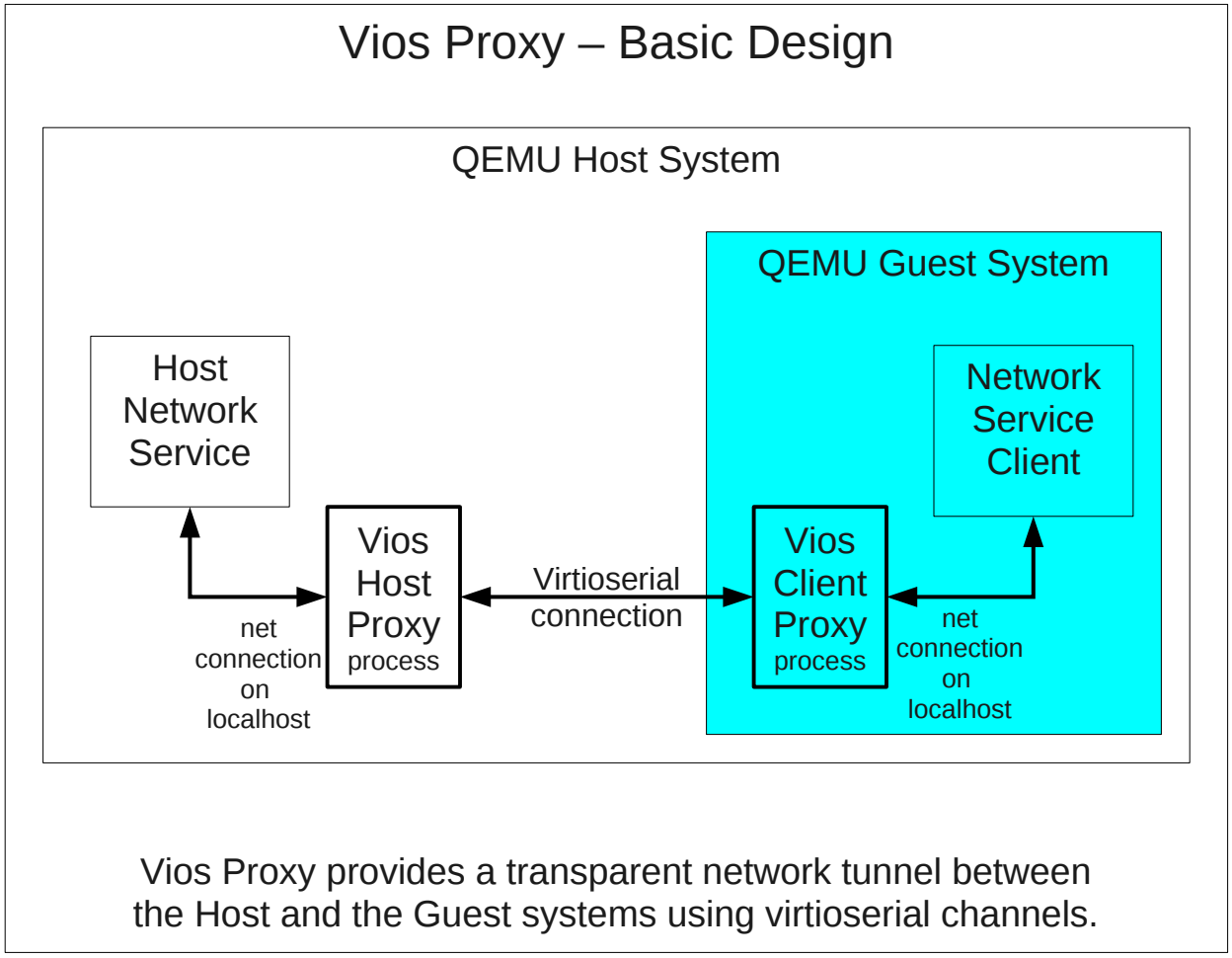
Use Case

Figure 1. Vios Proxy Use Case



Design Overview

Figure 2. Vios Proxy Design Overview



This design has the following features:

1. The QEMU Host system runs one instance of Vios Host Proxy for each proxied network service. This single proxy process provides the proxy service to many guest systems.
2. Each QEMU Guest system runs one instance of Vios Client Proxy for each proxied network service.
3. Each virtioserial connection provides one network connection at a time between the Guest client and the Host service.
4. Virtioserial connections are created when the Guest VM is started. Virtioserial connections are not created by the Vios proxy processes. Virtioserial connections may be created or destroyed dynamically by some other QEMU agent and the Vios proxy processes respond accordingly.

Understanding Virtioserial

VirtioSerial as required by Vios Proxy is supported as of Fedora F13. Please see http://fedoraprojectwiki/Features/VirtioSerial for details.

Virtioserial Connection Host Endpoint

The Virtioserial connection in a QEMU host is a unix domain socket (UDS).

The administrator creating QEMU VM's has complete control over the path name of the UDS.

Virtioserial Connection Guest Endpoint

The Virtioserial connection in a QEMU guest is a file.

The administrator creating QEMU VM's picks a name for the file and that name is created in /dev/virtio-ports.

Create Guests with Virtioserial Channels

Here is a script that launches two VM's, each with three virtioserial ports.

```
#!/bin/sh

#
# Define a common root directory for broker-side channels to guests.
#
QPID_BROKER_ROOT="/tmp/qpid"

#
# Launch some number of guests in a loop.
#   On the host the channels to the first guest shall be
#      /tmp/qpid/guest1/0
#      /tmp/qpid/guest1/1
#      /tmp/qpid/guest1/2
#   Successive guests increment guest1 to guest2, guest3, and so on.
#
#   On the guest the channels to the host shall be
#      /dev/virtio-ports/qpid.0
#      /dev/virtio-ports/qpid.1
#      /dev/virtio-ports/qpid.2
#   The guest-to-host channel names are the same for all guests.
#
#
for gNameSeq in `seq 1 2`
do
    GUEST_NAME="guest"$gNameSeq

    mkdir -p ${QPID_BROKER_ROOT}/${GUEST_NAME}

    qemu-kvm -name "${GUEST_NAME}" -m 192M -smp 2 -snapshot ~chug/v/images/v14a.img \
        -device virtio-serial \
        -chardev socket,path=${QPID_BROKER_ROOT}/${GUEST_NAME}/0,server,nowait,id=${GUEST_NAME}_0 \
        -chardev socket,path=${QPID_BROKER_ROOT}/${GUEST_NAME}/1,server,nowait,id=${GUEST_NAME}_1 \
        -chardev socket,path=${QPID_BROKER_ROOT}/${GUEST_NAME}/2,server,nowait,id=${GUEST_NAME}_2 \
        -device virtserialport,chardev=${GUEST_NAME}_0,name=qpid.0 \
        -device virtserialport,chardev=${GUEST_NAME}_1,name=qpid.1 \
        -device virtserialport,chardev=${GUEST_NAME}_2,name=qpid.2 \
        &
    PID=$!
    echo "QEMU ${GUEST_NAME} PID = $PID"
done
```

Launch vios-host-proxy

vios-host-proxy command line

```
[chug@localhost build]$ ./vios_host_proxy -h
usage: ./vios_host_proxy [guest_dir [service_port [log_level]]]
where
 guest_dir    - path containing directories of virtioserial endpoints to guests.
                Default = /tmp/qpid
 service_port - the service port on localhost that is proxied to the guests.
                Default = 5672
 log_level    - log verbosity setting.
                One of FATAL, ALERT, ERROR, WARN, NOTICE, INFO, DEBUG.
                Default = INFO
```

vios-host-proxy process details

The vios-host-proxy process must be privileged enough to access the host endpoint domain sockets.

Launch vios-client-proxy

vios-client-proxy command line

```
[chug@localhost build]$ ./vios_client_proxy -h
usage: ./vios_client_proxy [host_dir [service_port [log_level]]]
where
 host_dir     - path containing virtioserial endpoints to the host.
                Default = /dev/virtio-ports
 service_port - the service port on localhost that is proxied to the guests.
                Default = 5672
 log_level    - log verbosity setting.
                One of FATAL, ALERT, ERROR, WARN, NOTICE, INFO, DEBUG.
                Default = INFO
```

vios-client-proxy process details

The vios-client-proxy process must be privileged enough to access the guest endpoint files.

Vios Proxy Source Code

The source code is available on Red Hat Engineering Git.

http://git.engineering.redhat.com/?p=users/crolke/virtio-serial-work.git;a=summary

In the 'tree' view browse to qpid-vios-proxy, a very simple CMake project.

Vios Proxy Design Considerations

Channel Naming Conventions

Logging

Vios Framing Protocol

Performance

Unit Test