Содержание

Реферат

TODO: Написать реферат

Термины

Гетерогенная сеть - информационная сеть, в которой работают различные протоколы, используются технологии и оборудование различных фирм-производителей.

Интерфейс - формализованные правила, в соответствии с которыми взаимодействуют модули, реализующие протоколы соседних уровней модели сетевого взаимодействия (набор сервисов, предоставляемых данным уровнем соседнему).

Протокол - формализованные правила, определяющие последовательность и формат сообщений, которыми обмениваются сетевые компоненты, лежащие на одном уровне модели сетевого взаимодействия в разных узлах.

IP (Internet Protocol) — маршрутизируемый протокол сетевого уровня без установления соединения.

Сокет — пара IP-адрес: порт, однозначно определяющая сетевой процесс; точка доступа прикладного процесса к сети.

Обозначения и сокращения

В текущей расчетно-пояснительной записке применяется следующие сокращения и обозначения.

ОС — Операционная система

IP — Internet Protocol

OSI — Open Systems Interconnection

TCP — Transmission Control Protocol

IETF — Internet Engineering Task Force

BSD означает "Berkeley Software Distribution"

DNS Аббревиатуры - ОС, TCP/IP, LAN, UUCP, IPX Определения - фрагментации

Введение

Организация взаимодействия между устройствами и программами в сети является сложной задачей. Сеть объединяет разное оборудование, различные операционные системы и программы – это было бы невозможно без принятия общепринятых правил, стандартов. В области компьютерных сетей существует множество международных и промышленных стандартов, среди которых следует особенно выделить международный стандарт OSI и набор стандартов IETF.

В ОС такую задачу реализуют сетевая подсистема, что позволяет иметь широкий спектр сетевых возможностей. Сетевая подсистема, выполняющаяся в режиме ядра, естественным образом ответственна за управление сетевыми устройствами ввода-вывода, но кроме этого на нее также возложены задачи маршрутизации и транспортировки пересылаемых данных. Современные ОС Linux требует контроля по причине того, что безопасность ядра не идеальна в том числе и сетевая подсистема ядра [1, 2].

Обычно для отладки сетевых ошибок необходимо проверить все узлы, участвующие в сетевом взаимодействии: отправляющий, связующие и принимающий. Однако из-за сложной конфигурации, в сети возникают неочевидные связи и взаимодействий между различными элементами, что сильно осложняет процесс поиска источника неполадки даже в рамках одного узла. Неизбежно возникнет ситуация, в которой непонятно, с какой стороны подступиться проблеме. Тогда разработчику придется перебирать возможные причины возникновения ошибки, используя весь набор доступных для сетевой отладки инструментов и полагаясь лишь на свой профессиональный опыт или интуицию. Такой бессистемный подход приводит к высокой стоимости устранения сбоев.

Целью работы является провести анализ существующих средств мониторинга сетевых подсистем ядра ОС Linux.

Для достижения поставленной цели необходимо решить следующие задачи:

— провести анализ предметной области подсистем ядра ОС Linux;

— провести обзор существующих подсистем и средств сетевого мониторинга ядра ОС Linux;

— сформулировать критерии сравнения средств сетевого мониторинга ядра;

— классифицировать существующие подсистемы и средства сетевого мониторинга.

1 Анализ предметной области

Прежде чем приступить к обзору и анализу существующих средств сетевого мониторинга ядра Linux, необходимо определить что подразумевается под сетевым мониторингом ядра Linux

1.1 Ядро Linux

Ядро Linux — это основной внутренний компонент операционной системы Linux, отвечающие за распределением системными ресурсами, управление аппаратным обеспечением и обеспечение взаимодействие приложения с аппаратным обеспечением [3].

В Linux системах ядро является монолитным с модульной конструкцией, которое реализовано в виде одного большого процесса, выполняющий в одном адресном пространстве. Такие ядра обычно хранятся на диске в виде одного большого статического бинарного файла. Все службы ядра находятся и выполняются в одном большом адресном пространстве ядра. Взаимодействия в ядре осуществляются очень просто, потому что все, что выполняется в режиме ядра, выполняется в одном адресном пространстве. В отличие от микроядра, который разделяет службы ядра на несколько процессов, называемыми серверами. Ядро может вызывать функции непосредственно, как это делают пользовательские приложения.

Главным задачами ядра в первую очередь является:

1) обеспечение среды выполнения для программ в операционной системе;

2) взаимодействие с аппаратными компонентами и обслуживание их низкоуровневые элементы.

В современных системах с устройствами управления защищенной памятью ядро обычно занимает привилегированное положение по отношению к пользовательским программам. Это включает доступ ко всем областям защищенной памяти и полный доступ к аппаратному обеспечению. Системные переменные (system state) и область памяти, в которой находится ядро, вместе называются пространством ядра (kernel-space), или привилегированным

режимом, или также называется «режим ядра». Соответственно, пользовательские программы выполняются в пространствах задач (user-space), или в «пользовательском режиме».

1.2 Подсистемы ядра

Ядро Linux разделяется на ряд подсистем, рисунок 1.1, как на высоком, так и на низких уровнях. Такое разделение позволяет упростить разработку ядра и сделать его более гибким. Каждая подсистема выполняет реализует свой функционал, такие как управление памятью, управление и взаимодействие процессов, файловая система, сетевые стек, которое используется другими подсистемами.

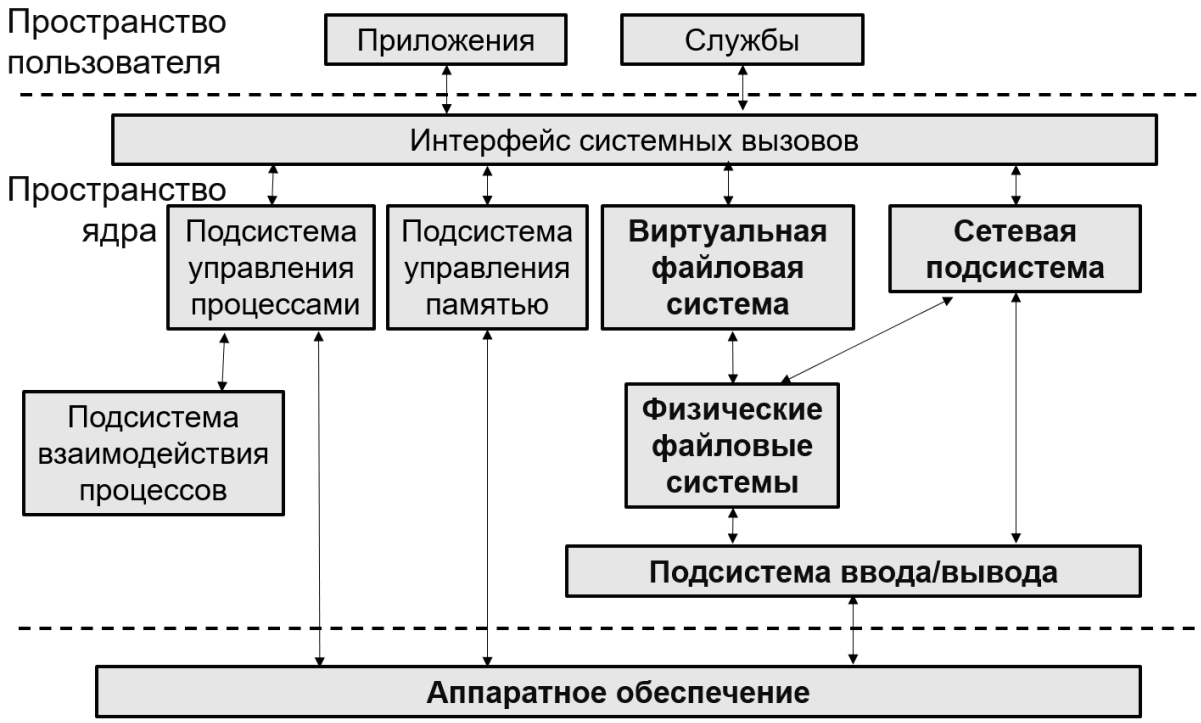

Рисунок 1.1 – Общая архитектура Linux

1.3 Сетевая подсистема ядра

Сетевая подсистема ядра Linux [4, 5] обеспечивает взаимодействие процессов, выполняющихся на разных узлах сети, то есть дает возможность сетевого взаимодействия между приложениями .

Сетевая подсистема ядра Linux реализует следующий функционал:

— поддержку взаимодействия процессов с помощью механизмов сокетов (sockets);

— реализацию стеков сетевых протоколов (TCP/IP, UDP/IP, IPX/SPX и другие);

— поддержку сетевых интерфейсов;

— обеспечение маршрутизации пакетов (routing);

— обеспечение фильтрации пакетов (netfilter).

Взаимодействие сетевой подсистемы с другими подсистема ядра показана на рисунке 1.1.

1.3.1 Многоуровневая модель

Для решения задачи сетевых взаимодействий применяется многоуровневый иерархический подход, заключающийся в разбиении процесса коммуникации на набор уровней с четко определенными способами взаимодействия уровней на одном узле и на соседних узлах. Таким образом в подсистеме существует сетевой стек, который является производной стека BSD. Сетевой стек хорошо оснащен добротным набором интерфейсов, которые варьируются от протоколо-независимых (protocol agnostic), таких как интерфейс уровня общих сокетов или уровня устройств, до специальных интерфейсов конкретных сетевых протоколов. Уровень сокетов представляет собой стандартный API к сетевой подсистеме. Он предоставляет пользовательский интерфейс к различным сетевым протоколам. Уровень сокетов реализует стандартизованный способ управления соединениями и передачи данных между конечными

точками, от доступа к «чистым» кадрам данных и блокам данных протокола IP/PDU, и до протоколов TCP/UDP.

В то время как работа в сети отсылается к модели сетевого взаимодействия по OSI, сетевой стек в Linux использует модель TCP/IP [6, 7], включающая в себя 4 уровня:

1) уровень сетевых интерфейсов (канальный уровень) относится к драйверам устройств, обеспечивающим доступ к физическому уровню, который может состоять из многочисленных сред, таких как последовательные каналы или устройства Ethernet, описываются как сетевые интерфейсы;

2) уровень межсетевого взаимодействия (сетевой уровень) обеспечивает работу базовой службы доставки пакетов по назначению;

3) транспортный уровень обеспечивает надежную доставку данных со сквозным обнаружением и устранением ошибок;

4) прикладной уровень отвечает за взаимодействие с приложениями и процессами на хостах, также определяются пользовательские интерфейсы процесса или приложения, наблюдается работа протоколов и служб — FTP, Telnet и другие.

Сетевая модель TCP/IP условно согласуется с моделью OSI, включающая в себя 7 уровней:

1) физический уровень;

2) канальный уровень;

3) сетевой уровень;

4) транспортный;

5) сеансовый уровень;

6) представительский уровень;

7) прикладной уровень.

На рисунке 1.2 показано как модели TCP/IP и OSI пересекаются между собой.

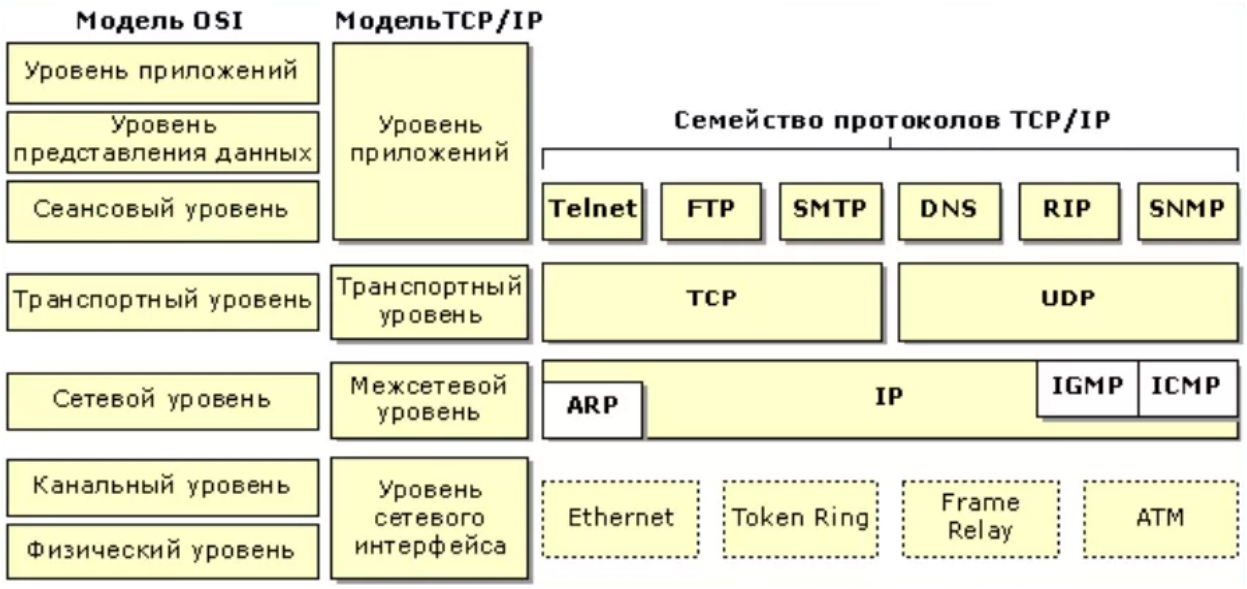

Рисунок 1.2 – Модель OSI и TCP/IP

1.3.2 Сетевой интерфейс

Сетевые интерфейсы являются основой сетевой подсистемы, иначе говоря абстракцией, используемые для представления связи устройств сети с протоколом TCP/IP и передачи данных через некоторые линии связи. Основными устройствами, позволяющими организовывать взаимодействие по сети, являются сетевые адаптеры (Ethernet-карты).

Интерфейс имеет набор параметров, большинство которых относятся к сетевому уровню (IP-адрес, маска сети и т.п.). Важным параметром сетевого интерфейса является аппаратный адрес. В Ethernet аппаратный адрес называется MAC-адрес и состоит из шести байтов, которые принято записывать в шестнадцатеричной системе исчисления и разделять двоеточиями.

TODO: возможно включить подробности

1.4 Сетевой мониторинг ядра

Сетевой мониторинг ядра Linux — это контролирование пакетов и нахождение наличие ошибок сетевых интерфейсов сетевой подсистемы, что относится вмешательством в работу подсистемы и ее сетевого стека.

TODO: подробнее описать ПОПРАВИТЬ Часто мониторинг сетевой подсистемы операционной системы заканчивается на счетчиках пакетов, октетов и ошибок сетевых интерфейсах. Но это только 2й уровень модели OSI! С одной стороны большинство проблем с сетью возникают как раз на физическом и канальном уровнях, но с другой стороны приложения, работающие с сетью оперируют на уровне TCP сессий и не видят, что происходит на более низких уровнях.

2 Существующие решения

В данном разделе рассмотрены основные подсистемы и средства сетевого мониторинга ядра Linux, которые используются в настоящее время.

2.1 Обзор методов сетевого мониторинга ядра Linux

2.1.1 Утилиты для сетевого мониторинга

Для современных Linux существует множество утилит для конфигурации и устранения неполадок сетевой подсистемы ядра. В основном это инструменты командой строки. Наиболее распространенные и часто используемые из них iproute2, ethtool, ping, traceroute, nslookup, netcat, iptables, tcpdump [**?**], с помощью которых можно узнать конфигурацию сетевых пакетов, проверить наличие соединения и работоспособность DNS, просмотреть таблицу маршрутизации, изучить содержание сетевых пакетов и многое другое, что может помочь определить сбои в сетевой подсистеме.

Плюсы:

1) данные утилиты оптимизированы и не могут навредить подсистемам ядра;

2) имеют документацию о задаче утилиты;

3) достаточно легки в использовании.

Недостатки:

1) такое множество средств заточено под определенные задачи и их функциональность ограничена программным интерфейсом, предоставленным ядром ОС;

2) для решение сложных задач необходимо и использовать комбинацию из множества утилит или же не могут справится с такой задачей.

2.1.2 Модификация кода ядра Linux

Открытость ОС Linux позволяет реализовать средства получения информации о событиях, происходящих в сетевой подсистеме ядра, путем модификации исходного кода ядра, что дает возможность находить сбои.

Модификация ядра Linux — это изменение или модификация кода, которое не несет изменение структур ядра Linux. Модификация с целью сетевого мониторинга [8] изменение кода сетевой подсистемы ядра Linux, путем добавление функционала вывода информации в системный журнал.

Плюсы:

1) данный подход позволяет решить любую задачу;

2) ...

Недостатки:

1) модификации ядра имеют доступ ко всему функционалу ядра Linux, включая системные вызовы, коммуникацию с устройствами и т.д. При неверном изменении кода ядра или же сложность модификации повышает вероятность возникновения критических ошибок, способных нарушить работоспособность системы, вследствие чего нельзя достичь необходимого уровня безопасность;

2) для добавления кода в ядро Linux необходимо строго уметь работать с зыком C и компилятором под данный язык, включая инструменты конфигурации ядра Linux. Необходимо знать интерфейс прикладного программирования ядра (API)[9]. что усложняет использование данного подхода и требует крайне высоких компетенций разработчика, а иногда команды разработчиков.

3) также как и при добавлении кода напрямую в ядро, если при работе модуля возникнет ошибка, то есть шанс, что система также экстренно завершит работу;

4) исходя из вышеуказанного процесс модификации довольно длителен.

2.1.3 Зондирование ядра Linux

Второй способ сетевого мониторинга ядра Linux, основывается на встраивании модулей ядра, берет свое начало аналоговой реализации от IBM — DProbes. DProbes включали в себя, помимо основного зондирующего механизма, обратный интерпретатор, обработчики зондов которого могут быть реализованы как простые функции на языке C, которые будут выполняться в контексте ядра, если они скомпилированы как модуль ядра или даже скомпилированы в ядро.

Использование зондирование ядра (от англ. kernel probe, kprobes) [10, 11] позволяет входить в работающее ядро для целей отладки, трассировки, оценки производительности, нахождение ошибок и т.п., путем установления точек останова. По достижению точки останова, возникает ловушка, регистры сохраняются, а управление передается к соответствующей функции написанного модуля ядра. После завершения данной функции работа ядра возвращается.

Большая часть функционала ядра поддерживает зондирование, поэтому с его помощью можно исследовать все ядро. Для того что бы отследить действия сетевой подсистемы необходимо написать модуль ядра для всех функций сетевой подсистемы. В каждом модуле обозначается имя соответствующей функции и описывается метод, который будет отвечать за мониторинг подсистемы.

Плюсы:

1) данный подход позволяет решить любую задачу;

2) реализация независимости от конкретной версии ядра упрощает за счет того, что для разных версий нужно адаптировать только название функций.

Недостатки:

1) ответственность за безопасность возрастает благодаря вынесению функциональности в отдельные модули ядра, что повышает вероятность нарушить работу ядра;

2) за счет динамической загрузки модулей ядра, появляется необходимость постоянной пересборки определенным разработчиком модулей при изменении, что осложняет работу;

3) для добавления модуля в ядро Linux необходимо строго уметь работать с зыком C и компилятором под данный язык, включая инструменты конфигурации ядра Linux. Необходимо знать интерфейс прикладного программирования ядра (API)[9]. что усложняет использование данного подхода и требует крайне высоких компетенций разработчика, а иногда команды разработчиков;

4) ...

2.1.4 Точки трассировки

2.1.5 ftrace

2.1.6 Berkeley Packet Filter

2.1.7 Extended Berkeley Packet Filter

2.2 Критерии сравнения методов сетевого мониторинга

В данном разделе будут описаны критерии, которые будут использоваться для сравнения подсистем и средств сетевого мониторинга ядра.

Таблица 2.1 – Критерии сравнения подсистем и средств сетевого мониторинга ядра

Критерий	**Описание**
Производительность	Быстрота работы программ.
Безопасность	Наличие гарантии, что внесенный код не вызовет остановку системы.
Скорость разработки	Скорость разработки , если оно требуется.
Гибкость	Возможность подстроиться под любые поставленные задачи.
Простота отладки	Является ли написанные модификации простыми в отладке, если отладка имеется.
Поддержка	Поддержка разработчиками ядра при его написании, зависимость от версий.
Простота развертывания	Насколько сложно развертывать средства мониторинга на большом количестве машин.
Простота использования	Насколько сложно использовать данное средство мониторинга.

2.3 Сравнение методов сетевого мониторинга

Заключение

TODO: заключение написать

Список использованных источников

1. Количество багов ядра по модулям [Электронный ресурс]. Режим доступа: https://bugzilla.kernel.org/report.cgi?y_axis_field=component&cumulate=0&z_axis_field=&format=pie&x_axis_field=&no_redirect=1&query_format=report-graph&short_desc_type=allwordssubstr&short_desc=&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&longdesc_type=allwordssubstr&longdesc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&keywords_type=allwords&keywords=&cf_kernel_version_type=allwordssubstr&cf_kernel_version=&deadlinefrom=&deadlineto=&bug_id=&bug_id_type=anyexact&emailassigned_to1=1&emailtype1=substring&email1=&emailassigned_to2=1&emailreporter2=1&emailcc2=1&emailtype2=substring&email2=&emailtype3=substring&email3=&chfieldvalue=&chfieldfrom=&chfieldto=Now&j_top=AND&f1=noop&o1=noop&v1=&action=wrap.

2. Количество багов версий ядра [Электронный ресурс]. Режим доступа: https://bugzilla.kernel.org/report.cgi?bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&cumulate=0&y_axis_field=cf_kernel_version&width=1024&height=600&action=wrap&format=table.

3. Лав Роберт. Ядро Linux: описание процесса разработки, 3-е изд. — Москва: ООО «И.Д. Вильямс», 2013. с. 496.

4. О.И. Цилюрик. Модули Linux ядра: учебное пособие. — Казань: Казанский университет, 2011. С. 89 – 98.

5. Benvenuti C. Understanding Linux Network Internals. — Sebastopol: O'Reilly Media, Inc., 2005. p. 1064.

6. Хант Крэйг. TCP/IP Сетевое администрирование 3-е издание. — Санкт-Петербург–Москва: Издательство «Симво», Серия «Бест-селлеры O'Relly», 2008. С. 18 – 42.

7. Лора А. Чеппел Эд Титтел. TCP/IP: учебное пособие. — Санкт-Петербург: Издательство «БХВ-Петербург», 2003. С. 11 – 77.

8. Rosen R. Linux Kernel Networking Implementation and Theory. — Apress: ISBN., 2014. p. 648.

9. Kernel API [Электронный ресурс]. Режим доступа: https://www.kernel.org/doc/html/latest/.

10. Kernel Probes (Kprobes) [Электронный ресурс]. Режим доступа: https://www.kernel.org/doc/Documentation/kprobes.txt.

11. Probing the Guts of Kprobes [Электронный ресурс]. Режим доступа: https://landley.net/kdocs/ols/2006/ols2006v2-pages-109-124.pdf.