virtio-vsock

Zero-configuration host/guest communication

Stefan Hajnoczi <stefanha@redhat.com>

KVM Forum 2015

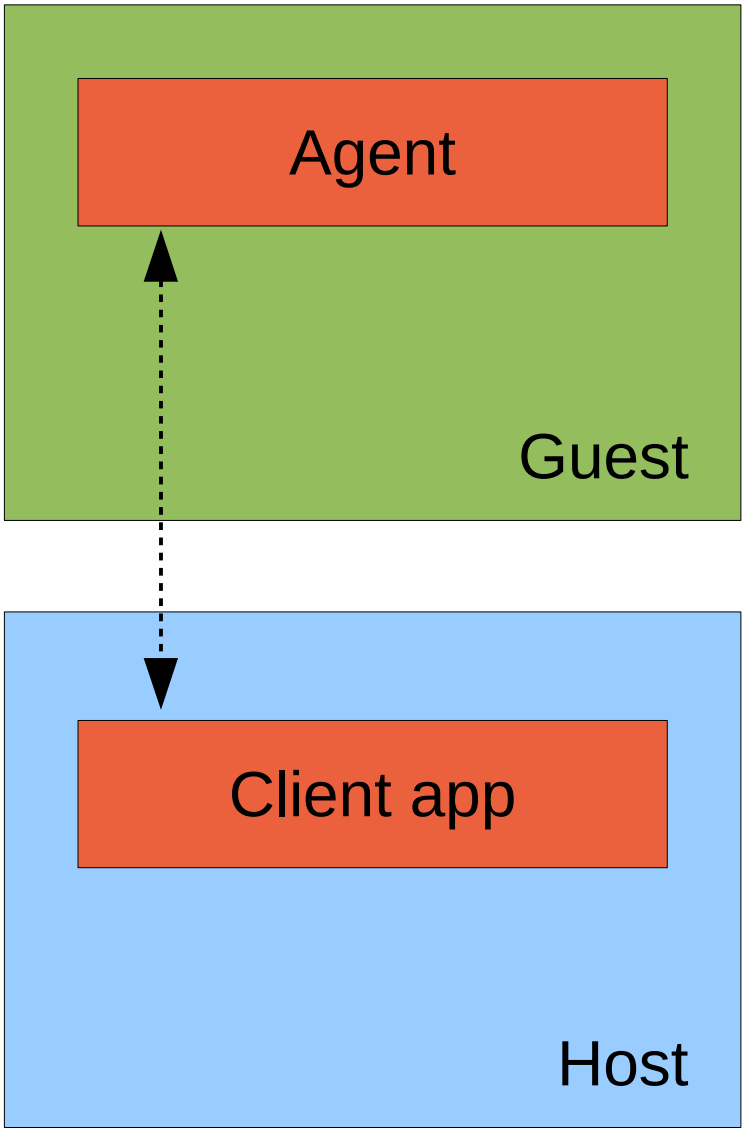
Agenda

- Host/guest communication use cases
- Overview of virtio-serial
- Desirable features that are missing
- Overview of virtio-vsock
  - AF_VSOCK address family
  - Communications protocol
- Porting code to virtio-vsock
- Status of virtio-vsock

Host/guest communication use cases

Communications channel between virtual machine and hypervisor.

- qemu-guest-agent
  - Backups, suspend, etc
- SPICE vdagent
  - Clipboard sharing, etc
- Custom agents
- Host services (file sharing)



virtio-serial overview

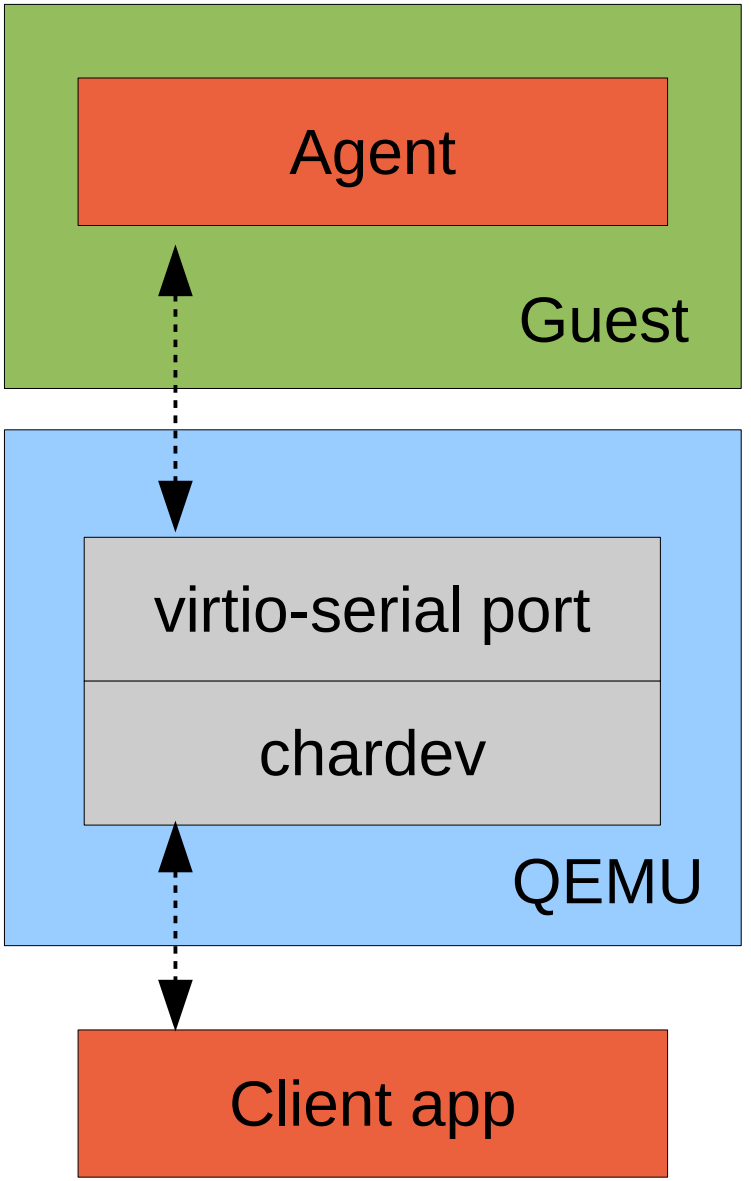
Virtio device for **serial ports** and console

Host can add/remove ports

Ports can be named (e.g. org.qemu.guest_agent.0)

Guest agent opens /dev/virtio-ports/<name> character device

Client app connects to QEMU chardev (UNIX domain socket, named pipe, etc)



Limitations of virtio-serial

- N:1 connections are clunky over 1:1 serial port
  - Applications have to multiplex over 1 stream
  - Libvirt has to arbitrate access between qemu-guest-agent clients
- Relatively low number of ports available (~512)
  - Limit is hardcoded by host
- Stream semantics (no message boundaries)
  - Ugly for datagram protocols
- Applications must use character devices instead of familiar sockets API...

Sockets API – standard for communication

The familiar Berkeley/POSIX sockets API:

- socket()/bind()/listen()/accept()/connect()/etc
- Used by many programs

Character devices do not support sockets API

- Can't share TCP/IP and virtio-serial code

Sounds like we want **networking**...

Possible solution: Ethernet

Pro: TCP/IP and NIC support already exists

Con:

- Adding & configuring guest interfaces is invasive
- Prone to break due to config changes inside guest
- Creates network interfaces on host that must be managed

No other hypervisor uses Ethernet for host/guest communication...they hit the same problems.

AF_VSOCK in Linux

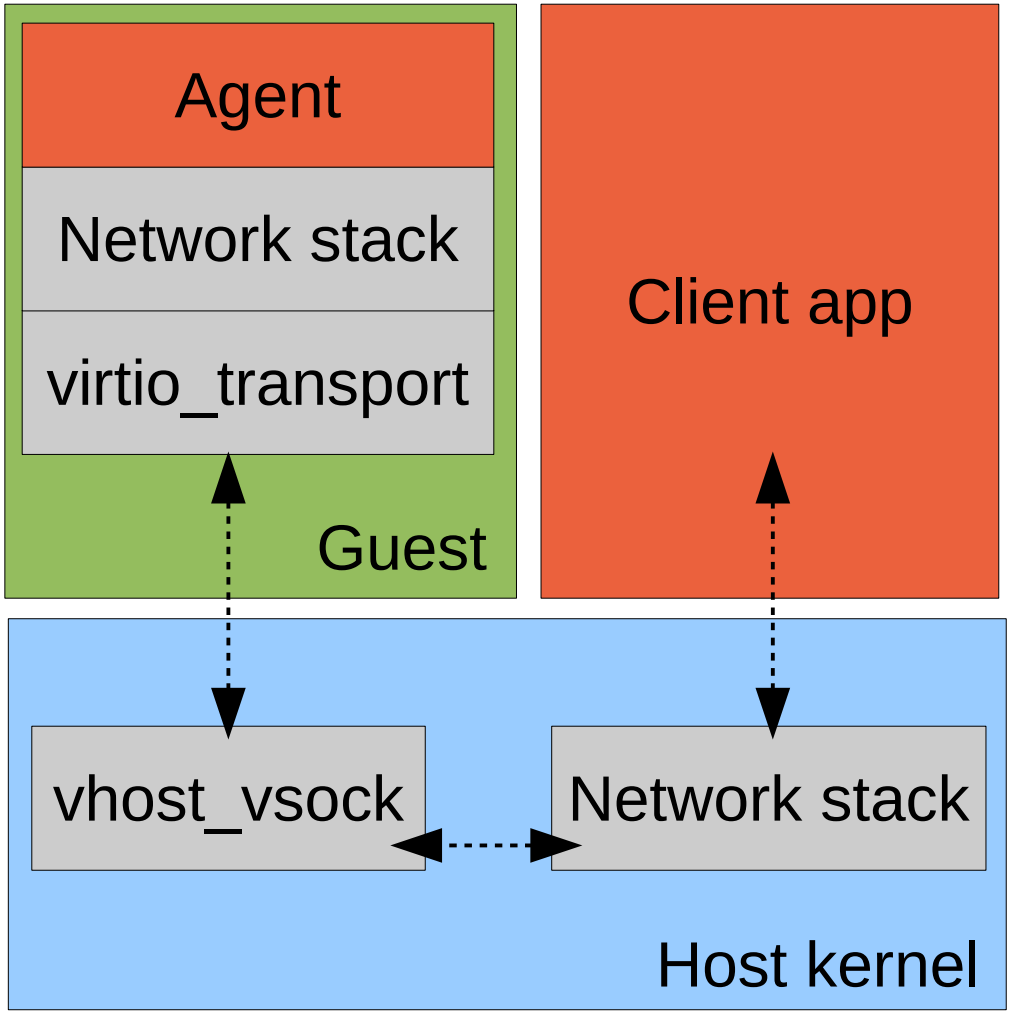
- New socket address family for host/guest communication
- Supports datagram and stream semantics
- Addresses are <u32 cid, u32 port>
  - Each guest has unique cid
  - Host has well-known cid
- Contributed to Linux by VMware in 2013
- Gerd Hoffmann and Asias He prototyped a virtio transport for vsock

vhost-vsock architecture

Uses vhost driver framework to integrate with host network stack

Both guest and host applications use sockets API

Socket types:
SOCK_STREAM (connection-oriented, reliable, ordered)
SOCK_DGRAM (connectionless, unreliable, unordered)



virtio-vsock device design

rx/tx virtqueue pair for data exchange

- Each packet has a header with addressing and state information
- Packets from all flows use single rx/tx virtqueue pair
- Credit-based flow control for reliable connections
- Best-effort for unreliable connections

Control virtqueue

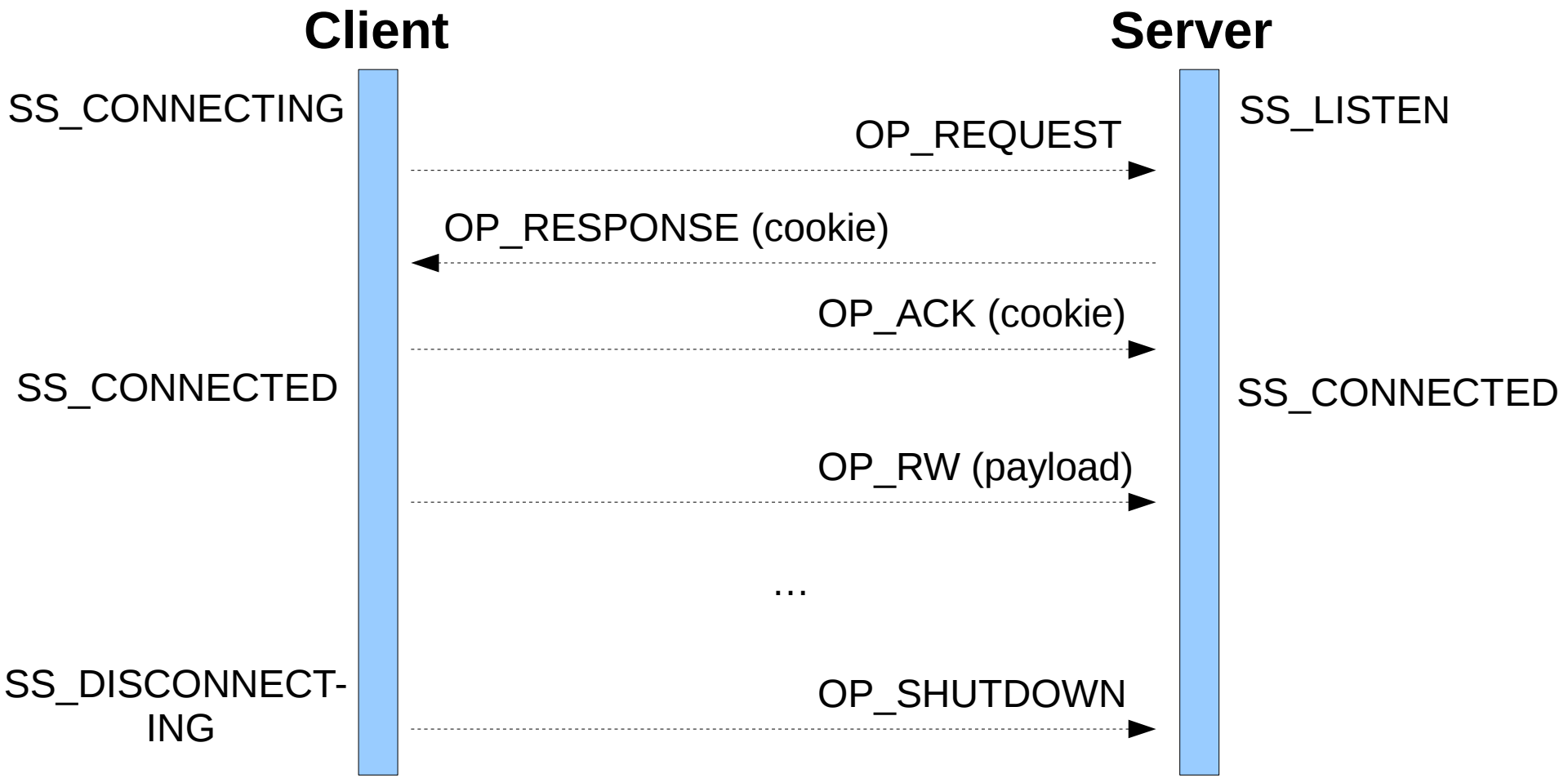
- Reserved for future use

Configuration space

- Guest cid field (filled in by host)

virtio-vsock communications protocol

Example lifecycle of stream socket type:



Porting code to AF_VSOCK

Similar challenges to IPv6 porting:

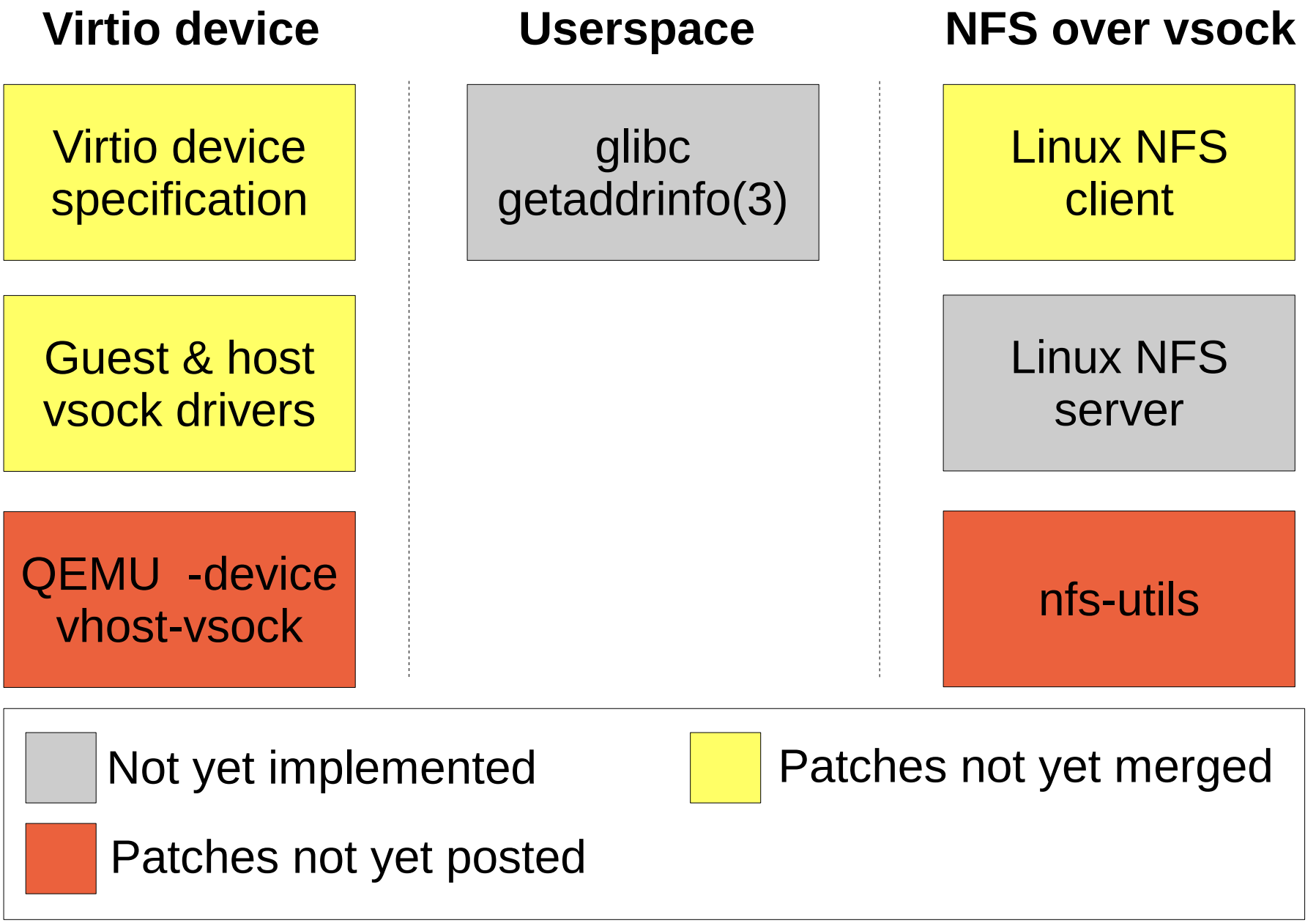
1) Use `socket(AF_VSOCK, type, 0)`

2) Use vsock addressing:

```
struct sockaddr_vm sa = {
    .svm_family = AF_VSOCK,
    .svm_cid    = VMADDR_CID_HOST,
    .svm_port   = 1234,
};
```

The rest is standard sockets API usage.

Status of patches



Questions?

Email: stefanha@redhat.com

IRC: stefanha on #qemu irc.oftc.net

Blog: http://blog.vmsplice.net/

Specification: http://goo.gl/mi6LCR

Code:

- https://github.com/stefanha/linux vsock
- https://github.com/stefanha/qemu vsock

Slides available on my website: http://vmsplice.net/