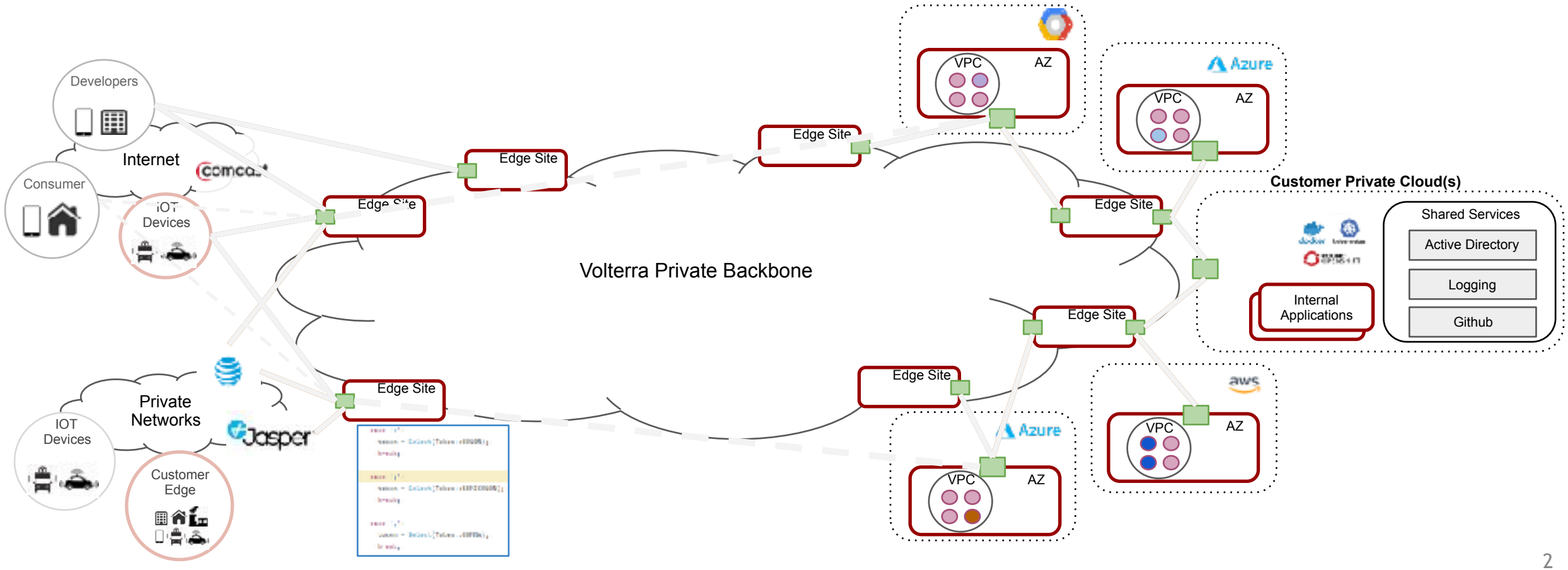
Integrating Envoy with DPDK-based virtual networks

Raja Sivaramakrishnan
Volterra

Volterra infrastructure



Linux kernel overhead

- System calls
- Interrupt processing
- Copying data between user and kernel space
- Context switches when blocking on I/O

Few hundred thousand packets per second

DPDK design

- Polling (eliminates interrupt overhead)
- User-space IO (eliminates user-kernel copies)
- Huge pages (reduces TLB misses)
- Lockless synchronization across cores
- Processor affinity
- Batch processing of packets

Millions of packets per second

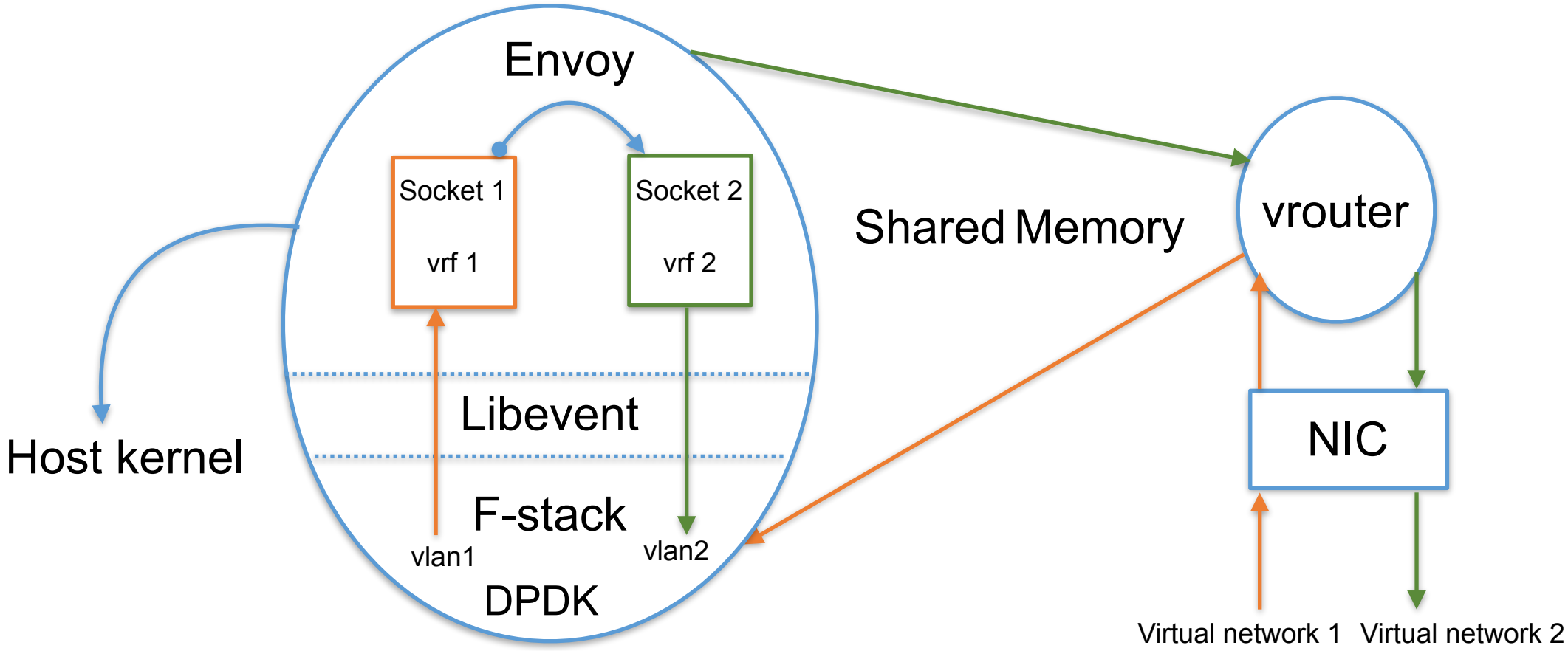
DPDK-based virtual networking

- Multi-tenancy using virtual networks
- Each virtual network has its own VRF
- Overlay tunnels to avoid any impact on physical network infrastructure
- Virtual router (vrouter) module integrated with DPDK
- Applications run on top of vrouter, in a container or VM
- Performance is best when application uses DPDK too
- Envoy is run as an application over vrouter

User-space network stack

- Envoy modified to use user-space network stack instead of kernel network stack
- Network stack based on F-stack (FreeBSD stack+ DPDK)
- Envoy linked with F-stack and modified libevent libraries to run as a single process in its own container
- Shared memory between vrouter and Envoy process implemented using user-space vhost
- Could be extended to have multiple instances of Envoy processes running over single vrouter instance

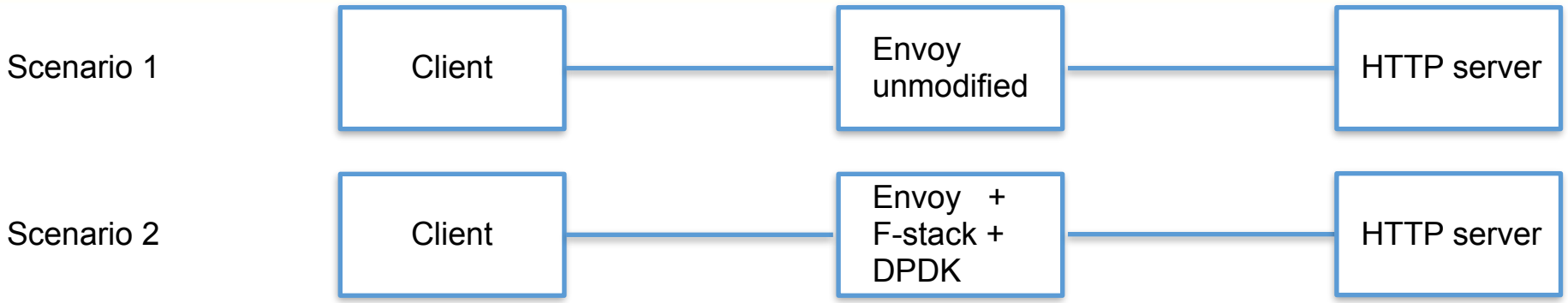
Packet forwarding path



Envoy/library modifications

- Bazel external repositories for libevent, F-stack and DPDK
- libevent polls continuously for packets by calling into F-stack (instead of sleeping in epoll/kqueue)
- libevent calls into F-stack to listen/accept connections
- Support for af_packet and virtio interfaces in F-stack config
- F-stack APIs to add a socket/interface to a vrf
- Support for multiple vrfs in Envoy
- Support for polling in Envoy dispatcher
- Support for alternative network stack on a per-socket basis
- Listen socket creation moved to worker thread from main thread

Performance numbers



VMs in Azure cloud

	HTTP requests per second	Latency
Scenario 1	172	4.23 ms
Scenario 2	504	2.07 ms

Summary

- Main contributions are:
  - Support for multi-tenancy in Envoy
  - Improved Envoy performance using DPDK and F-stack
- Plan to publish a blog in the near future
- Other applications
- More information at www.ves.io

Thank you


envoycon