Network Stack as a Service in the Cloud

Zhixiong Niu\textsuperscript{1}, Hong Xu\textsuperscript{1}, Dongsu Han\textsuperscript{2}, Peng Cheng\textsuperscript{3}, Yongqiang Xiong\textsuperscript{3}, Guo Chen\textsuperscript{3}, Keith Winstein\textsuperscript{4}

\textsuperscript{1}City University of Hong Kong
\textsuperscript{2}KAIST
\textsuperscript{3}Microsoft Research Asia
\textsuperscript{4}Stanford University
Imagine you’re a tenant. You want to deploy a new stack.
Motivation: Tenants

I heard that BBR is great. Let’s deploy it to my VMs!
Motivation: Tenants

I heard that BBR is great. Let’s deploy it to my VMs!
Motivation: Tenants

I heard that BBR is great. Let’s deploy it to my VMs!
Motivation: Tenants

I heard that BBR is great. Let’s deploy it to my VMs!

Problem: cannot deploy a stack across OSes
Motivation: Tenants

I heard that BBR is great. Let’s deploy it to my VMs!

Problem: cannot deploy a stack across OSes
Motivation: Tenants
Motivation: Tenants

No. of commits of mTCP and F-stack in 2017
Motivation: Tenants

Problem: high deployment and maintenance cost
Motivation: Tenants

No. of commits of mTCP and F-stack in 2017

Problem: high deployment and maintenance cost
So your life as a tenant sucks. What about the cloud provider?
Motivation: Provider

I know that BBR is great.
Let me deploy it for my tenants!
Motivation: Provider

I know that BBR is great. Let me deploy it for my tenants!
Motivation: Provider

I know that BBR is great. Let me deploy it for my tenants!

Problem: can’t touch the tenant stack
Motivation: Provider

I know that BBR is great. Let me deploy it for my tenants!

Problem: can't touch the tenant stack
So what’s wrong here?
Current architecture
Network stack is coupled to the guest OS

Current architecture
Interface unchanged (BSD sockets, etc.)
Packets handled in the NSM

Tenant

Provider

Interface unchanged (BSD sockets, etc.)

Network stack module

Network Stack

Networking API

APP1

APP2

VM
Vision: Network Stack as a Service

Tenant

Provider

Network Stack module

Network Stack

VM

APP1

APP2

Networking API

Packets handled in the NSM

Interface unchanged (BSD sockets, etc.)
What’re the benefits?
Flexibility for Tenants

![Diagram showing the flexibility for tenants with VMs and NSM connections through mTCP and BBR.]
Flexibility for Tenants

- Stack independent of the guest OS
Flexibility for Tenants

- Stack independent of the guest OS
- No deployment or maintenance cost
Efficiency for Provider
Efficiency for Provider

- Offer meaningful SLAs

<table>
<thead>
<tr>
<th>NSM</th>
<th>Capacity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>mTCP</td>
<td>25Mpps</td>
<td>$2/hr</td>
</tr>
<tr>
<td>mTCP</td>
<td>50Mpps</td>
<td>$4/hr</td>
</tr>
<tr>
<td>F-Stack</td>
<td>20Mpps</td>
<td>$2/hr</td>
</tr>
</tbody>
</table>
Efficiency for Provider

- Offer meaningful SLAs
- Optimize resource utilization

<table>
<thead>
<tr>
<th>NSM</th>
<th>Capacity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>mTCP</td>
<td>25Mpps</td>
<td>$2/hr</td>
</tr>
<tr>
<td>mTCP</td>
<td>50Mpps</td>
<td>$4/hr</td>
</tr>
<tr>
<td>F-Stack</td>
<td>20Mpps</td>
<td>$2/hr</td>
</tr>
</tbody>
</table>

BBR NSM
Efficiency for Provider

- Offer meaningful SLAs
- Optimize resource utilization

<table>
<thead>
<tr>
<th>NSM</th>
<th>Capacity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>mTCP</td>
<td>25Mpps</td>
<td>$2/hr</td>
</tr>
<tr>
<td>mTCP</td>
<td>50Mpps</td>
<td>$4/hr</td>
</tr>
<tr>
<td>F-Stack</td>
<td>20Mpps</td>
<td>$2/hr</td>
</tr>
</tbody>
</table>

- Easier to assert coordination and control
Efficiency for Provider

- Offer meaningful SLAs
- Optimize resource utilization
- Easier to assert coordination and control

<table>
<thead>
<tr>
<th>NSM</th>
<th>Capacity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>mTCP</td>
<td>25Mpps</td>
<td>$2/hr</td>
</tr>
<tr>
<td>mTCP</td>
<td>50Mpps</td>
<td>$4/hr</td>
</tr>
<tr>
<td>F-Stack</td>
<td>20Mpps</td>
<td>$2/hr</td>
</tr>
</tbody>
</table>

BBR NSM

pHost
NUM Fabric
mon.
Accelerate Innovation

mTCP Oct 2017

mTCP Nov 2017

mTCP Dec 2017

…
Accelerate Innovation

- Allow stack to evolve independently with the guest OS
- Write once, run everywhere
Accelerate Innovation

- Allow stack to evolve independently with the guest OS
- Write once, run everywhere

Not possible in current architecture
NetKernel

VM
- APP1
- APP2
- Network API

NSM
- Network Stack
- vNIC

Hypervisor
- Virtual Switch / Embedded Switch (SR-IOV)

Physical NICs
- pNICs
NetKernel

VM

- APP1
- APP2
- Network API
- GuestLib

NSM

- Network Stack
- vNIC

Hypervisor

- Virtual Switch / Embedded Switch (SR-IOV)

Physical NICs

- pNICs
NetKernel

Hypervisor

Virtual Switch / Embedded Switch (SR-IOV)

Physical NICs

pNICs

Socket API

GuestLib

Network API

APP1

APP2

VM

Huge page

NSM

Network Stack

ServiceLib

vNIC

Physical NICs

pNICs
NetKernel

VM

APP1

APP2

Network API

GuestLib

NSM

Network Stack

ServiceLib

CoreEngine

Hypervisor

Virtual Switch / Embedded Switch (SR-IOV)

Physical NICs

pNICs

Socket API

Data

Huge page

Queue
Microbenchmark

- 3000 lines of C code, in user space
- QEMU KVM 2.5.0, Linux Kernel 4.9
- Intel Xeon CPU E5-2618L v3 @ 2.30GHz x 2

Communication between ServiceLib and GuestLib
(Random read and copy)

<table>
<thead>
<tr>
<th>Chunk size</th>
<th>64B</th>
<th>512B</th>
<th>1KB</th>
<th>2KB</th>
<th>4KB</th>
<th>8KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>8ns</td>
<td>64ns</td>
<td>117ns</td>
<td>214ns</td>
<td>425ns</td>
<td>809ns</td>
</tr>
</tbody>
</table>
Microbenchmark

- 3000 lines of C code, in user space
- QEMU KVM 2.5.0, Linux Kernel 4.9
- Intel Xeon CPU E5-2618L v3 @ 2.30GHz x 2

Communication between ServiceLib and GuestLib
(Random read and copy)

<table>
<thead>
<tr>
<th>Chunk size</th>
<th>64B</th>
<th>512B</th>
<th>1KB</th>
<th>2KB</th>
<th>4KB</th>
<th>8KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>8ns</td>
<td>64ns</td>
<td>117ns</td>
<td>214ns</td>
<td>425ns</td>
<td>809ns</td>
</tr>
</tbody>
</table>

64Gbps       81Gbps
Windows VM + BBR NSM

Throughput (Mbps)

- Win + NSM BBR: 12 Mbps
- Linux BBR: 12 Mbps
- Windows CTCP: 6 Mbps
- Linux CUBIC: 3 Mbps

Beijing to California with 350ms rtt and 12Mbps Uplink.
Takeaway

- **Vision: Network Stack as a Service**
  - Decouple the network stack from the guest OS
  - Better flexibility and efficiency, and faster innovation

- **NetKernel as a solution**
  - GuestLib, ServiceLib, CoreEngine
Research Agenda
Research Agenda

- NSM form
  - VM? unikernel-based VMs? containers? hypervisor modules?
Research Agenda

- NSM form
  - VM? unikernel-based VMs? containers? hypervisor modules?

- Support for containers
  - Currently a container has to use the host stack
  - Different containers on the same host use different stacks
Research Agenda

- NSM form
  - VM? unikernel-based VMs? containers? hypervisor modules?

- Support for containers
  - Currently a container has to use the host stack
  - Different containers on the same host use different stacks
Research Agenda

- NSM form
  - VM? unikernel-based VMs? containers? hypervisor modules?

- Support for containers
  - Currently a container has to use the host stack
  - Different containers on the same host use different stacks

- Network stacks to NSMs

...
Open Questions

- Any downsides?
- Other use cases in a production cloud?
- How about a private data center?
- What’s the right abstraction boundary of the network stack?