Белгородский государственный технологический университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №5 по теме: «Протоколы ARP/RARP»

**Выполнил:**  
студент группы ПВ-31  
Адаменко И. И.

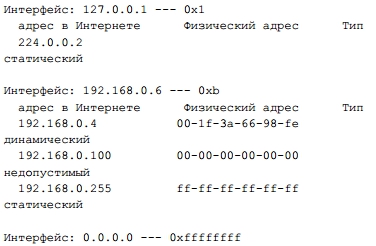
**Проверил:**старший преподаватель  
Федотов Е. А.

Белгород  
2015

**Цель работы:** изучить протоколы ARP/RARP.

**Задание:**

1. Программно реализовать вывод ARP-таблицы в следующем виде:



1. Программно реализовать добавление записи в ARP-таблицу.
2. Программно реализовать удаление записи из ARP-таблицы.
3. Программно реализоваться получение MAC-адреса по IP-адресу.
4. Программы должны быть написаны на языке программирования C#.

# Теоретическая часть

ARP (Address Resolution Protocol — протокол определения адреса) — протокол канального уровня, предназначенный для определения MAC-адреса (адреса канального уровня) по известному IP-адресу (адресу сетевого уровня). Наибольшее распространение этот протокол получил благодаря распространению сетей IP, построенных поверх Ethernet, поскольку практически в 100 % случаев при таком сочетании используется протокол ARP.

Протокол ARP работает различным образом в зависимости от того, какой протокол канального уровня работает в данной сети — протокол локальной сети (Ethernet, Token Ring, FDDI) с возможностью широковещательного доступа одновременно ко всем узлам сети, или же протокол глобальной сети (X.25, frame relay), как правило не поддерживающий широковещательный доступ.

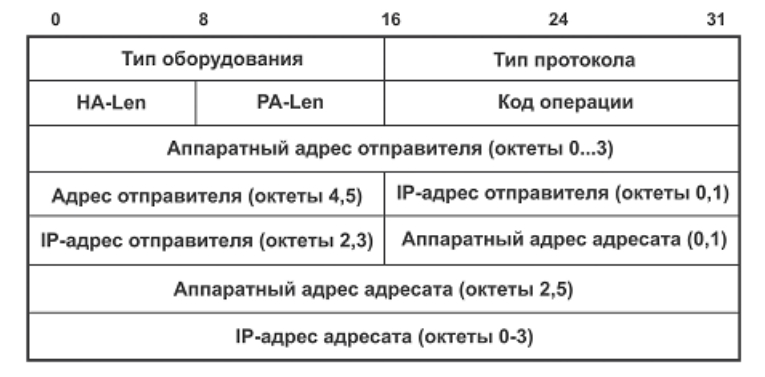
Для того чтобы уменьшить количество посылаемых запросов ARP, каждое устройство в сети, использующее протокол ARP, должно иметь специальную буферную память. В ней хранятся пары адресов (IP-адрес, физический адрес) устройств в сети. Всякий раз, когда устройство получает ARP-ответ, оно сохраняет в буферной памяти соответствующую пару. Если адрес есть в списке пар, то нет необходимости посылать ARP-запрос. Эта буферная память называется ARP-таблицей.

В ARP-таблице могут содержаться как статические, так и динамические записи. Динамические записи добавляются и удаляются автоматически, статические заносятся вручную. Так как большинство устройств в сети поддерживает динамическое разрешение адресов, то администратору, как правило, нет необходимости собственноручно указывать записи протокола ARP в таблице адресов.

Кроме того, ARP-таблица всегда содержит запись с физическим широковещательным адресом (0xFFFFFFFFFFFF) для локальной сети. Эта запись позволяет устройству принимать широковещательные ARP-запросы. Каждая запись в ARP-таблице имеет свое время жизни, например, для операционной системы Microsoft Windows 2000 оно составляет 10 минут. При добавлении записи для нее активируется таймер. Если запись не востребована в первые две минуты, она удаляется. Если используется — будет существовать на протяжении 10 минут. В некоторых реализациях протокола ARP новый таймер устанавливается после каждого обращения к записи в ARP-таблице.

Сообщения протокола ARP при передаче по сети инкапсулируются в поле данных кадра. Они не содержат IP-заголовка. В отличие от сообщений большинства протоколов, сообщения ARP не имеют фиксированного формата заголовка. Это объясняется тем, что протокол был разработан таким образом, чтобы он был применим для разрешения адресов в различных сетях. Фактически протокол способен работать с произвольными физическими адресами и сетевыми протоколами.

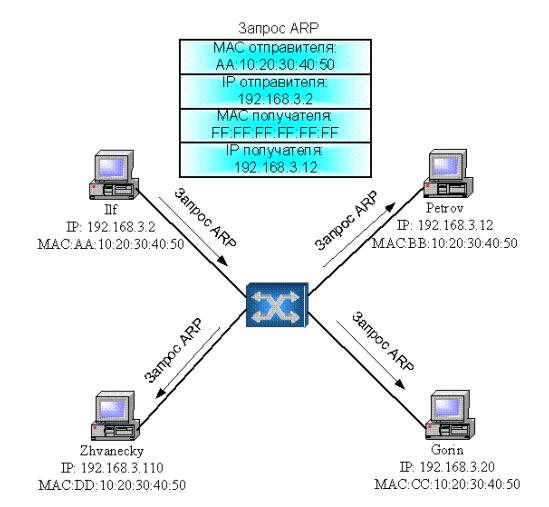
На рисунке ниже показана структура запросов и ответов ARP и RARP.



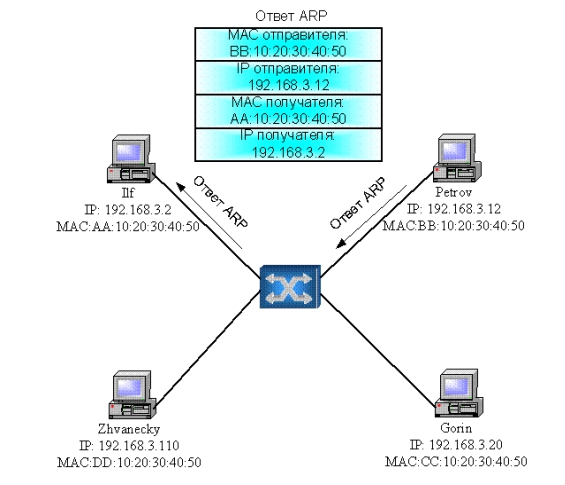
В поле типа оборудования для сетей Ethernet указывается значение 1. Поле типа протокола позволяет использовать пакеты ARP не только для протокола IP, но и для других сетевых протоколов. Для IP значение этого поля равно 0800 16. Длина локального адреса для протокола Ethernet равна 6 байтам, а длина IP-адреса — 4 байтам. В поле операции для ARP запросов указывается значение 1 для протокола ARP и 2 для протокола RARP.

Узел, отправляющий ARP-запрос, заполняет в пакете все поля, кроме поля искомого локального адреса (для RARP-запроса не указывается искомый IP-адрес). Значение этого поля заполняется узлом, опознавшим свой IP-адрес.

Рисунки ниже показывают принцип работы протокола ARP. Анализируя структуру запроса, можно увидеть, что компьютер с адресом 192.168.3.2 делает попытку узнать МАС-адрес компьютера с IP -адресом 192.168.3.12. Для этого он посылает широковещательный запрос, содержащий IP-адрес, с МАС-адресом, установленным в FF:FF:FF:FF:FF:FF .



Когда компьютер с адресом 192.168.3.12 получает этот широковещательный запрос, он анализирует IP-адрес, для которого выполняется разрешение. Определив, что его адрес совпадает с искомым, он формирует ответ протокола ARP, где указывает свой МАС-адрес.



Ответ посылается уже не широковещательно — отправитель знает МАС-адрес инициатора запроса и поэтому передает пакет целенаправленно.

В ходе обычной работы сетевая программа, например, TELNET, отправляет прикладное сообщение, пользуясь транспортными услугами TCP. Модуль TCP посылает соответствующее транспортное сообщение через модуль IP. В результате составляется IP-пакет, который должен быть передан драйверу Ethernet. IP-адрес места назначения известен прикладной программе, модулю TCP и модулю IP. Необходимо на его основе найти Ethernet-адрес места назначения. Для определения искомого Ethernet-адреса используется ARP-таблица.

ARP-таблица заполняется автоматически модулем ARP, по мере необходимости. Когда с помощью существующей ARP-таблицы не удается преобразовать IP-адрес, то происходит следующее:

1. По сети передается широковещательный ARP-запрос.
2. Исходящий IP-пакет ставится в очередь.

Каждый сетевой адаптер принимает широковещательные передачи. Все драйверы Ethernet проверяют поле типа в принятом Ethernet-кадре и передают ARP-пакеты модулю ARP.

Каждый модуль ARP проверяет поле искомого IP-адреса в полученном ARP-пакете и, если адрес совпадает с его собственным IP- адресом, то посылает ответ прямо по Ethernet-адресу отправителя запроса.

Этот ответ получает машина, сделавшая ARP-запрос. Драйвер этой машины проверяет поле типа в Ethernet-кадре и передает ARP-пакет модулю ARP. Модуль ARP анализирует ARP-пакет и добавляет запись в свою ARP-таблицу.

Полностью порядок преобразования адресов выглядит так:

1. По сети передается широковещательный ARP-запрос.
2. Исходящий IP-пакет ставится в очередь.
3. Возвращается ARP-ответ, содержащий информацию о соответствии IP- и Ethernet-адресов. Эта информация заносится в ARP-таблицу.
4. Для преобразования IP-адреса в Ethernet-адрес у IP-пакета, поставленного в очередь, используется ARP-таблица.
5. Ethernet-кадр передается по сети Ethernet.

Т.е., если с помощью ARP-таблицы не удается сразу осуществить преобразование адресов, то IP-пакет ставится в очередь, а необходимая для преобразования информация получается с помощью запросов и ответов протокола ARP, после чего IP-пакет передается по назначению.

Если в сети нет машины с искомым IP-адресом, то ARP-ответа не будет и не будет записи в ARP-таблице. Протокол IP будет уничтожать IP-пакеты, направляемые по этому адресу. Протоколы верхнего уровня не могут отличить случай повреждения сети Ethernet от случая отсутствия машины с искомым IP-адресом.

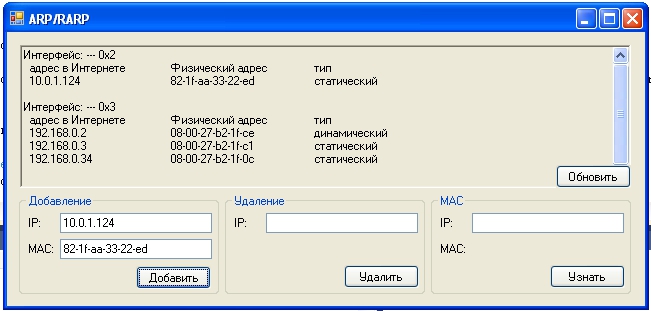
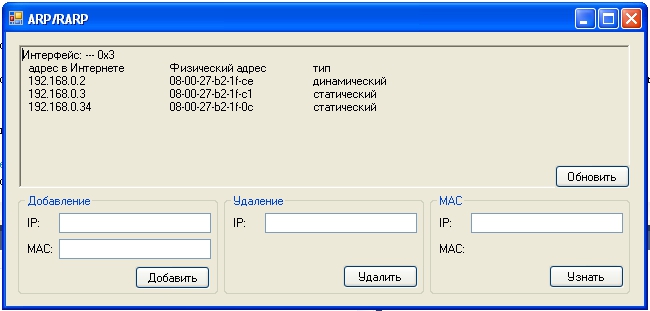
Некоторые реализации IP и ARP не ставят в очередь IP-пакеты на то время, пока они ждут ARP-ответов. Вместо этого IP-пакет просто уничтожается, а его восстановление возлагается на модуль TCP или прикладной процесс, работающий через UDP. Такое восстановление выполняется с помощью таймаутов и повторных передач. Повторная передача сообщения проходит успешно, так как первая попытка уже вызвала заполнение ARP-таблицы.

Следует отметить, что каждая машина имеет отдельную ARP-таблицу для каждого своего сетевого интерфейса.

# Используемые функции

* ExecuteCommandLine — выполняет команду в командной строке Windows;
* MACFormat — форматирует массив байт в строку вида FF-FF-FF-FF-FF-FF;
* IPFormat — форматирует число в строку вида 255.255.255.255;
* ShowARPTable — отображает на форме текущее состояние таблицы ARP.

# Скриншоты приложения

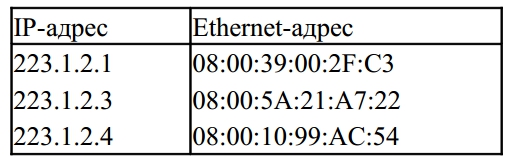


# Код программы

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Windows.Forms;
5. **using** System.Runtime.InteropServices;
6. **using** System.IO;
7. **using** System.Diagnostics;
8. **using** System.Net;
10. **namespace** Lab5
11. {
12. **public** partial **class** MainForm : Form
13. {
14. **public** MainForm()
15. {
16. InitializeComponent();
17. ShowARPTable();
18. }
20. #region events
22. **private** **void** addARP\_Click(**object** sender, EventArgs e)
23. {
24. var arpStream = ExecuteCommandLine("arp",  
     "-s " + addIP.Text + " " + addMAC.Text).ReadToEnd();
25. **if** (arpStream == "")
26. {
27. ShowARPTable();
28. }
29. **else**
30. {
31. arpTable.Clear();
32. arpTable.Text += arpStream;
33. }
34. }
36. **private** **void** delARP\_Click(**object** sender, EventArgs e)
37. {
38. var arpStream = ExecuteCommandLine("arp", "-d " + delIP.Text).ReadToEnd();
40. **if** (arpStream == "")
41. {
42. ShowARPTable();
43. }
44. **else**
45. {
46. arpTable.Clear();
47. arpTable.Text += arpStream;
48. }
49. }
51. **private** **void** ARPTableRefresh\_Click(**object** sender, EventArgs e)
52. {
53. ShowARPTable();
54. }
56. **private** **void** findMAC\_Click(**object** sender, EventArgs e)
57. {
58. **try**
59. {
60. var dst = IPAddress.Parse(findIP.Text);
61. var macAddr = **new** **byte**[6];
62. var macAddrLen = (**uint**)macAddr.Length;
63. var intIP = BitConverter.ToInt32(dst.GetAddressBytes(), 0);
65. **if** (SendARP(intIP, 0, macAddr, **ref** macAddrLen) != 0)
66. **throw** **new** InvalidOperationException("SendARP failed.");
68. var str = **new** **string**[macAddrLen];
70. **for** (**int** i = 0; i < macAddrLen; i++)
71. str[i] = macAddr[i].ToString("x2");
73. findMACLabel.Text = **string**.Join(":", str);
74. findMACLabel.Visible = **true**;
75. }
76. **catch** (Exception ex)
77. {
78. findMACLabel.Text = "неизвестен.";
79. findMACLabel.Visible = **true**;
80. }
81. }
83. #endregion
85. #region imports etc
87. [StructLayout(LayoutKind.Sequential)]
88. **public** **struct** MIB\_IPNETROW
89. {
90. **public** **int** Index;
91. **public** **int** PhysAddrLen;
92. [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
93. **public** **byte**[] PhysAddr;
94. **public** **int** Addr;
95. **public** **int** Type;
96. }
98. **public** **enum** MIB\_IPNET\_TYPE
99. {
100. OTHER = 1,
101. INVALID = 2,
102. DYNAMIC = 3,
103. STATIC = 4
104. }
106. [DllImport("iphlpapi.dll")]
107. **public** **extern** **static** **int** GetIpNetTable(IntPtr pTcpTable, **ref** **int** pdwSize,  
      **bool** bOrder);
109. [DllImport("iphlpapi.dll", ExactSpelling = **true**)]
110. **public** **static** **extern** **int** SendARP(**int** destIp, **int** srcIP,  
      **byte**[] macAddr, **ref** **uint** physicalAddrLen);
112. [DllImport("iphlpapi.dll", ExactSpelling = **true**)]
113. **public** **static** **extern** **int** CreateIpNetEntry(IntPtr pArpEntry);
115. [DllImport("iphlpapi.dll", ExactSpelling = **true**)]
116. **public** **extern** **static** **int** DeleteIpNetEntry(IntPtr pArpEntry);
118. #endregion
120. **public** **static** StreamReader ExecuteCommandLine(String file, String arguments = "")
121. {
122. ProcessStartInfo startInfo = **new** ProcessStartInfo();
123. startInfo.CreateNoWindow = **true**;
124. startInfo.WindowStyle = ProcessWindowStyle.Hidden;
125. startInfo.UseShellExecute = **false**;
126. startInfo.RedirectStandardOutput = **true**;
127. startInfo.FileName = file;
128. startInfo.Arguments = arguments;
130. Process process = Process.Start(startInfo);
132. **return** process.StandardOutput;
133. }
135. **private** **string** MACFormat(**byte**[] m)
136. {
137. **return** **string**.Format("{0:x2}-{1:x2}-{2:x2}-{3:x2}-{4:x2}-  
      {5:x2}", m[0], m[1], m[2], m[3], m[4], m[5]);
138. }
140. **private** **string** IPFormat(**int** ip)
141. {
142. var b = BitConverter.GetBytes(ip);
143. **return** **string**.Format("{0}.{1}.{2}.{3}", b[0], b[1], b[2], b[3]);
144. }
146. **public** **void** ShowARPTable()
147. {
148. var type = **new** **string**[] { "", "другой", "недопустимый", "динамический",  
      "статический" };
149. var arp = **new** List<MIB\_IPNETROW>();
150. var size = 0;
151. GetIpNetTable(IntPtr.Zero, **ref** size, **true**);
152. var p = Marshal.AllocHGlobal(size);
154. **if** (GetIpNetTable(p, **ref** size, **true**) == 0)
155. {
156. var num = Marshal.ReadInt32(p);
157. var ptr = IntPtr.Add(p, 4);
159. **for** (**int** i = 0; i < num; i++)
160. {
161. arp.Add((MIB\_IPNETROW)Marshal.PtrToStructure(ptr,  
      **typeof**(MIB\_IPNETROW)));
162. ptr = IntPtr.Add(ptr, Marshal.SizeOf(**typeof**(MIB\_IPNETROW)));
163. }
165. Marshal.FreeHGlobal(p);
166. }
168. arpTable.Clear();
170. **foreach** (var i **in** arp.GroupBy(x => x.Index))
171. {
172. arpTable.Text +=   
      **string**.Format("Интерфейс: --- 0x{0:x}", i.Key) + '\n';
173. arpTable.Text +=  
      "  адрес в Интернете\t Физический адрес\t тип" + '\n';
174. **foreach** (var n **in** arp.Where(x => x.Index == i.Key))
175. {
176. arpTable.Text += **string**.Format("  {0,-15}\t\t {1}\t\t {2}",
177. IPFormat(n.Addr),
178. MACFormat(n.PhysAddr),
179. type[n.Type]) + '\n';
180. }
182. arpTable.Text += '\n';
183. }
184. }
185. }
186. }

# Ответы на вопросы

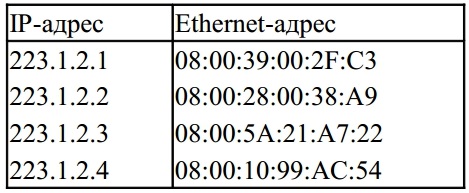
1. **Какие задачи решает протокол ARP?**Функциональность протокола ARP сводится к решению двух задач. Одна часть протокола определяет физические адреса при посылке дейтаграммы, другая отвечает на запросы устройств в сети. Протокол ARP предполагает, что каждое устройство «знает» как свой IP-адрес, так и свой физический адрес.
2. **Что такое ARP-таблица? Почему она является необходимым элементом?**Преобразование адресов выполняется путем поиска в таблице. Эта таблица, называемая ARP-таблицей, хранится в памяти и содержит строки для каждого узла сети. В двух столбцах содержатся IP- и Ethernet-адреса. Если требуется преобразовать IP-адрес в Ethernet-адрес, то ищется запись с соответствующим IP-адресом (пример ниже).



Принято все байты 4-байтного IP-адреса записывать десятичными числами, разделенными точками. При записи 6-байтного Ethernet-адреса каждый байт указывается в 16-ричной системе и отделяется двоеточием.

ARP-таблица необходима потому, что IP-адреса и Ethernet-адреса выбираются независимо, и нет какого-либо алгоритма для преобразования одного в другой. IP-адрес выбирает менеджер сети с учетом положения машины в сети интернет. Если машину перемещают в другую часть сети интернет, то ее IP-адрес должен быть изменен. Ethernet-адрес выбирает производитель сетевого интерфейсного оборудования из выделенного для него по лицензии адресного пространства. Когда у машины заменяется плата сетевого адаптера, то меняется и ее Ethernet-адрес.

1. **Типы записей ARP-таблицы**Таблица состоит их двух столбцов: IP и Ethernet. Например:



1. **Опишите процесс преобразования IP-адреса в локальный**Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель указывает свой локальный адрес. ARP-запросы и ответы используют один и тот же формат пакета. Так как локальные адреса могут в различных типах сетей иметь различную длину, то формат пакета протокола ARP зависит от типа сети.
2. **Как может работать протокол в глобальных сетях?**В глобальных сетях администратору сети чаще всего приходится вручную формировать ARP-таблицы, в которых он задает, например, соответствие IP-адреса адресу узла сети X.25, который имеет смысл локального адреса. В последнее время наметилась тенденция автоматизации работы протокола ARP и в глобальных сетях. Для этой цели среди всех маршрутизаторов, подключенных к какой-либо глобальной сети, выделяется специальный маршрутизатор, который ведет ARP-таблицу для всех остальных узлов и маршрутизаторов этой сети. При таком централизованном подходе для всех узлов и маршрутизаторов вручную нужно задать только IP-адрес и локальный адрес выделенного маршрутизатора. Затем каждый узел и маршрутизатор регистрирует свои адреса в выделенном маршрутизаторе, а при необходимости установления соответствия между IP-адресом и локальным адресом узел обращается к выделенному маршрутизатору с запросом и автоматически получает ответ без участия администратора.
3. **Что представляет собой протокол RARP?**Протокол RARP — это протокол, решающий обратную задачу — нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP — RARP (Reverse Address Resolution Protocol). Reverse ARP (или обратное разрешение) работает аналогично протоколу ARP за исключением того, что в его задачи входит определение физического адреса по известному адресу сетевого уровня. Этот протокол требует наличия в сети сервера RARP, подключенного к тому же сегменту сети, что и интерфейс маршрутизатора.
4. **В каких целях может быть использован протокол RARP?**Он используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера. Наиболее часто протокол Reverse ARP используется для запуска бездисковых рабочих станций.