Network Working Group
Request for Comments: 903

Finlayson, Mann, Mogul, Theimer
Stanford University
June 1984

Протокол RARP

A Reverse Address Resolution Protocol

Ross Finlayson, Timothy Mann, Jeffrey Mogul, Marvin Theimer
Computer Science Department
Stanford University
June 1984

Статус документа

В этом документе описывается метод, позволяющий станциям динамически определять свой адрес протокольного[1] уровня (например, IP) по известному аппаратному адресу.

Документ содержит спецификацию стандарта, предложенного сообществу ARPA Internet и является приглашением к дискуссии в целях развития стандарта.

I. Введение

Хосты типа бездисковых станций зачастую не знают свой протокольный адрес в момент загрузки, но им обычно известен аппаратный адрес сетевого интерфейса. Для взаимодействия с протоколами вышележащих уровней (например, IP) хост должен узнать свой протокольный адрес из некого внешнего источника. Проблема заключается в отсутствии стандартного механизма получения адреса.

Разработанный Пламмером (Plummer) протокол ARP[2] [1] предназначен для решения обратной задачи – определения аппаратного адреса хоста по его протокольному адресу. В данном документе описывается протокол обратного преобразования адресов RARP[3]. Как и в случае ARP, предполагается широковещательная сетевая среда, подобная Ethernet.

II. Вопросы реализации протокола

Ниже приведены рекомендации по реализации протокола RARP.

A) Преобразования ARP и RARP отличаются одно от другого. ARP предполагает, что каждый хост знает связь между своими аппаратными и протокольными адресами. Информация о связях между аппаратными и протокольными адресами других хостов собирается в специальном кэше. Все хосты сети с точки зрения ARP имеют одинаковый статус (не требуется деление на серверы и клиентов).
Для преобразования RARP требуется один или несколько серверов, поддерживающих базу данных о связях между аппаратными и протокольными адресами хостов и отвечающих на запросы адресов от клиентских хостов.

B) Как было отмечено, преобразование RARP требует наличия серверов, поддерживающих большие базы данных. Нежелательно, а в некоторых случаях просто невозможно поддерживать такие базы данных в ядре операционной системы хостов. Таким образом, большинство реализаций будут требовать того или иного взаимодействия с программой, не входящей в ядро.

C) Для протокола важна простота реализации и минимальное воздействие на существующие программы. Было бы ошибкой предлагать протокол, который потребует модификации программ каждого хоста, независимо от того, будет ли этот протокол использоваться на данном хосте.

D) Желательно использование фрагментов кода существующих программ в целях минимизации усилий по разработке и снижения расходов.

III. Предлагаемый протокол

Предлагается выполнять преобразование RARP с помощью отдельного протокола канального уровня. Например, для сред Ethernet пакеты RARP будут иметь значение поля Ethertype (которое будет выделено для этого протокола), отличающееся от ARP. Это будет указывать, что преобразования ARP и RARP различаются фундаментально и поддерживаются хостами по-разному. Воздействие на существующие системы минимально – существующие серверы ARP не будут вводиться в заблуждение пакетами RARP. Этот делает RARP самостоятельным протоколом, который может применяться для отображения аппаратных адресов на адреса любых протоколов вышележащего уровня.

---

[1] Сетевого уровня в современной терминологии. *Прим. перев.*

[2] Address Resolution Protocol - протокол преобразования адресов.

[3] Reverse Address Resolution Protocol.

Это приближение обеспечивает простоту реализации клиентской части RARP, но разработка RARP-серверов потребует более значительных усилий. Однако, эти трудности не являются непреодолимыми, как показано в Приложении A, где даны наброски двух вариантов реализации протокола для 4.2BSD Unix.

RARP использует такой же формат пакетов, как протокол ARP, а именно:

```
ar$hrd (пространство аппаратных адресов) -  16 битов
ar$pro (пространство протокольных адресов) -  16 битов
ar$hln (размер аппаратного адреса) - 8 битов
ar$pln (размер протокольного адреса) - 8 битов
ar$op  (код операции) - 16 битов
ar$sha (аппаратный адрес отправителя) - n[1] байтов
ar$spa (протокольный адрес отправителя) - m[2]  байтов
ar$tha (аппаратный адрес получателя) - n байтов
ar$tpa (протокольный адрес получателя) - m байтов
```

Поля arhrd, arpro, ar$hln и ar$pln имеют такой же смысл, как для протокола ARP (см. [1]).

Предположим, что в качестве аппаратного используется 48-битовый адрес Ethernet, а в качестве протокольного – 32-битовый адрес IP. Нам требуется определить адрес IP, соответствующий известному адресу Ethernet. В этом случае для каждого пакета RARP выполняются равенства ar$hrd = 1 (Ethernet), ar$pro = 2048[3] (Ethertype для пакетов IP), ar$hln = 6, ar$pln = 4.

Протокол использует два кода операций: 3 (request reverse – запрос обратного преобразования) и 4 (reply reverse – отклик на запрос обратного преобразования). Коды операций 1 и 2 имеют такой же смысл, как указано в [1] – пакеты с такими кодами могут передаваться с использованием обычных программ ARP. Пакеты с любыми другими кодами операций являются ошибочными. Как и ARP, данный протокол не использует пакетов not found (не найдено) или error (ошибка), поскольку серверы RARP не обязаны отвечать на полученные запросы. Отправитель запроса RARP должен задавать для себя время ожидания (тайм-аут) отклика на переданный запрос.

Поля arsha, arspa, artha и ar$tpa в пакетах RARP интерпретируются следующим образом:

Для пакетов с кодом операции 3 (request reverse):

```
ar$sha – аппаратный адрес отправителя пакета.
ar$spa – не определено.
ar$tha – аппаратный адрес получателя[4].
ar$tpa – не определено.
```

Для пакетов с кодом операции 4 (reply reverse):

```
ar$sha – аппаратный адрес отвечающего хоста.
ar$spa – протокольный адрес отвечающего хоста[5].
ar$tha – аппаратный адрес получателя[6].
ar$tpa – протокольный адрес получатель (то, что было запрошено).
```

IV. Литература

[1] Plummer, D., "An Ethernet Address Resolution Protocol", RFC 826[7], MIT-LCS, November 1982.

Приложение A. Два примера реализации протокола для 4.2BSD Unix

Ниже приведены два наброска различных вариантов реализации сервера RARP для 4.2BSD.

A) Обеспечивается доступ к пакетам канального уровня за пределами ядра. Сервер RARP полностью реализуется за пределами ядра и взаимодействует с ядром только для приема или передачи пакетов RARP. Ядро изменяется так, чтобы обеспечивался требуемый доступ к пакетам RARP; в настоящее время ядро 4.2 обеспечивает доступ только для пакетов IP. Одним из существующих механизмов, которые обеспечивают возможность доступа к пакетам является псевдо-драйвер пакетного фильтра CMU. Этот драйвер успешно используется в CMU[8] и Стэнфорде (Stanford) для реализации подобных серверов в пользовательском пространстве.

B) Создается кэш записей базы данных внутри ядра. Полная база данных сервера RARP поддерживается пользовательским процессом за пределами ядра. Сам сервер RARP реализуется в ядре и использует в откликах содержимое кэша записей. Кэш может быть таким же, какой используется для пересылки ARP.

   Кэш заполняется записями из базы данных с использованием двух новых операций ioctl (они подобны SIOCIFADDR в том, что реально не связываются с конкретным сокетом). Одна из операций обеспечивает нахождение в “спящем” режиме при отсутствии транзакций и при появлении запросов передает их пользовательскому процессу, а вторая заносит запись этой транзакции в таблицу ядра. Таким образом, если ядро не может найти запись в кэше, оно помещает запрос в (глобальную) очередь и вызывает функцию wakeup(). Реализация первой операции ioctl включает вызов sleep(), считывание из очереди первого объекта и возврата его пользовательскому процессу.

---

[1]Значение n определяется из поля ar$hln.

[2]Значение m определяется из поля ar$pln.

[3]Десятичное значение.

[4]В случаях, когда отправитель хочет узнать свой протокольный адрес, это поле, подобно ar$sha, содержит аппаратный адрес отправителя.

[5]Отметим, что указание в поле ar$spa в пакетах с кодом 4 протокольного адреса отправителя включено лишь для удобства. Например, система, в которой одновременно используется ARP и RARP, может использовать полученную пару адресов (arspa, arsha) и не передавать впоследствии запрос ARP для этого адреса.

[6]Должен совпадать с адресом отправителя запроса

[7]На сайте www.protocols.ru имеется перевод этого документа на русский язык.

[8]Carnegie-Mellon University — университет Карнеги-Мэллона. *Прим. перев.*

Поскольку ядро не может находится в состоянии ожидания на уровне прерывания, пока пользовательский процесс ответит, оно может отказаться от обработки запроса (в предположении, что клиент повторит запрос) или, если вторая операция ioctl передаст копию запроса обратно в ядро, сформировать и передать отклик на этот запрос.

Перевод на русский язык

Николай Малых

nmalykh@bilim.com