Анализ производительности сетевой подсистемы микроядерного окружения Genode

Сартаков Василий, Александр Тарасиков

{sartakov,tarasikov}@ksyslabs.org

Tools & Methods of Program Analysis, 2013

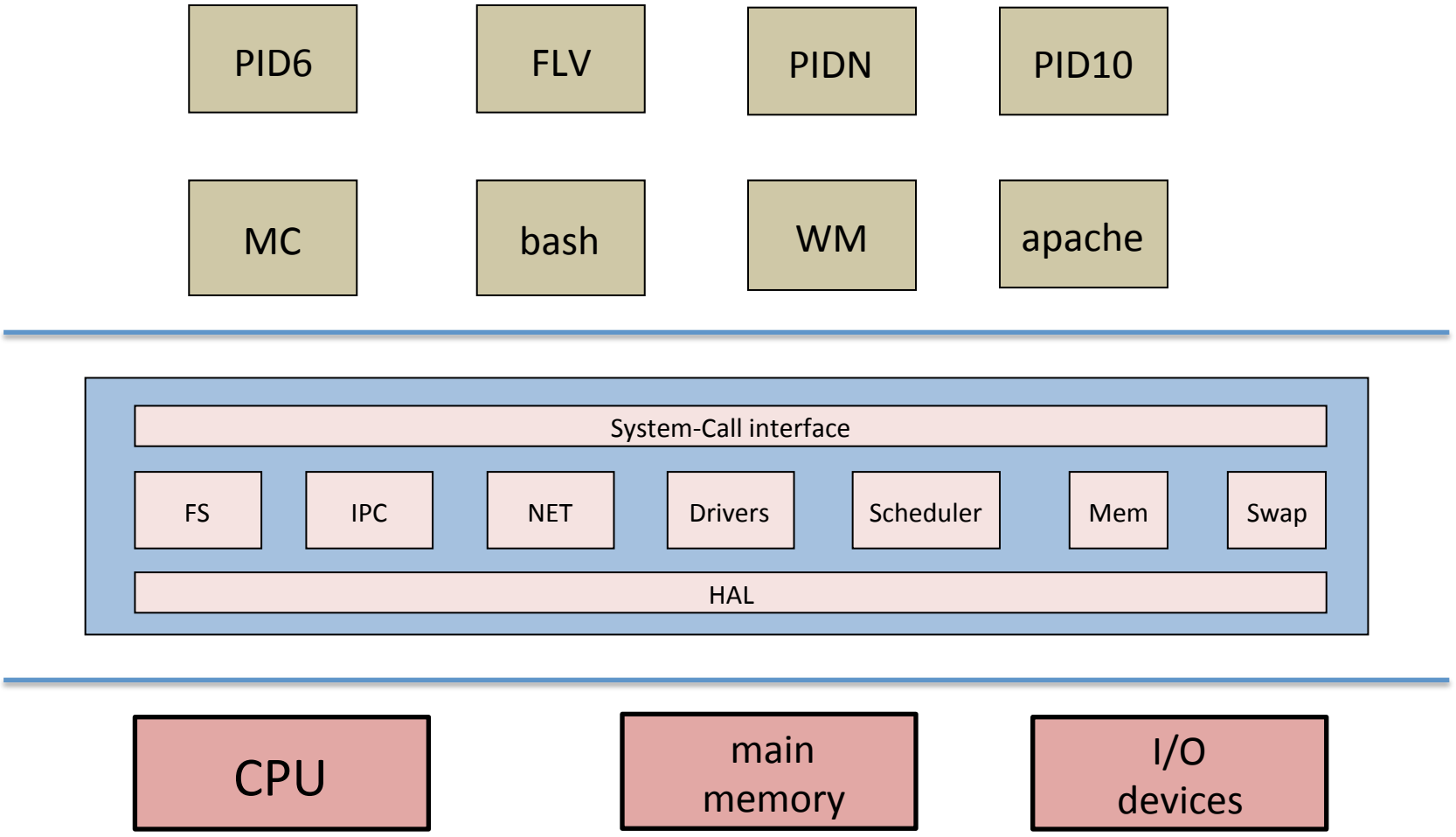
Agenda

- Intro
- Микроядра и их окружения
- Пример использования – Шлюз
- Анализ производительности
- Заключение

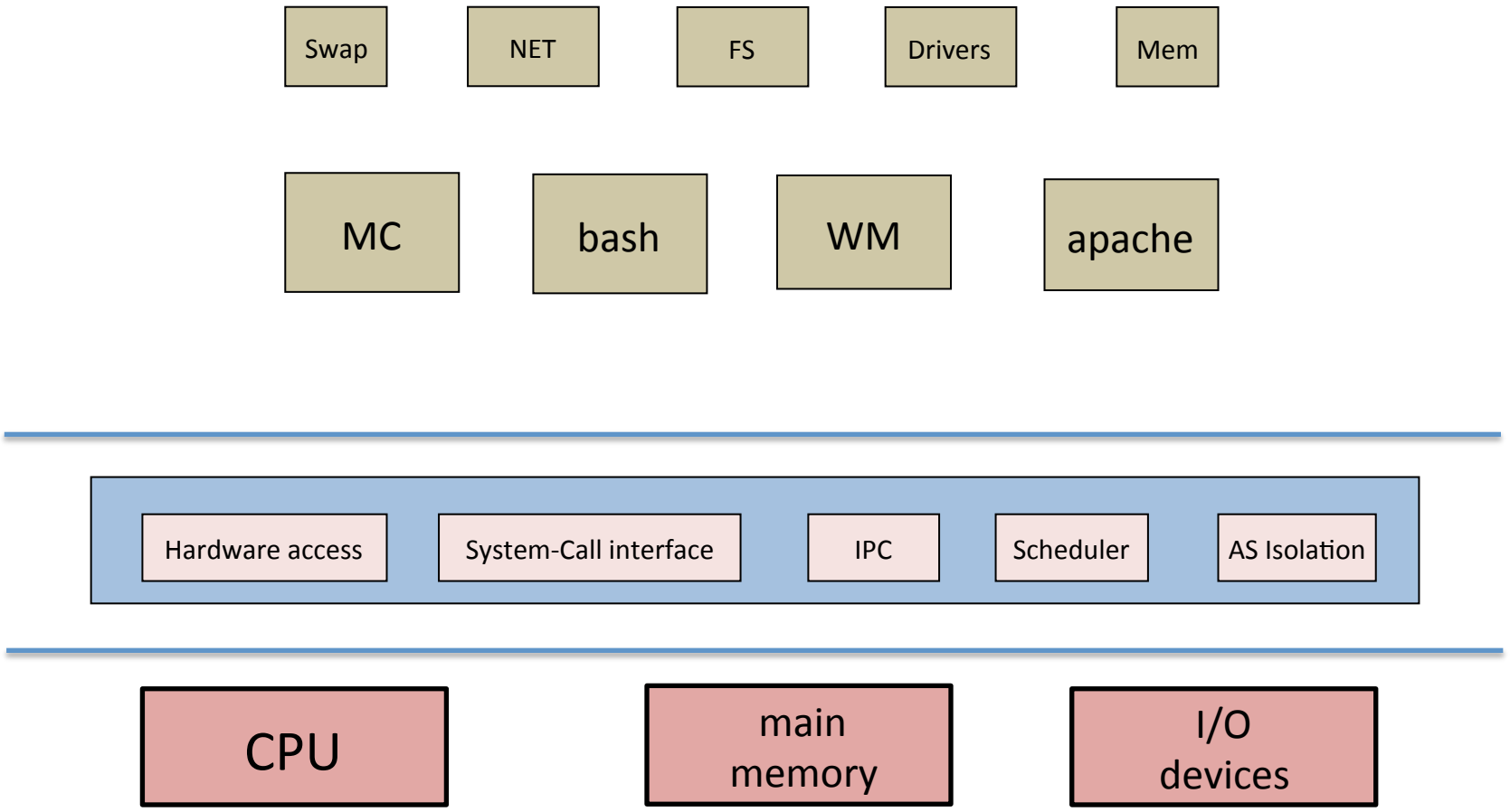
Операционные Систем

- Управляют ресурсами
  - Hardware
  - Software
- Предоставляет интерфейс к ресурсам
- Обеспечивает изоляцию и совместное использование ресурсов

Монолитно-модульный Linux



Микроядро


Swap
NET
FS
Drivers
Mem
MC
bash
WM
apache
Hardware access
System-Call interface
IPC
Scheduler
AS Isolation
CPU
main memory
I/O devices

История

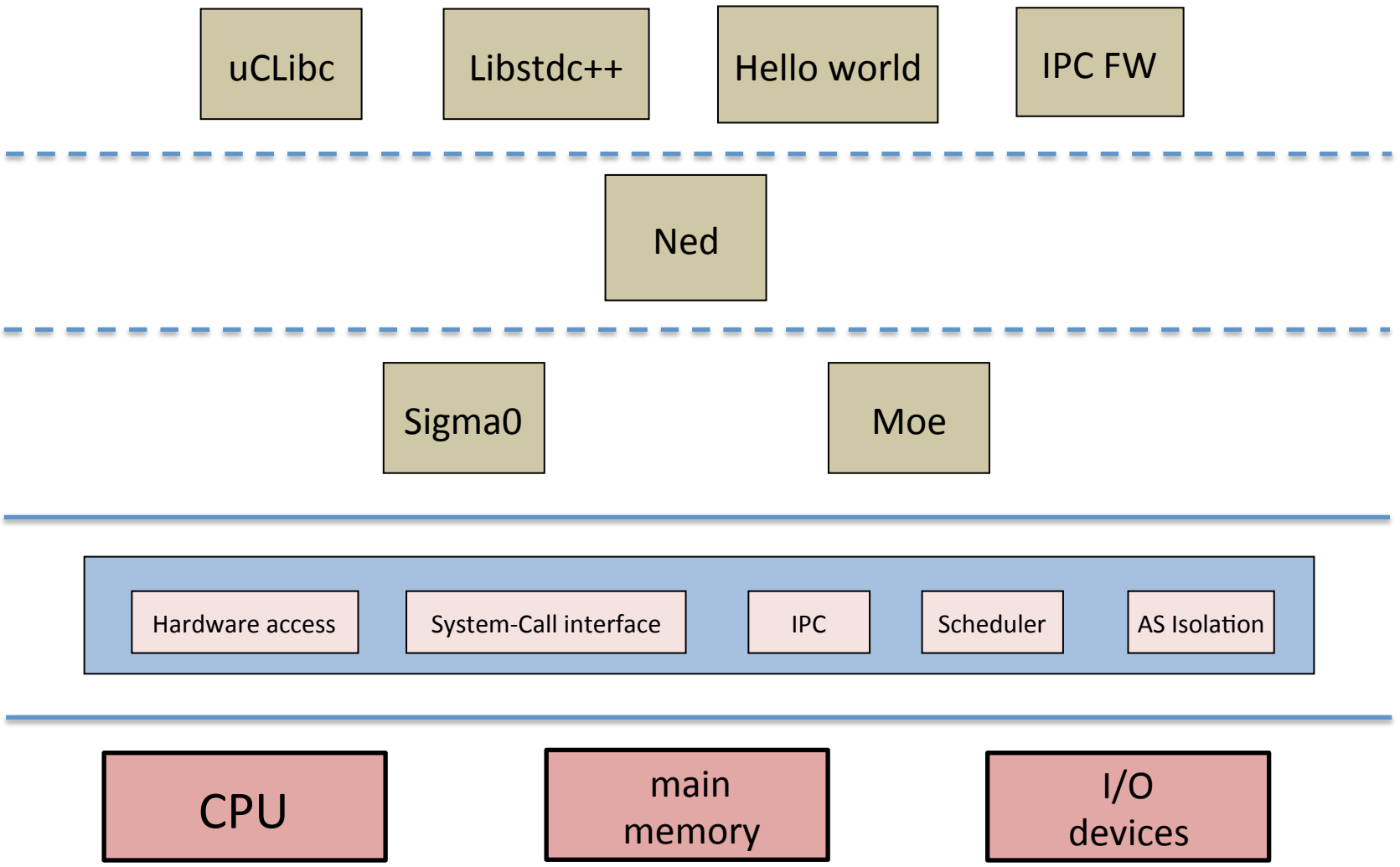
- Первое поколение: Mach (CMU 1985-1994, GNU/Mach, OS X)
- Второе поколение: Minix3 (VU Amsterdam)
- Третье поколение: Семейство L4
  - L4Ka::Pistachio
  - L4/Fiasco
  - Fiasco.OC
  - SeL4

  ….

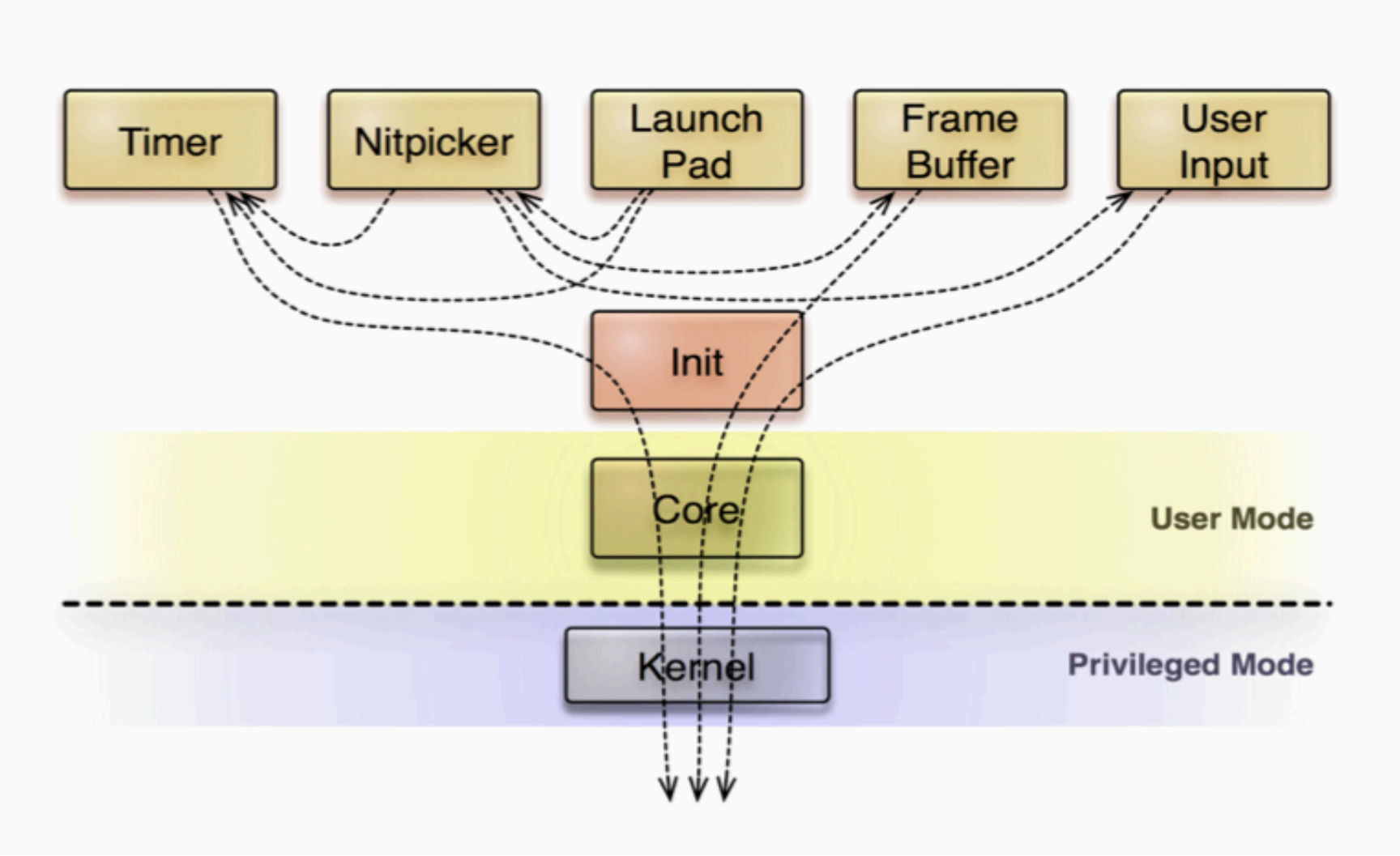
Fiasco.OC

- C++
- Object-capability uKernel
  - Everything is an object
  - Capabilities (контролируемые ссылки на объекты)
- Fiasco.OC это не операционная система, она требует окружения:
  - L4Re
  - Genode

L4Re


uCLibc
Libstdc++
Hello world
IPC FW
Ned
Sigma0
Moe
Hardware access
System-Call interface
IPC
Scheduler
AS Isolation
CPU
main memory
I/O devices

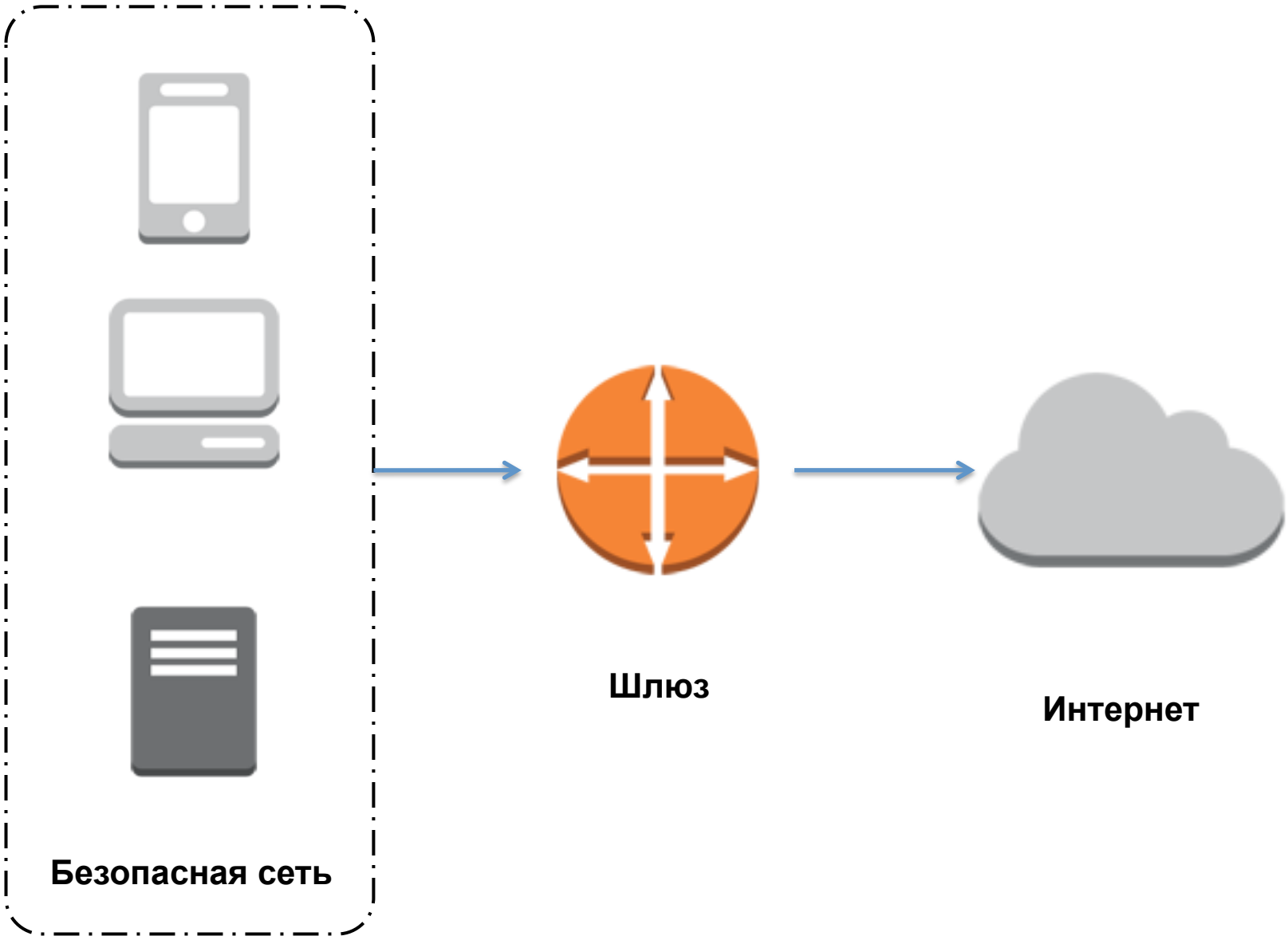
Genode


Timer
Nitpicker
Launch Pad
Frame Buffer
User Input
Init
Core
User Mode
Kernel
Privileged Mode

Компоненты Окружений

- «Стандартные» библиотеки (uCLibc, stdc++)
- Драйвера: iPXE_kit, dde_kit
- Паравиртуализированный L4Linux
- Компоненты системы (Ned, IO, Moe, Sigma, init)
- Портированные приложения – Qt, LwIP

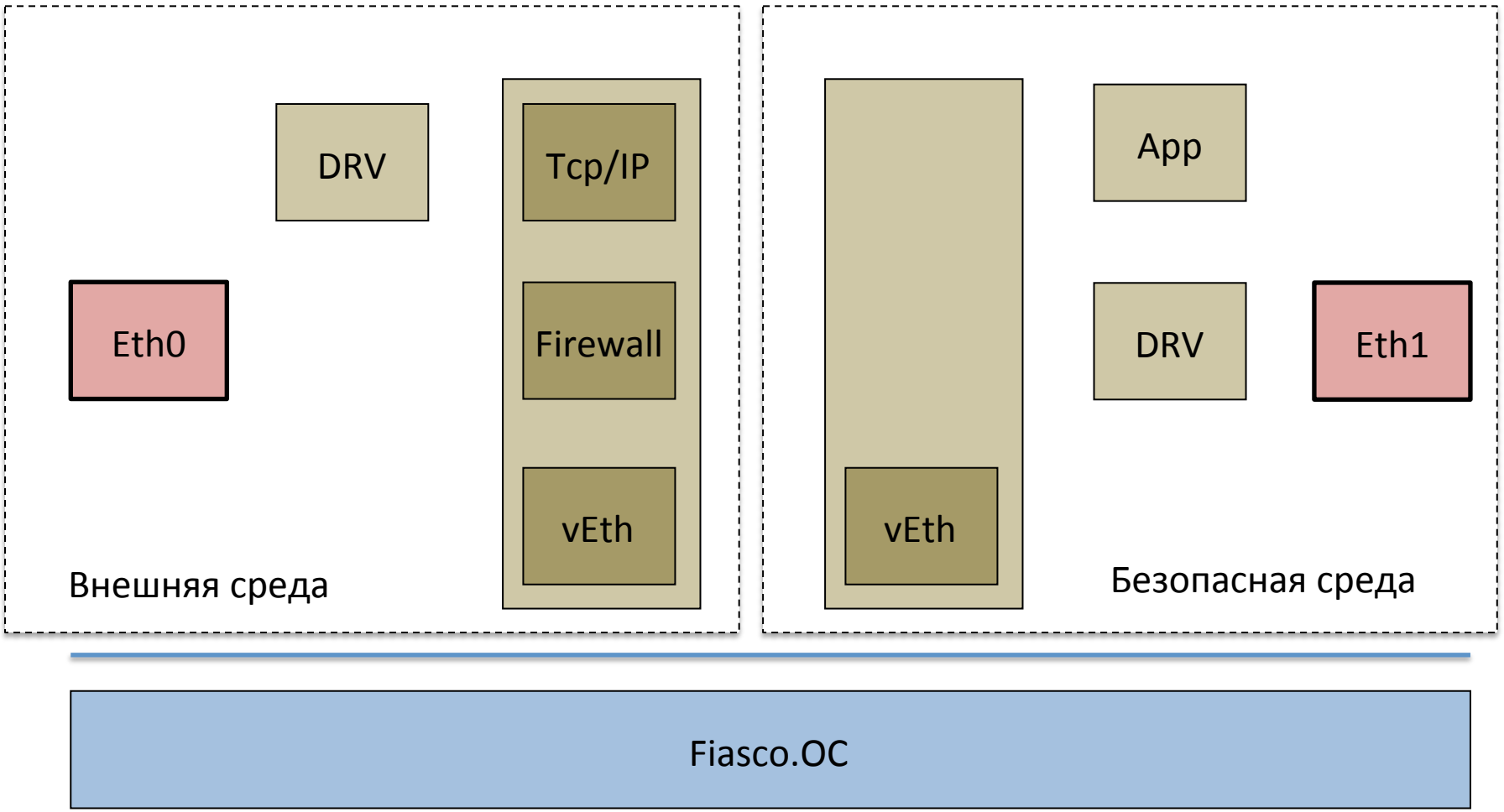
Gateway



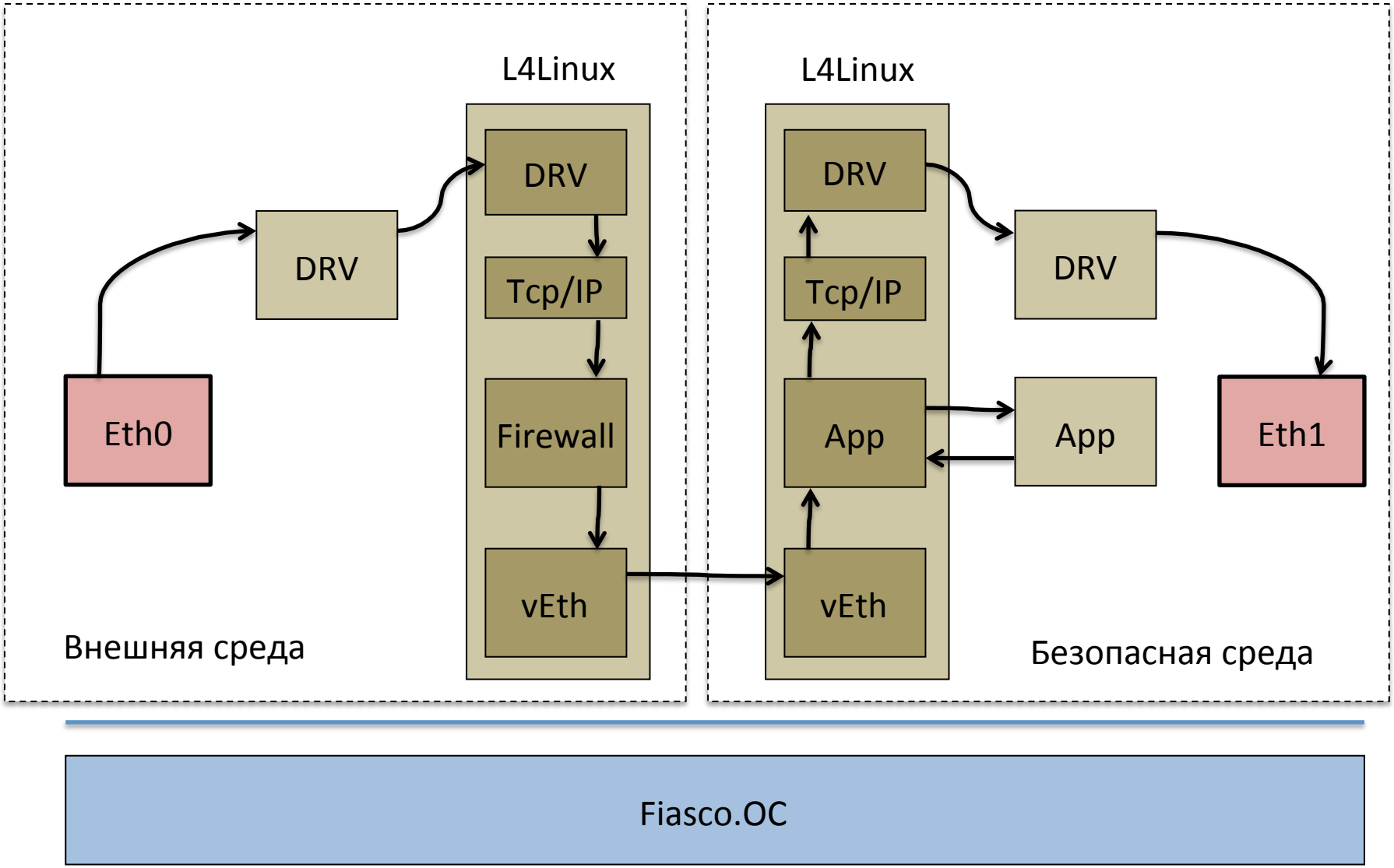
Gateway::Монолит

- Уязвимость в драйвере: Специально сформированный пакет -> срыв -> доступ к памяти ядра
- «Умное» устройство: специально сформированный пакет -> активация закладки в сетевом устройств -> кража данных из памяти
- И это все не смешно

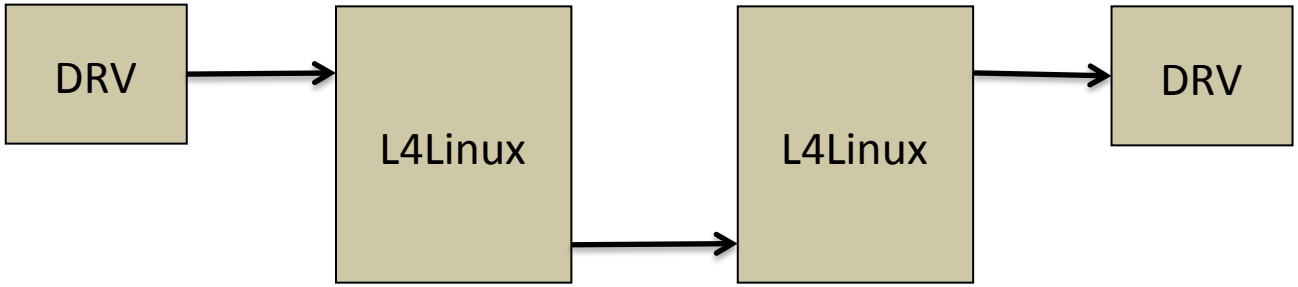
Gateway


DRV
Tcp/IP
Eth0
Firewall
vEth
Внешняя среда
App
DRV
Eth1
vEth
Безопасная среда
Fiasco.OC

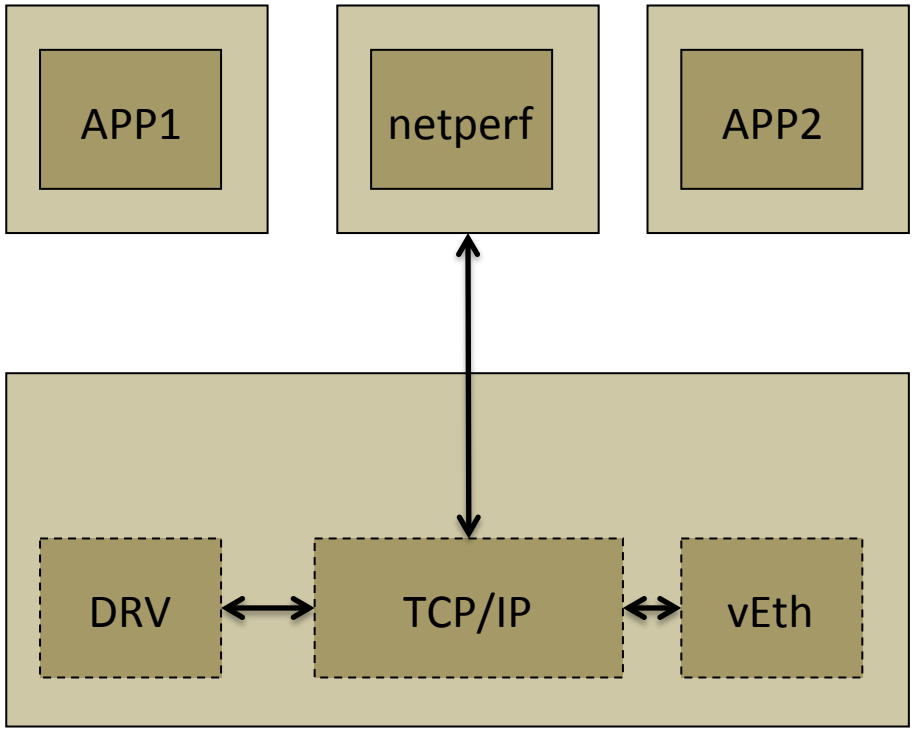
Gateway


L4Linux
L4Linux
DRV
DRV
DRV
DRV
Tcp/IP
Tcp/IP
Eth0
Firewall
App
App
Eth1
vEth
vEth
Внешняя среда
Безопасная среда
Fiasco.OC

Упрощенная схема

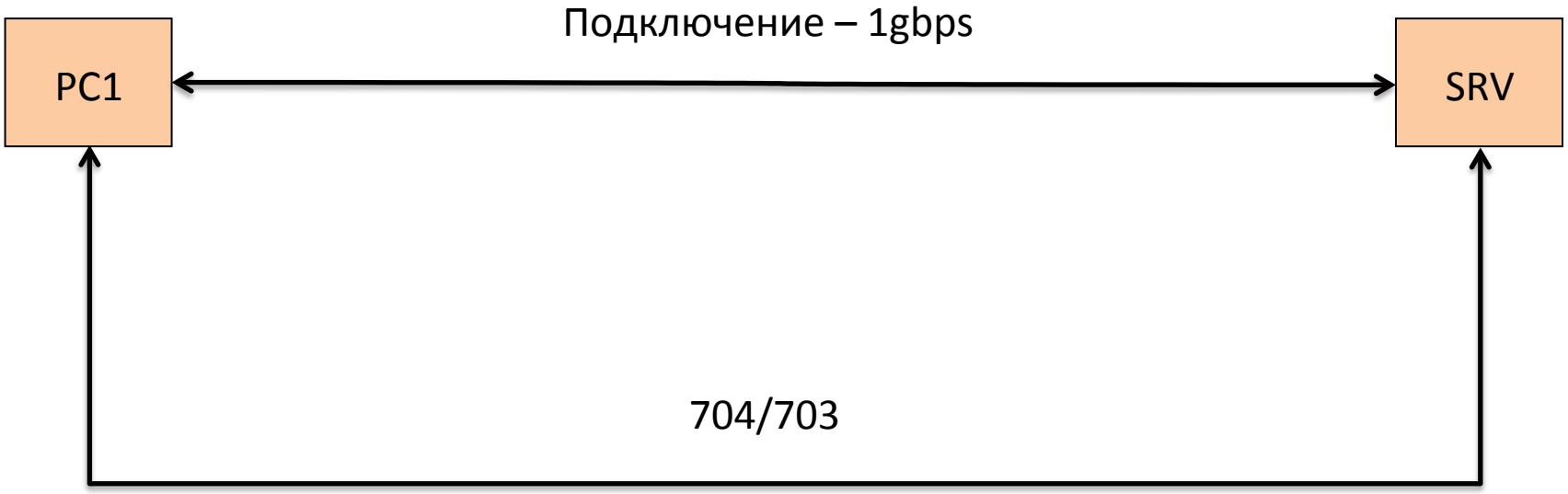


L4Linux

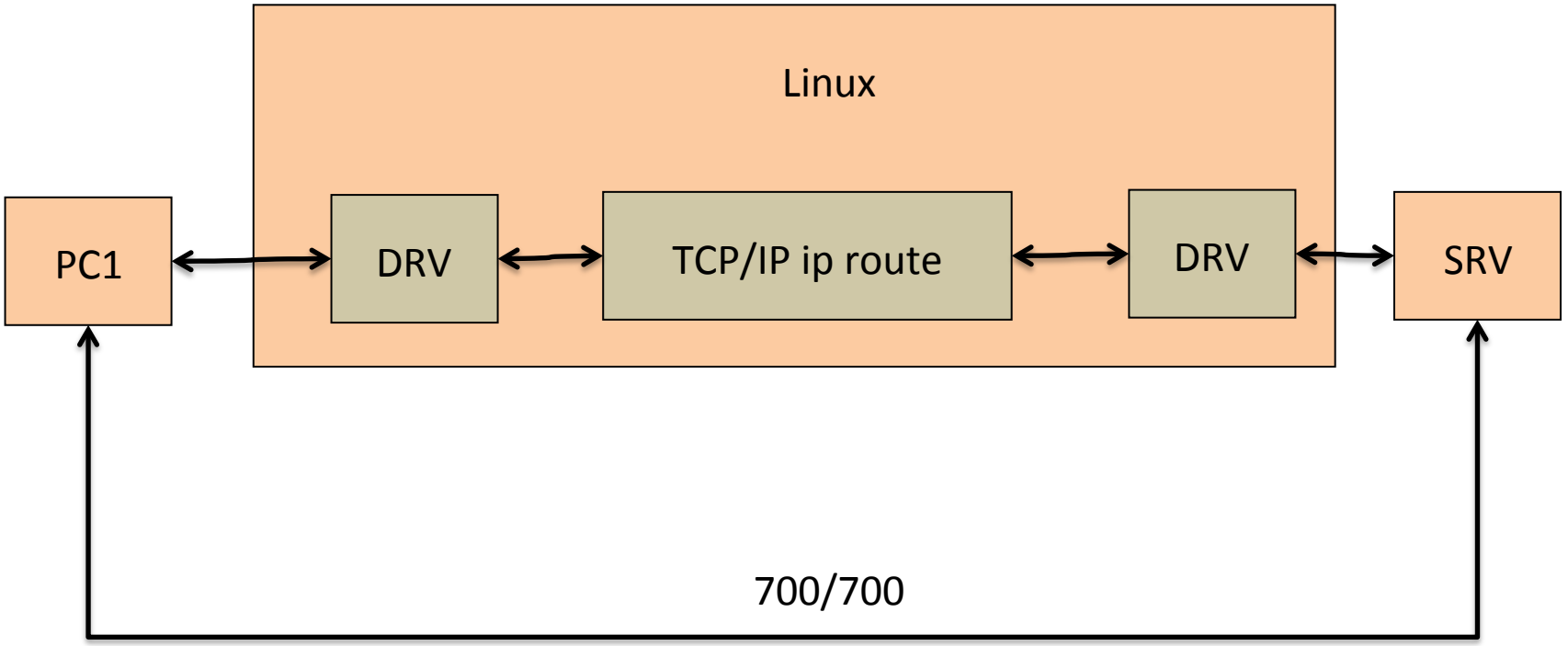


L4Linux kernel

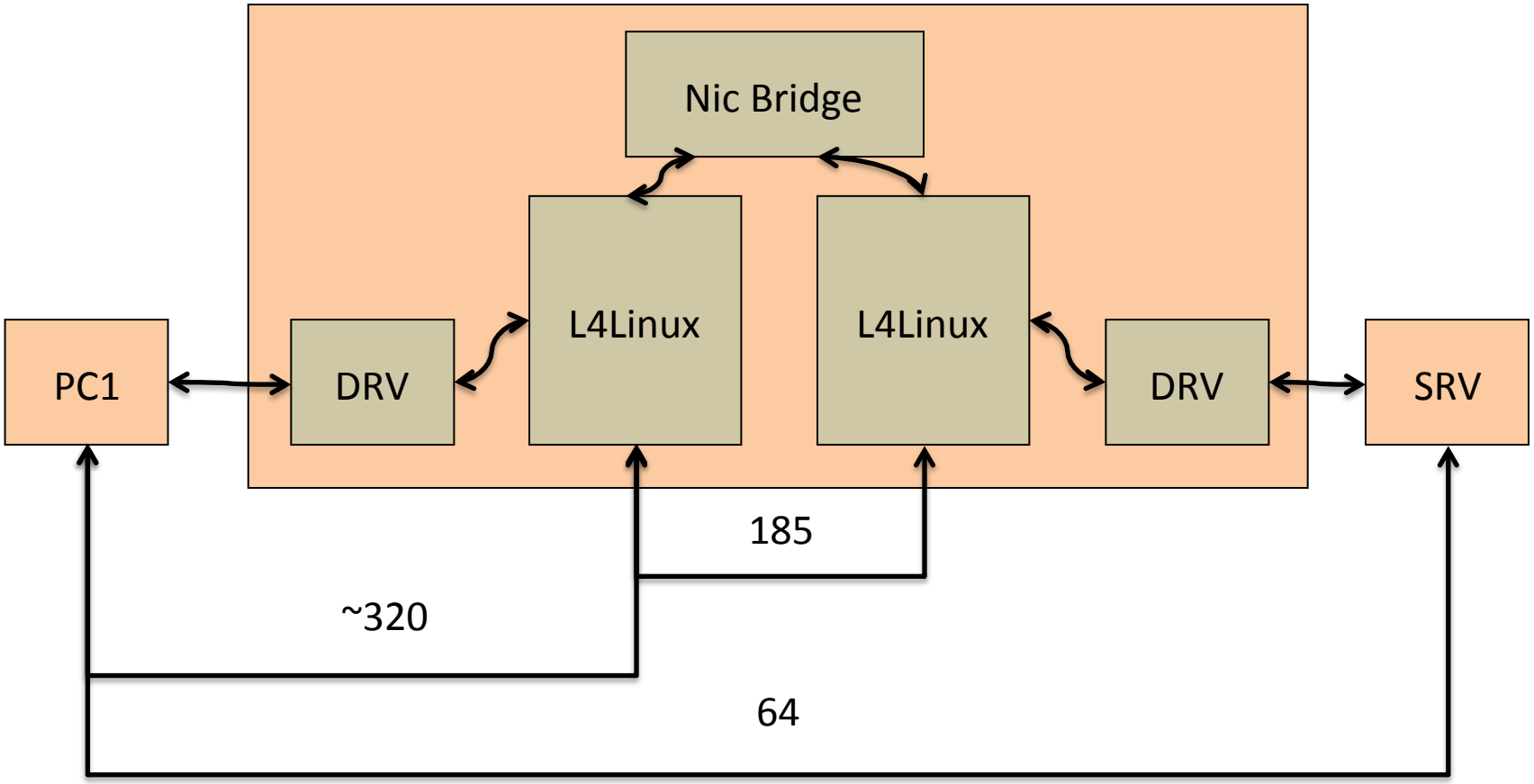
Netperf: PC <-> SRV



Netperf: PC <-> Linux PC <->SRV



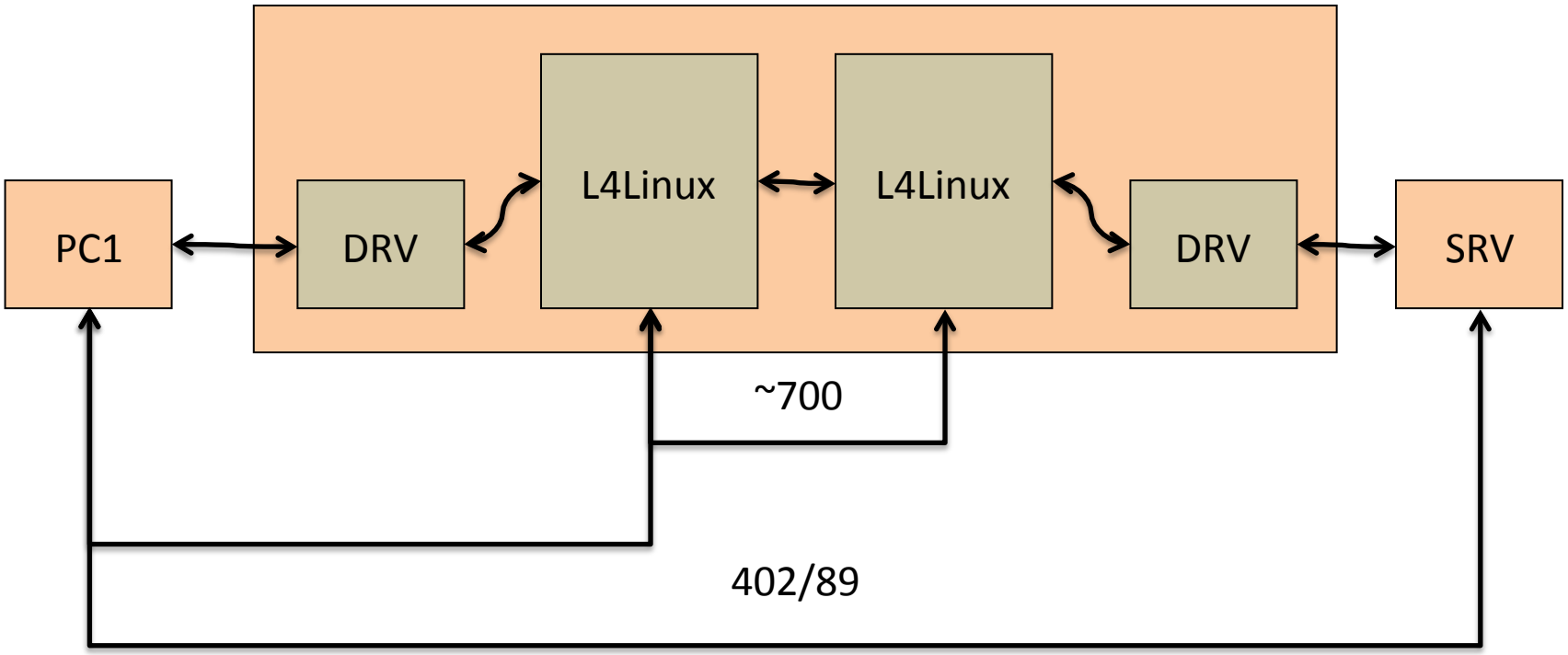
PC <-> DRV<-> L4LInux <-> L4Linux <-> DRV <-> SRV



Первый Анализ

- Отказ от Nic Bridge. Вместо него кольцевой буффер с передачей сигналов напрямую между L4Linux.
- Одна и та же память используется в DDE_iPXE для приема и отправки данных.

PC <-> DRV<-> L4LInux <-> L4Linux <-> DRV <-> SRV



Профилирование

- Два подхода к профилированию:
  - Oprofile, Профилирование на уровне ядра. (kernel + userspace)
  - Gprof, профилирование программ (gcc + lib)

- Первый затратный по реализации, бесполезен (как оказалось в будущем)
- Требует POSIX совместимости

Профилирование

- Профилировщик как внутрисистемный (внутриядерный) отладчик – измерение частоты (количества) и продолжительности
- Констатировал частое нахождения в функциях связанных с системными вызовами
- Использование памяти ядра, поскольку «микроядерный» дизайн профилировщика негативно сказывался на производительности системы – вносил большую погрешность
- Разработали профилировщик для окружения:
  - Линкуется на старте
  - Сохраняет данные в памяти, выгружает по http, поскольку в системе отсутствует носитель

Результаты (в тиках процессора)

Процедура	Тики процессора	модуль
dde_kit_sem_up	13	lib/nic.c [dde_ipxe]
dde_kit_sem_down	59	lib/nic.c [dde_ipxe]
memcpy	23	lib/nic.c [dde_ipxe]
get_acked	46	nic/component.h [Nic]
submit_packet	190-160k	nic/component.h [Nic]
packed_descriptor	10	nic/component.h [Nic]
get_packet	8	lib/l4lx/genode_net.cc [L4Linux]
acknowledge_packet	254	lib/l4lx/genode_net.cc [L4Linux]
submit_packet	65	lib/l4lx/genode_net.cc [L4Linux]
memcpy	25	lib/l4lx/genode_net.cc [L4Linux]

Анализ

- submit_packet - помещает пакет в циклический буфер и делает IPC запрос к другому процессу.
- Спародически может увеличить время выполнения до 190K
- Возможные причины – реализация драйвера, управления памятью, управление процессами.

Причины::Драйвера

- Драйвера:
  - Не поддерживает MSI-X, как следствие приходится использовать прерывание PCIe, возникает конкуренция с другими устройствами
  - Не объясняет задержки в выполнении

Причины::Память

- Netperf последовательно увеличивает размер пакетов. Чем дольше работает тест, тем больший разброс значений в submit_packet.
- Гипотеза:
  - Медленная аллокация памяти, зависящая от размеров региона.

Genode::allocator

- SLAB Allocator.
  - Список регионов по 1KB, 2KB,16KB
- Кольцевой буфер между сервисами:
  - При освобождении памяти одним регионом происходит добавление его в список свободных регионов, в частности доступный второму сервису.
- При передаче данных от одного сервиса другому происходит unmap
- И в этот момент показалось что проблема найдена

Но нет

- Причиной всему оказались блокировки.

Genode::mutex

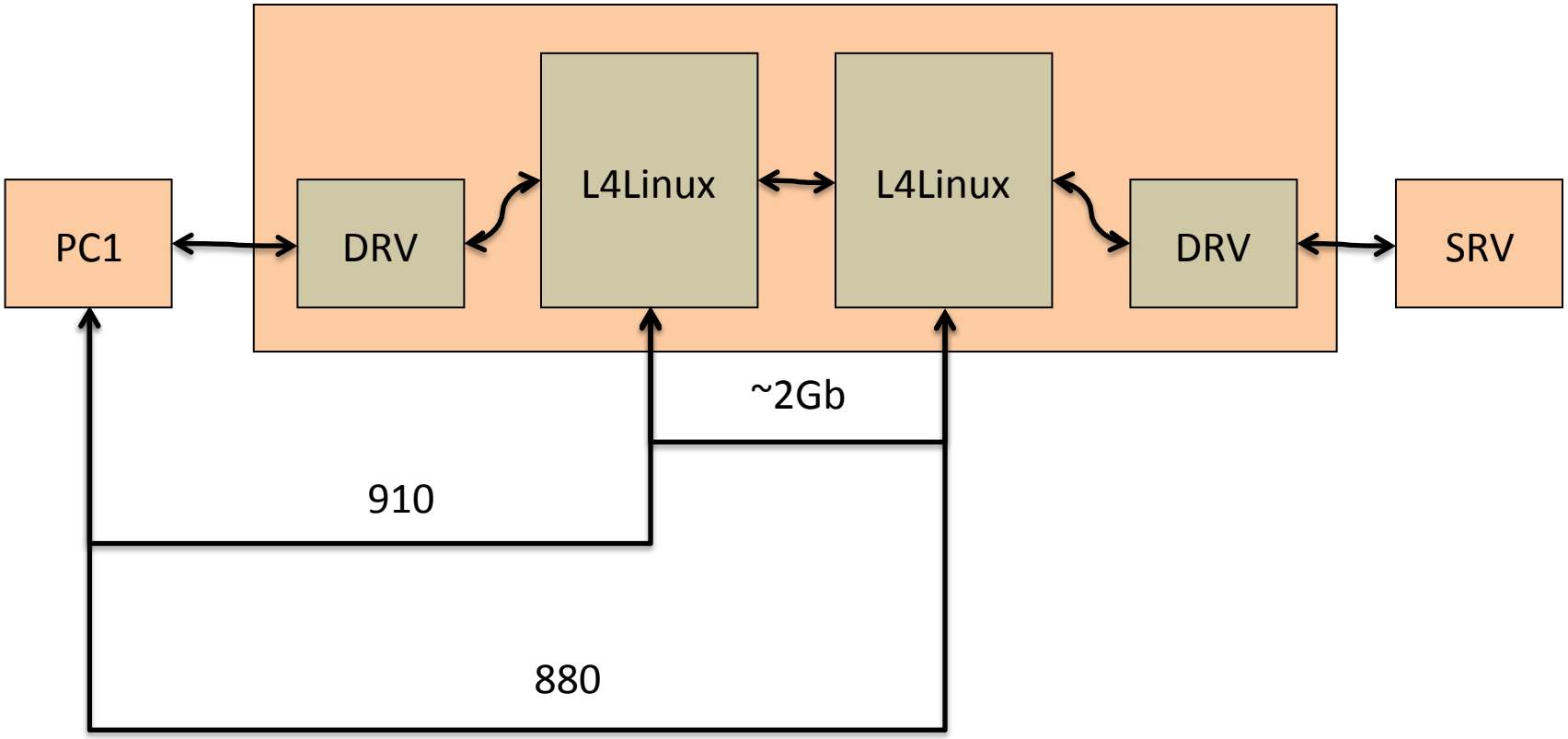
- SMP реализация отличается от реализации для однопроцессорной системы
- mutex + messaging = slow
- idle 30% (?!?!)

Решения

- Хорошие решения:
  - Уменьшить количество threads в драйверах
  - Lockless environment (WIP)
    - Вместо блокировок – использование задержек по времени на подобии Read-Copy-Update

- Плохие решения:
  - Dataspace (shared memory)
  - Прямой доступ L4Linux к устройству

Lockless L4Re



Выводы

- Сами по-себе переключения контекста, количество которых значительно в сравнении с монолитно-модульными ядрами, не всегда приводят к деградации производительности.
- L4 окружения требуют адаптацию прикладного ПО, не смотря на наличие DDE китов
- Нет предела совершенству

Future work

- Lockless окружение
- Custom userspace tcp/ip stack, virtual switch, driver kit

Спасибо.