Network Working Group
Request For Comments: 896

John Nagle
Ford Aerospace and Communications Corporation

6 January 1984

Контроль перегрузок в сетях TCP/IP

Congestion Control in IP/TCP Internetworks

В этом документе обсуждаются некоторые аспекты контроля насыщения в сетях TCP/IP[1]. Целью является стимулирования размышлений и дальнейшего обсуждения этой темы. Хотя документ содержит конкретные предложения по реализации усовершенствованного контроля перегрузок, он не задаёт каких-либо стандартов.

Введение

Контроль насыщения является признанной проблемой сложных сетей. Мы обнаружили, что протокол Internet Министерства обороны (IP[2]), «чистый» протокол передачи дейтаграмм и протокол управления передачей (TCP[3]) - протокол транспортного уровня при совместном использовании могут порождать необычную проблему насыщения (перегрузки), вызываемую взаимодействием между уровнями транспорта и дейтаграмм. В частности, шлюзы IP[4] подвержены явлению, которое мы назвали «коллапсом насыщения» (congestion collapse), особенно при соединении через такие шлюзы сетей с существенно разной пропускной способностью. Мы разработали решение, предотвращающее такой коллапс.

Эти проблемы не являются общепризнанными, поскольку указанные протоколы чаще всего применяются в сетях, использующих технологию ARPANET IMP. В сетях на основе ARPANET IMP традиционно используется одна пропускная способность и идентичные узлы коммутации, которые имеют ощутимо избыточную «ёмкость». Эта избыточная ёмкость и способность систем дросселировать передачу от хостов обеспечивает для большинства хостов и сетей TCP/IP адекватную обработку насыщения. В результате недавнего разделения ARPANET на две соединённых между собой сети и роста числа других сетей с разными свойствами, подключённых к ARPANET свойств системы IMP стало недостаточно для того, чтобы хосты могли быстро и надёжно обмениваться данными. Требуется усовершенствованный контроль насыщения для успешной работы сети при высоком уровне нагрузки.

Ford Aerospace and Communications Corporation и её родительская компания - Ford Motor Company имеют единственную на сегодняшний день частную сеть TCP/IP на основе каналов значительной протяжённости (long-haul network). Эта сеть соединяет четыре объекта (один в Мичигане, два в Калифорнии и один в Англии) и на некоторых имеются свои локальные сети. Эта сеть имеет соединение с ARPANET но использует свои устройства для протяжённых линий — трафик между объектами Ford передаётся через арендованные линии, включая спутниковый канал для связи через Атлантику. Все коммутационные узлы являются «чистыми» коммутаторами дейтаграмм IP без сквозного контроля насыщения, а на всех хостах применяются программы, написанные или значительно изменённые в Ford или Ford Aerospace. Пропускная способность каналов в этой сети варьируется от 1200 бит/с до 10 Мбит/с. В целом мы не может позволить себе избыточную пропускную способность, как в ARPANET, и каналы дальней связи сильно загружены в периоды пикового трафика. Переходные процессы продолжительностью в несколько секунд являются для сети нормальным явлением.

Поскольку сеть полностью ориентирована на работу с дейтаграммами, имеет значительную нагрузку и существенно различающиеся по пропускной способности каналы, мы вынуждены были решать проблемы, которые сообщество ARPANET/MILNET только начинает осознавать. Наша сеть чувствительна к не вполне оптимальному поведению реализаций TCP на хостах внутри сети и за её пределами. Мы приложили значительные усилия для изучения поведения TCP в разных условиях и решили некоторые широко распространённые проблемы TCP. Здесь представлены две проблемы и их решения. Эти проблемы характерны для многих реализаций TCP - если для данной реализации пропускная способность через шлюз ARPANET/MILNET хуже пропускной способности в рамках одной сети, это говорит о высокой вероятности наличия одной или обеих проблем в реализации TCP.

Коллапс насыщения

До того, как начать обсуждение двух конкретных проблем и их решений, опишем, что может случиться, если эти проблемы своевременно не решить. В сильно загруженных сетях, основанных на передаче дейтаграмм, со сквозным повтором передачи как только коммутирующие узлы достигают насыщения, время кругового обхода через сеть возрастает и вместе с ним растёт число дейтаграмм, находящихся на пути через сеть. Это нормальное поведение под нагрузкой. Пока в пути находится только одна копия каждой дейтаграммы, можно считать, что насыщение контролируется. Но как только начинается повторная передача ещё не доставленных дейтаграмм, могут возникать серьёзные проблемы.

Предполагается, что реализации TCP на хостах повторяют передачу пакетов несколько раз с возрастающими интервалами, пока не будет достигнут некий верхний предел интервала повторной передачи. Обычно этого механизма достаточно для предотвращения серьёзных проблем при возникновении перегрузки в сети. Даже при использовании более совершенных адаптивных алгоритмов повторной передачи внезапно возросшая загрузка в сети может приводить к тому, что время кругового обхода будет расти быстрее, нежели передающий хост способен обновить значение интервала кругового обхода. Такие нагрузки возникают при начале передачи большого объёма данных (например, копирования файла через сеть) со стартовым окном большого размера. Если время кругового обхода превысит интервал повторной передачи какого-либо из хостов, этот хост начнёт передавать в сеть новые копии дейтаграмм. В сети возникнут серьёзные проблемы. В конце концов все доступные буферы узлов коммутации будут заполнены и начнётся отбрасывание пакетов. Время кругового обхода для доставленных пакетов достигает максимума. Хосты

[1] В оригинале используется термин «IP/TCP Internetworks». *Прим. перев.*
[2] Internet Protocol.
[3] Transmission Control Protocol.
[4] Маршрутизаторы в современной терминологии. *Прим. перев.*

передают каждый пакет несколько раз и в конце концов некоторые копии каждого пакета попадают к адресату. В сети возникает коллапс насыщения.

Описанная ситуация устойчива. При возникновении в сети перегрузки, если алгоритм выбора пакетов для отбрасывания является беспристрастным, ухудшение работы сети будет сохраняться. В таких случаях каждый пакет будет передаваться несколько раз и пропускная способность сети будет составлять лишь малую часть обычной производительности. Мы создавали такие условия в нашей сети для эксперимента и наблюдали стабильность поведения. Период кругового обхода может стать настолько большим, что отдельные хосты будут разрывать соединения по тайм-ауту.

В системе ARPANET/MILNET обычно не возникает коллапса и критической перегрузки, поскольку здесь обеспечивается существенная избыточная ёмкость. Если соединение не проходит через шлюзы IP, механизмы управления потоком трафика от IMP к хостам обычно предотвращают возникновение коллапса, в частности, благодаря хорошей временной синхронизации реализаций TCP в случае «чистой» ARPANET. Однако нет никаких механизмов предотвращения коллапса, за исключением сообщений ICMP Source Quench, для случая работы TCP в сети ARPANET/MILNET при отбрасывании пакетов шлюзами. Следует отметить, что достаточно нескольких хостов с некорректным поведением, чтобы перегрузить шлюзы и лишить другие хосты возможности передачи трафика. Мы наблюдали такие проблемы неоднократно с некоторыми хостами ARPANET (с администраторами которых было приватное общение).

Увеличение объёма памяти на шлюзах не решает проблему. При расширении памяти неприемлемый рост времени кругового обхода должен наступать раньше начала отбрасывания пакетов. Это позволяет отодвинуть возникновение коллапса во времени, но после того, как коллапс произойдёт, ситуация усугубляется тем, что большая часть пакетов будет находиться в сети и хосты-отправители будут создавать больше дубликатов.

Две проблемы

Были отмечены две ключевых проблемы, связанных с устройством реализаций TCP — мы назвали их «проблемой мелких пакетов» (small-packet problem) и «проблемой гашения источника» (source-quench problem). Вторая проблема была решена несколькими разработчиками, первую обычно (и неправильно) считают решаемой в будущем. Мы обнаружили, что после решения проблемы мелких пакетов проблема source-quench существенно ослабляется. По этой причине сначала рассматривается проблема мелких пакетов и её решение.

Проблема мелких пакетов

С мелкими пакетами связано несколько проблем. При использовании TCP для передачи 1-символьных сообщений от клавиатуры обычно создаются пакеты размером 41 (1 байт данных и 40 байтов заголовка) при каждом нажатии клавиши. Накладные расходы в 4000% неприятны, но в сетях с незначительной загрузкой не создают проблем. Однако в сетях с высокой загрузкой вызываемое указанными накладными расходами насыщение сети может приводить к потере дейтаграмм и повторам передачи, поскольку перегрузка коммутационных узлов и шлюзов будет приводить к значительному увеличению времени доставки. На практике пропускная способность может снизиться до того, что соединения TCP будут разрываться.

Эта «классическая» проблема известна давно и была отмечена в сети Tymnet ещё в конце 1960-х годов. В этом случае решением проблемы стало ограничение числа дейтаграмм, генерируемых в единицу времени. Это ограничение реализовалось путём задержки передачи мелких пакетов на короткое (200 — 500 мсек) время в надежде на то, что за это время будет введён ещё один символ и можно будет передать их в одном пакете. Другим свойством, повышающим удобство работы пользователей, была отмена задержки при передаче управляющих символов (например, возврат каретки).

Этот метод применяется в NCP Telnet, X.25 PAD и TCP Telnet. Преимуществами метода являются простота понимания и реализации, а недостаток состоит в сложности выбора параметра задержки, который бы устроил всех. Малое время обеспечит быстрый отклик в сетях Ethernet 10 Мбит/с, но слишком мало для предотвращения коллапса насыщения в сильно загруженных сетях с периодом кругового обхода в 5 секунд. И наоборот, достаточно большая задержка в соответствии с высокой загрузкой сети не будет удовлетворять пользователей сети Ethernet.

Решение проблемы мелких пакетов

Очевидно, что для решения проблемы желательна адаптивная модель. Одним из вариантов может быть адаптивный межпакетный интервал в зависимости от времени кругового обхода, наблюдаемого TCP. Хотя такой механизм можно реализовать, он не является необходимым. Было найдено более простое и элегантное решение.

Решение заключается в запрете отправки новых сегментов TCP, пока не будут подтверждены все отправленные ранее данные. Запрет является безусловным — не применяется каких-либо таймеров, проверки размера пакетов или выполнения иных условий. Для реализации обычно требуется лишь пара строк кода в программном модуле TCP.

На первый взгляд кажется, что такое решение кардинально изменит поведение TCP. Однако это лишь кажется. В результате все работает правильно. Давайте посмотрим, почему это происходит.

Когда пользовательский процесс «пишет» в соединение TCP, модуль TCP получает некие данные. Он может удержать их для будущей отправки или передать пакет незамедлительно. Если передача откладывается, данные обычно передаются после получения входящего пакета и связанной с этим смены состояния системы. Состояние меняется одним из двух способов — входящий пакет подтверждает получение отправленных данных удаленным хостом или удалённый хост сообщает о доступности буферного пространства для получения новых данных (второй случай называется «обновлением окна»). Всякий раз при получении данных в соединении модуль TCP должен проверить своё текущее состояние и, возможно, передать некоторые пакеты. Таким образом, когда мы откладываем передачу данных, полученных от пользователя, мы просто задерживаем передачу до момента прихода следующего сообщения от удалённого хоста. Сообщение должно прийти достаточно скоро, если соединение не было до этого в состоянии бездействия и связь с удаленным хостом не была потеряна. В первом случае (бездействие соединения) наша схема будет приводить к передаче пакета всякий раз, когда пользователь что-то пишет в соединение TCP. Таким образом, бездействие соединение не ведёт к блокировке. Во втором случае, когда связь с удаленным хостом потеряна,

передача дополнительных данных все равно приведёт к отказу. Отметим, что мы никак не изменили обычную логику повтора передачи TCP, поэтому потеря сообщений не создаёт проблемы.

Проверка поведения этой схемы в разных условиях показывает её работоспособность при любых обстоятельствах. Первая из проверок связана с отмеченной выше проблемой, которую хочется решить, а именно — ориентированные на передачу символов соединения Telnet. Предположим, что пользователь передающего модуля TCP вводит новый символ каждые 200 мсек, а соединение организовано через сеть Ethernet со временем кругового обхода (включая программную обработку) 50 мсек. Без какого-либо механизма предотвращения насыщения мелкими пакетами будет передаваться один пакет на каждый введённый пользователем символ и отклик будет оптимальным. Накладные расходы составят 4000%, но это приемлемо для сети Ethernet. Классический таймер с ограничением 2 пакета/с приведёт к передаче в каждом пакете 2 или 3 символов. Отклик в результате замедлится, хотя в широкополосной сети Ethernet это не нужно. Накладные расходы снизятся до 1500%, но для сети Ethernet это не будет хорошим компромиссом. В нашей схеме при каждом вводе символа пользователем TCP будет в состоянии бездействия и символ будет передаваться сразу, как при отсутствии контроля насыщения. Пользователь не заметит какой-либо задержки. Таким образом, наша схема будет работать, как схема без контроля перегрузок и обеспечит более быстрый отклик по сравнению с использованием таймера.

Во второй проверке выполнялся такой же тест Telnet, но через протяжённый канал со временем кругового обхода 5 секунд. Без какого-либо механизма предотвращения перегрузки мелкими пакетами каждые 5 секунд будет передаваться 25 новых пакетов.[1] Издержки составят 4000%. В классической схеме с таймером и тем же ограничением (2 пакета в секунду) число остающихся в сети и вносящих вклад в насыщение пакетов составит 10. Передача множества пакетов не улучшит время кругового обхода, а, в общем случае, ухудшит, поскольку остающиеся в сети пакеты занимают время на линии. Издержки снизятся до 1500%. В нашей схеме при вводе первого символа соединение TCP окажется бездействующим и пакет будет передан незамедлительно. Следующие 24 символа, приходящие от пользователя с интервалом 200 мсек, будут удерживаться до приёма сообщения от удалённого хоста. При поступлении через 5 секунд подтверждения ACK от удалённой стороны для первого переданного пакета будет отправлен 1 пакет с 24 символами. В результате наша схема снизит издержки до 320% без ухудшения времени отклика. Обычно в нашей схеме время отклика будет лучше, поскольку снизятся накладные расходы — в данном случае в 4,7 раз по отношению к схеме с таймером. Насыщение будет снижено во столько же раз, а задержка, связанная со временем кругового обхода, резко уменьшится. Для этой ситуации наша схема обеспечивает заметные преимущества перед любой другой моделью.

Мы используем нашу схему для всех соединений TCP, а не только для Telnet. Посмотрим, как будет происходить передача файлов при использовании этого метода. Снова рассмотрим два крайних случая.

Как и раньше, рассмотрим сначала случай сети Ethernet. Пользователь записывает данные в TCP блоками по 512 байтов с той скоростью, которую может поддерживать TCP. Первая запись пользователя в TCP начинает процесс и первая дейтаграмма будет иметь размер 512+40 или 552 байта. Вторая запись от пользователя не вызовет передачи пакета и блок будет помещён в буфер. Предположим, что пользователь заполнит выходной буфер TCP до того, как вернётся первое подтверждение ACK. Когда пакет ACK придёт, все данные из очереди, вплоть до размера окна, будут переданы. С этого момента окно будет находиться в заполненном состоянии, поскольку каждый пакет ACK будет вызывать передачу данных из очереди. Таким образом, после первого интервала кругового обхода, в течение которого передаётся единственный блок, наша схема будет работать с максимальной пропускной способностью. Стартовая задержка составит около 50 мсек для Ethernet, поэтому она не окажет значительного влияния. Все три схемы в этой ситуации обеспечивают эквивалентную производительность.

Далее рассмотрим передачу файла через соединение с пятисекундным интервалом кругового обхода. Как и раньше, будет передан только один пакет, пока не поступит первое подтверждение ACK. После этого окно будет заполнено и сохранит это состояние. Поскольку время кругового обхода составляет 5 секунд, в течение первых пяти секунд будет передано лишь 512 байтов данных. В предположении окна размером 2K после прихода первого ACK будет передано 2K данных и далее будет поддерживаться передача 2K в течение каждого интервала кругового обхода (5 секунд). Это единственный случай, когда наша схема уступает варианту с таймером и различие заметно только на начальном этапе, а установившаяся пропускная способность одинакова. Наивная схема и ограничение по таймеру будут приводить к затрате 250 секунд на передачу 100 кбайт при описанных выше условиях, а наша схема требует 254 секунд — разница составляет 1,6%.

Таким образом, во всех протестированных случаях наша схема обеспечивает производительность не менее 98% от двух других схем, но обеспечивает существенное преимущество при организации соединений Telnet по каналам с большими интервалами кругового обхода. Мы применяем эту схему в Ford Aerospace Software Engineering Network и позволяет запускать экранные редакторы через сеть Ethernet и разговаривать с удалёнными хостами TOPS-20 с повышенной производительностью в обоих случаях.

Контроль насыщения с помощью ICMP

Решив проблему перегрузки сети мелкими пакетами и вместе с ней проблему избыточного числа мелких пакетов в сети, мы обратили внимание на проблему контроля насыщения в целом. Поскольку наша сеть работает исключительно на основе дейтаграмм без управления потоком данных между парами узлов, единственным доступным в рамках протокола IP механизмом являются сообщения ICMP Source Quench. Мы обнаружили, что при аккуратной обработке этих сообщений обеспечивается адекватный способ предотвращения серьёзных проблем при перегрузках. Мы считаем, что нужно уделять серьёзное внимание поведению наших хостов и коммутационных узлов по отношению к сообщениям Source Quench.

[1]Такой ситуации не возникает в «чистой» сети ARPANET, поскольку IMP будет блокировать хост при большом числе неподтвержденных пакетов, но при вовлечении «чистой» локальной сети на основе дейтаграмм (типа Ethernet) или «чистого» шлюза дейтаграмм (типа шлюза ARPANET/MILNET) возможно нахождение в сети большого числа (неподтвержденных) крошечных пакетов.

Когда передавать ICMP Source Quench

Действующий стандарт ICMP[1] указывает, что сообщения ICMP Source Quench следует передавать всякий раз при отбрасывании пакета и можно также передавать их в тех случаях, когда шлюз отмечает у себя нехватку ресурсов. Здесь есть некоторая двусмысленность, но очевидно, что отбрасывание пакетов без отправки сообщений ICMP является нарушением стандарта.

Наше базовое допущение состоит в том, что пакеты не должны отбрасываться при нормальной работе сети. Следовательно, мы хотим «пригасить» отправителей до того, как они перегрузят коммутационные узлы и шлюзы. Все наши коммутационные узлы передают сообщения ICMP Source Quench до того, как на узле закончится буферная ёмкость, не дожидаясь того, что пакеты придётся отбрасывать в результате нехватки буферов. Как показано в анализе проблемы мелких пакетов простое увеличение размера буферов не решает проблему. Для общего случая наш опыт показывает, что сообщения Source Quench следует передавать после того, как занято около половины буферного пространства — это допущение не основано на исчерпывающих экспериментах, но является разумным с технической точки зрения. Можно говорить об адаптивной схеме с управляемым порогом «гашения», но эксперименты показали, что такая система пока не требуется.

Имеющиеся реализации шлюзов генерируют сообщения Source Quench лишь после первого факта отбрасывания пакета. Мы сочли такой подход нежелательным, поскольку любая система контроля насыщения, основанная на отбрасывании пакетов, приводит к дополнительному расходу пропускной способности и может быть подвержена коллапсу при высоком уровне загрузки сети. Мы приняли решение о генерации сообщений Source Quench с большой неохотой, понимая, что пакеты подтверждения тоже могут «гаситься» и это будет приводить к отказам соединений. Ниже показано, что аккуратная обработка сообщений Source Quench на хостах снимает это опасение.

Действия при получении ICMP Source Quench

Мы информируем TCP или любой другой протокол этого уровня о приёме сообщения ICMP Source Quench. Базовым действием наших реализаций TCP в этом случае является снижение количества остающихся в сети данных для хоста, указанного в Source Quench. Такой контроль заставляет передающий модуль TCP вести себя так, будто размер окна на удалённом хосте был снижен. Наша первая реализация была простейшей, но эффективной — при получении Source Quench наш модуль TCP вёл себя так, будто размер окна стал нулевым, даже если окно не пусто. Такое поведение сохранялось до получения некоторого числа (в настоящее время 10) пакетов ACK, после чего TCP возвращался в обычный режим.[2] David Mills из Linkabit Corporation реализовал похожий, но более эффективный механизм дросселирования на основе счётчика остающихся в сети пакетов для своих систем DCN. Дополнительное усложнение механизма не даёт, по видимому, существенного результата, но это не подвергалось формальной проверке. Обе реализации эффективно предотвращают коллапс насыщения на коммутационных узлах.

Source Quench, таким образом, эффективно ограничивает число остающихся в сети сообщений (в предельном случае до одного). В результате связь сохраняется, но падает скорость, что в точности соответствует желаемому эффекту.

Важным свойством этой схемы является то, что Source Quench не препятствует отправке подтверждений и повторов. Реализация Source Quench полностью на уровне IP обычно не приводит к успеху, поскольку уровень IP не имеет достаточной информации для корректного управления соединением. Удержание подтверждений ведёт в повторам передач и созданию ненужного трафика. Удержание повторов может приводить к разрыву соединений по тайм-ауту повтора. В нашей схеме соединения сохраняются даже при пиковой нагрузке и снижается лишь их пропускная способность.

Другим протоколам, работающим на одном уровне с TCP, также следует отвечать на сообщения Source Quench. В каждом случае мы предлагаем «притормаживать» новый трафик, но подтверждения должны передаваться, как обычно. Единственная серьёзная проблема связана с протоколом UDP, но обычно с ним связана небольшая часть трафика. Мы пока не реализовали какого-либо механизма торможения этого трафика и сообщения Source Quench просто игнорируются для UDP.

Самозащита шлюзов

Как было отмечено, шлюзы уязвимы к некорректному управлению перегрузками на хостах. Некорректное поведение хоста в части генерации избыточного трафика может создавать проблемы не только для этого хоста, но и для других хостов, трафик которых не связан с данным хостом. Проблему можно решать на уровне хоста, но, поскольку один некорректно работающий хост, оказывает негативное влияние на работу других, шлюзам в будущем следует поддерживать возможность защиты самих себя от такого поведения некорректно работающих или злонамеренных хостов. Мы предлагаем некоторые базовые способы такой самозащиты.

Однажды, в конце 1983 года, ошибка TCP на хосте ARPANET вынудила этот хост на отчаянную передачу повторов одной и той же дейтаграммы с частотой, которую сеть ARPANET могла воспринять. Шлюз, который соединял нашу сеть с ARPANET оказался перегружен и лишь малая часть полезного трафика могла пройти через него, поскольку пропускная способность соединения шлюза с ARPANET превышала пропускную способность в нашу сеть. Шлюз безостановочно передавал сообщения ICMP Source Quench, но породивший проблему хост игнорировал их. Это продолжалось в течение нескольких часов, пока вызвавший проблему хост не рухнул. В течение этого срока наша сеть была фактически отключена от ARPANET.

Шлюз, вынужденный отбросить пакет, выбирает жертву по своему усмотрению. Классическим вариантом является выбор последнего полученного последним пакета или пакета в конце самой длинной очереди. Мы считаем целесообразным отбрасывать последний пакет, полученный от хоста, который является источником наибольшего числа пакетов в очередях шлюза. Такая стратегия обеспечивает распределение пропускной способности между хостами, использующими этот шлюз. Мы ещё не испытали эту стратегию, но она представляется разумной стартовой точкой для самозащиты шлюзов.

---

[1] ARPANET RFC 792 является действующим стандартом. Мы предупреждены Агентством оборонных коммуникаций (Defense Communications Agency) о том, что описание ICMP в документе MIL-STD-1777 не полное и будет удалено из новых версий этого стандарта.

[2] Это соответствует афоризму: «Не думай о пропорциональном управлении, пока работает дискретное».

Другая стратегия заключается в отбрасывании недавно прибывшего пакета, если он дублирует уже имеющийся в очереди пакет. Связанные с этим расчёты не вызывают проблем, если используются методы хэширования. Такая проверка не обеспечивает защиты от злонамеренных хостов, но обеспечит некоторую защиту от реализаций TCP с неподобающим контролем повторной передачи. Шлюзы между скоростными локальными сетями и узкополосными протяжёнными сетями могут счесть такую проверку полезной, если локальные хосты настроены на эффективную работу в ЛВС.

В идеальном случае шлюзу следует детектировать некорректно работающие хосты и подавлять их, однако такое детектирование в системах на основе дейтаграмм достаточно сложно. Хотя отказ реагировать на сообщения ICMP Source Quench можно считать основанием для шлюза отключить соответствующий хост. Обнаружение таких отказов является нетривиальной задачей, но это благодатная тема для исследования.

Заключение

Проблемы контроля насыщения в сетях на основе дейтаграмм весьма сложны, но имеют эффективные решения. Если сеть TCP/IP работает с высокой нагрузкой, реализации TCP должны решать несколько важных вопросов не менее эффективно, нежели описано здесь.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru