Implementation of a User-Level-Firewall in Linux

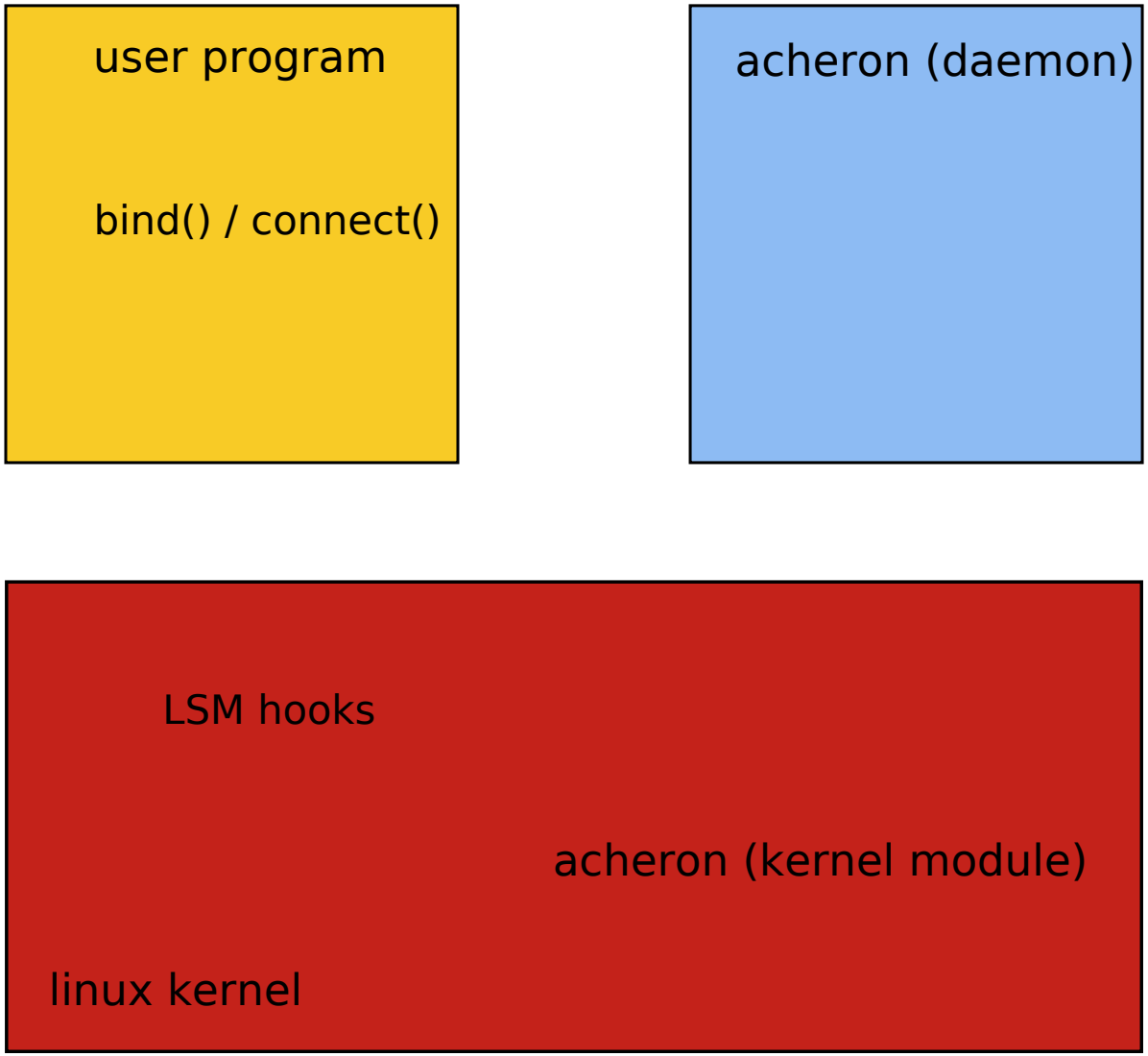
Johannes Bauer, Severin Strobl

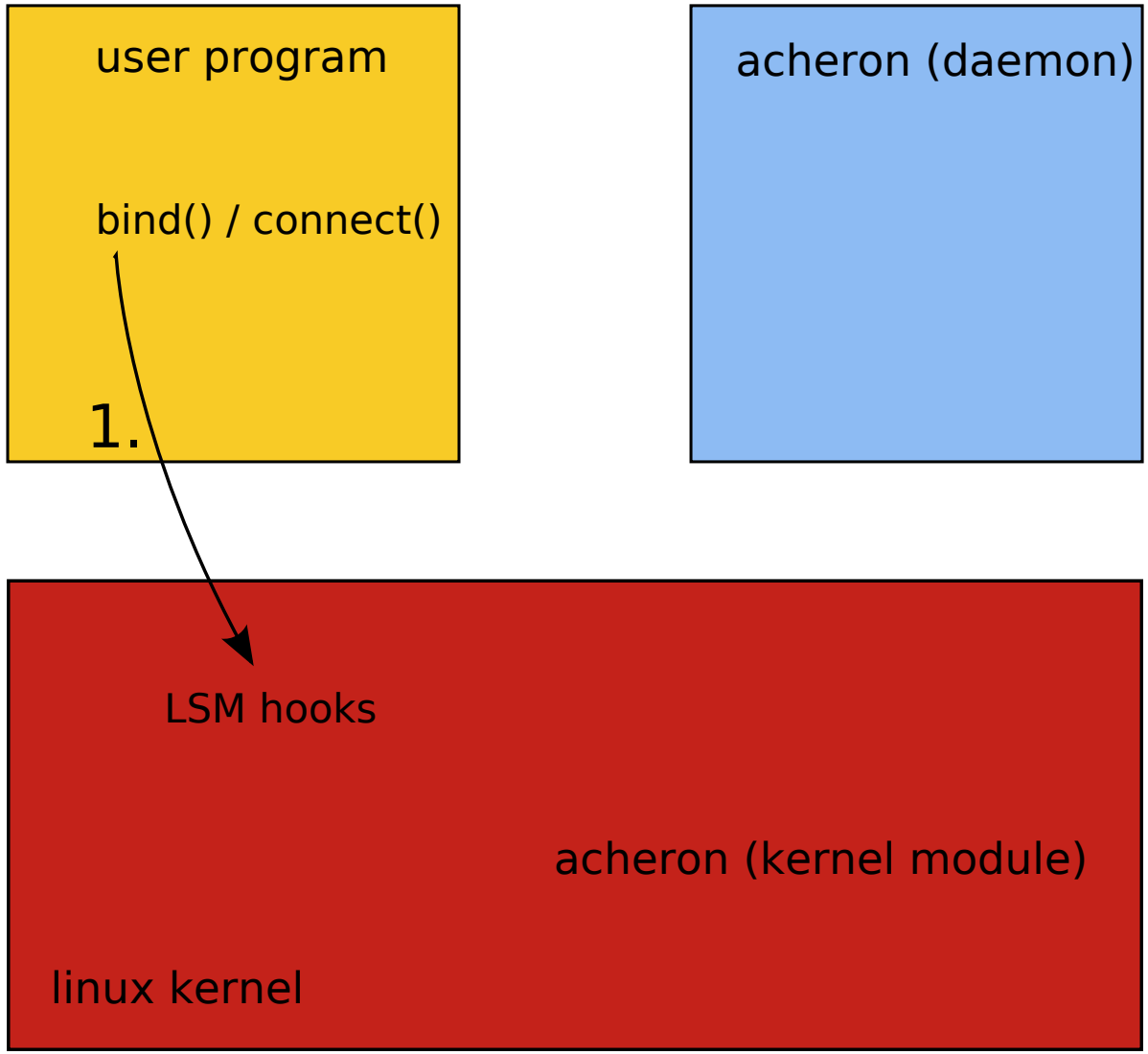
Department of Computer Science 4

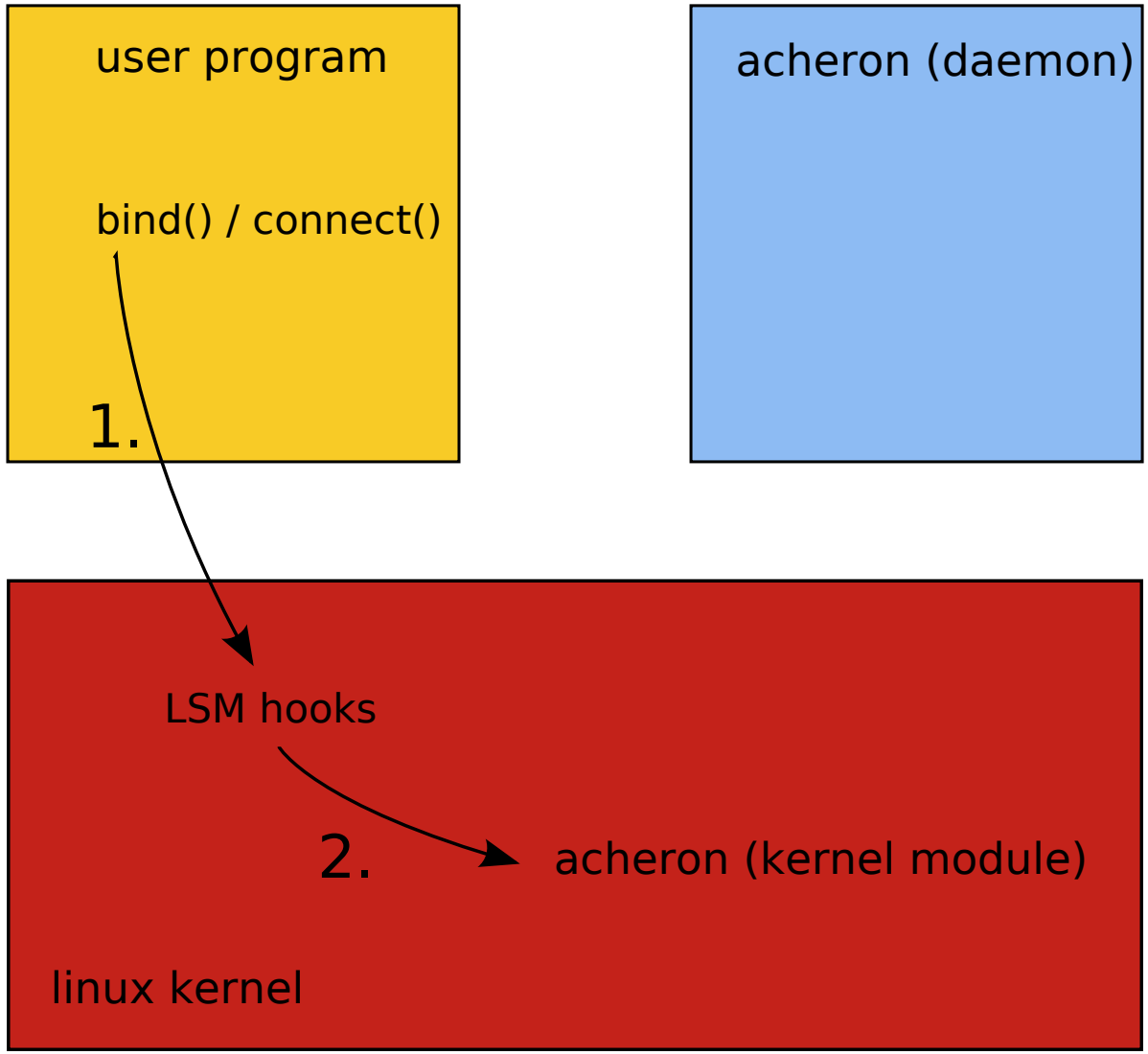
22nd of February 2008

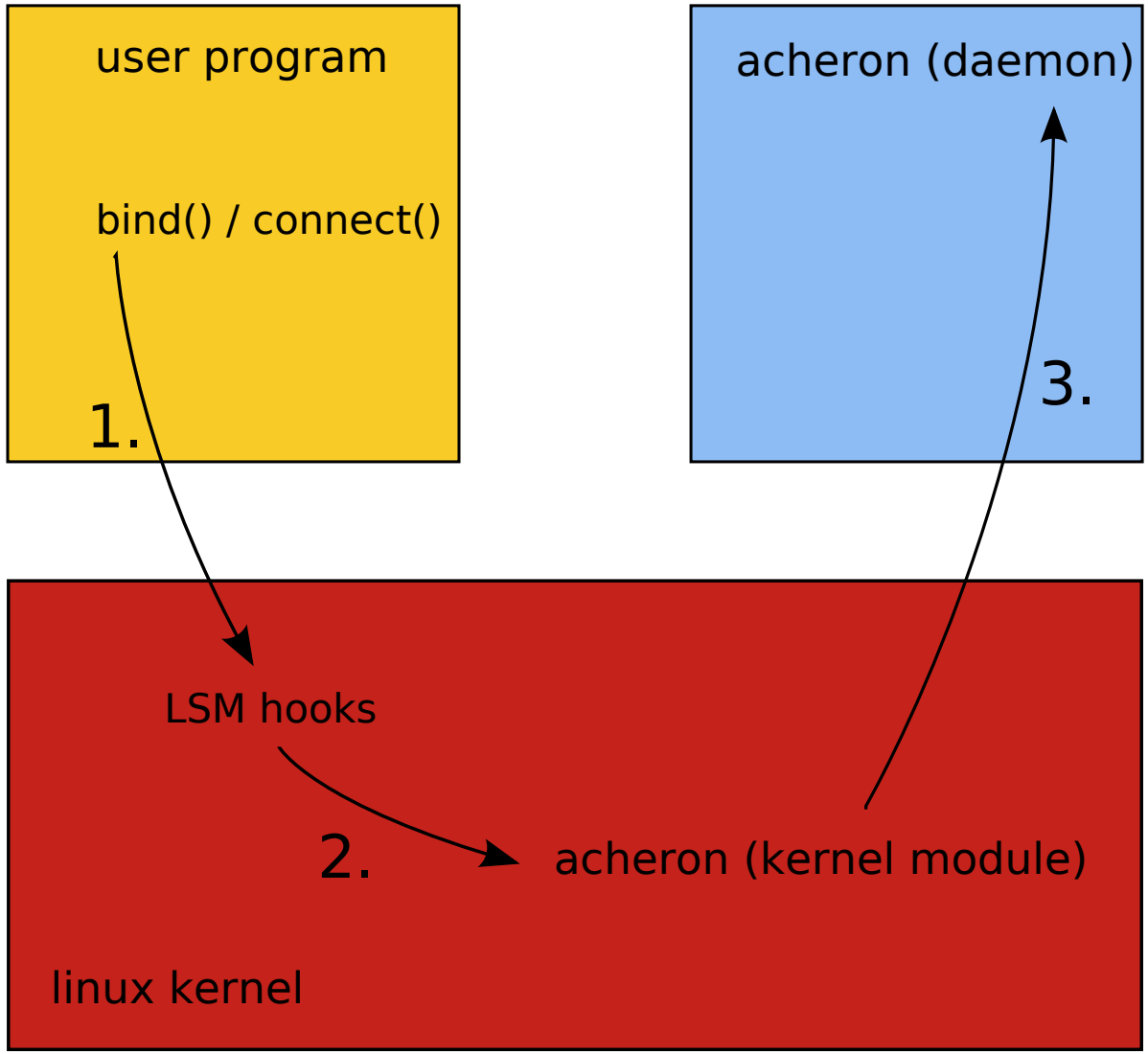
What does it do?

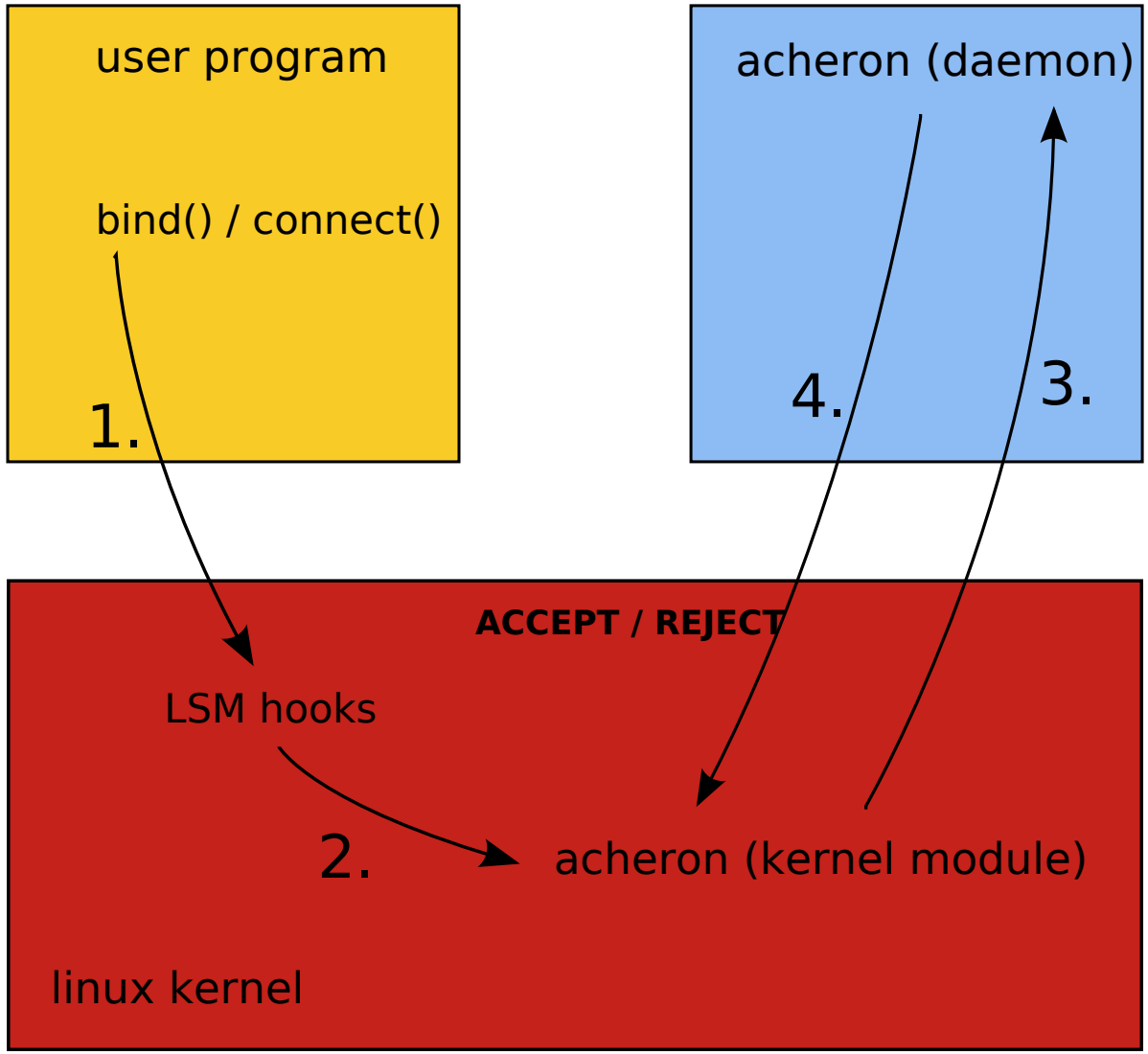
- Why do we need it? What is this all about?
- Integration into LSM (Linux Security Modules)
- Kernel Level Interaction
- User Space interaction

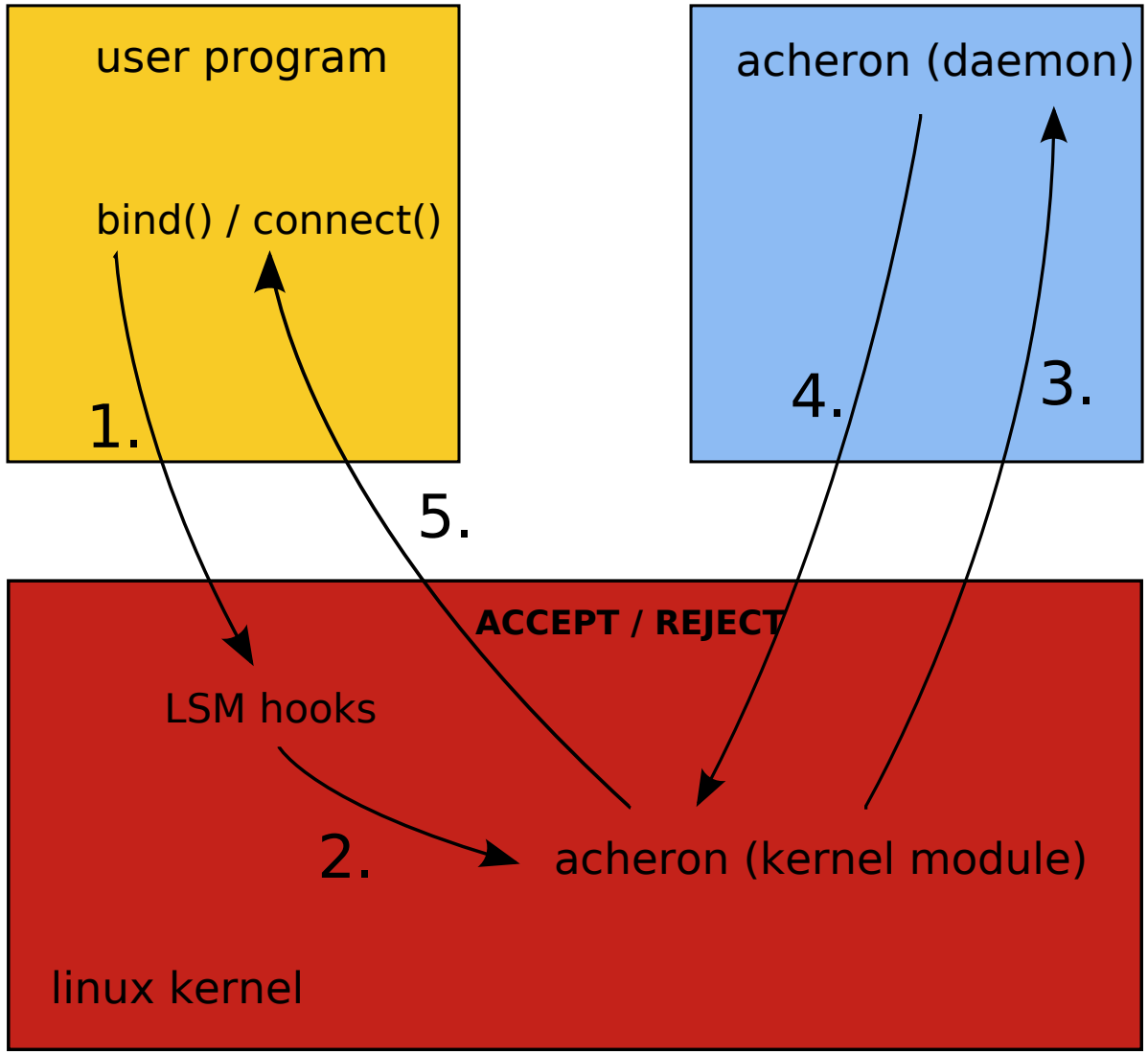

user program
bind() / connect()
acheron (daemon)
LSM hooks
acheron (kernel module)
linux kernel

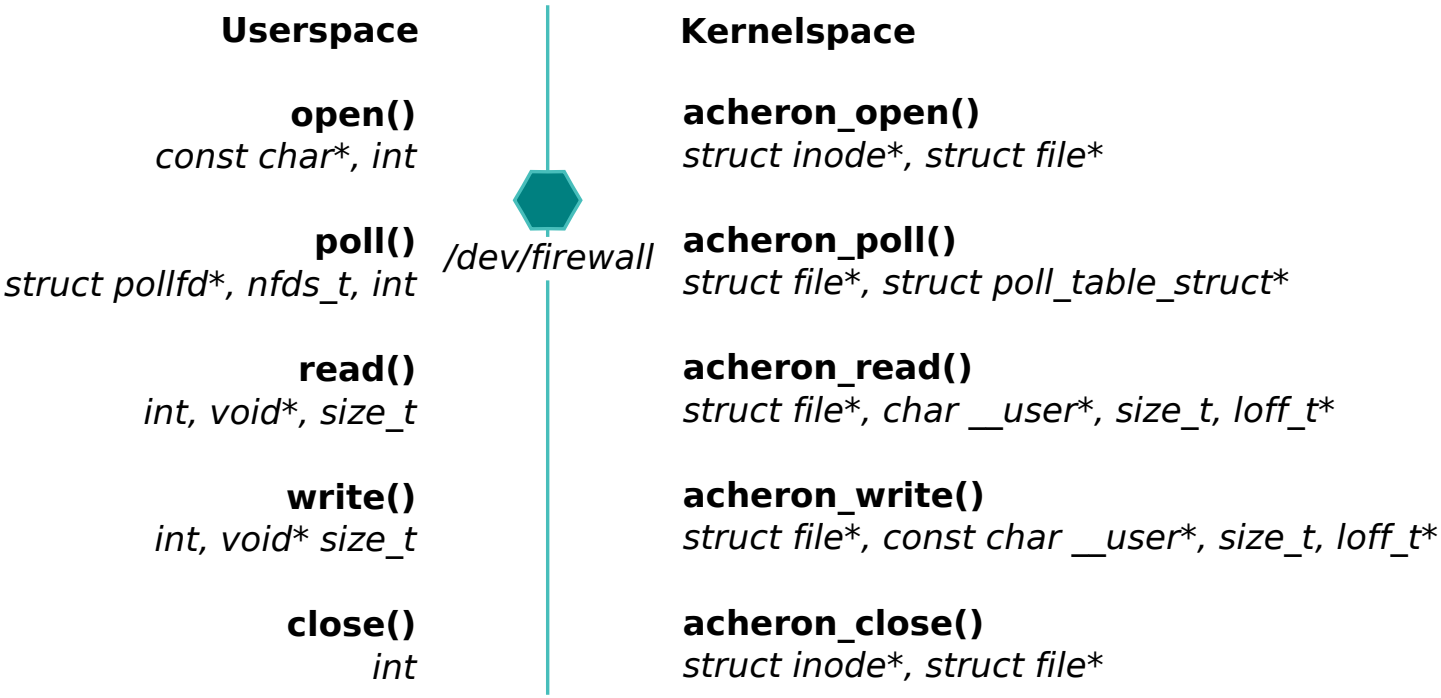

user program
bind() / connect()
1.
acheron (daemon)
LSM hooks
acheron (kernel module)
linux kernel

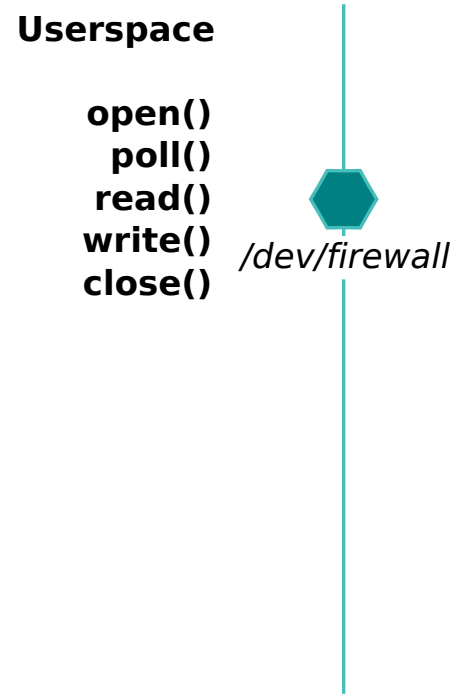

user program
bind() / connect()
1.
acheron (daemon)
LSM hooks
2.
acheron (kernel module)
linux kernel

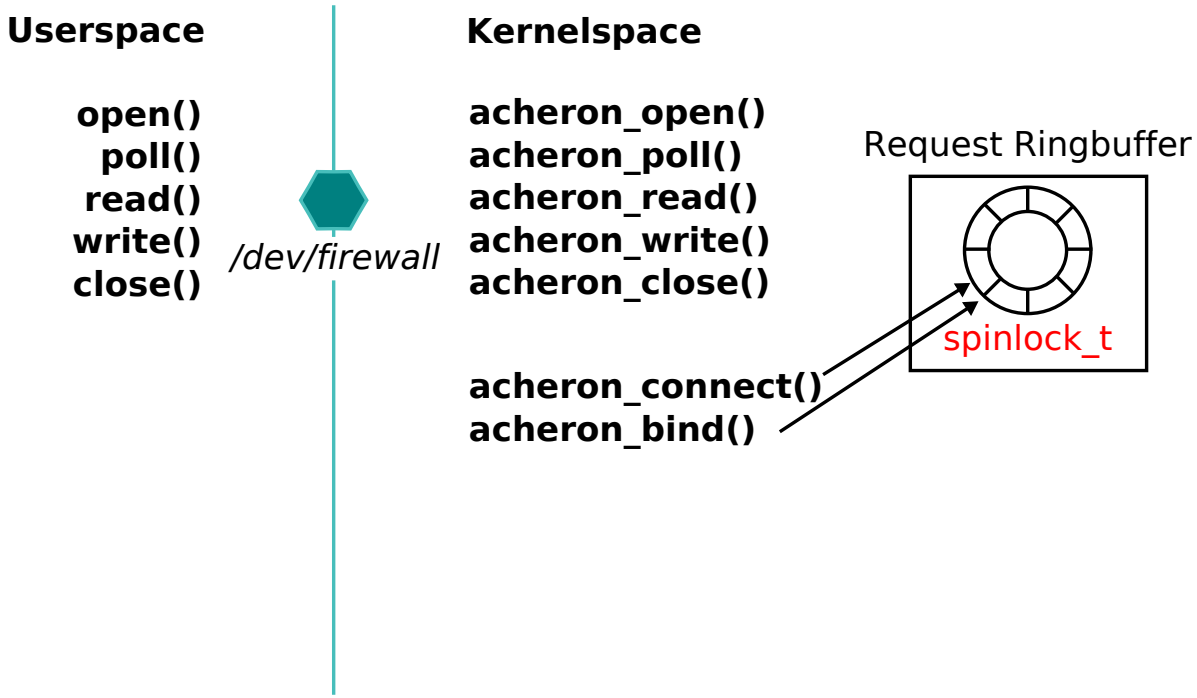

user program
bind() / connect()
1.
acheron (daemon)
3.
LSM hooks
2.
acheron (kernel module)
linux kernel

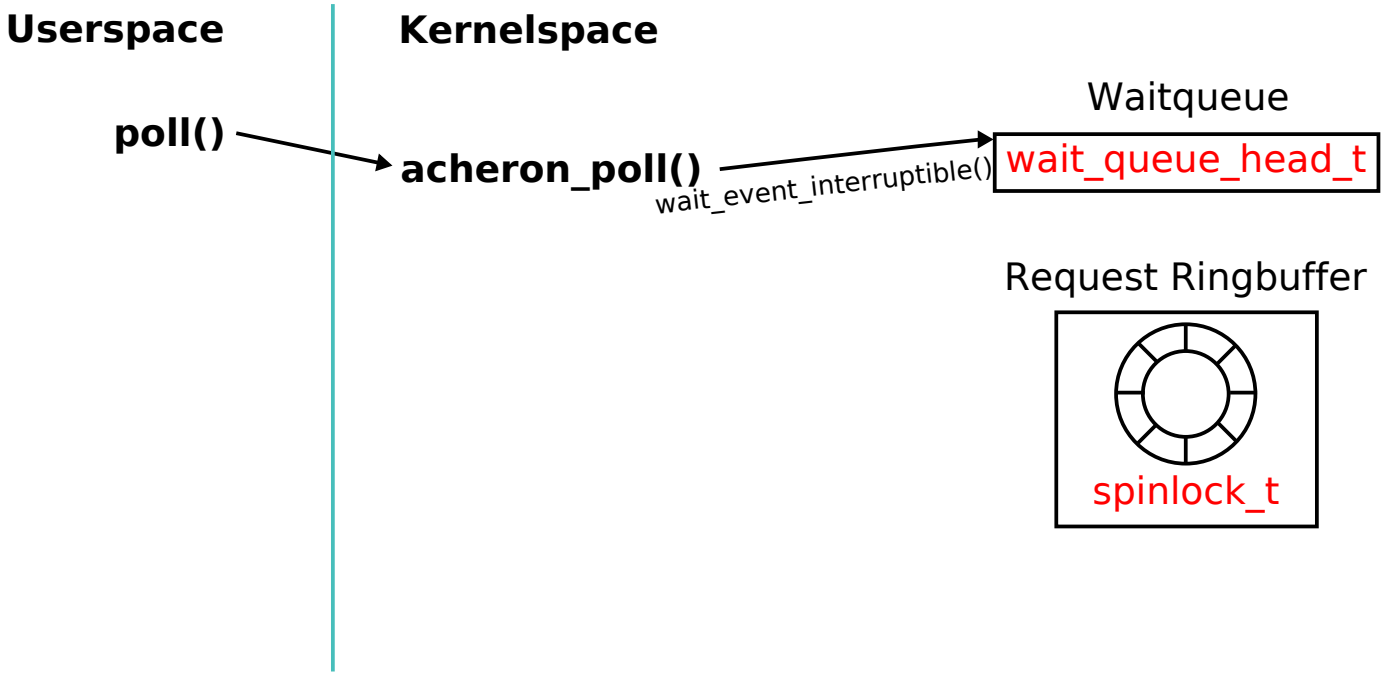

user program
bind() / connect()
1.
acheron (daemon)
4.
3.
ACCEPT / REJECT
LSM hooks
2.
acheron (kernel module)
linux kernel

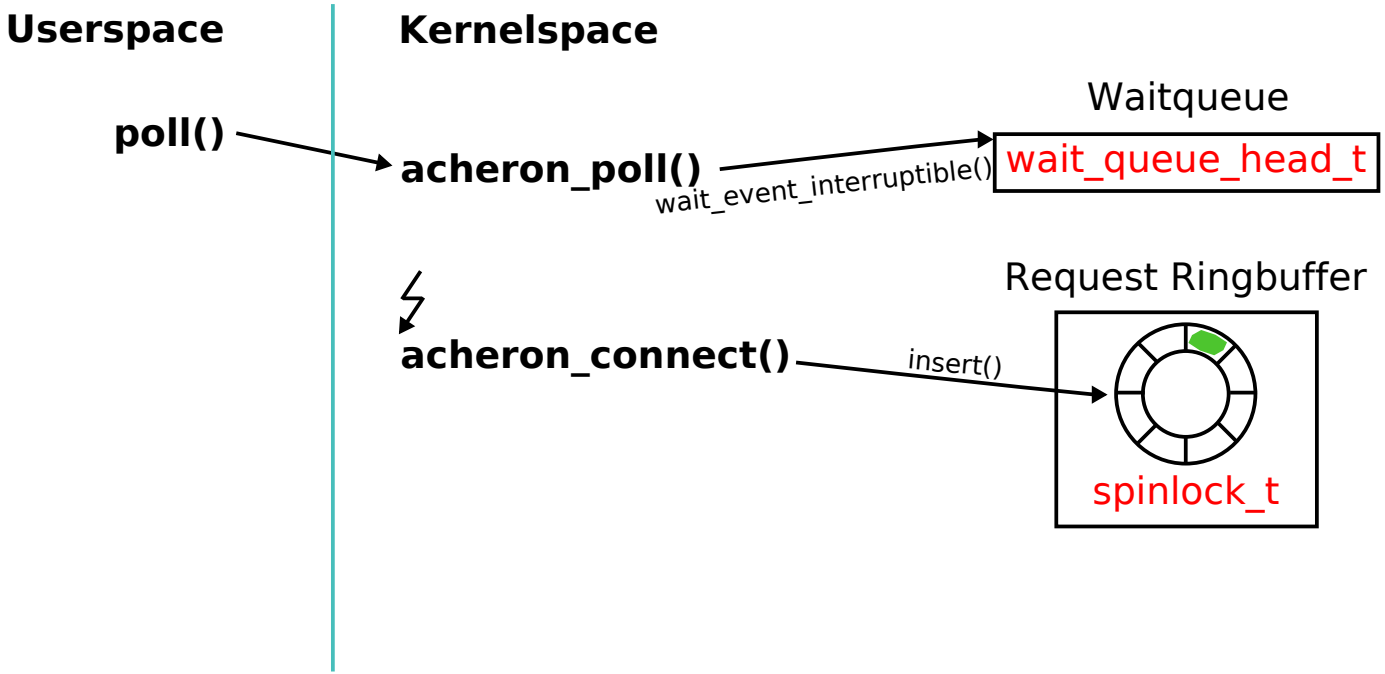

user program
bind() / connect()
1.
5.
acheron (daemon)
4.
3.
ACCEPT / REJECT
LSM hooks
2.
acheron (kernel module)
linux kernel

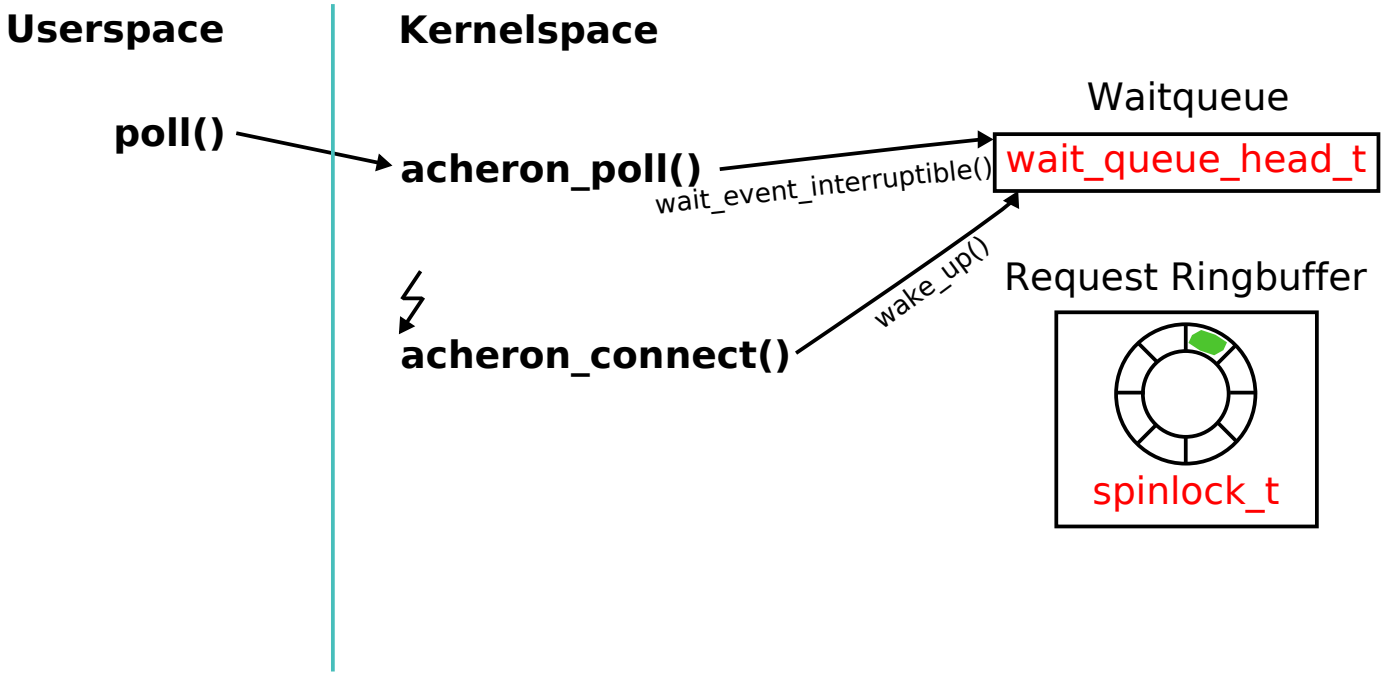

Userspace
open()
const char*, int
poll()
struct pollfd*, nfds_t, int
read()
int, void*, size_t
write()
int, void* size_t
close()
int
/dev/firewall
Kernelspace
acheron_open()
struct inode*, struct file*
acheron_poll()
struct file*, struct poll_table_struct*
acheron_read()
struct file*, char __user*, size_t, loff_t*
acheron_write()
struct file*, const char __user*, size_t, loff_t*
acheron_close()
struct inode*, struct file*

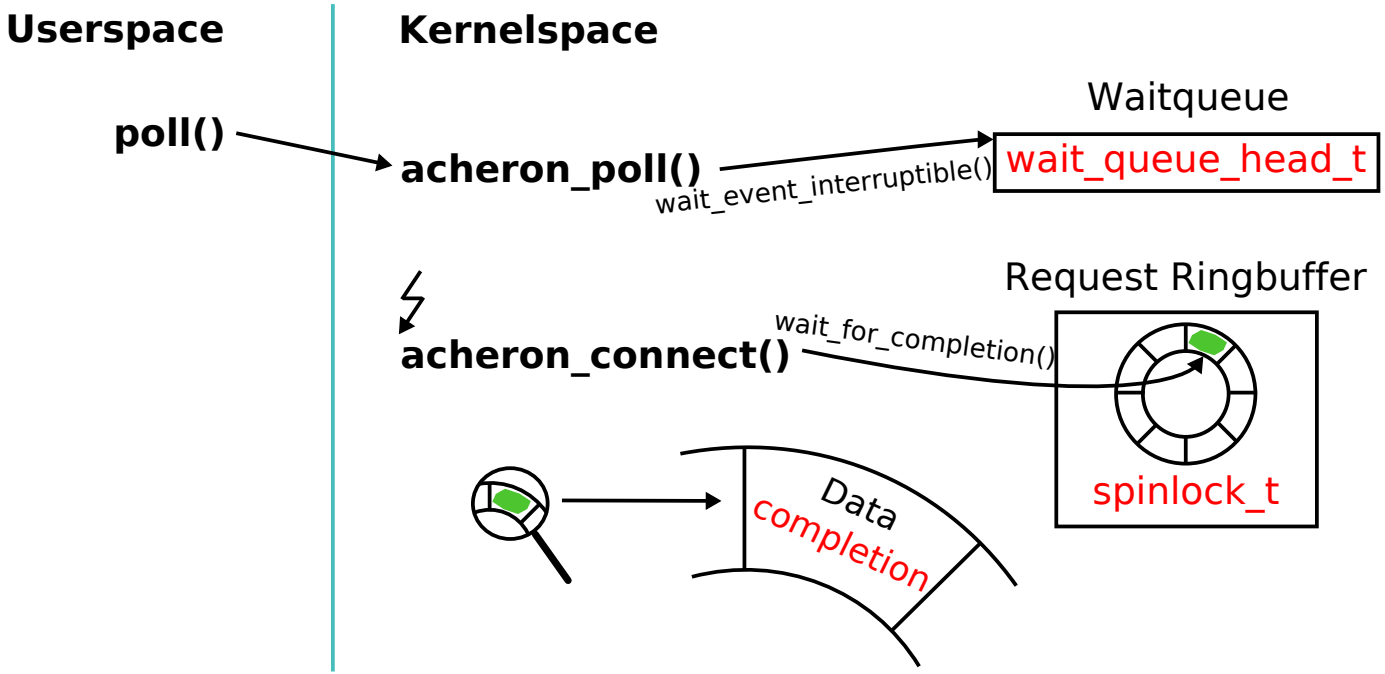

Userspace
open()
poll()
read()
write()
close()
/dev/firewall
Kernelspace
acheron_open()
acheron_poll()
acheron_read()
acheron_write()
acheron_close()
acheron_connect(struct socket*, struct sockaddr*, int)
acheron_bind(struct socket*, struct sockaddr*, int)

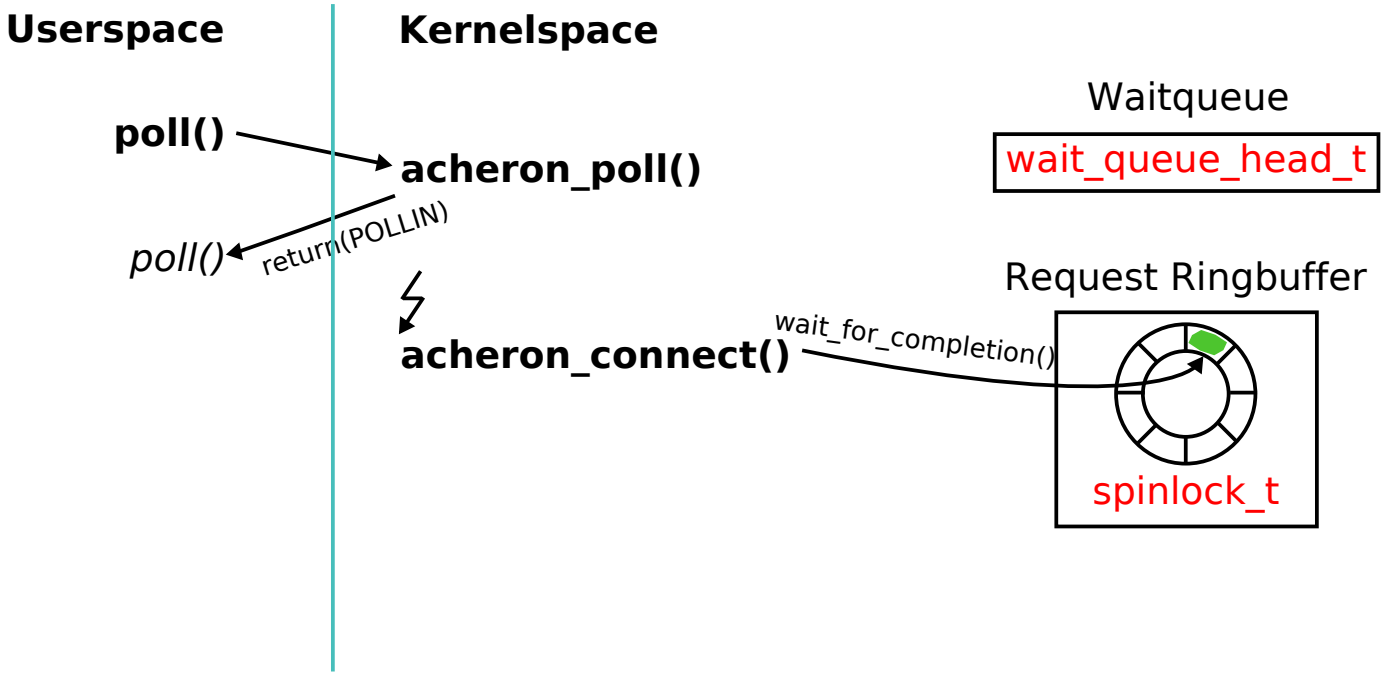

Userspace
open()
poll()
read()
write()
close()
/dev/firewall
Kernelspace
acheron_open()
acheron_poll()
acheron_read()
acheron_write()
acheron_close()
acheron_connect()
acheron_bind()
Request Ringbuffer
spinlock_t

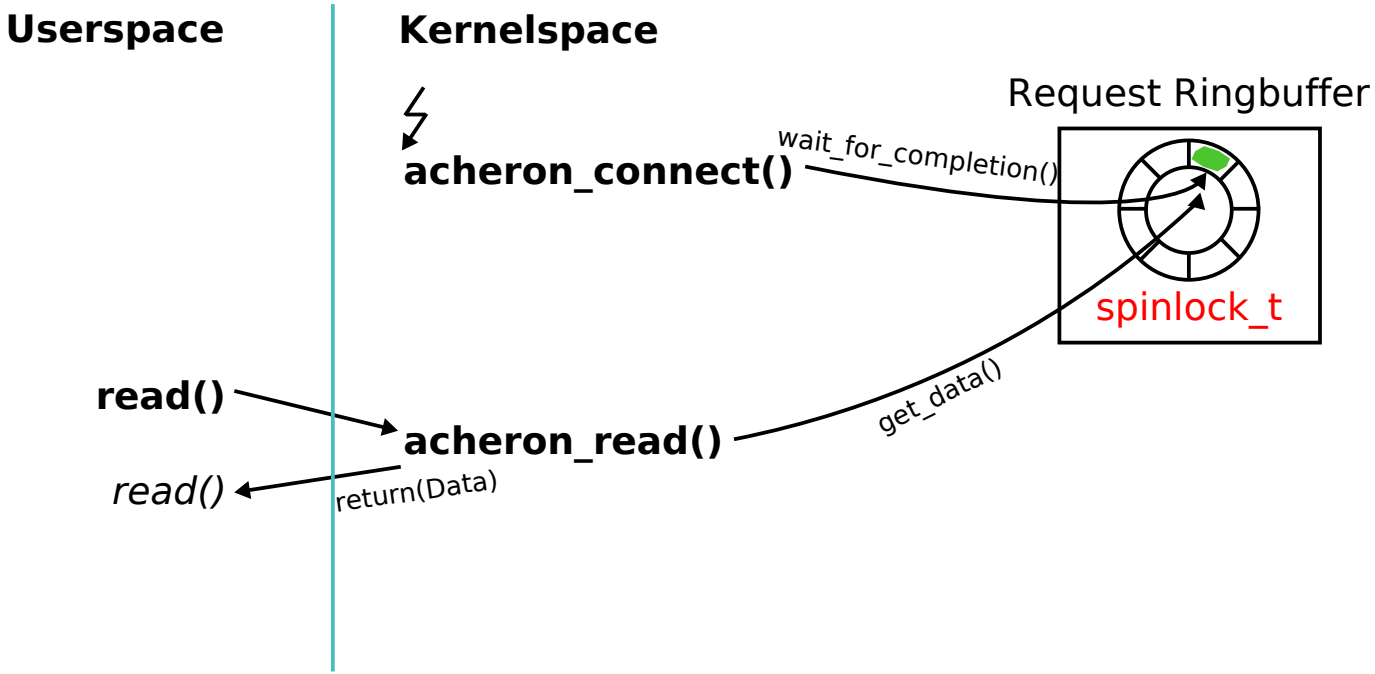

Userspace
Kernelspace
poll()
acheron_poll()
wait_event_interruptible()
Waitqueue
wait_queue_head_t
Request Ringbuffer
spinlock_t

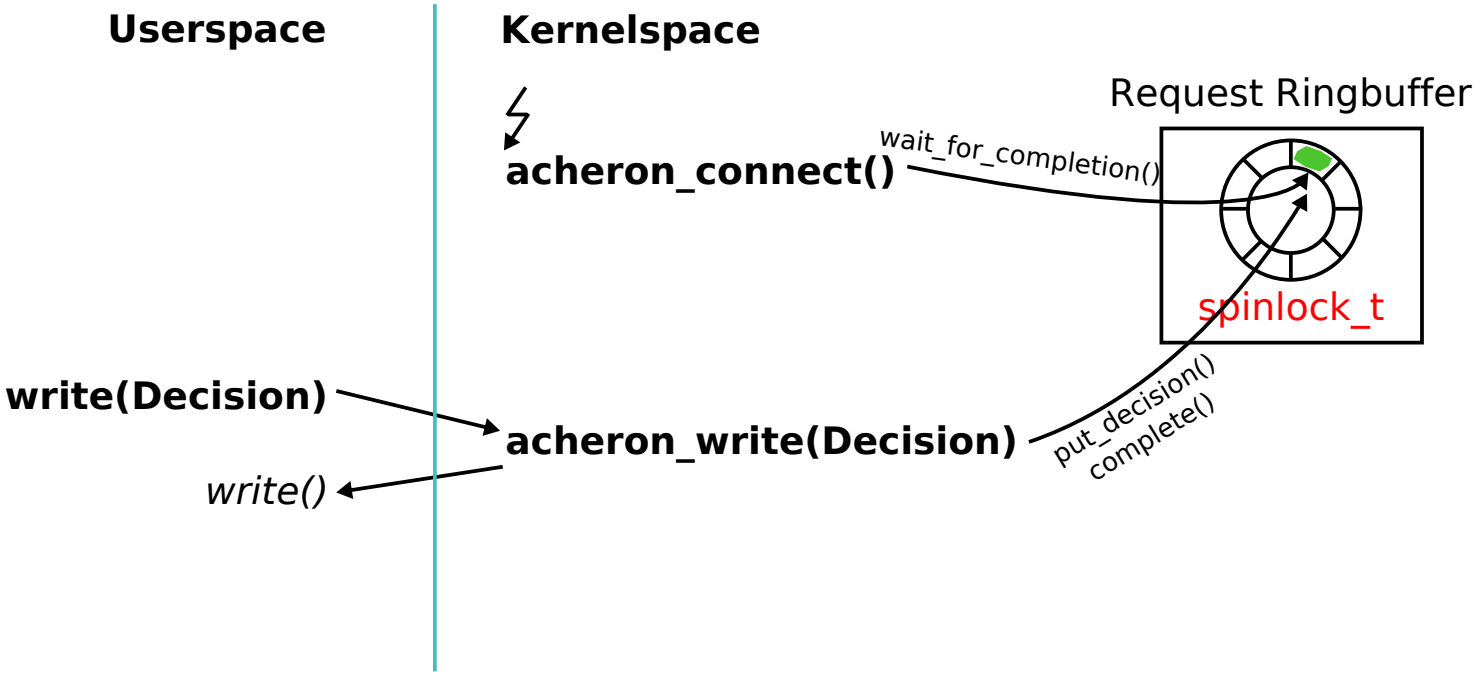

Userspace
Kernelspace
poll()
acheron_poll()
Waitqueue
wait_event_interruptible()
wait_queue_head_t
Request Ringbuffer
acheron_connect()
insert()
spinlock_t


Userspace
Kernelspace
poll()
acheron_poll()
wait_event_interruptible()
Waitqueue
wait_queue_head_t
wake_up()
Request Ringbuffer
acheron_connect()
spinlock_t


Userspace
Kernelspace
poll()
acheron_poll()
Waitqueue
wait_event_interruptible()
wait_queue_head_t
Request Ringbuffer
acheron_connect()
wait_for_completion()
spinlock_t
Data
completion

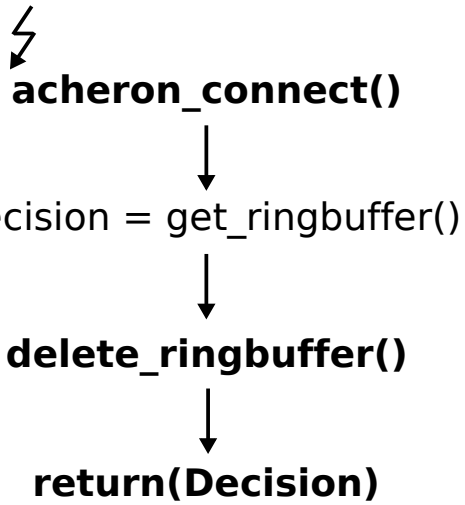

Userspace
Kernelspace
poll()
acheron_poll()
poll()
return(POLLIN)
acheron_connect()
Waitqueue
wait_queue_head_t
Request Ringbuffer
wait_for_completion()
spinlock_t


Userspace
Kernelspace
Request Ringbuffer
acheron_connect()
wait_for_completion()
spinlock_t
read()
acheron_read()
get_data()
read()
return(Data)

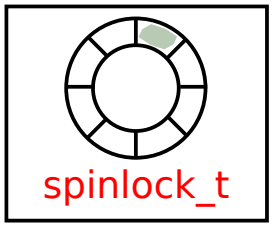

Userspace
Kernelspace
Request Ringbuffer
acheron_connect()
wait_for_completion()
spinlock_t
write(Decision)
acheron_write(Decision)
put_decision()
complete()
write()

Userspace

Kernelspace

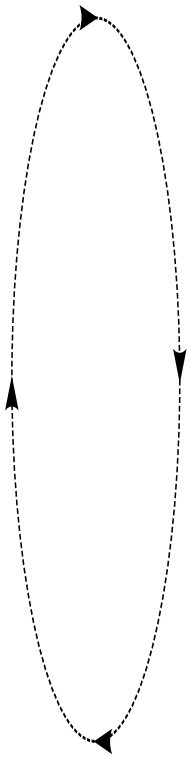


Request Ringbuffer



Device FW_Device

RuleSet Rules



poll(FW_Device)

blocked in kernel until message arrives

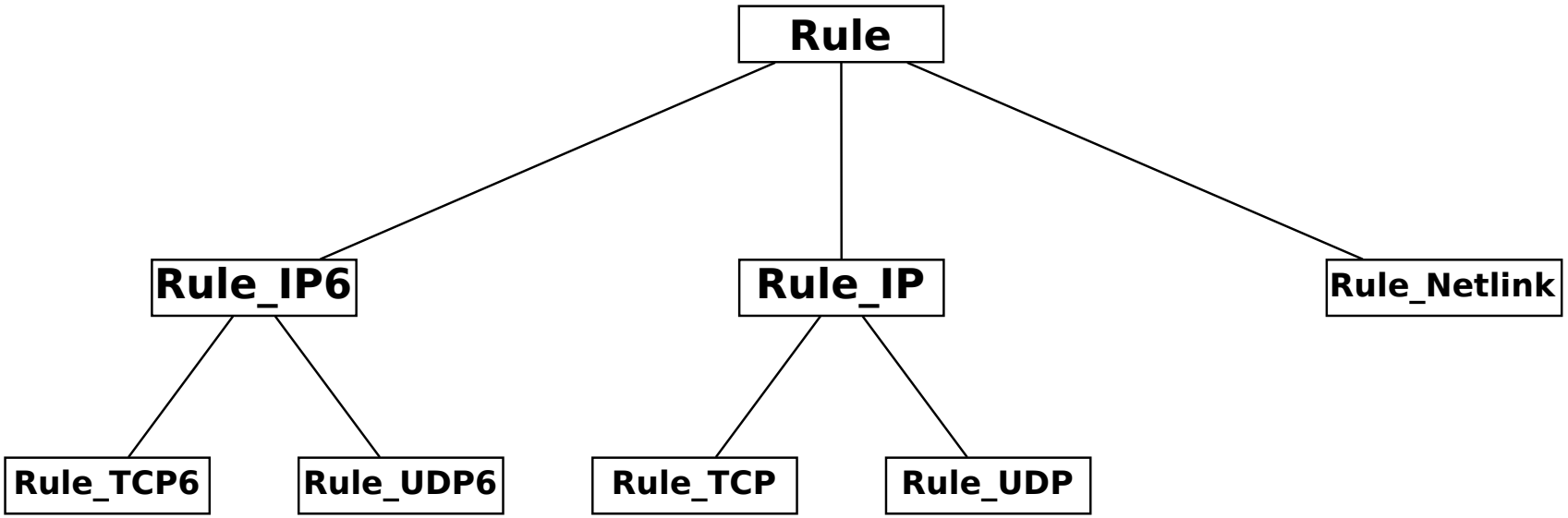
Kernel_Notification KNot

read(FW_Device, KNot)

Request Req = GenerateRequest(KNot)

Response Resp = Rules.CheckRequest(Req)

write(FW_Device, Resp)


Rule
Rule_IP6
Rule_IP
Rule_Netlink
Rule_TCP6
Rule_UDP6
Rule_TCP
Rule_UDP

Rule

- user
- group
- application
- time
- process id

policy

Rule_IP / Rule_IP6

- destination
- destination port
- protocol (implicit)

policy

Netlink

- netlink pid
- netlink group

policy

Rule_TCP(6) / Rule_UDP(6)

- protocol (implicit)

policy

Are there any more...

Questions?