

## NTK\_RFC 0002

### Bandwidth measurement

#### Измерение пропускной способности

В этом документе описаны изменения протокола Npv7. Текст будет включен в окончательную документацию, а пока его можно изменять, контактируя с разработчиками.

#### Проблема измерения качества каналов

В текущей версии Npv7 радар (Radar) измеряет качество канала, используя лишь время кругового обхода  $rtt^1$  и потери пакетов. Это явно не оптимально, поскольку пропускная способность не учитывается и канал с малой пропускной способностью (например, 20 Кбит/с), но с небольшой задержкой окажется лучше широкополосного канала с умеренной задержкой.

#### Улучшение

Узел должен включать в пакеты трассировки (Tracer Packet) не только  $rtt$  пройденных каналов, но и текущую их пропускную способность (полосу). Это позволит формировать трафик с учетом реальной пропускной способности каналов.

#### Измерение пропускной способности

Измерение включает 2 фазы. В первой измеряется общая полоса новых каналов узлами перехвата и назначения, а вторая фаза обеспечивает постоянный мониторинг пропускной способности. Отслеживание используемой полосы выполняется с помощью библиотеки `librcar`.

#### Общая доступная полоса

$A \leftrightarrow B \leftrightarrow C$

Узел В «ловит» трафик между узлами А и С. В конце ловушки В измеряет общую доступную полосу каналов В→С и В→А. Имеются разные методы измерения пропускной способности каналов. Первый и простейший заключается в передаче неопределенного числа случайных пакетов в течение нескольких секунд по адресу получателя на канале. Скорость передачи контролируется с помощью `librcar` и регистрируется максимальное число переданных в секунду байтов как доступная для выгрузки (upload) полоса данного канала. Этот метод очень эффективен и измеренная полоса является реальной, а не аппроксимированной. Побочным эффектом является полная загрузка канала на несколько секунд.

Другой вариант более изыскан и использует методы Packet Pair и Packet Train, описанные в <http://perform.wpi.edu/downloads/wbest/README> и <http://www-static.cc.gatech.edu/fac/Constantinos.Dovrolis/bw.html>.

Поскольку каналы могут быть асимметричными, измерения повторяются также для А→В и С→В. В результате будет получена пропускная способность каналов А→В, В→А, С→В, В→С.

#### Контроль полосы в реальном масштабе времени

С помощью библиотеки `librcar` узел В может отслеживать использование полосы каналов.

##### *Max\_link\_bw*

Общая доступная пропускная способность канала.

##### *nflows*

Текущее число потоков в канале. Поток считается множество (stream) пакетов, связанных с конкретным соединением.

Текущую доступную полосу канала можно аппроксимировать выражением

$link\_avail\_bw = Max\_link\_bw / (nflows + 1)$

Такая аппроксимация выбрана потому, что метод управления трафиком TCP похож на алгоритм max-min и при большом числе потоков  $link\_avail\_bw$  является хорошим приближением.

Если радар сообщает о значительных вариациях текущей доступной полосы, передается новый запрос QSPN.

#### Задержка rtt

Каждый узел сети будет задерживать пересылку полученного пакета Трасег (TP) на время, обратно пропорциональное  $link\_avail\_bw$ . Таким образом, пакеты TP будут приниматься для эффективности. Побочным эффектом этого правила является игнорирование крайних случаев, когда маршруты с очень низким значением  $rtt$  будут иметь очень малую полосу  $bw$  или маршруты с оптимальной полосой  $bw$  будут иметь очень большое значение  $rtt$ . Однако в реальном мире такие ситуации достаточно редки, поскольку  $rtt$  и  $bw$  зачастую связаны между собой.

Дополнительная информация об эффективности пакетов TP приведена в документе [QSPN v2](#).

<sup>1</sup>Round-trip time.

## Пропускная способность маршрута

Пропускная способность маршрута из S в D обозначается  $bw(S \rightarrow D)$  и равна пропускной способности наилучшего канала в пути. Например, для маршрута

S --64Mb/s--> R --64Mb/s--> T --32Mb/s--> O --100Mb/s--> I --100Mb/s--> D  
 пропускная способность составит  $bw(S \rightarrow D) = 32$  Мбит/с.

Пакет TP должен запомнить единственное значение пропускной способности (`_one_bw`) и это будет  $bw()$  текущего маршрута. Например, если S передает TP, то по получении этого пакета узлом T, сохраненная в нем пропускная способность будет  $bw(S \rightarrow R \rightarrow T) = 64$ , но при достижении узла O, она сменится на  $bw(S \rightarrow R \rightarrow T \rightarrow O) = 32$ .

Отметим, что каждое значение  $bw$  приблизительно соответствует доступной полосе соответствующего канала (`link_avail_bw`). Например, запись S --64Mb/s--> R показывает, что текущая пропускная способность канала S-->R составляет приблизительно 64 Мбит/с.

## Асимметричные маршруты

QSPN v1 предоставляет каждому узлу наилучший маршрут загрузки трафика (download) с каждого другого узла. Рассмотрим в качестве примера узлы S и D.



Маршруты типа 1 являются лучшими маршрутами выгрузки (upload) из D в S, а маршруты типа 2 - лучшими для противоположного направления. QSPN дает S только маршруты типа 1, а D знает только маршруты типа 2. Если маршруты не симметричны, S при выгрузке в D будет использовать маршрут типа 1, который может иметь низкую производительность.

Решение очень просто - S при выгрузке большого блока данных будет запрашивать у D его маршруты D->S (тип 2), т. е. лучшие маршруты выгрузки из S в D.

Демон Netsukuku, используя `libscap`, будет отслеживать пакеты, исходящие от `localhost`. Если пакеты отправлялись более 16 секунд, демон будет запрашивать у целевого узла лучшие маршруты выгрузки (upload) и добавлять их в свою таблицу маршрутизации. Таким образом, маршруты выгрузки будут запрашиваться лишь при необходимости.

Отметим, что в [QSPN v2](#) имеется встроенная поддержка асимметричной маршрутизации.

## Задержка и полоса

Может возникнуть ситуация, когда канал с хорошей пропускной способностью имеет высокую задержку. Для некоторых приложений малая задержка важна - к таким приложениям относится, например, ssh, где не так важна высокая скорость как малая задержка, чтобы получать быстрый отклик при вводе команд.

Узлам Netsukuku следует поддерживать три типа таблиц маршрутизации:

- маршруты с наилучшей пропускной способностью;
- маршруты с наименьшей задержкой;
- маршруты с эффективной комбинацией пропускной способности и задержки.

Если протокол приложения использует значения TOS в своих пакетах IP, можно добавить правила маршрутизации и выбрать для протокола подходящую таблицу (например, вторую для ssh).

Возможен вариант хранения в ядре лишь усредненной (третьей) таблицы, поскольку она применяется наиболее часто. Конкретные маршруты с наилучшей пропускной способностью и задержкой можно добавлять в ядро лишь в момент их использования. Например, при запуске bittorrent для загрузки с узлов A,B,C монитор ntkd будет добавлять в ядро маршруты с наилучшей пропускной способностью для A,B,C.

## IGS

Шлюз inet-gw, позволяющий совместно использовать свое соединение с Internet, замеряет свою доступную полосу Internet-соединения. Максимальная пропускная способность в обоих направлениях известна, поскольку пользователь должен задать ее в конфигурационном файле `netsukuku.conf`. Таким образом, отслеживание устройства, используемого принятым по умолчанию маршрутом в Internet, позволяет легко узнать доступную полосу.

## Балансировка нагрузки

Балансировка нагрузки будет применяться всегда, поскольку маршруты добавляются в ядро с использованием опции `multipath`. Пакеты, передаваемые узлу, будут использовать разные маршруты.

Балансировка нагрузки внутри Netsukuku не зависит от проблем балансировки на IGS. Узлы ntk согласованы между собой и не используют NAT для взаимодействия внутри сети.

Перевод на русский язык

Николай Малых

[nmalykh@protocols.ru](mailto:nmalykh@protocols.ru)

[www.protocols.ru](http://www.protocols.ru)