XenSummit Design Session

XenSockets: Past, Present and Future

Alexander Jung <alexander.jung@neclab.eu>
PhD Student <a.jung@lancs.ac.uk>

School of Computing & Communications | Lancaster University

Orchestrating a brighter world
NEC

This discussion

- XenSocket and existing systems towards IpVMC;
- The design and implementation of Unikraft's `posix-socket` module to provide this socket abstraction (thus achievable I*p*VMC of unikernels with minimal application alteration);
- Applications and use cases.

Existing systems

- Zhang, Xiaolan, et al. "**XenSocket: A high-throughput interdomain transport for virtual machines.**" Springer, Berlin, Heidelberg, 2007.

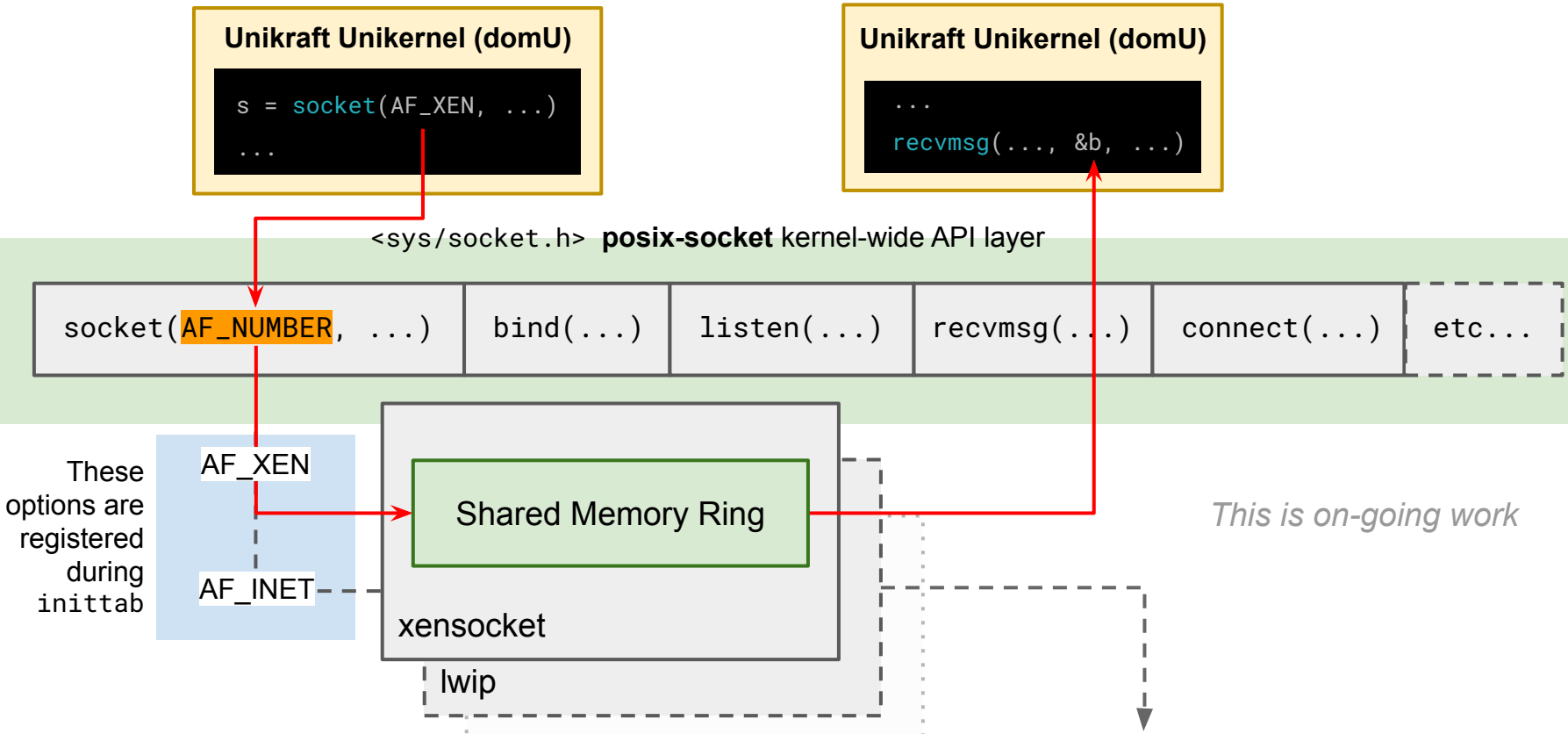
- Rizzo, Luigi. "**Netmap: a novel framework for fast packet I/O.**" 21st USENIX Security Symposium (USENIX Security'12). 2012.
  - ClickOS

- Jeong, EunYoung, et al. "**mTCP: a highly scalable user-level TCP stack for multicore systems.**" 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI'14). 2014.

- Wiesböck, Johannes, Johannes Naab, and Henning Stubbe. "**Virtio-Vsock-Configuration-Agnostic Guest/Host Communication.**" Network 73 (2019).

- *Argo: Hypervisor-Mediated Exchange (HMX) for Xen*

Unikraft's `posix-socket` module for I*p*VMC



Registration of a socket implementation

```c
#include "my_socket.h"

int
mysock_glue_create(struct posix_socket_driver *d,
         int family, int type, int protocol)
{
  return my_socket(etc, family, type, protocol);
}
```

- Allows for “non-POSIX/-standard” implementations (*see LwIP, picoTCP…*)
- Additional state can be abstracted away

```c
struct posix_socket_driver {
  /* The AF family ID */
  const int af_family;
  const char *libname;
  /* The interfaces for this socket */
  const struct posix_socket_ops *ops;
  /* The memory allocator for this socket driver */
  struct uk_alloc *allocator;
  /* Entry for list of socket drivers. */
  UK_TAILQ_ENTRY(struct posix_socket_driver) _list;
};
```

- Custom memory allocator can be set for the implementation.

Registration of a socket implementation

```
static struct posix_socket_ops mysock_ops = {
  .init        = mysock_lib_init,
  .create      = mysock_glue_create,
  .accept      = mysock_glue_accept,
  .bind        = mysock_glue_bind,
  .shutdown    = mysock_glue_shutdown,
  .connect     = mysock_glue_connect,
  .listen      = mysock_glue_listen,
  .recv        = mysock_glue_recv,
  .recvfrom    = mysock_glue_recvfrom,
  .recvmsg     = mysock_glue_recvmsg,
  .send        = mysock_glue_send,
  ...
};

POSIX_SOCKET_FAMILY_REGISTER(AF_INET, &mysock_ops, NULL);
```

Unikraft uses a similar registration semantic to LKMs.

With Unikraft, instead of using a special AF family number (in the case of `AF_XEN` or `AF_VSOCK`), we can replace the stack entirely.

Possible end use-cases / discussion points

- But *what is a socket?*[1]
- FaaS and VNF
- Orchestrated systems
  - Unikernels in Kubernetes pods?
  - What are the limitations of container use cases as comparison?
- Mobile Edge Computing? (Xen on Arm)

1. Chan, Michael, Heiner Litz, and David R. Cheriton. "**Rethinking network stack design with memory snapshots.**" Presented as part of the 14th Workshop on Hot Topics in Operating Systems. 2013.