11th Central and Eastern European Software Engineering Conference in Russia - CEE-SECR 2015

October 22 - 24, Moscow

Кооперативная виртуализация сети в промышленных серверных приложениях на Линуксе

Василий Толстой

EMC²

V. 0.4 pdf

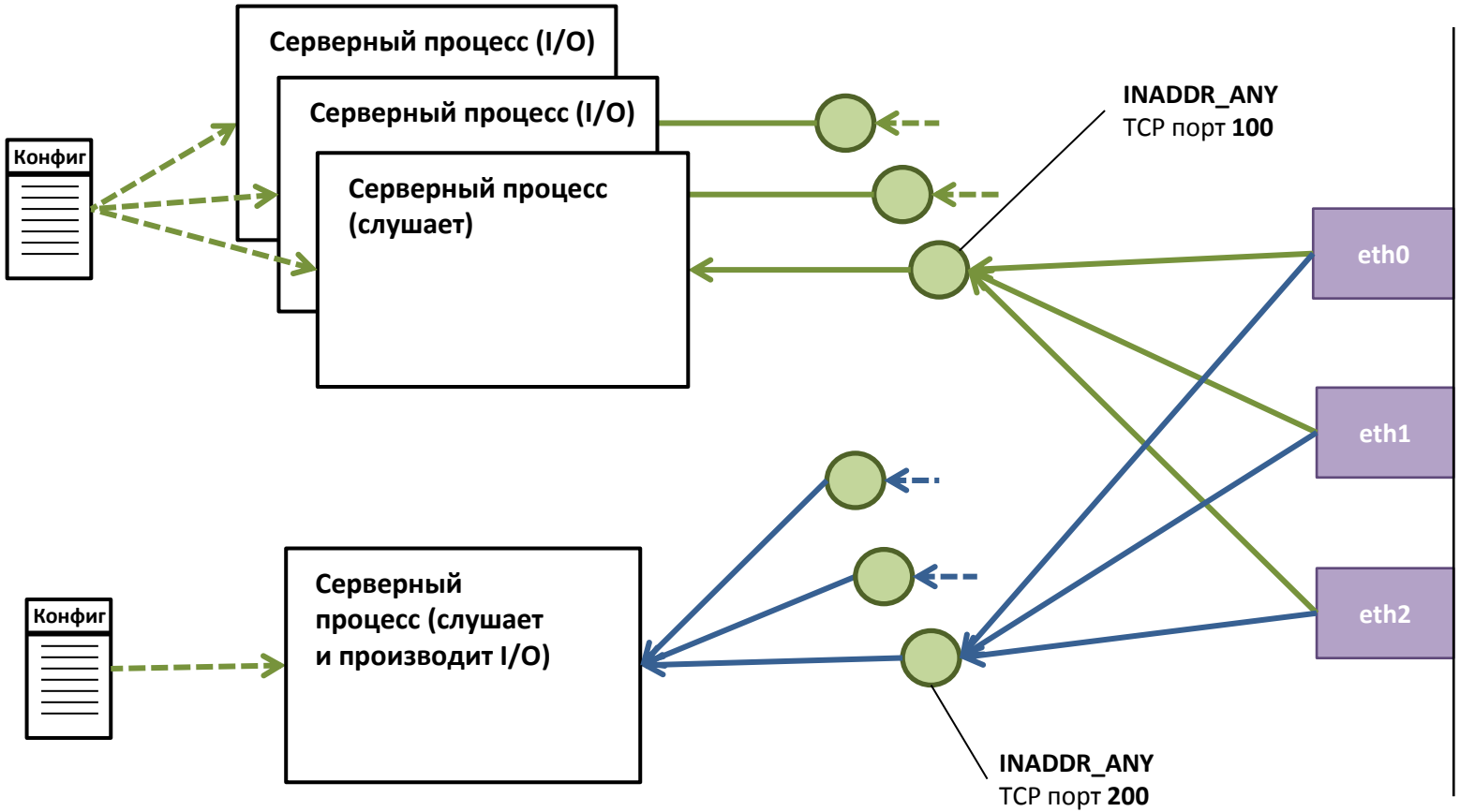
Виртуальные серверы внутри физического сервера

- Что такое сервер?
- Это такая коробка
- Это такое приложение
- Мощность компьютеров растет
- Много серверов внутри одной коробки
- Вместе со своим блоком конфигурации – это виртуальный сервер (ВС)
- Как им ужиться вместе и остаться независимыми?
- В этом докладе: сложный случай, промышленное монолитное серверное приложение

Классическое серверное приложение

- Один процесс, порождающий свои копии
- Другой конфиг => другой головной процесс
- Не различает линки
- Не различает IP своей платформы , работает со всеми одинаково
- Использует основную (main) роутинговую таблицу
- Обнаружение (discovering) через первое пригодное соединение
- Полагается на вспомогательные сервисы (DNS, NTP), у которых один конфиг

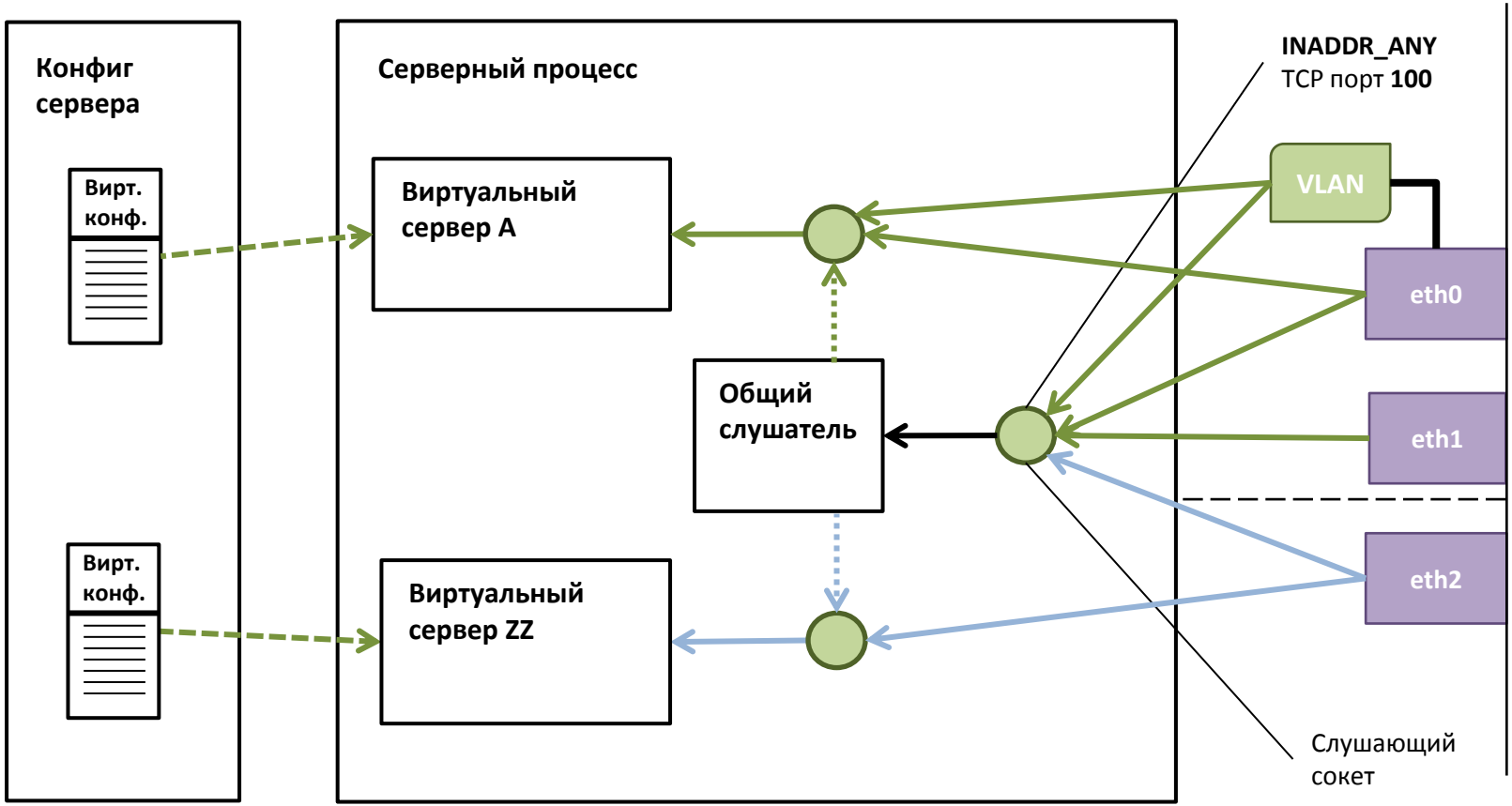
Классический сервер на Linux



Промышленное серверное приложение

- Один многопоточный процесс
- Обслуживает несколько физических линков
- Обслуживает несколько LAN/VLAN/VxLAN => несколько подсетей IP
- Представляет несколько виртуальных серверов (ВС)
- Использует отдельный набор IP-адресов для каждого ВС
- Разделяет вспомогательные сервисы для каждого ВС
- Использует TCP/iSCSI разгрузку (offload)
- И т.д.

Промышленный сервер на Linux



Кооперативная виртуализация: способ адаптировать старую кодовую базу к новым реалиям

- Код уже написан!
- «Память кода»: важный актив
- Виртуализация пришла и не уйдет
- Пример: система хранения данных (СХД):
  - серверное приложение
  - постоянное развитие
  - тесно связанная система
- Линукс уже стоит
- И что делать?

Всё-таки, почему один процесс?

- Меньше переключений контекста и памяти
  - Выше производительность
  - Ниже задержка (latency)
  - Меньше расход памяти
- Проще реализовать обработку без копирования (zero copy)
- Проще построить обработку IP в пользовательском пространстве (user space IP stack)
- Горизонт дедупликации (deduplication scope): чем шире, тем лучше

Кооперативная виртуализация

Кооперативная виртуализация: с помощью самого приложения позволить увидеть его как набор виртуальных серверов

Виртуальные машины	**Policy Based Routing**	**Firewall**
Контейнеры	**Пространства имен**	**Хитрый дизайн приложения**

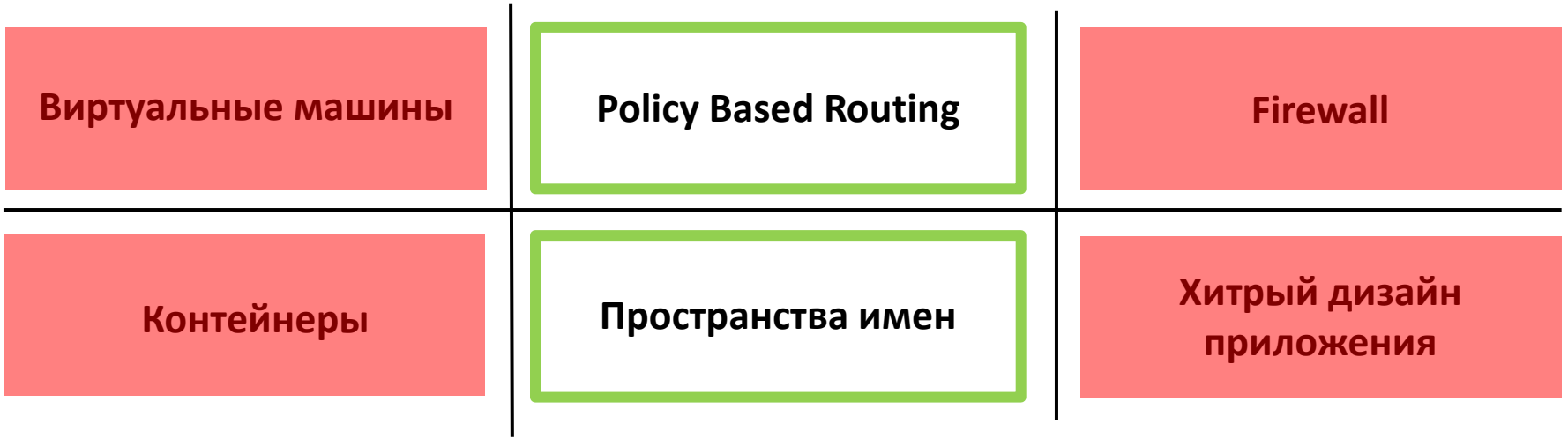
Кооперативная виртуализация

Кооперативная виртуализация: с помощью самого приложения позволить увидеть его как набор виртуальных серверов

Виртуальные машины	**Policy Based Routing**	**Firewall**
Контейнеры	**Пространства имен**	**Хитрый дизайн приложения**

Кооперативная виртуализация

Кооперативная виртуализация: с помощью самого приложения позволить увидеть его как набор виртуальных серверов



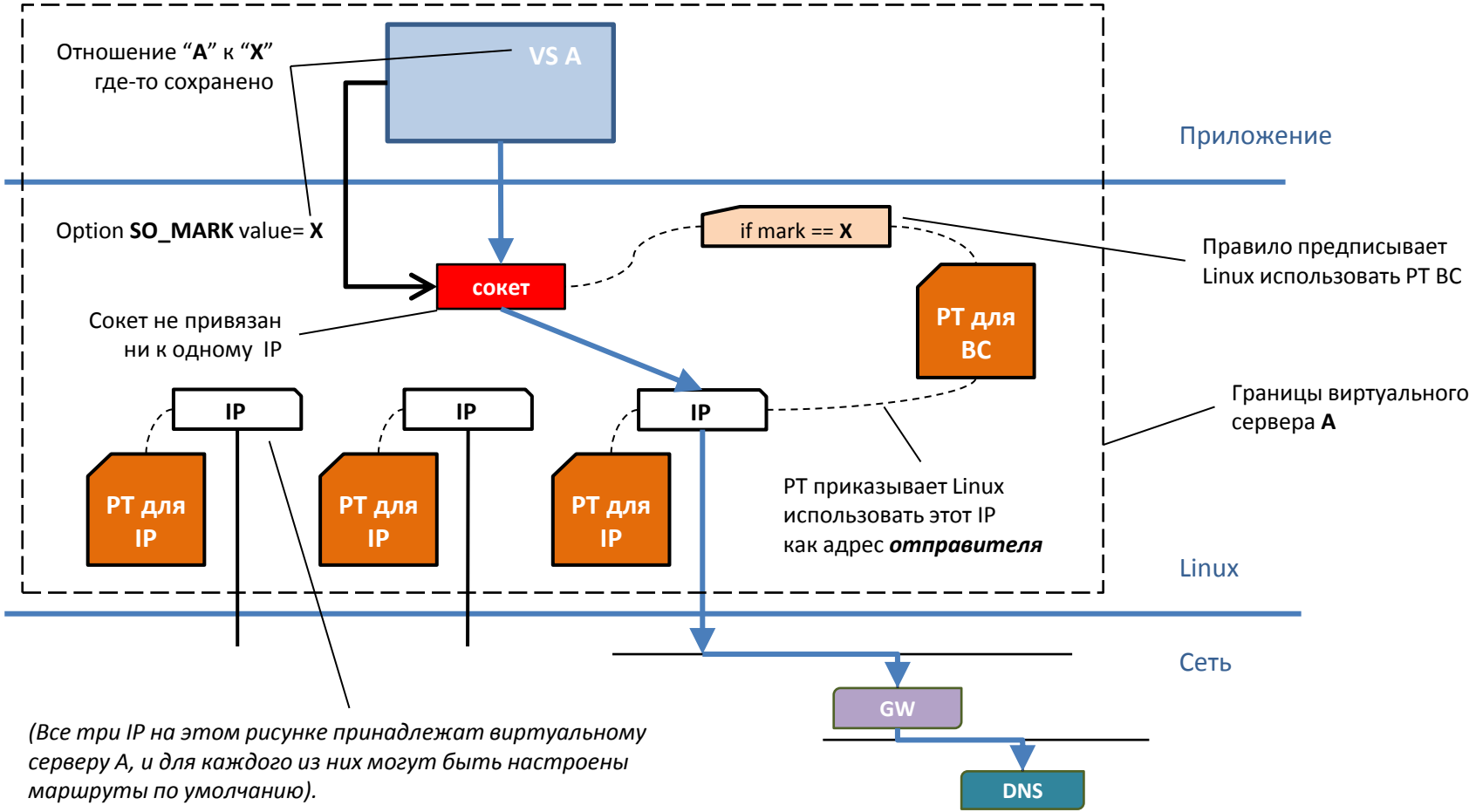
Ключевой момент: разделение траффика

- Надо рассортировать входящий траффик по разным коробкам
- Нет критериев – нет сортировки
- Комбинации из провода+VLANa+IP+порта (?) кажется достаточно
- Но надо сортировать исходящий траффик по тем же коробкам
- В идеале всё сетевое – распихать по тем же коробкам
- «Виртуальный сервер»: что это для сетевого стека?

Policy Based Routing

- Проще всего использовать IP отправителя как критерий в правиле PBR
- А если у ВС несколько IP?
- Входящие TCP соединения – OK. TCP их привязывает к IP
- Личная библиотека разрешения имен – тоже не проблема
- Исходящие соединения – вот проблема!
- Как ограничить исходящие соединения подмножеством IP?
- Множество критериев PBR: может там что-то есть?
- Firewall mark
- fwmark == SO_MARK value: можно пометить сам сокет!

Независимая роутинговая таблица для каждого ВС

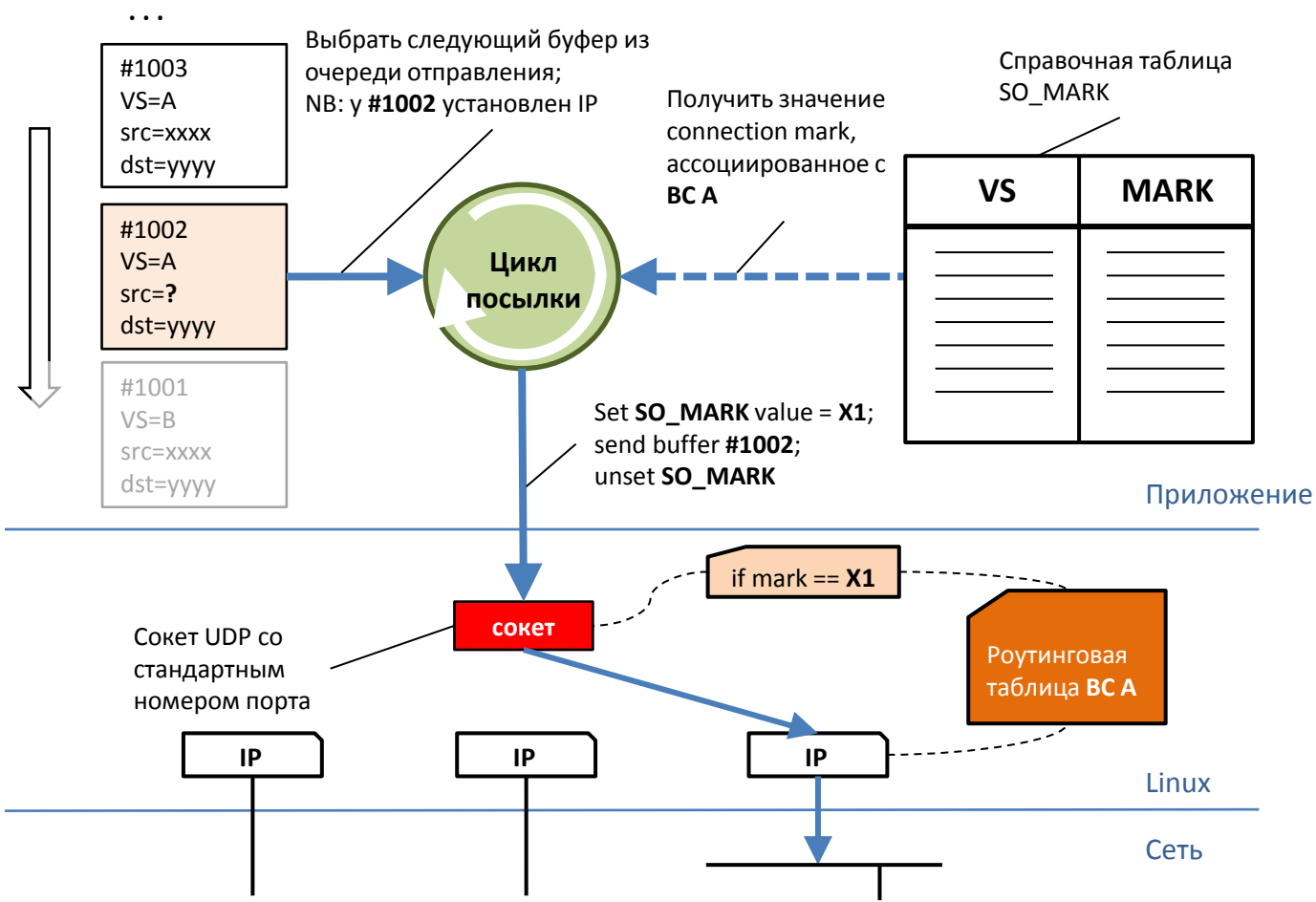


(Все три IP на этом рисунке принадлежат виртуальному серверу A, и для каждого из них могут быть настроены маршруты по умолчанию).

Policy Based Routing: проблема с UDP

- Незадача: соединения UDP не порождают отдельные сокеты
- Дополнительная незадача: клиент ожидает стандартный (well-known) порт отправителя
- Нужно делить сокет между разными ВС
- Что делать с socket mark?
- Переключать на ходу?
- Да! Время переключения ~микросекунд, этого достаточно.
- В идеале – как-то разделить на отдельные сокеты (см. ipvlan)

Последовательное применение опции SO_MARK к датаграммам UDP

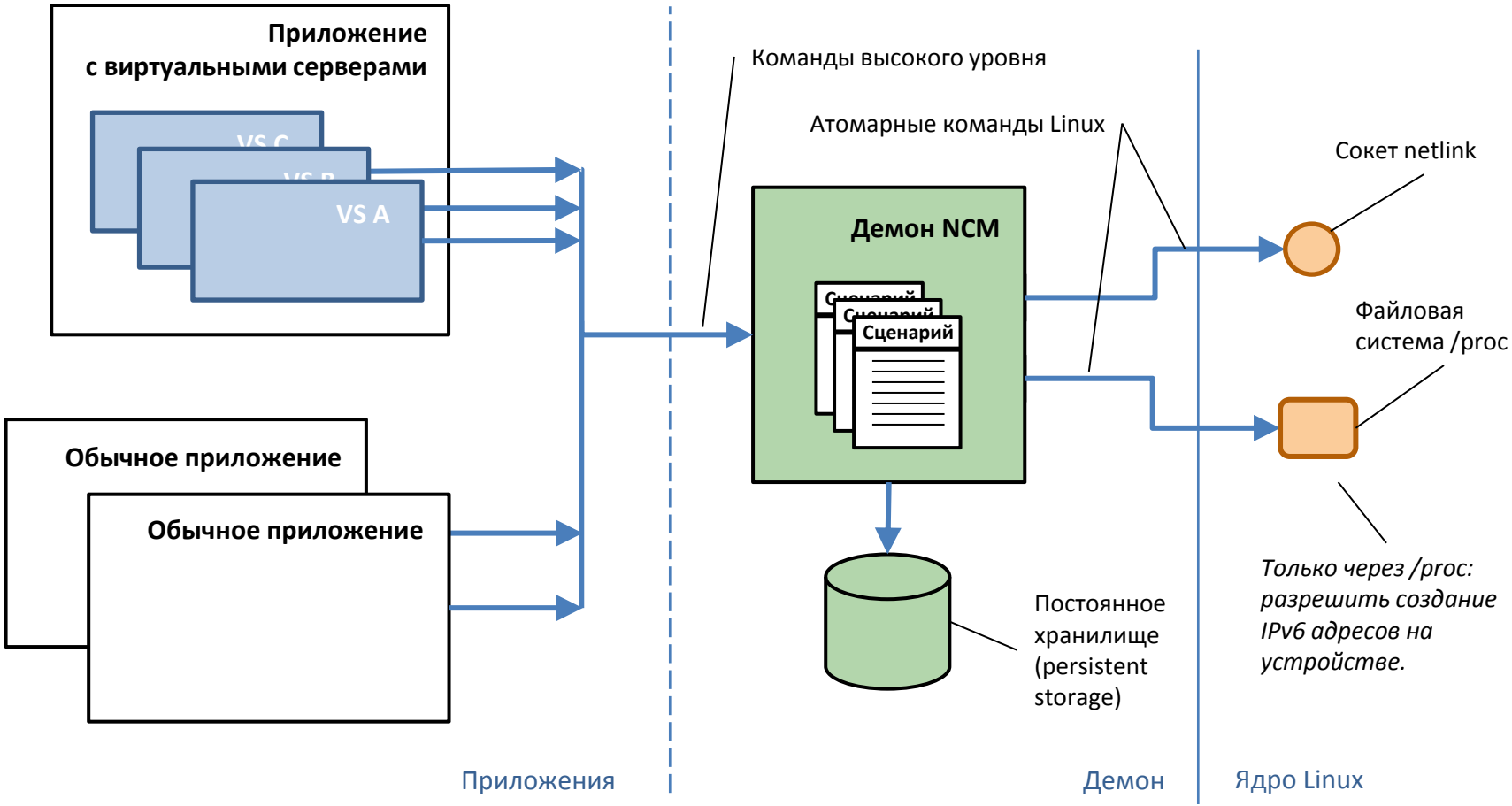


NB: буферы этой очереди отправляются разными виртуальными серверами, не только сервером **A**

Координация сетевой конфигурации: демон NCM

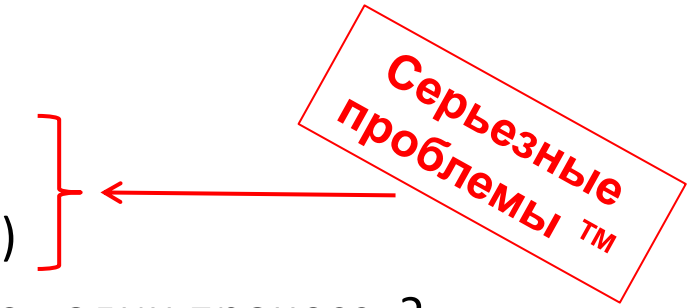
- Сложную сетевую конфигурацию легко сломать
- Приложение не желает вникать в детали
- Принцип суперпозиции (каждый ВС конфигурирует сеть независимо)
- Нужна точка координации запросов на конфигурацию
- Демон NCM (Network Configuration Management Daemon)
- netlink API + /proc
- Пример сложной операции: добавить IP

Демон сетевой конфигурации



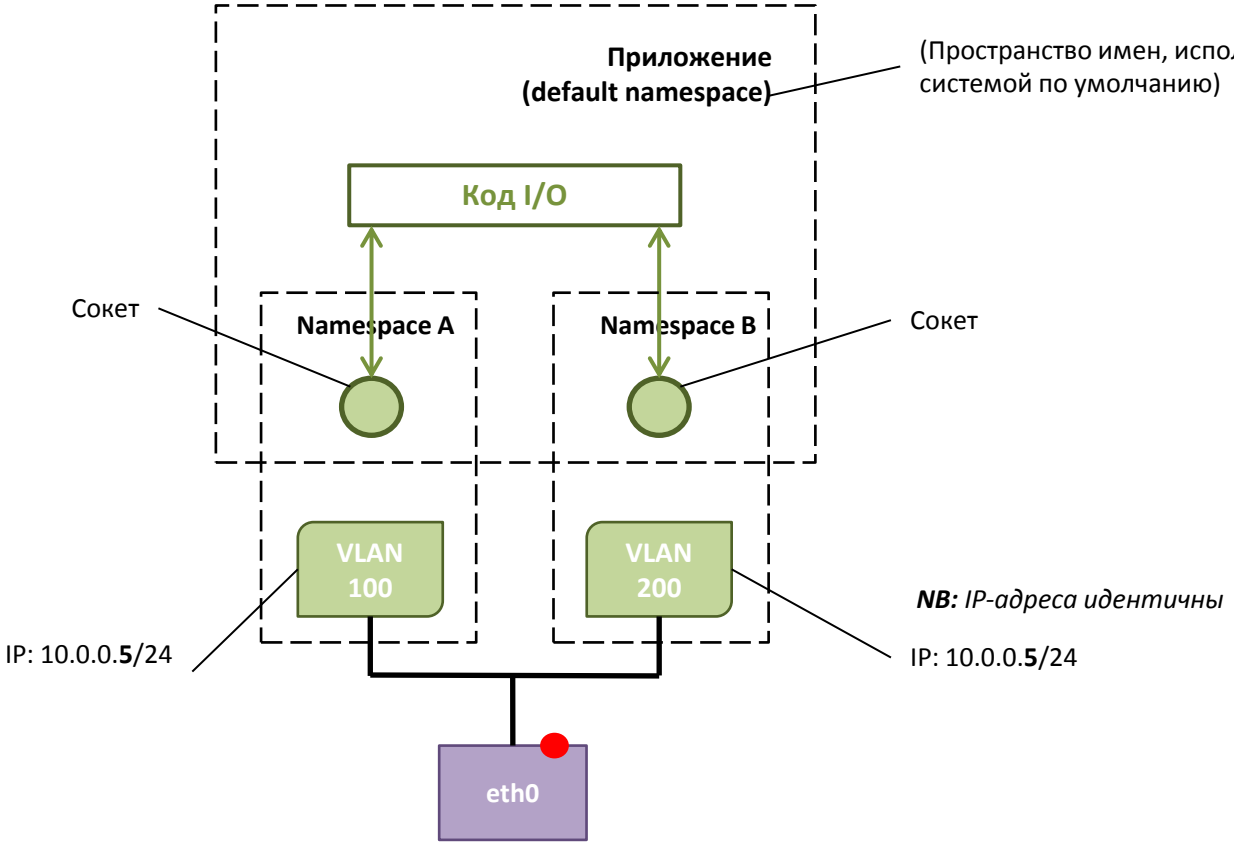
Сетевые пространства имен (network namespaces)

- В PBR нет:
  - Независимой адресации IP
  - Независимых настроек брандмауэра (firewall)

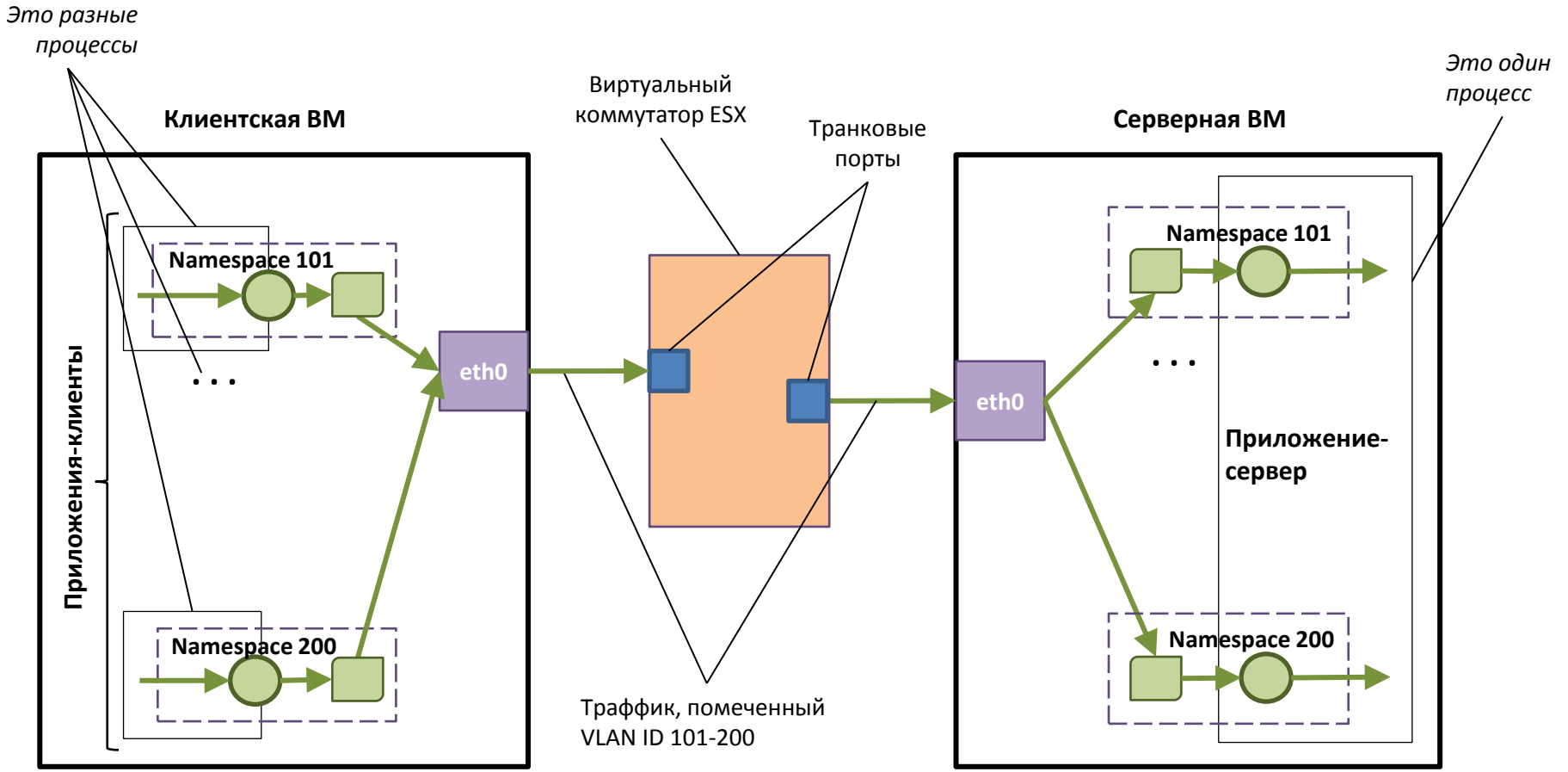


- Пространства имен (namespaces)! Но как же «один процесс»?
- Поток (thread) может перейти в пространство имен и вернуться
- Поток одновременно может работать с сокетами из разных пространств имен (и *epoll()*, и *accept()*, и т.д.)
- Прототип
- Тесты производительности и масштабирования

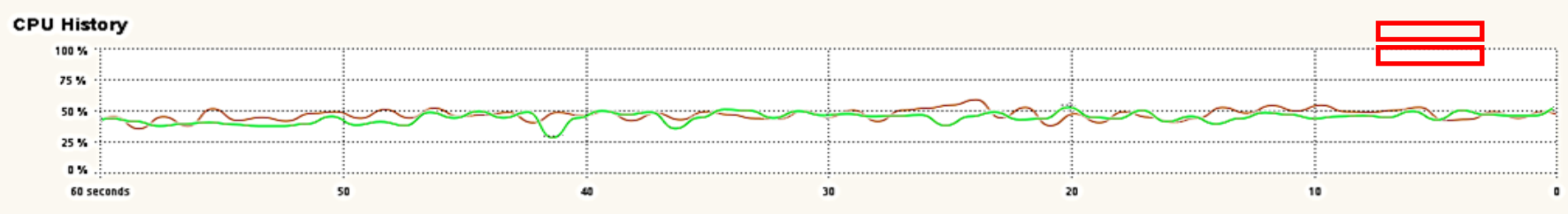
Пространства имен: концепция



Пространства имен: тест производительности и масштабирования



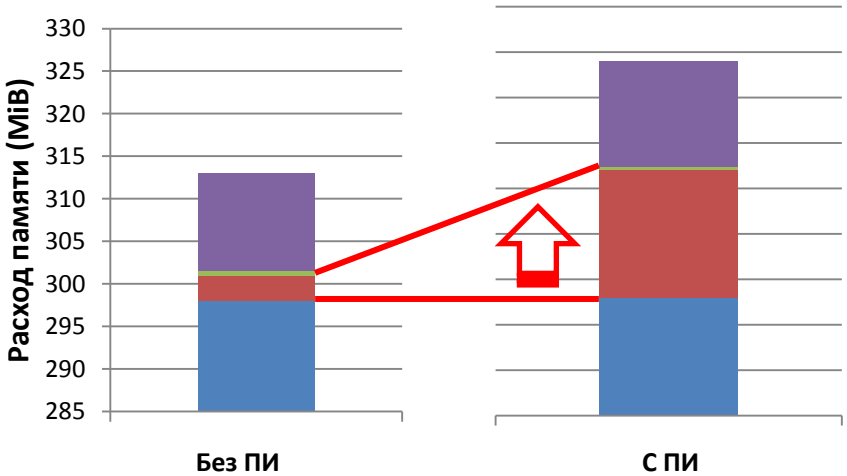
Пространства имен: результаты теста производительности и масштабирования



Без пространств имен

С пространствами имен

Загрузка CPU: не изменилась.



+ I/O

+ 100 слушающих сокетов

+ IP +VLAN +ПИ

База

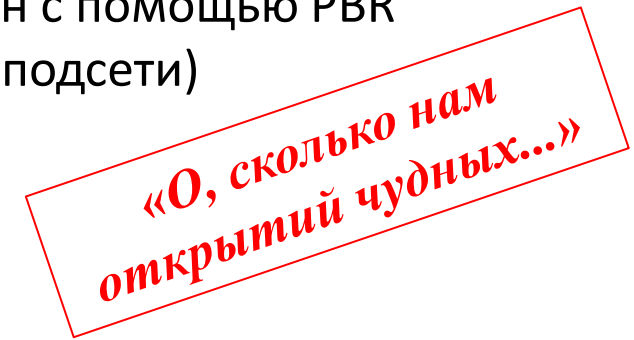
(А в новых версиях ядра – чуть ли не втрое меньше!)

Дополнительный расход памяти: 80 – 120 KB/namespace.

Сетевые пространства имен: проблемы

– It’s alive!!!
– Гм, не совсем.

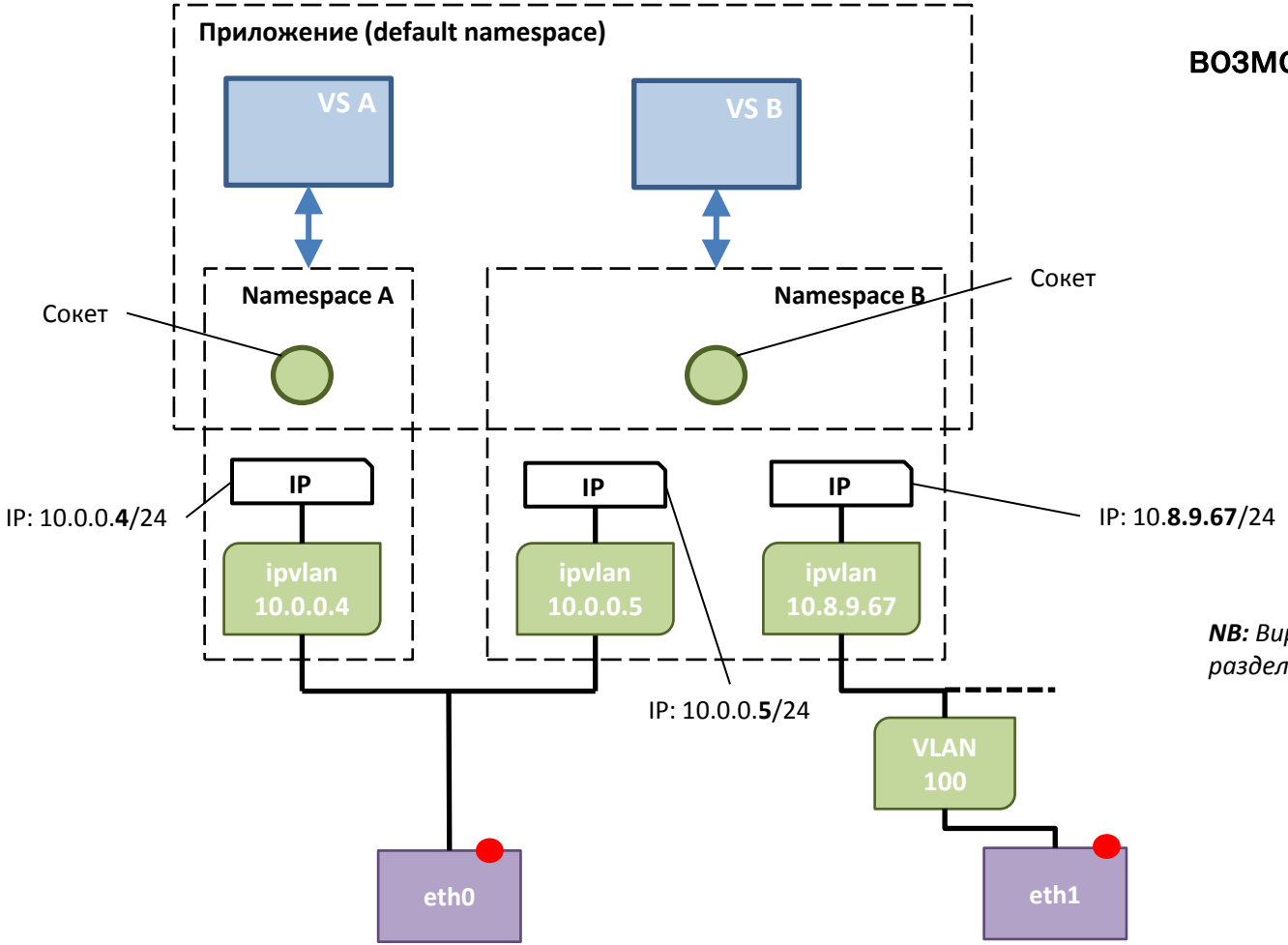
- Нет BM и контейнеров: всё нужно делать самим
- Время (NTP): одно на всю машину
- RPC: нужен portmapper в разных пространствах имен
- Нужно нарезать каждое пространство имен с помощью PBR (потому что два ВС могут иметь IP в одной подсети)
- Этот список – неполон



Разделение траффика по адресам: ipvlan

- Долой магию PBR! Каждому ВС – личное пространство!
  - Умеем делить на пространства только по целым устройствам
  - А если IP двух ВС на одном устройстве?
- Идея: виртуальные устройства для IP!
- Встречаем: ipvlan
- Проблема с широковещательными и групповыми (broadcast and multicast) пакетами: копировать иль не копировать?
  - Если приложение слушает два адреса в одной подсети, получит два пакета
  - Копировать вообще «фу»: в наши-то времена zero copy!

Пространства имен: возможная конфигурация с ipvlan



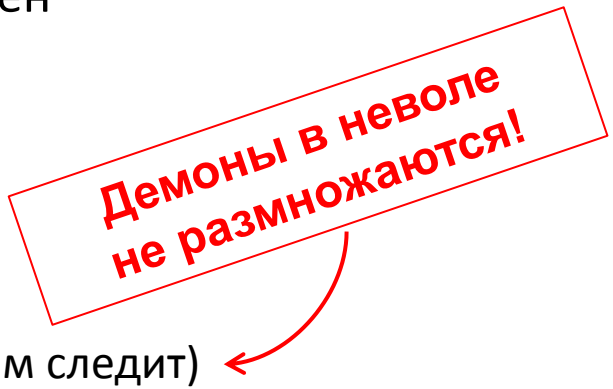
– действующий MAC

– слушающие сокеты

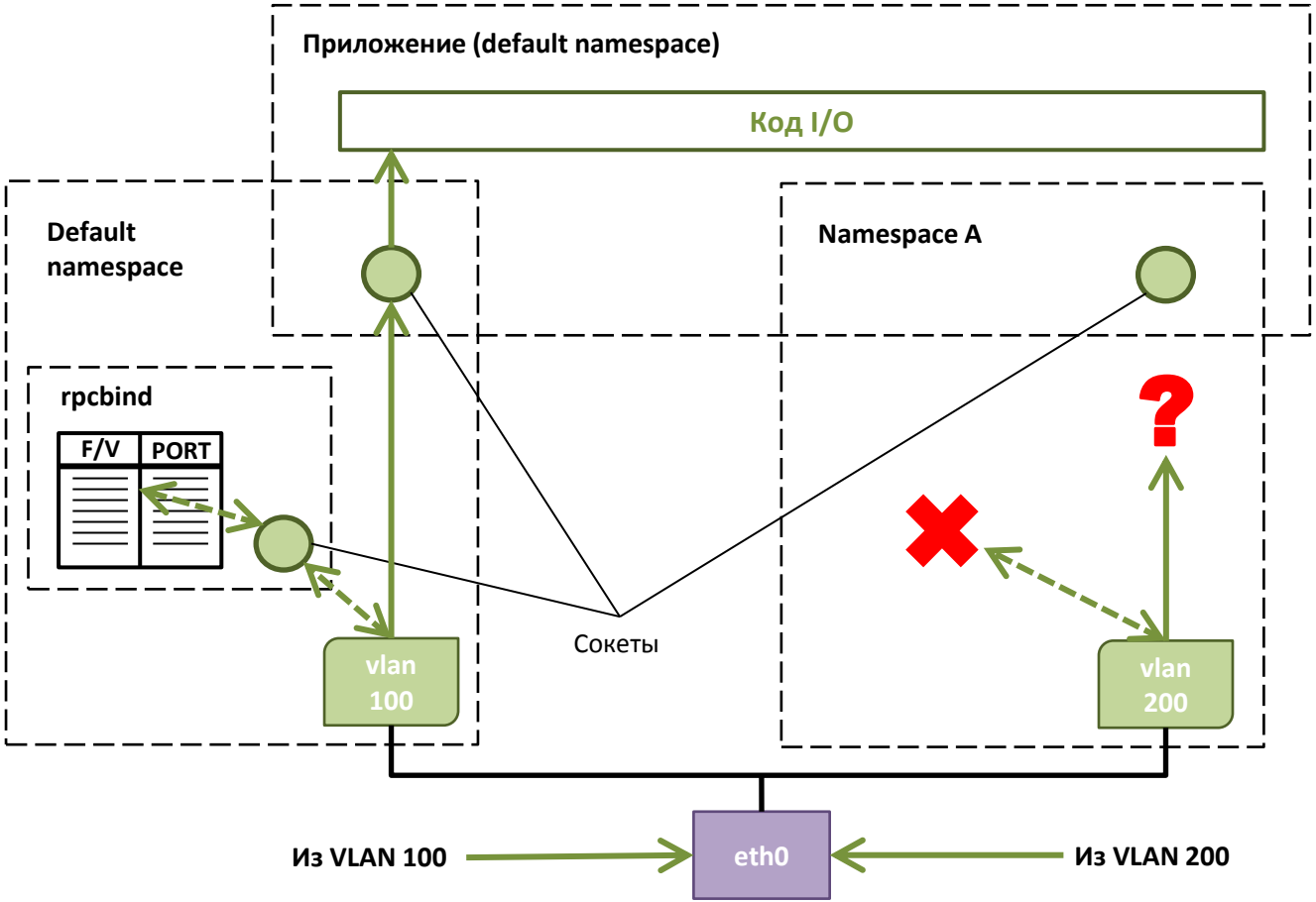
NB: *Виртуальные устройства VLAN разделяются между пространствами имен.*

Демоны и пространства

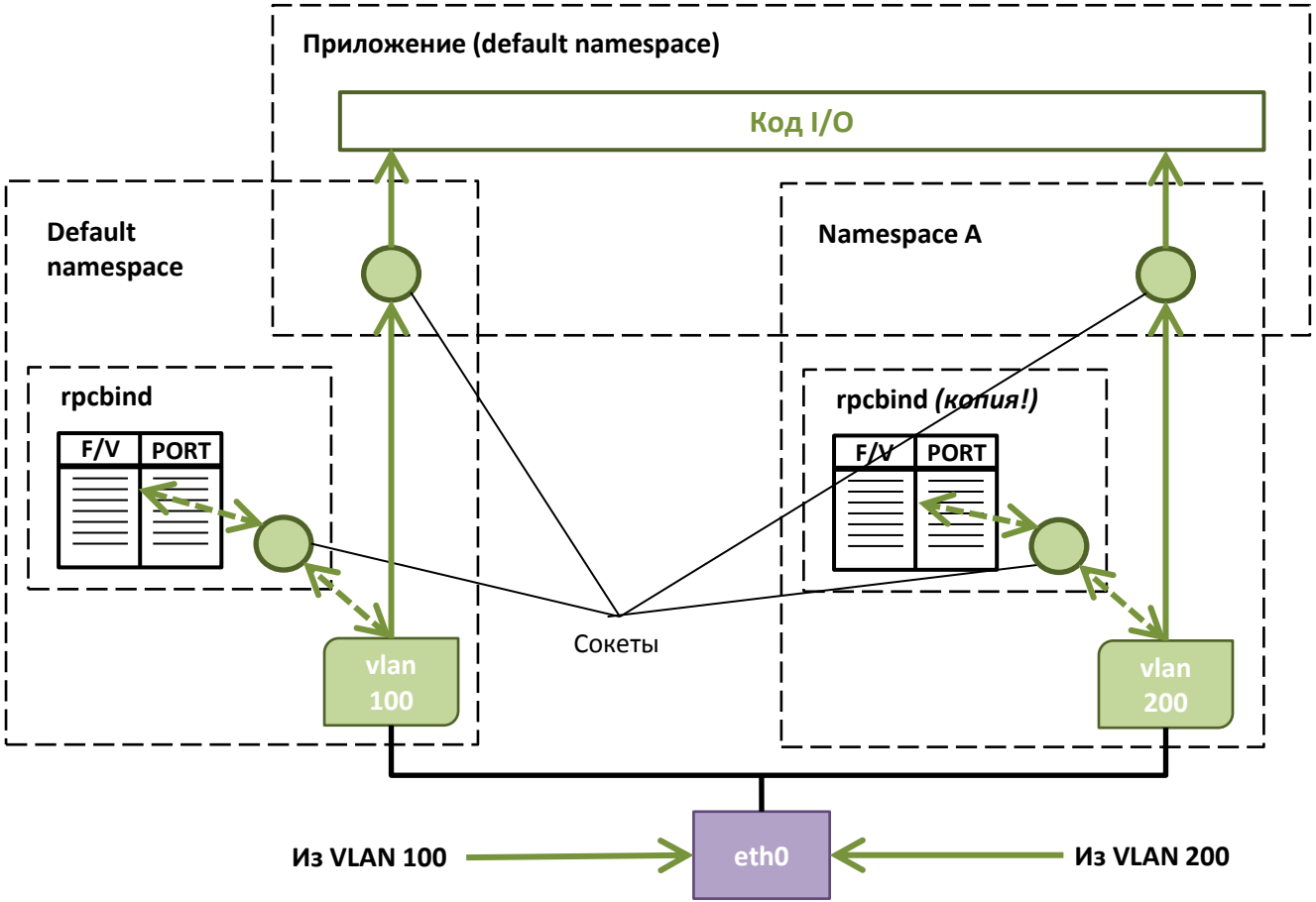
- Библиотека разрешения имен (resolver): копия для каждого процесса
- Демона «не видно» в другом пространстве имен
- Варианты:
  - По демону в пространстве
  - Демон работает с несколькими пространствами
- Проблемы:
  - У демона – единственный экземпляр (и он за этим следит)
  - Мы создаем и уничтожаем пространства на лету
- Хороший пример: RPC portmapper (rbcbind)



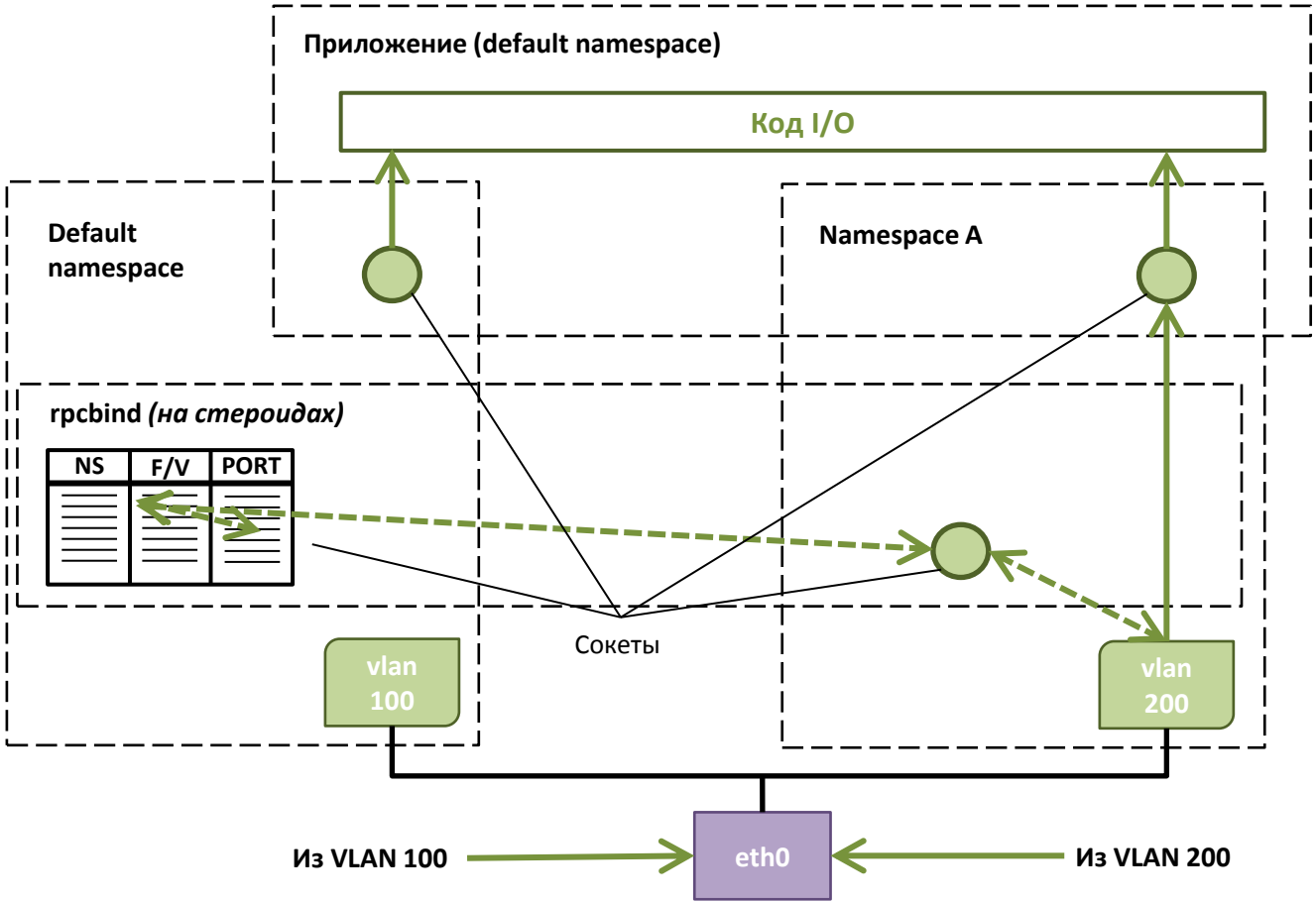
RPC в пространствах имен: стандартный демон



RPC в пространствах имен: размножение демона



RPC в пространствах имен: модификация демона



Итоги на сегодня

- Промышленное, монолитное серверное приложение как набор независимых виртуальных серверов: получилось!
- Модификация кода приложения: нужна, но это приемлемо
- Несколько известных трюков в Linux
- «Из коробки» Linux работает иначе
- Демон сетевой конфигурации
- Разделение вспомогательных сервисов: требует усилий
- Интересные перспективы развития

2015 CEE-SEC(R) Разработка ПО

Спасибо!

Василий Толстой

Центр разработок EMC

Россия, Санкт-Петербург

tolstv@emc.com

EMC²

Дополнительные слайды

Промышленное серверное приложение: «а также»

Дополнительно в таком приложении:

- Поддерживается зеркало траффика: ответ через то же устройство и по тому же пути, откуда получен запрос
- Исходящие соединения ограничены «горизонтом» виртуального сервера
- Максимальное переиспользование сетевого стека Линукса (это упрощает поддержку кода)
- Быстрый старт и быстрое конфигурирование сети
- Максимальное использование аппаратной разгрузки (hardware offload) для TCP и iSCSI

Процедура установки IP-адреса демоном NCM

Пример входных параметров:

VS conmark: 7
Port: eth0
VLAN: 1077
IP: 10.22.33.56
Mask: 255.255.255.0
Gateway: 10.22.33.1

Шаги:

- Прочитать состояние сетевого стека.
- Сгенерировать и зарезервировать свободный номер роутинговой таблицы для добавляемого адреса.
- Проверить, есть ли на запрошенном устройстве (eth0) нужное виртуальное устройство mod8021q (eth0.1077).
- Если такого нет, создать его.
- Создать на устройстве eth0.1077 IP-адрес (10.22.33.56).
- Создать правило для роутинговой таблицы этого адреса (ip rule add...) с зарезервированным номером.
- Заполнить таблицу с этим номером, добавив туда два роута:

  10.22.33.0/24 --> eth0.1077;

  default via 10.22.33.1

- Удалить из таблицы main роут в подсеть, созданный для нашего адреса системой.
- Найти таблицу с указанным номером VS conmark.
- Если в ней отсутствует роут в подсеть, соответствующую нашему адресу, добавить такой роут.
- Объявить о новом адресе во внешней сети, послав с соответствующего устройства принудительный ARP-ответ.

Примечания:

Приведенное описание неполно: в нем не разобраны некоторые особые случаи, например создание двух адресов одной подсети или отсутствие шлюза по умолчанию.

Установка адреса IPv6 требует дополнительных шагов с использованием /proc и альтернативного протокола объявления адреса.

Реализация этих шагов на практике потребует базы данных для постоянного и временного хранения состояний системы, реализации алгоритма отката всей процедуры при ошибках и т.д.