САН–ХУАН – Как это работает: Основы DNS
Понедельник, 12 марта 2018 года, 17:00 – 18:30 по AST
ICANN61 | Сан-Хуан, Пуэрто-Рико

КАТИ ПЕТЕРСЕН (CATHY PETERSEN): Здравствуйте, добро пожаловать на конференцию «Основы DNS – Как это работает». Сейчас выступит Мэтт Ларсон (Matt Larson), наш докладчик и вице-президент по исследовательской деятельности в офисе технического директора ICANN. Как видно, людей сегодня в зале немного, и если вы не против, переместитесь вперед и начнем. Мэтт?

МЭТТ ЛАРСОН: Добрый день! Добро пожаловать на конференцию «Основы DNS – Как это работает». Нас сегодня немного, так что если у вас есть вопросы, пожалуйста, поднимайте руки. У нас достаточно времени на рассмотрение материала, так что мы можем делать перерывы и отвечать на вопросы.

IP-адреса просты для машин и сложны для людей, и как раз по этой причине мы в первую очередь поговорим о DNS. Когда существовали только адреса IPv4, была определенная надежда на то, что люди смогут запомнить один или пару адресов. Но вот, для примера, адрес IPv6 с относительно небольшим количеством

символов, но он в большей степени зависит от количества нулей в адресе, и запоминать адреса v6 практически невозможно. Однако смысл здесь заключается в том, что люди должны использовать имена. Компьютеры и роутеры используют цифры, а людям нужны имена.

В ранние годы существования Интернета имена были простыми. У нас были так называемые «имена одной метки». Это были имена без точек, они не являлись доменными именами, так как доменные имена еще не были приняты. Любое имя на раннем этапе существования Интернета должно было умещаться в пространстве из 24 символов. Это относилось ко всем именам всех компьютеров в любом месте в Интернете, и мы называли их «имена хостов». Хост – это просто условное обозначение компьютера.

Преобразование этих имен в IP-адреса, чтобы люди могли использовать имена, а программы, компьютеры или роутеры и прочие устройства могли использовать цифры, называлось «разрешением имен». Для разрешения имен на раннем этапе существования Интернета, до появления DNS, использовался файл узла, который назывался «host.text». Вы не обязаны знать об этом – это представляет просто исторический интерес. Он выполнял те же функции, но немного отличался по формату от современного файла узла Esty,

если он знаком вам по системам UNIX или Lynx. В целом это просто файл, текстовый файл с именами и адресами, то есть именами узлов и соответствующими IP-адресами.

Их работа обеспечивалась централизованно организацией под названием Сетевой Информационный Центр (NIC), у которой был контракт с правительством США на выполнение определенных задач сетевого администрирования на раннем этапе существования Интернета. Одна из этих задач представляла собой поддержку работы файлов узлов.

Это было в те времена, когда Интернет был намного меньше распространен, он даже назывался не Интернетом, а сетью ARPA, объединяющей ресурсы нескольких ЭВМ, это был экспериментальный проект Министерства обороны США. В то время существовали десятки тысяч узлов Интернета и было целесообразна централизованная поддержка работы файла узла, а механизм обновления был слабо развитым в плане технологии и реализовался через электронную почту.

Когда администратор сети добавлял компьютер в сеть или удалял его из сети, изменял его имя или IP-адрес, он отправлял письмо по электронной почте в NIC, в котором писал: «Вот что я сделал, измените IP-адрес этого компьютера на такой-то». После этого NIC сохранял

оригинал файла «host.text» и выпускал новый один раз в неделю. Сетевые администраторы должны были загружать новый файл, если они решали, что он устарел или мог устареть. Замена происходила не быстро и вы могли какое-то время оставаться без обновления. Загрузка осуществлялась по протоколу FTP.

Все это представляло собой технологическое решение крайне низкого уровня. Очевидно, что с этим существовали проблемы, которые можно было предвидеть, немного подумав. Одна из этих проблем была связана с нехваткой имен. Если имеется всего 24 символа для задания имени компьютера, а сеть разрастается и в ней появляется больше устройств, то возникает нехватка имен и становится труднее избежать дубликатов, и что еще хуже, этот файл поддерживался реально очень простым способом – NIC редактировали его в текстовом редакторе, базы данных для таких файлов не было – были просто файлы и текстовый редактор. Эффективного метода предотвращения появления дубликатов не существовало и дубликаты появлялись повсеместно, создавая проблемы.

К другой очевидной проблеме относилась символизация. Ни у кого и никогда не было одной и той же версии файла, у вас она была всегда устаревшей. Другую проблему создавали трафик и нагрузка. В конце периода, когда использовался файл host.text, он начал

становиться таким большим, что для его загрузки требовалась очень широкая полоса пропускания Интернета. Это было время, когда скорость соединения 64 килобит в секунду считалась высокой. Это имело место незадолго до моего начала работы в Интернете, но мне говорили, что в конце эры использования файла host.text время его загрузки превышало периодичность обновления этого же файла.

Другими словами, почти никогда нельзя было иметь последнюю версию файла, потому что ко времени окончания загрузки файла уже появлялась новая версия для загрузки. Стало очевидным, что централизованная поддержка этого файла узла не соответствовала масштабам сети и надо было что-то с этим делать.

Осуждения на тему замены файла host.text начались в начале 10980-х годов и они преследовали две основные цели. Одна из них связана с увеличением масштаба сети, о чем говорилось чуть ранее, а другая – с упрощением рассылки по электронной почте, на чем я собираюсь остановиться подольше. Когда люди думают о DNS и мотивации к использованию DNS, они прежде всего вспоминают о проблемах масштаба сети, но они могут не вспомнить или вообще не знают о проблемах с рассылкой по электронной почте, что также являлось проблемой. В результате, что не удивительно, мы пришли к системе доменных имен.

Это мой слайд со сводной информацией по DNS. Если и можно свести во едино информацию по DNS, то она может быть представлена на этом слайде. По сути DNS представляет собой распределенную базу данных. Данные в этой базе обслуживаются локально, то есть у всех есть своя часть базы данных и они должны обслуживать свои данные, но эти данные доступны глобально, так как сама база данных распределена по всему миру, по всему Интернету. Вы храните свои собственные данные на локальном уровне, но у вас есть возможность просматривать чьи-то еще данные.

В DNS используется модель взаимодействия «клиент-сервер». Средства преобразования находятся на стороне клиента, и в отношении них необходимо учитывать один аспект – они отправляют запросы. Серверы имен находятся на стороне серверов, и в отношении них важно учитывать один аспект – они отвечают запросы на запросы. Преобразователи отправляют запросы, на которые отвечают серверы имен.

Есть определенные способы оптимизации – DNS использует кэширование для повышения эффективности работы, и что я подразумеваю под этим, раз уж мы говорим о распределенной по всему миру базе данных – такой быстротой отличается только скорость света, так что когда вы обращаетесь к этой базе данных, вам

может понадобится сделать несколько запросов, которые пересекут весь мир и вернутся обратно, а это занимает определенное время. Это очень удобно. На самом деле, очень важно помнить не только конечный результат обращения к базе данных, но и все промежуточные результаты, а также запоминать их и записывать в кэш, что ускорит этот процесс в следующий раз.

В DNS также используется репликация для обеспечения резервирования и распределения нагрузки. Говоря о том, что все хранят свои локальные копии собственных данных, но используется также и репликация, я имею в виду, что имеется не просто еще одна копия, есть множество копий их данных, и это обеспечивает резервирование. Другими словами, если бы у вас была всего одна копия ваших данных и с ней что-нибудь случилось бы, никто не смог бы просматривать то-либо в ней, но если у вас есть несколько копий, то обеспечивается резервирование. Это также способствует распределению нагрузки. Если большое количество людей просматривает базу данных и у вас есть несколько ее копий, то создаваемая запросами нагрузка может быть распределена среди нескольких копий.

Вот различные составные компоненты DNS, представленные на рисунке на одном слайде. Мы

поговорим о них более подробно в ходе конференции, но думаю, что стоит показать их все вместе и сразу, чтобы понимать, куда мы движемся. Давайте начнем с нижнего левого угла и пройдемся по всему слайду. Внизу слева у нас есть устройство, подключенное к Интернету, в данном случае это телефон, но в реальности любое устройство, подключенное к Интернету, для которого требуется преобразование имен в адреса, то есть любое устройство, использующее DNS, должно иметь простой DNS-клиент, называемый оконечным преобразователем.

Конечные преобразователи выполняют роль связующего звена между приложением, например, в данном случае у меня есть ярлык веб-браузера, и конечные преобразователи являются мостиком между приложением, таким как браузер, и остальной DNS. Оконечный преобразователь принимает запросы от приложений, например, на преобразование имени в адрес, и затем он преобразует его в DNS-запрос и отправляет так называемому рекурсивному преобразователю.

Оконечный преобразователь является очень простым, все что он умеет – это прием запроса от приложения, преобразование его в DNS-запрос,отправка этого запроса рекурсивному преобразователю и затем – ожидание ответа. В отличие от него, рекурсивный преобразователь является более сложным. Он умеет

связываться с разными так называемыми «полномочными серверами доменных имен», где хранятся и просматриваются данные в DNS.

Рекурсивному преобразователю может потребоваться связь с несколькими полномочными серверами для нахождения ответа. Он может связаться с одним из серверов и получить ответ: «Нет, у меня нет окончательного ответа на этот запрос, но я могу перенаправить его другому серверу имен». Затем он может связаться с этим другим сервером имен и тот ответит: «У меня нет ответа, но я могу перенаправить запрос еще куда-либо поближе к цели».

Так что рекурсивный преобразователь обладает достаточным интеллектом для навигации по этим полномочным серверам и нахождения ответа. Если посмотреть на блок рекурсивного преобразователя, то можно увидеть, что фактически он состоит из сервера имен и преобразователя. Если вы помните, серверы имен отвечают на запросы, и таким образом, компонент «сервер имен» рекурсивного преобразователя отвечает на запросы от конечного преобразователя, но что касается компонента «преобразователь», следует помнить, что преобразователи отправляют запросы, и этот компонент отправляет запросы полномочным серверам имен.

Я также показал здесь кэш и все что получает преобразователь, каждый ответ от полномочных серверов, который он размещает в своем кэше, чтобы использовать его для ответа на другие запросы в будущем. Это «экосистема» DNS, если позволите, на укрупненном уровне.

Я хотел бы дать определения некоторых важных связанных с DNS терминов и концепций. Первое из них – это то, что мы называем «пространство имен». Как я уже сказал, DNS – это распределенная база данных, а структура этой базы данных есть то, что мы называем пространством имен. Как вам может быть известно, когда я говорю о базе данных, вы можете подумать о реляционной базе данных, если вы с ней знакомы, а в структуре реляционной базы данных имеется множество таблиц, в которых находятся строки. В каждой строке есть столбцы и данные. Такова структура реляционной базы данных.

Данные DNS, пространство имен имеют совершенно иную структуру, которую мы называем обращенным деревом. У меня на слайде есть пример очень малой части пространства имен DNS. В обращенном дереве корень находится вверху, а ветви растут вниз.

Это дерево специалиста по информационным технологиям, и поэтому вас не должно удивлять, что

этот специалист представляет его вверх ногами, с корнем вверху и растущими вниз ветвями. Каждый узел этого дерева, как и каждый блок схемы на слайде, каждый узел, имеет имя, имеет метку. Корневой узел является особенным. Корневой узел находится в самом верху дерева, он фактически не имеет метки или его метка является нулевой, то есть она ничего из себя не представляет. Иногда вы видите нечто, представленное как у меня на слайде в виде точки в кавычках, которое показано для указания на то, что там ничего нет. Это пространство имен.

Мы часто обращаемся к этим узлам в пространстве имен, к их положению относительно корня. Вверху находится корень, а сразу под ним, если посмотреть на левую часть, находятся так называемые узлы верхнего уровня, они находятся непосредственно под корнем. На уровне ниже верхнего находятся узлы второго уровня и так далее, вниз по пространству имен с перемещением вниз по дереву. Иногда мы называем узлы обозначающими родство терминами, такими как родительский и дочерний элемент.

В данном примере корень является родительским по отношению к .com, а .com является родительским для примера, но пример является дочерним элементом .com, если использовать отношения «родитель - дочерний элемент». Каждая из этих меток имеет ограниченный

ряд символов, которые могут использоваться. Мы обозначаем его как заголовок (LTH) в отношении букв, цифр и дефиса, и это единственные символы, которые могут использоваться в именах меток DNS, а максимальная длина метки составляет 63 символа. Еще один важный аспект, который необходимо знать – имена меток не чувствительны к регистру. Вы можете смешивать символы верхнего и нижнего регистра и они будут равнозначными.

Каждый из этих узлов имеет доменное имя, а доменное имя предназначено для указания на место узла в пространстве имен. Определение доменного имени является очень простым – начните с узла и вы доберетесь до корня, пишите имена и ставьте точки между ними. Вы можете видеть выделенное примечание здесь внизу – мы пишем www и ставим точку, затем мы переходим к родительскому элементу, это наш пример, мы берем имя примера и ставим точку и затем com с точкой, и мы у корня.

Особый тип доменного имени мы называем «полностью определенное доменное имя», которое не связано ни с каким другим доменом. Полностью определенное доменное имя точно указывает на местоположение узла в пространстве имен, и это FQDN (полностью определенное доменное имя) заканчивается точкой, а точка фактически является разделителем между

доменными именами верхнего уровня, в данном случае – .com и корнем. Вы берете доменное имя верхнего уровня, ставите точку и метку корня, но корень не имеет метки, метка корня является нулевой, а это означает, что доменное имя оканчивается точкой.

Если все это кажется вам знакомым, вы также можете быть знакомы с файловой системой Unix или, кстати, с файловой системой Windows, которые представляют собой пример структуры данных, которую также можно представить в виде перевернутого дерева. В файловой системе узлы представляют собой файлы или директории, а вместо доменного имени есть путь доступа. Пути к файлам показывают, где находится файл или директория в системе, практически как и доменное имя, которое показывает местоположение узла в пространстве имен.

Здесь у нас появился еще один важный термин – домен. Определение домена очень простое – это просто узел в пространстве имен и все, что находится ниже него. Например, я выделил домен .com. Так, .com представляет собой узел .com и все, что находится ниже. Я показываю здесь три имени .com, а фактически их 131 миллион. Очевидно, что часть их на слайде не показана. Домен .com огромен, это все что находится под .com, любое доменное имя, оканчивающееся на .com в домене .com.

Я хочу сравнить это с термином «зона», и это действительно очень важный термин, так как вы часто слышите его и важно понимать, что это такое. Помните, что мы рассматриваем это потому, и DNS у нас есть потому, что нам требуется распределенное администрирование, централизованное управление данными по именам узлов, и IP-адреса не решили эту проблему, и потому возникла идея о распределении, чтобы все управляли своими данными по своим именам узлов и адресам.

Поэтому пространство имен разделено, чтобы обеспечивалось распределенное администрирование, а эти административные подразделения и называются зонами. У всех есть своя собственная зона. Это как маленькая песочница, в которой можно играть. Все могут делать любые изменения в своей зоне и никому не мешают, они отвечают за свою зону и обслуживают ее.

Зоны создаются путем делегирования. Делегирование выполняется с более высокого уровня в пространстве имен на более низкий уровень этого пространства. Делегирующая зона называется родительской, а создаваемая зона – дочерней, и этот процесс начинается от корня и продвигается вниз.

Позвольте привести пример. Здесь мы снова видим пространство имен и если посмотреть на него с этой

точки зрения, то у нас недостаточно информации для определения границ зоны. Если посмотреть на пространство имен здесь, то мы не знаем, как оно разделено для целей администрирования.

Рассматривая приведенный мной пример, представьте себе, что вы смотрите со спутника или космической станции и видите, что находится в Северной Америке. Рассматривая Северную Америку, вы не можете определить, что в реальности там находятся три страны – Канада, США и Мексика – вы этого не знаете.

Вы можете рассматривать Северную Америку весь день и все равно не определите ее административных границ или политических границ, потому что они не показаны и для этого вам нужна дополнительная информация. Также дело обстоит и с зонами. Вы можете также смотреть на пространство имен, но пока вы не знаете, где происходит делегирование, вы не можете определить границы зоны.

Давайте я нарисую некоторые границы зоны. Я знаю, где существует делегирование в пространстве имен, поэтому я могу обозначить границы зоны. В самом верху пространства имен находится корневая зона, а от корневой зоны выполняется делегирование к зонам верхнего уровня. Представим это таким образом, что в зоне имеется информация о делегировании, которая

создает или указывает на делегируемую зону и этот процесс делегирования продвигается вниз по пространству имен.

В данном примере корневая зона делегирует администрирование зоне .com, которая далее в данном примере делегирует администрирование в зоны пример.com, bar.com и foo.com. Рассматривая относительные размеры этих зон, давайте начнем с корневой зоны. Здесь, когда я смотрел сюда последний раз, было 1543 доменных зон верхнего уровня, и корневая зона относительно мала, в ней должна содержаться только информация по делегированию 1543 зон под ней.

Давайте спустимся ниже к .com, с другой стороны .com, если вы просмотрели материалы, предоставленные Verisign во время регистрации, в их последнем отчете говорится, что в зоне .com имеется около 131 миллиона имен. Эта зона com огромна, она самая большая из всех и должна содержать информацию о делегировании по 131 миллиону зон .com – зон второго уровня ниже нее. Делегирование может распространяться ниже второго уровня, на этом слайде у меня нет примера, но оно определенно может распространяться так. Делегирование может распространяться в пространстве имен на неограниченную глубину.

Если вы помните, минуту назад я сказал, что в DNS используется репликация для обеспечения резервирования и повышения эффективности работы – вот об этой репликации мы и говорим сейчас. Как вы помните, серверы имен отвечают на запросы, и можно сказать, что сервер имен является полномочным в отношении зоны, если на нем имеется вся информация по этой зоне.

Идея заключается в том, что полномочный сервер имен знает, что находится в зоне и может дать определенный ответ на запрос об этом. Он может ответить так: «Да, вы спрашивали о чем-то в этой зоне – вот информация». Или он может выдать следующее: «Вы спрашивали о чем-то, но это имя не существует, оно не находится в данной зоне, поэтому я могу сказать, что оно не существует».

Зоны должны иметь несколько полномочных серверов, и как я уже сказал, для этого используется репликация, которая обеспечивает резервирование и распределение нагрузки. Для каждой зоны должен быть по меньшей мере один полномочный сервер имен, а в реальности их должно быть несколько, оптимальный подход предполагает наличие как минимум двух полномочных серверов, так как если у вас есть только один такой сервер и с ним что-нибудь случится, то никто не сможет посылать запросы в вашу зону.

При наличии нескольких полномочных серверов вы должны обеспечить синхронизацию данных между ними. Идея заключается в том, что информация о зоне на всех полномочных серверах должна быть одинаковой. Как вам это сделать? Хорошая новость заключается в том, что протокол DNS имеет встроенные средства для этого.

Синхронизация данных по зоне между полномочными серверами является встроенной функцией и существует процесс, называемый «передача зоны», который позволяет перемещать данные по зоне в другие места. Вы назначаете один полномочный сервер в качестве основного и там вы вносите изменения по зоне, и у вас есть другие полномочные серверы, которые являются второстепенными или ведомыми – термины равнозначны, и эти полномочные серверы выгружают данные по зоне с основного сервера, то они связываются с ним и выполняют определенное действие, которое называется передачей зоны, и происходит копирование зоны с основного на второстепенный сервер.

Важно отметить, что все серверы зоны, все полномочные серверы, являются одинаковыми и в них хранятся одинаковые данные. Единственная разница между основным и второстепенным серверами заключается в том, откуда они получают данные. Основной сервер выгружает данные по зоне со своего

диска, а второстепенный сервер выгружает их с основного сервера, но данные на основном и второстепенном серверах одни и те же.

Конечно, какое-то короткое время они могут быть не синхронизированы, так как когда вы вносите изменения на основном сервере, на нем хранится более актуальная информация, но затем она распространяется на второстепенные серверы и данные становятся синхронизированными. Замечательно, что эта функция встроена в DNS и вам самим не нужно беспокоиться о синхронизации серверов имен – она происходит сама по себе.

Теперь я хочу перейти на уровень ниже. Мы говорили о зонах, теперь давайте заглянем внутрь зоны и поговорим о находящихся в ней данных. Помните, что каждый узел в пространстве имен, каждый блок на показанных мной схемах, каждый из них имеет доменное имя, которое соответствует пути к нему, и следует понимать, что заданное доменное имя может содержать информацию различных видов и может быть связано с данными разных типов.

Наиболее распространенный вид информации – это IP-адрес. Мы можем связать IP-адрес с доменным именем. Такой тип информации связан с доменом и мы называем ее ресурсными записями. Ресурсные записи являются

содержащимися в DNS данными и существуют разные виды ресурсных записей для хранения данных различных типов. Наиболее распространенные виды ресурсных данных используются для хранения IP-адресов. В этом отношении версия IP и версия IP 6 являются разными типами записей, но они являются ресурсными записями для хранения данных другого типа.

Зона состоит просто из набора ресурсных записей и все записи зоны сведены в файл, который мы называем файлом зоны. Для каждой зоны имеется файл зоны и записи по нескольким зонам никогда не объединяются в один файл. Зона – это всего лишь набор ее ресурсных записей.

Позвольте мне показать вам, как выглядят эти ресурсные записи. В принципе, существует способ составления записей, стандартный способ записи в виде текста. Ресурсные записи имеют пять полей – нам нет необходимости рассматривать их все. Здесь важно понять, что это ресурсные записи разных типов и они содержат данные соответствующих типов. Существует несколько наиболее распространенных типов ресурсных записей.

Я уже говорил, что имеется тип записи для адресов IPv4, который называется «запись или адресная запись», и

есть тип записей, используемых для хранения адресов IPv6, который мы называем «ресурсная запись AAAA» для четырех «A». Кроме этого есть несколько других типов записей, о которых я расскажу вкратце. В данный перечень входят наиболее распространенные типы, но в реальности существует много других типов ресурсных записей.

Когда я последний раз заглядывал туда, там было 84 различных типов записей и реестр IANA, который назывался... у меня было это на слайде, не буду читать это вам, и есть URL для этого. Если вы зайдете на этот сайт, то вот как оно выглядит. Основываясь на размере поля ресурсной записи, мы можем иметь до 65 000 записей, но сейчас мы очень далеки от этого и имеем всего 84.

Если вы задумались о хранении чего-то нового в DNS, то вы можете зайти в IFT, создать проект интернет-документа и убедить людей в том, что ваша идея должна воплотиться в новый тип DNS, а вы можете создать его и разместить в регистратуре, а затем у вас появятся новые вещи, которые вы сможете разместить в DNS. Люди все время делают так. Не так часто – у нас всего 84 записи, но люди реально думают о новых вещах, которые они хотят разместить в DNS и создают новые типы записей для хранения новых типов данных. Тем не

менее очевидно, что наиболее распространенный вид данных – это IP-адреса.

Наиболее частый способ использования DNS – это преобразование доменных имен в адреса, и здесь у нас снова появляются эти две записи типа адресов, которые я показывал. Вот пример реального представления адресной записи в виде текста и записи типа AAAA. Слева у нас доменное имя, например, .com, затем тип, представленный первой буквой A для адреса, а затем – фактический адрес.

Это пример ресурсной записи в файле зоны, например, зоны .com, и в этой ресурсной записи просто указано, что наш пример.com имеет следующий IP-адрес. Под ним находится запись из четырех A, в которой указано, что пример.com имеет такой-то адрес IPv6. Большая часть DNS состоит из записей в формате A и четырех A по той причине, как я постоянно говорю, что основной задачей DNS является преобразование доменных имен в IP-адреса.

Существуют и другие типы, и что мне кажется интересным, большая часть этих типов используется людьми, которые используют информацию DNS, которые просматривают информацию в DNS, потому что им нужно сделать что-то, например, подключиться к веб-

серверу, и поэтому их веб-браузеру нужно найти имя для преобразования адреса.

При этом некоторые типы записей используются самой DNS, и основными примерами этого являются запись NS и запись SOA, которые мы очень кратко рассмотрим, и это такие типы, которые не зависят ни от чего, кроме DNS. Чем они интересны – это типы записей, которые не отличаются от других типов, и я хотел бы представить их в виде склада, если можно сравнить их со складом.

Например, вы арендуете склад и вам нужно разместить в нем множество вещей. Вы же не подъезжаете к нему на грузовике и просто начинаете закидывать в него коробки – чтобы пользоваться складом, вам нужно сделать какие-то полки, и после этого взять коробки, товары в коробках, и разместить их на полках.

Склад не пригоден для использования без полок, и DNS имеет сходство с ним в том плане, что записи NS и записи SOA представляют собой полки, и они должны существовать для того, чтобы DNS работала, но за пределами DNS этим практически никто не интересуется, всем интересны другие типы записей, такие как A и AAAA.

Позвольте мне рассказать вам о записях NS. Они используются так же, как и полномочные серверы имен для зоны. На данном примере показаны две записи NS и

что они обозначают в пример.com – у этой зоны есть два полномочных сервера имен, один из них называется NS1.EXAMPLE.COM, а другой – NS2.EXAMPLE.COM. Левая сторона – это имя зоны, а правая – имя сервера имен.

Теперь записи NS становятся немного сложнее, так как в реальности они появляются в двух местах. Они появляются в родительской зоне и в дочерней зоне. В этом блоке у меня находится фактический перечень записей NS для зоны .com, для этой зоны есть 13 полномочных серверов имен и они называются, как вы видите, посмотрите на правую часть – с A.gTLD-SERVERS.NET по M.gTLD-SERVERS.NET, это перечень записей NS и эти 13 записей появляются в двух местах.

Давайте я расскажу об этом немного подробнее. Они появляются в корневой зоне, и в данном случае корневая зона является родительской, и эти записи, эти записи NS в родительской зоне фактически выполняют делегирование, что сообщает нам, что остальная часть DNS ниже корневой зоны относится к зоне .com, а затем перечень записей NS также появляется снова в зоне, которая в данном случае имеет имя .com. Записи NS зоны .com появляются в корне, в родительской зоне и в самой зоне .com.

Говоря о том, как вы ищете данные в DNS, можно увидеть, как важно, чтобы эти записи NS появлялись в родительской зоне, потому что... Я забегу вперед и заранее скажу, что разрешение имен работает в DNS таким образом, что когда вы ищете что-то в DNS, вы начинаете с корня, а затем вы продвигаетесь по этим указателям делегирования, по записям NS. Если вы ищете что-то ниже зоны .com, вы начинаете с корня и в корневой зоне вы увидите делегирование на .com, а затем вы можете перейти к имени сервера .com и увидите делегирование под ним, и так далее, но об этом чуть позже.

Записи NS включают в себя всего лишь имена, и если вы посмотрите на представленный мной пример, пример.com, то увидите, что один из серверов имен представлен NS1.EXAMPLE.COM, но одного только этого недостаточно, потому что затем вам потребуется IP-адрес NS1.EXAMPLE.COM, если вы действительно хотите с ним связаться. В некоторых случаях информация по делегированию также должна включать в себя записи адреса, которые мы называем "связующие записи". Если вы когда-нибудь слышали что-то о связующих записях, то это и есть запись адреса для сервера имен.

В каждой зоне существует и другая запись, которая называется «запись SOA», но я не хочу говорить о ней

подробно. Я всего лишь хотел упомянуть, что такая запись существует. Есть одна из таких записей SOA для каждой зоны и она находится в верхней части или в том, что мы называем «вершиной» зоны. Большинство значений здесь связаны с переносом зоны, о котором я говорил ранее. Они указывают полномочным серверам, как выполнять синхронизацию и как часто выполнять синхронизацию зоны.

Давайте вернемся ко второму назначению DNS. Вторая проблема, которую была призвана решить DNS, связана с рассылкой по электронной почте. DNS должна решить проблему доставки почты на адрес электронной почты. В прежние времена, до существования DNS, у нас были адреса электронной почты в виде «пользователь @ имя узла», который как раз являлся одним из тех имен, которые должны были иметь максимум 24 символа.

Таким был ваш адрес электронной почты, и до появления DNS ваша электронная почта должна была направляться на узел, имя которого указывалось в правой стороне адреса электронной почты. Не было способов указывать адрес электронной почты в виде, например, MATT@FOO, в действительности приходилось отправлять сообщение машине, которая называлась Bar, в другое место. Нет, если ваш адрес электронной почты был MATT@FOO, то ваша

электронная почта отправлялась машине, которая называлась FOO, и там вы должны были читать ее.

Одной из задач DNS было разъединение этой цепи, чтобы можно было сказать: «Вот мой адрес электронной почты и вот сюда должна попадать моя электронная почта, вам не нужно отправлять ее куда-то еще». DNS обеспечивает гибкость в этом отношении. Существует запись, которая называется «запись обмена электронной почтой», которая указывает, куда должна направляться электронная почта в домене, и вот пример этого. Эти записи MX в примере с «.com доменное имя» указывают, куда должна направляться электронная почта.

Для любого адреса «пользователь @ пример .com», эти записи MX указывают на то, что почта должна отправляться машине с именем MAIL.EXAMPLE.COM. Там есть индекс предпочтения, это число 10 и число 20, и это немного противоречит логике, так как чем меньше этот индекс, тем более предпочтительным является почтовый сервер. Эти две записи MX относятся к любой электронной почте, адресованной кому-либо на «пример.com», вы должны отправить ее на MAIL.EXAMPLE.COM, но если по какой-то причине вы этого сделать не можете, то вы должны попробовать отправку на адрес MAIL-BACKUP.EXAMPLE.COM.

Любой почтовый сервер, который должен доставить электронную почту, должен быть способен находить записи MX для доменных имен. Между DNS и электронной почтой на базе протокола SMTP существует очень тесная связь. Когда почтовый сервер принимает сообщение, которое он должен доставить, он ищет записи MX для адреса электронной почты, и таким образом он узнает, куда следует отправить сообщение.

До этого момента мы говорили о преобразовании имен в IP-адреса, что является важной задачей. Иногда вам необходимо преобразование в обратном направлении. Преобразование имени в IP-адрес мы называем обратным преобразованием, но что если потребоваться сделать что-то исходя из IP-адреса, если имя известно? Иногда возникают случаи, когда вам это необходимо.

Представьте, например, что существует инструмент для устранения неполадок в сети с названием Trace Route, который позволяет показывать все роутеры между вами и IP-адресом, на который вы хотите перейти. Когда вы видите IP-адрес каждого роутера, вам может быть интересен сам IP-адрес, но вам также может быть интересно, где он находится. Кто его использует? Какое у него имя? Это пример обратного преобразования, при котором берется IP-адрес и находится соответствующее имя.

Вспомним о том, что у нас была таблица хостов. Если у вас есть просто файл со списком имен и IP-адресов, и вы хотите выполнить прямое преобразование, то это очень просто – у вас есть имя, и вы хотите преобразовать его в IP-адрес, вы просматриваете этот файл до тех пор, пока не найдете имя и соответствующий IP-адрес, вот и все. Но что если вам требуется выполнить обратное преобразование? Это так же просто. Вы просматриваете файл, находите IP-адрес и соответствующее ему имя. Таблица хостов работает прекрасно. Что вы делаете с DNS?

Давайте вернемся к примеру пространства имен. Структура пространства имен такова, что она позволяет легко находить доменные имена, но в этом случае поиск IP-адресов невозможен. Если мне нужно найти доменное имя, например WWW.EXAMPLE.COM, то я начинаю от корня и иду вниз, пока не доберусь до WWW. Но что мне делать, если я начинаю с IP-адреса? Ответ заключается в том, что вы не можете сделать это в DNS так, как я показал вам – вы не сможете найти IP-адрес.

Что должно было произойти: у нас должен был быть способ преобразования IP-адресов в доменные имена, чтобы можно было находить их как доменные имена, что в реальности мы и имеем. Существует еще один тип записи – запись PTR для указателя, и тут IP-адреса преобразуются в особые доменные имена, в которые

заносятся записи PTR и это позволяет вам искать IP-адреса как доменные имена и затем находить соответствующие им имена. Адреса IPv4 находятся под доменным именем IN-ADDER для интернет-адреса «.arpa», а адреса IPv6 – под доменным именем IP6.ARPA.

Позвольте мне привести пример этого. Это дерево пространства имен. Я просто показываю вам его новую часть, о существовании которой вы могли не подозревать. Справа у нас здесь EXAMPLE.COM, о котором мы знаем, но здесь у нас есть домен INADDER.ARPA, и в этом случае, если посмотреть внизу, то можно увидеть, где находится запись PTR для зоны EXAMPLE.COM. Вы видите, что справа у нас находится запись адреса, в которой указано, что IP-адрес EXAMPLE.COM – это 192.0.2.7, и если вы хотите найти EXAMPLE.COM, то вы видите запись адреса на EXAMPLE.COM и определяете по ней адрес.

Как я узнаю имя, если доменное имя соответствует IP-адресу 192.0.2.7? Что вам нужно сделать – преобразуйте этот IP-адрес в доменное имя, и для этого вы берете IP-адрес, преобразовываете его и добавляете к нему IN-ADDER.ARPA, а затем находите запись PTR. Это работает только потому, что все понимают правила. Все понимают то, что я вам описываю. Если вы имеете выделенный для вас диапазон IP-адресов, то

региональные интернет-регистратуры оказывают содействие в управлении доменом IN-ADDER.ARPA и вы можете получить зону, соответствующую выделенным вам IP-адресам.

Например, если вам выделено 192.0.2/24, то все, что начинается с 192.0.2 будет иметь вид 2.0.1. Домен 192.INADDER.ARPA выделен вам и вы должны будете разместить там записи PTR, если вы хотите, чтобы люди могли выполнять обратное преобразование ваших IP-адресов. Это работает. Это несколько неудобно, но работает.

Думаю, что большинство людей согласились бы с тем, что обратное преобразование на самом деле не так важно, как прямое преобразование. Прямое преобразование позволяет вам вводить доменное имя в браузере и переходить на сайт. Обратное преобразование больше используется для диагностики и устранения неполадок. Скорее всего, обычных людей, если они не являются сетевыми инженерами или системными администраторами, обратное преобразование не интересует.

Как я уже сказал, существует много ресурсных записей других типов, вот пример других записей, просто чтобы имели понятие о других видах данных, которые люди

могут разместить в DNS. Вот пример того, как может выглядеть файл зоны для очень маленькой зоны.

Вот файл зоны для нашей гипотетической зоны EXAMPLE.COM и я знаю, что не рассказал подробно обо всех этих записях, но это небольшая зона, похожая на большинство зон в Интернете, потому что, если подумать, в отношении большинства доменных имен в Интернете вам скорее всего захотите сделать две вещи. Вы хотите иметь веб-сервер, вы хотите иметь сайт и вы хотите принимать электронную почту.

Но сейчас, что очевидно, существуют домены, в которых намного больше вещей, чем это, любые виды имен, большое число доменных имен, как мое личное доменное имя, например, которое я использую для электронной почты – мне нужна электронная почта и небольшой сайт. Моя личная зона для моего доменного имени выглядит приблизительно так же, и это все, что вам нужно для поддержки этих приложений.

У нас есть IP-адрес для EXAMPLE.COM, по которому будет находиться наш веб-сервер. Вы можете просмотреть эту зону и догадаться, что 192.0.2.7 – это IP-адрес сайта EXAMPLE.COM, потому что у нас есть запись, которая преобразует EXAMPLE.COM в IP, и затем вы увидите некоторые записи MS – они указывают, куда отправлять почту для EXAMPLE.COM.

Сейчас я бы поговорить о процессе разрешения имен. Вот как мы ищем что-то в DNS. Компоненты DNS, которые я показал в начале этого заседания на рисунке, конечный преобразователь, рекурсивные преобразователи и полномочные серверы имен, все они взаимодействуют между собой для поиска данных в пространстве имен.

Важно знать, что запрос DNS всегда имеет три параметра – доменное имя, такое как WWW.EXAMPLE.COM, тип данных которые вы ищете, в данном случае «A» для адреса, а также есть значение, которое называют классом – я пропустил его и не говорил о нем. Класс это то, что люди раньше собирались использовать для расширения DNS на сети других типов, в реальности это не использовалось, но это оно встроилось в DNS и мы имеем его там.

В данном случае класс всегда будет чем-то, что называют «классом Интернета», но вам не нужно задумываться об этом. В данном случае важны только доменное имя и тип. Если вы собираетесь отправить запрос DNS, если вы собираетесь задать серверу имен вопрос, то вы всегда должны указывать доменное имя и тип.

Существует два типа таких запросов, а также конечные преобразователи – помните – конечный

преобразователь есть в таких вещах, как ваш телефон, ваш тостер, ваш холодильник, ноутбук и все что угодно, что соединяется с Интернетом и должно преобразовывать имена в адреса или другие виды данных, все эти устройства имеют конечный преобразователь. Конечные преобразователи отправляют так называемые рекурсивные запросы, а рекурсивный запрос служит сигналом для рекурсивного преобразователя, который сообщает ему: «Эй, я конечный преобразователь и мне всего лишь нужно, чтобы ты выдал мне ответ или ошибку. Я не смогу принять от тебя ничего больше. Я не могу принять частичный ответ, мне нужен полный ответ на мой вопрос».

С другой стороны, рекурсивные преобразователи умнее и они могут принимать эти частичные ответы, которые являются ссылками. Они отправляют тип запроса, который указывает, как в данном случае, что они могут принимать ссылку в качестве ответа. Как я уже сказал, если вы собираетесь искать что-то в DNS, вы начинаете с корневой зоны и двигаетесь вниз по указателям делегирования.

Есть полномочные серверы, которые имеют полномочия в отношении корневой зоны, они имеют всю информацию в корневой зоне – это и означает «полномочный», и мы называем их корневыми

серверами имен. Если вы собираетесь начать разрешение имен в корневой зоне, то у вас должна быть возможность связываться с корневым сервером имен, а как вы определите, что является корневыми серверами имен? Ответ заключается в том, что должна быть настроена их конфигурация, нет способа их обнаружения – должна быть настроена их конфигурация на каждом рекурсивном сервере имен. Это отличается от остальных параметров сети.

Когда мой телефон подключился к сети WIFI, к сети WIFI ICANN в конференц-центре, он использовал протокол, который называется «протокол динамического выбора конфигурации хоста». DHCP и мой телефон выдали запрос сети: «Эй, я новое устройство в сети, мне нужен IP-адрес». И сеть ответила – мой телефон ничего не знал об этой сети – сеть ответила: «Хорошо, вот твой IP-адрес и вот еще некоторые другие параметры конфигурации, которые нужны тебе, включая IP-адрес рекурсивного сервера имен, который нужно использовать».

Это пример, когда устройство может не знать ничего, а сеть сообщает ему все необходимое. С рекурсивным сервером имен дело обстоит иначе – вы не можете просто включить его без настройки конфигурации, он должен знать корневые серверы имен и их IP-адреса. Существует специальный файл, который нужен всем

рекурсивным серверам имен. Он называется «файл корневых подсказок» и содержит имена и IP-адреса корневых серверов имен.

Хорошо, когда вы устанавливаете рекурсивный сервер имен, например, на машине с системой Linux, и кто-то уже поработал с Linux-пакетом, включил в него ПО для рекурсивного сервера имен, файл корневых подсказок и некоторые рекурсивные преобразователи, и даже имена и IP-адреса корневых серверов в составе самого исходного кода ПО. Но вы можете найти файл корневых подсказок в URL и вот как он выглядит.

Есть 13 корневых серверов имен. Есть 13 серверов, полномочных в отношении корневой зоны. Точка слева означает корневую зону, а затем у нас есть 13 записей NS, которые являются именами серверов имен для корневой зоны, и как вы видите, они имеют имена с A.ROOT-SERVERS.NET по M.ROOT.SERVERS.NET, а ниже вы видите их адреса IPv4 и адреса IPv6. Каждый корневой сервер имен имеет адрес IPv4 и адрес IPv6. Рекурсивный сервер имен для разрешения имен должен иметь эту информацию, должен знать имена и IP-адреса корневых серверов имен.

Давайте немного отвлечемся и поговорим о корневой зоне и о том, как в нее попадает информация. Помните, что находится в корневой зоне? У нас есть информация

о зонах доменов верхнего уровня. У нас есть записи NS для доменов верхнего уровня и администрирование корневой зоны в известном смысле является сложным. Есть две организации, которые работают вместе, ICANN выполняет роль, которая называется «Исполнитель функций IANA» и подразделение ICANN PTI осуществляет эту роль, а другой организацией является Verisign, роль которого называется «обслуживание корневой зоны».

Эта схема довольно старая, она возникла в начале 1990-х годов, когда Verisign называлась Network Solutions, которую компания Verisign купила в 2000 году. До создания ICANN роль исполнителя функций IANA выполнял Университет Южной Калифорнии. Это очень старая исторически сложившаяся схема, и она в определенном смысле сложная, но такова уж корневая зона.

Эти две организации, ICANN и Verisign, сотрудничали в сфере размещения данных в корневой зоне, создания файла корневой зоны, и кроме того, нам нужны полномочные серверы в корневой зоне, корневые серверы имен. Есть 12 организаций, которые осуществляют управление полномочными серверами имен для корневой зоны. Это несколько необычно, так как для большинства зон есть одна организация, которая управляет всеми полномочными серверами.

Возьмем, к примеру, .com. Я работал в компании Verisign, поэтому знаю немного о том, как это работает, компания Verisign управляет всеми полномочными серверами для .com. Возьмем другую зону – что делает несколько компаний – они могут управлять некоторыми из полномочных серверов сами или привлекать стороннего провайдера, а может быть, нескольких сторонних провайдеров для обеспечения резервирования, но не 12 организаций – это необычно.

Есть 13 имен корневых серверов имен. Хотя имя выглядит как A.ROOT-SERVERS.NET, для краткости люди используют имена с AROOT по MROOT, и это организации,которые управляют серверами с A по M. Это интересная группа организаций, и они не имеют между собой ничего общего, кроме того, что они управляют корневым сервером имен. Если посмотреть на их список, то мы увидим коммерческие организации, образовательные учреждения, общественные организации, интернет-провайдеров, департаменты правительства США, всего понемногу, и так сложилось давно, лет 20 назад.

Данный список операторов был статичным, он оставался таким же в течение 20 лет и с этим связано множество запутанных историй, на которые у нас нет времени. У нас есть 13 корневых серверов и 12 организации. Их 12, потому то компания Verisign управляет двумя – AROOT и

JROOT. Это корневая зона, корневые серверы имен и операторы корневых серверов.

Если вы хотите узнать подробнее об операторах корневых серверов и корневых серверах, то вы можете посетить сайт ROOT-SERVERS.ORG и там вы узнаете, где находятся все корневые серверы и получите немного информации по ним.

Позвольте мне дать вам представление на реально высоком уровне о том, как работает процесс изменения для корневой зоны. Вся информация корневой зоны связана с доменами верхнего уровня. Если операторы доменов верхнего уровня хотят внести изменение, например, добавить полномочный сервер для своих доменов верхнего уровня, удалить полномочные серверы для своих доменов верхнего уровня или изменить имя или IP-адрес одного из своих серверов, они должны представить это изменение исполнителю функций IANA, которым является компания PTI, подразделение ICANN, и PTI выполняет ряд проверок и обновляет имеющуюся у них базу данных корневой зоны, а затем направляет запрос управляющему корневой зоны, которым является компания Verisign.

Компания Verisign выполняет дополнительные проверки, обновляет свою базу данных корневой зоны, создает

файл корневой зоны, открывает доступ к нему и затем 13 корневых серверов загружают этот файл и открывают доступ к нему. Это версия процесса очень, очень высокого уровня. Я просто представляю это, чтобы показать как различные организации взаимодействуют для обеспечения работы всего этого, но процесс намного сложнее, чем я тут показываю. Это было небольшое отступление для ознакомления с корневой зоной.

А теперь давайте поговорим о том, как работает разрешение имен. Допустим, в левом нижнем углу у нас есть телефон и кто-то открыл веб-браузер на телефоне и набрал там WWW.EXAMPLE.COM. Веб-браузер направляет вызов конечному преобразователю, а конечный преобразователь представляет собой очень простой код. Оранжевым цветом у нас тут показано, что это фактически API-вызов, одна часть программы веб-браузера вызывает конечный преобразователь, другую программу или часть программы, а иногда это просто функция вызова, и говорит: «Мне нужен адрес WWW.EXAMPLE.COM».

Как вы помните, конечный преобразователь очень простой, все что он знает – это IP-адрес рекурсивного сервера, или, может быть, нескольких рекурсивных серверов, которым он посылает запрос. Конечный преобразователь преобразует его в запрос DNS, запрос

от веб-браузера к рекурсивному серверу имен с заданной конфигурацией, что в данном случае представлено 4.2.2.2, и это фактически реальный IP-адрес рекурсивного сервера. Используемый интернет-провайдером сервер имен называется сервером 3 уровня и такие рекурсивные серверы называют открытыми, так как кто угодно может послать запрос такому рекурсивному серверу, и если так можно сказать, это просто хороший IP-адрес, который можно использовать для примера.

Конечный преобразователь выдает запрос на преобразование рекурсивному преобразователю, запрашивает у него адрес WWW.EXAMPLE.COM, чтобы этот пример стал еще интереснее, я говорил – помните, что рекурсивные преобразователи имеют кэш, но чтобы это было интересно, мы предположим, что данный рекурсивный преобразователь только что включили и в его кэше ничего нет, все что он знает – это имена и IP-адреса корневых серверов. Рекурсивный сервер выбирает один из корневых серверов, например LROOT, и задает ему тот же вопрос, который он только что получил от конечного преобразователя: каков IP-адрес WWW.EXAMPLE.COM?

На данном этапе корневой сервер не может ответить на запрос напрямую, корневой сервер не знает IP-адрес WWW. EXAMPLE.COM, он не знает ничего о

EXAMPLE.COM, но он знает что-то о .com, потому что в корневой зоне есть данные по делегированию в .com. Корневой сервер может вернуть то, что мы называем ссылкой на .com, он может ответить так: «Вот серверы имен и их IP-адреса для .com».

Теперь рекурсивный сервер кэширует этот ответ, чтобы он мог использовать эту информацию в будущем, и затем он выполняет то, что мы называем «переход по ссылке», он выбирает один из серверов .com и посылает ему такой же запрос. Реальный сервер .com имеет название C.gTLD-SERVERS.NET, компания Verisign решила называть серверы .com также, как и корневые серверы.

Есть ROOT-SERVERS.NET с A по M, и так же есть серверы gTLD-SERVERS.NET с A по M, но это имена серверов .com. Наш рекурсивный преобразователь выбрал один из серверов .com, C.gTLD-SERVERS.NET, и выдает ему тот же запрос, который он получил от конечного преобразователя и который был отправлен корневому серверу: «Каков IP-адрес WWW.EXAMPLE.COM?»

В этот момент сервер .com также не знает IP-адрес WWW.EXAMPLE.COM, но он знает полномочные серверы имен, например, .com, и он отправляет в ответ

этот список, он отправляет в ответ ссылку, например, на .com.

Теперь наш рекурсивный сервер кэширует это и переходит по ссылке, отправляет такой же запрос в третий раз одному из полномочных серверов, например, .com, который представлен NS1.EXAMPLE.COM.

Теперь NS1.EXAMPLE.COM является полномочным по отношению к .com, к примеру, он знает IP-адрес WWW.EXAMPLE.COM и может вернуть его рекурсивному серверу, который кэширует его, отправляет обратно конечному преобразователю, который направляет его обратно приложению, и теперь приложение знает IP-адрес сервера и может связаться с ним, загрузить страницу в Интернете и продолжить работу с ней. Это процесс решения имен в самом простом варианте. Как и многое из того, что связано с DNS, он может быть сложнее, чем этот упрощенный вариант. Здесь важно отметить, что вы начинаете с корня и продвигаетесь вниз.

Вы не всегда должны начинать с корня, так как существует кэширование. Я выделял все моменты кэширования по мере описания процесса. Что произойдет, если кто-то зайдет сюда и сразу же после этого захочет перейти на FTP.EXAMPLE.COM?

Конечный преобразователь компилирует запрос DNS и отправляет его рекурсивному серверу, как и до этого, но теперь он получает все, что хранится в кэше рекурсивного сервера, он получает информацию о серверах .com, о серверах пример.com, поэтому ему не надо начинать от корня, ему не надо обращаться к .com, он может обращаться напрямую к полномочному серверу пример.com и задавать ему вопрос, получать ответ, так же кэшировать его и возвращать конечному преобразователю, который возвращает его приложению. Вы видите, насколько кэширование все ускоряет. Если бы не было кэширования, все в Интернете было бы гораздо медленнее.

На последнем моем слайде представлена схема высокого уровня. Мы говорили только о DNS, но если подумать о доменных именах в контексте ICANN, то мы рассматриваем нечто большее, чем полномочные DNS, мы представляем себе более масштабную картину, которая включает в себя владельцев доменов, регистраторов и регистратуры.

Я показываю эту более крупную картину просто чтобы дать представление о том, где находятся другие «действующие лица» мира имен. У нас есть владельцы доменов, которые общаются с регистраторами, обычно через сайты, по вопросам покупки доменных имен или внесения изменений. Затем регистратор связывается с

регистратурами. Основной частью регистратуры является база данных, в которой регистрируются доменные имена и хранится информация о них. Регистратура должна сделать доступной информацию в базе данных, которая связана с DNS, сделать ее доступной на полномочных серверах имен и доступной для запросов от рекурсивных преобразователей. Надеюсь, это немного поможет понять, как это все вписывается в более масштабную картину.

Вот и закончились все мои слайды. У нас осталось немного времени. Буду рад ответить на любые ваши вопросы если они у вас есть.

КАТИ ПЕТЕРСЕН (CATHY PETERSEN): В качестве напоминания, сообщите свое имя и название организации, с которой Вы связаны, если такая есть. Спасибо!

МЭТТ ЛАРСОН: Есть вопросы в зале Adobe? Хорошо. Да, все проголодались. Я тоже немного проголодался.

МАЛИСА РИАРДС (MALISA RICHARDS): У меня есть вопрос. Малиса Ричардс, стипендиат из [неразборчиво]. Какие критерии вы используете при установке сервера корневой зоны?

МЭТТ ЛАРСОН: Для разных операторов они разные. У каждого оператора своя политика. Я не говорил об этом, но сейчас все корневые серверы выполняют любое приведение типа, что означает, что есть не просто один сервер для определенного IP-адреса корневого сервера, но есть несколько серверов, и в любом способе приведения типа используется лежащая в его основе система маршрутизации Интернета, которая позволяет серверу с одним и тем же IP давать ответы на запросы во многих местах в сети.

Все операторы использовали алгоритм «в любом случае» для обеспечения наличия множества экземпляров, как мы их называем, их конкретных адресов корневых серверов, и у разных операторов есть разные политики. Я знаю что для ICANN, для LROOT, практически всем нам нравится использовать LROOT, например, и это именно одиночный сервер. Вы просите купить сервер и потом вы его покупаете, подключаете его к сети и обеспечиваете стойко-место, питание и подключение, выделяете полосу и затем управляете им, а он работает также, как и все остальное в LROOT. У разных операторов корневых серверов есть разные политики.

МАЛИСА РИАРДС: Еще один вопрос. Я сейчас просматриваю страницу в Интернете, которую вы порекомендовали, и я заметила, что в Карибском бассейне, в частности, не так много корневых серверов по сравнению с Южной Африкой или Северной Америкой. Можете ли вы объяснить, почему так получилось?

МЭТТ ЛАРСОН: Не могу ответить за других операторов, но это может быть так потому, что операторы различных сетей не связывались с операторами корневых серверов для установки там своих корневых серверов. По крайней мере, со стороны ICANN ограничения очень незначительны, и если вы хотите купить сервер, то мы будем рады установить его для вашей сети. Спасибо всем. Да, продолжайте.

НОРМАН УОРПУТ (NORMAN WARPUT): Меня зовут Норман, я из Вануату, стипендиат ICANN. Спасибо за презентацию. У меня вопрос по процессу изменения в корневой зоне. Применимо ли это к национальным доменам верхнего уровня?

МЭТТ ЛАРСОН: Похоже, что я не совсем понял ваш вопрос.

НОРМАН УОРПУТ: Применим ли процесс изменения корневой зоны к национальным доменам верхнего уровня?

МЭТТ ЛАРСОН: Да, конечно. С точки зрения DNS, TLD есть TLD. Мы ввели категории TLD, такие как ccTLD и gTLD, и далее спонсированные и неспонсированные gTLD, но это все дополнительные категории, на которые мы их разделили. С точки зрения DNS, TLD есть TLD независимо от того, как мы его называем, будь то ccTLD, gTLD или что-то еще. Есть еще вопросы?

ЭНДРЮ ФРЕЙЗЕР (ANDREW FRASER): Где определено, как переходить на рекурсивный сервер или как дать указание на него? На каком уровне это происходит, через вашего интернет-провайдера или где-то еще? Это в связи с тем, что есть множество рекурсивных серверов, которые мы можем выбирать.

МЭТТ ЛАРСОН: Да, прежде всего, он настроен на каждом устройстве, и когда ваши устройства подключаются к сети, сеть должна дать вам эти данные. Когда вы получаете IP-адрес, любая сеть в дополнение к этому должна

сообщить вам эти данные – она дает IP-адрес рекурсивного сервера. Все, кто обслуживает сеть, должны иметь рекурсивный сервер.

ЭНДРЮ ФРЕЙЗЕР: То есть мой телефон будет иметь заранее заданную конфигурацию в зависимости от моего провайдера?

МЭТТ ЛАРСОН: Ну я бы не сказал что это заранее заданная конфигурация – я бы выразился так: когда телефон подключается к сети вашего провайдера, ваш провайдер должен ответить ему так: «Вот твой IP-адрес, и вот некоторые другие настройки конфигурации, включая имя рекурсивного сервера, который необходимо использовать». Так они работают. При этом, если вы хотите, вы можете заблокировать его и выбрать другой. Есть так называемые сторонние провайдеры рекурсивных DNS или иногда встречаются провайдеры систем DNS общего пользования или открытые преобразователи, они имеют множество имен и являются компаниями, управляющими рекурсивными серверами, которые могут использоваться кем угодно.

Первой компанией, которая сделал это в достаточно большом масштабе, насколько мне известно, была Open DNS, и я помню, что когда они сделали это, моя реакция,

как и реакция многих людей, была следующей: «Зачем нам это может понадобиться?» Служба рекурсивных серверов выглядит как служба базовой сети, зачем мне использовать нечто такое важное, от чего я буду зависеть, когда мне нужно будет сделать что-то в сети и почему я должен полагаться на сервер имен вне моей сети? Предложенные компанией Open DNS преимущества заключались в том, что можем пользоваться дополнительными услугами, мы можем использовать фильтрацию контента на основании имен.

Вы могли сказать: я не хочу преобразовывать имена, соответствующие, например, азартным играм или контенту для взрослых, или чему-то подобному, и они также могли делать такие вещи, не преобразовывать доменные имена, которые соответствуют сайтам, на которых размещено вредоносное ПО или которые неблагонадежны с точки зрения безопасности. Помню, что я подумал, что создание Open DNS было нелепой идеей, но потом Open DNS была куплена Cisco более чем за 600 миллионов долларов, которая, как известно, знает, что делает. После Open DNS была система DNS общего пользования корпорации Google, например. Google заявляла, что она была предназначена для того, чтобы обеспечить надежную службу рекурсивных серверов общего пользования, в которой применялась валидация DNSSEC.

Мне не очень нравится DNSSEC, но это добавляет криптографическую аутентификацию к DNS. Google имела и имеет свое четкое мнение о DNSSEC и она также заявляла следующее: «Вот служба, которую мы предлагаем, она может использоваться бесплатно и вы знаете, что вы получите дополнительную защиту с помощью DNSSEC». Есть и другие системы DNS общего пользования, есть такая у Verisign, и есть система Quad 9. При желании вы можете изменить конфигурацию ваших устройств и использовать эти службы.

Есть еще вопросы? Хорошо, всем спасибо.

КАТИ ПЕТЕРСЕН: Спасибо всем. Спасибо, Мэтт. Спасибо нашим писцам и переводчикам, нашей группе технической поддержки. Прекрасная работа! Еще раз спасибо вам.

[КОНЕЦ СТЕНОГРАММЫ]