Network Stack as a Service in the Cloud

HotNets-XVI Dialogue

Luigi Rizzo and Jennifer Rexford

LR: This paper looks at a crowded space: whether and how to provide alternative implementations of network stacks and transport protocols, particularly for virtual machines.

JR: Indeed, the paper is part of an ongoing debate about how TCP functionality should be (re)factored, and where different parts of this functionality should run—a theme that runs across several of the papers in this year's HotNets program. The questions are timely, as multi-tenant data centers have multiple stakeholders, including individual tenants (that typically run a full TCP stack of their choosing within their virtual machines) and the cloud provider (that is ultimately responsible for dividing network resources across connections, VMs, and tenants). The never-ending march toward higher link speeds may also lead us to different answers to the question of "what functionality should run where?" over time.

LR: So, do you "buy" the paper's motivations for offering the network stack as a service?

JR: The main question I have is whether tenants really want to pick their own network stack, rather than just inherit the state-of-the-art choices the cloud provider makes. If tenants do not care to pick their own network stack, why not just offer them a container model in the first place, and let the cloud provider enforce a particular network stack in that way? Do tenants care because of traffic that goes from servers running in the tenant's VM to remote client hosts, carrying traffic with unique performance requirements or using network paths that may have unique performance properties? Or, is there some more mundane reason that the VM model is easier to use than containers?

LR: Containers are generally cheaper in terms of resource usage, but much less flexible. Users may need to run a completely different operating system, or simply prefer to have stricter control on the scheduling of processes. For these and other use cases, VMs are easier to use or may even be the only choice.

LR: Do you see other potential uses of this approach?

JR: The NetKernel approach of running the network stack as a service may have some other benefits, beyond the ones emphasized in the paper. First, a network-stack service could simplify network troubleshooting, compared to traditional settings where the network stack lies outside the cloud provider's visibility and control. By running the network stack, the cloud provider could leverage techniques like Web10G to monitor TCP performance effectively—something that is easy today in private clouds (where the provider controls the network stack) but not in public clouds (where the tenant controls the network stack). Second, NetKernel could allow the cloud provider to forgo adding another layer of packet shaping or scheduling to enforce fairness across connections, virtual machines, or tenants. These functions could be integrated directly in NetKernel rather than needing to run a second layer of functionality in (say) the hypervisor or the network interface card.

LR: I also see other interesting applications of the NetKernel approach. The FUSE userspace filesystem uses exactly the same technique, with a kernel module that intercepts all file system operations and redirects them to userspace. This allows access to all sort of services as filesystems. In a similar way, the NetKernel approach would make it easier to experiment with solutions that integrate multiple functions (say, encryption, tunnels, shaping and congestion control) without having to perform them at the packet level.

JR: Do you find the performance results compelling enough to argue for using NetKernel in real, operational settings?

LR: Perhaps we should first define “performance.” :(Is that speed, latency, CPU usage, or more generally how well the system interacts with the environment? The paper has an interesting experiment showing that on a poor network path, using NetKernel to replace the native TCP congestion control almost doubles the throughput, though still otherwise limited to 12 Mbit/s. This to me is a good performance improvement, and I think the vast majority of VM users need relatively low speeds, so any potential CPU overhead caused by the extra hop through NetKernel will be negligible.

JR: But what about “power users” who need to run at full link speed? The paper does show that the network stack module can achieve 40Gbit/s. Will it remain able to achieve line rate even with faster interfaces?

LR: I am afraid we do not have enough data. Surely the extra memory copies and hops through processes in the host do not come for free. A benchmark that saturates one of the components of the system (the NIC in this case) does not tell us anything about the others—CPU usage, memory bandwidth, cache footprint. For NetKernel, those are key questions. Will there be enough spare memory bandwidth to support the extra copying? How much extra CPU are we using, also considering that we are going to pollute the cache and reducing the overall performance of the CPU?

JR: Do you expect that cloud providers will embrace this idea?

LR: It will depend on whether they have enough incentives and find a suitable economic model. You already mentioned some features that cloud providers may find interesting for themselves or for the users. Offloading additional work to the host (to run the guest’s network protocol stack) will require the provider to allocate the extra CPU cycles and memory, possibly charging customers for usage, and to protect itself from malicious users.