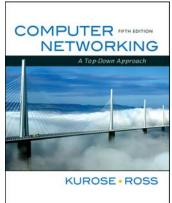
Wireshark Lab: Ethernet and ARP



Computer Networking: A Topdown Approach, 5th edition.

Version: 2.0 © 2009 J.F. Kurose, K.W. Ross. All Rights Reserved

In this lab, we'll investigate the Ethernet protocol and the ARP protocol. Before beginning this lab, you'll probably want to review sections 5.5 (Ethernet), 5.4.1 (link-layer addressing) and 5.4.2 (ARP) in the text. RFC 826 (<u>ftp://ftp.rfc-editor.org/in-notes/std/std37.txt</u>) contains the gory details of the ARP protocol, which is used by an IP device to determine the IP address of a remote interface whose Ethernet address is known.

1. Capturing and analyzing Ethernet frames

Let's begin by capturing a set of Ethernet frames to study. Do the following¹:

- First, make sure your browser's cache is empty. To do this under Mozilla Firefox V3, select *Tools->Clear Private Data* and check the box for Cache. For Internet Explorer, select *Tools->Internet Options->Delete Files*. Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-ethereal-lab-file3.html Your browser should display the rather lengthy US Bill of Rights.

¹ If you are unable to run Wireshark live on a computer, you can download the zip file

<u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u> and extract the file *ethernet--ethereal-trace-1*. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the ethernet-ethereal-trace-1 trace file. You can then use this trace file to answer the questions below.

• Stop Wireshark packet capture. First, find the packet numbers (the leftmost column in the upper Wireshark window) of the HTTP GET message that was sent from your computer to gaia.cs.umass.edu, as well as the beginning of the HTTP response message sent to your computer by gaia.cs.umass.edu. You should see a screen that looks something like this (where packet 4 in the screen shot below contains the HTTP GET message)

🗖 (Un	titled) - Wir	eshark																	[- 🗆 ×
Eile	<u>E</u> dit	<u>V</u> iew	<u>G</u> o <u>C</u> a	apture	<u>A</u> nalyze	<u>S</u> tatisti	ics <u>F</u>	<u>t</u> elp													
	4	0	@		D		×	¢,	—	9	4	⇔	Ŵ	æ :	⊉		3	•	Θ,	0	**
Eilter:											•	<u>E</u> xpressio	n <u>⊂</u> lea	ar <u>A</u> pp	oly						
No. +	1	ïme		Source	e			Destir	nation			Protoco	ol Info								<u>_</u>
		000			.168.2 .119.2					245.1 2.145		TCP TCP					Seq= ACK]				
	3 0	.050	729	192.	.168.2	.145		128.	119.	245.1	2	TCP	2038	8 > h'	ttp	[ACK]	Seq=	1 Ack	=1 Wi	n=65	535
		.055 .128			.168.2 .119.2					245.1 2.145		HTTP TCP					lbs/HT Seq=				
	6 0	.134	167	128.	.119.2	45.12		192.	168.	2.145		TCP	[тсі	P seg	ment	of a	i reas	sembl	ed PC	0]	
).150). 150			.119.2 .168.2					2.145 245.1		TCP TCP	203				reas				in=6
).213			.119.2 .119.2					2.145		TCP TCP					neas ment				nont
	11 ().215).215	947	192.	.168.2	.145		128.	119.	245.1	.2	TCP	2038	8 > h	ttp	[ACK]	Seq=	453 Ā	ck=32	14 Ŵ	in=6
).231).232			.119.2 .168.2					2.145 245.1		HTTP TCP					ion] H Seq=				
	14 C	.320	470	192.	.168.2	.145		128.	119.	245.1	.2	HTTP	GET	/fav	icon	.ico	HTTP/	1.1			
).403).423			.119.2 .168.2					2.145 2.224		HTTP TCP					Found Seg=				60
	17 C	.579	522	192.	.168.2	.145		128.	119.	245.1	.2	TCP	2038	8 > h	ttp	[ACK]	Seq=	793 A	ck=62	35 W	in=6
		.383 .392			.168.2 .168.2					.2.224 .2.224		TCP TCP					Seq= Seq=				
	201	0.38	9131	128.	.119.2	45.12		192.	168.	2.145		TCP	http	p > 2	038	[FIN,	ACK]	Seq=	6235	Ack=	793 '
	<u> </u>	.0.38	9210	192,	.168.2	.14]		120.	TTA.	245.1	2	ТСР	2038	8 > h	стр	[ACK]	seq=	793 A	CK=02	.50 W	
∢					<u> </u>																▶
					n wire taear						a.ed) Dst	: ∟ink	SVSG	45.0	au • a 8	(00.1	16.41	45.9	0.98)	
													128.1							0.00)	
							snc	Port	: 203	38 (20	38),	DSt P	ort: h	nttp ((80),	seq	: 1,)	Ack: 1	., Le	n: 45	2
					rotoco - /utto		7	7	641.	.	1	no /1 -1	\\								
			esnari t Metl		S/HTTP GET	-ethe	real	- lab-	-1116	23.110	II HI	19/1.1	Ar. An								
					reshar	k-lab:	s/HT	тр-е	there	eal-la	b-fi	le3.ht	ml								
					HTTP/																
					s.edu∖ ≥∕s_o		owe •		vinda	NVE NT	- 5 1	- an-U	s; rv:	1 0 1	- AN	cock	o /2001	70515	Fire	fov /7	0.0.
													xt/htm								
		•			n-us,e			•••		,		,	,	, -, -	,	,		, -,	,	5-, 1	5, ,
					zip,de																
			harse ve: 3		0-8859 n	-1,ut	F-8;	q=0.	7,*;c	¦=0.7∖	∖r∖n										
					⊓ live∖r	١n															
	\r\r																				
ब																					•
			n in	~ ~			20.0		-05	/2 UJ				7.811							
0040	68 74	61 7 68 6		2d 6d 65 61	: 61 6. . 6c 2(2 73 d 6c		48 54 52 2c		50 2d 69 6c		hark- there	-lab s, eal- la	/HTTP ab_fi	-е 1е						
0060	33	Ze 6	58 74	6d 6c	: 20 48	3 54	54 !	50 2f	31	2e 31	0d	3.htm	n] H T'	TP/1.∶	1.						
0070	0a 6d	48 6	5f 73 73 73	74 3a 2e 65	1 20 61 64 71		69 0 0a !		63 65	73 2e 72 2d	75 41	Host mass.	t:ga .edu	ia.cs .User	.u -A						
0090) 4d 6			5c 6c			Że	gent	Mo z	illa/	5.						
												P: 21 D: 3	21 M: 0 Dr	rops: 0							

• Since this lab is about Ethernet and ARP, we're not interested in IP or higherlayer protocols. So let's change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select *Analyze->Enabled Protocols*. Then uncheck the IP box and select *OK*. You should now see an Wireshark window that looks like:

_			reshar			Analy			ine	Uele																		<u>_ ×</u>
File	Edit	⊻iew	<u>G</u> o	⊆apti		<u>A</u> naly	20 . >	<u>5</u> tatist	×	Help ¢კ		<u> </u>	Q	à	4	\$	4	8	ዥ	ļ	ŀ			3		€,	Q	0
Eilter:														-	ΓĒ	xpress	ion	. <u>c</u> i								-	-	-
No. +	<u>,</u>	ïme			Sourc					Des	;tinati	00		_		Proto	rol	Info										<u> </u>
			0000				_61:	8e:	6d	_		/sG_	45:	90:	a8	0×08			_	_	_		_					
	3 0).050).050).055	729	1	Netg	jear.	_61:	5:90 8e: 8e:	6d	Li	nkīsy		45:	90:	a8	0×08 0×08 0×08	300	IΡ										
	5 C).128	3700	l	∟ink	sys	5_45	5:90	:a8	Ne	tgea	ir_6	1:8	e:6	d	0×08	300	IΡ										
).134).150						5:90 5:90				ar_6 ar_6				30×0 0×08												
		.150		1	Neto	jear.	_61:	8e:	6d	Li	nƙsy	/sG_	45:	90:	a8	0x08												
).213).219						5:90 5:90				ar_6 ar_6				0×08 0×08												
).215						8e:				/SG_																
).231).232						5:90 8e:				ar_6 /sG_				30×0 30×0												
		.320		1	Netō	jear.	_61:	8e:	6d	Li	nksj	/sG_	45:	90:	a8	0×08	300	IΡ										
).403).423						5:90 8e:				ar_6 /sG_				30×0 30×0												
		.579		1	Netō	jear.	_61:	8e:	6d	Li	nksj	/sG_	45:	90:	a8	0×08	300	IΡ										
		1.383 1.392						8e: 8e:				/sG_ /sG_				0×08 0×08												
			89131 89258	l	Link	sys	5_45	5:90 8e:	:a8	Ne	tgea	ar_6	1:8	e:6	d	0×08 0×08												
4																												
⊞ Fr	ame	4 (5	06 b	yte:	s or	ı wi	re,	506	byt	es	capt	ture	ed)															<u></u>
🗆 Et						-				-					-	, Ds	t:	Lin	iksy:	sG_	45:	90:8	a8	(00)	:0c	41:	45:90):a8)
			ion:												0													
L			s: L												ual	add	res	5 (uni	cas	tι							
			.0.																			ory	de	fau	lt)			
E :			Netg												-							-						
			s: N												_													
			0 .0.																			orv	de	faul	1+2			
· ·			· (0x			••	• • • •	• • •	••••	- LG	D10	G	100	aii	yu	niqu	e a	aur	255	C)	act	JUNY	ue	au	, c)			
			byte																									-
4	_			_		_		_	_	_	_	_	_		_			_	_			_		_	_	_		
0000			41 45 87 e9									8 01 2 91				AE												A
0020	f5	0c	07 f0	5 00) 50	7a	74	⊂4	58	7d a	a6 2	7 9	0 50) 18	3		. P2	zt .	.×}.	÷.1	۰.							_
0030			3a 90 72 6b							2f : 48 !		97) 45(:. hark												
0050	74	68	65 72	2 65	i 61	6c	2d	6c	61	62 2	2d 6	6 6	9 60	: 65	5	ther	'ea'	1- 1	l'ab-	fi	le							
0060			68 74 6f 73					54 61				1 20 3 73				3.ht .Hos												-
, File: "C	\DOC	UME~	1\PAUL	AW~:	1\LOC	ALS~	1\Tem	np\eth	erXXX	Xa026	520" 8	584 B	ytes I	00:	P:			-										

In order to answer the following questions, you'll need to look into the packet details and packet contents windows (the middle and lower display windows in Wireshark).

Select the Ethernet frame containing the HTTP GET message. (Recall that the HTTP GET message is carried inside of a TCP segment, which is carried inside of an IP datagram, which is carried inside of an Ethernet frame; reread section 1.5.2 in the text if you find this nesting a bit confusing). Expand the Ethernet II information in the packet details window. Note that the contents of the Ethernet frame (header as well as payload) are displayed in the packet contents window.

Answer the following questions, based on the contents of the Ethernet frame containing the HTTP GET message. Whenever possible, when answering a question you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout to explain your answer. To print a packet, use *File->Print*, choose *Selected packet only*, choose *Packet summary line*, and select the minimum amount of packet detail that you need to answer the question.

- 1. What is the 48-bit Ethernet address of your computer?
- 2. What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is *no*). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 468-469 in the text and make sure you understand the answer here.]
- 3. Give the hexadecimal value for the two-byte Frame type field. What do the bit(s) whose value is 1 mean within the flag field?
- 4. How many bytes from the very start of the Ethernet frame does the ASCII "G" in "GET" appear in the Ethernet frame?
- 5. What is the hexadecimal value of the CRC field in this Ethernet frame?

Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

- 6. What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is *no*). What device has this as its Ethernet address?
- 7. What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?
- 8. Give the hexadecimal value for the two-byte Frame type field. What do the bit(s) whose value is 1 mean within the flag field?
- 9. How many bytes from the very start of the Ethernet frame does the ASCII "O" in "OK" (i.e., the HTTP response code) appear in the Ethernet frame?
- 10. What is the hexadecimal value of the CRC field in this Ethernet frame?

2. The Address Resolution Protocol

In this section, we'll observe the ARP protocol in action. We strongly recommend that you re-read section 5.4.2 in the text before proceeding.

ARP Caching

Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your computer The *arp* command (in both MSDOS and Linux/Unix) is used to view and manipulate the contents of this cache. Since the *arp* command and the ARP protocol have the same name, it's understandably easy to confuse them. But keep in mind that they are different - the *arp* command is used to view and manipulate the ARP protocol defines the format and meaning of the messages sent and received, and defines the actions taken on message transmission and receipt.

Let's take a look at the contents of the ARP cache on your computer:

- **MS-DOS.** The *arp* command is in c:\windows\system32, so type either "*arp*" or "*c:\windows\system32\arp*" in the MS-DOS command line (without quotation marks).
- Linux/Unix. The executable for the *arp* command can be in various places. Popular locations are /sbin/arp (for linux) and /usr/etc/arp (for some Unix variants).

The *arp* command with no arguments will display the contents of the ARP cache on your computer. Run the *arp* command.

11. Write down the contents of your computer's ARP cache. What is the meaning of each column value?

In order to observe your computer sending and receiving ARP messages, we'll need to clear the ARP cache, since otherwise your computer is likely to find a needed IP-Ethernet address translation pair in its cache and consequently not need to send out an ARP message.

- **MS-DOS.** The MS-DOS *arp* –*d* * command will clear your ARP cache. The –*d* flag indicates a deletion operation, and the * is the wildcard that says to delete all table entries.
- Linux/Unix. The *arp* –*d* * will clear your ARP cache. In order to run this command you'll need root privileges. If you don't have root privileges and can't run Wireshark on a Windows machine, you can skip the trace collection part of this lab and just use the trace discussed in footnote 1.

Observing ARP in action

Do the following²:

- Clear your ARP cache, as described above.
- Next, make sure your browser's cache is empty. (To do this under Netscape 7.0, select *Edit->Preferences->Advanced->Cache* and clear the memory and disk cache. For Internet Explorer, select *Tools->Internet Options->Delete Files.*)
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-lab-file3.html Your browser should again display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture. Again, we're not interested in IP or higher-layer protocols, so change Wireshark's "listing of captured packets" window so that it shows information only about protocols below IP. To have Wireshark do this, select *Analyze->Enabled Protocols*. Then uncheck the IP box and select *OK*. You should now see an Wireshark window that looks like:

² The *ethernet-ethereal-trace-1* trace file in <u>http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip</u> was created using the steps below (in particular after the ARP cache had been flushed).

📶 eth	erne	ethe	real-tra	ice-1 - '	Wiresha	rk															- D X
Eile	<u>E</u> dit	<u>V</u> iew	<u>G</u> o ⊆	apture	<u>A</u> nalyze	<u>S</u> tatis	itics <u>F</u>	<u>t</u> elp										1			
	ë.	0	e i	M	D	K	×	e9	<u> </u>	٩	4	⇔	¢	Ŧ	⊉)	•	Θ	Ū,	**
<u>F</u> ilter:											▼ <u>E</u>	xpression	<u>⊂</u> l	ear <u>A</u>	pply						
No. +		lime -		Sour				Destin				Protocol									<u> </u>
	2 (3 4 5 8 5 8 7 1 8 1 9 1 11 1 12 1 13 1 14 1 15 1 16 1	.7.52	018 028 850 488 2974 4423 5902 5927 6468 8935 0025 0025 0069 7057 7422	Lin Amb Amb Tel Amb Lin Amb Lin Lin Lin Lin	it́Mic_ ksysG_ ksysG_	da:af a9:30 a9:30 a9:30 3:8d: a9:30 da:af da:af da:af da:af da:af da:af da:af	:73 1:68 1:68 1:68 1:68 1:68 1:68 1:68 1:68	Ambi Link Link Broa Link Ambi Link Ambi Ambi Link Ambi Ambi	tMic_ sysG_ sysG_ sysG_ tMic_ sysG_ tMic_ tMic_ tMic_ tMic_ tMic_ tMic_	a9:3c da:af da:af da:af da:af da:af da:af da:af a9:3c da:af a9:3c da:af a9:3c	73 73 73 73 73 68 73 68 68 68 68 68 68 68 68 68 68 68 68 68	0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080 0x080	192 0 IP 0 IP 0 IP 0 IP 0 IP 0 IP 0 IP 0 IP	2.168	3.1.1	is a	1.1? t 00:'	06:25	:da:a	f:73	.1.104
T	<pre>15 17.527057 LinksysG_da:af:73 AmbitMic_a9:3d:68 0x0800 IP 16 17.527422 LinksysG_da:af:73 AmbitMic_a9:3d:68 0x0800 IP 17 17.527457 AmbitMic_a9:3d:68 LinksysG_da:af:73 0x0800 IP</pre> Image: Frame 1 (42 bytes on wire, 42 bytes captured) Image: Frame 1 (42 bytes on wire, 42 bytes captured) Image: Ethernet II, Src: AmbitMic_a9:3d:68 (00:d0:59:a9:3d:68), Dst: Broadcast (ff:ff:ff:ff:ff:ff) Image: Address Resolution Protocol (request) Hardware type: Ethernet (0x0001) Protocol type: IP (0x0800) Hardware size: 6 Protocol size: 4 Opcode: request (0x0001) Sender MAC address: AmbitMic_a9:3d:68 (00:d0:59:a9:3d:68) Sender IP address: 192.168.1.105 (192.168.1.105) Target IP address: 192.168.1.1 (192.168.1.1)																				
0000 0010 0020		00 1	06 04	00 01	= 00 di L 00 di) c0 ai	0 59		3d 68 3d 68						(.=h. (.=h.							
File: "C	:\Doc	uments	and Sett	ings\Pa	ula Wing\M	ly Docur	ments\\	Wireshar	k\traces	- ethere	eal(P: 17 D:	17 M: 0)							11.

In the example above, the first two frames in the trace contain ARP messages (as does the 6^{th} message). The screen shot above corresponds to the trace referenced in footnote 1.

Answer the following questions:

- 12. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?
- 13. Give the hexadecimal value for the two-byte Ethernet Frame type field. What do the bit(s) whose value is 1 mean within the flag field?
- 14. Download the ARP specification from <u>ftp://ftp.rfc-editor.org/in-notes/std/std37.txt</u>. A readable, detailed discussion of ARP is also at <u>http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html</u>.
 - a) How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?
 - b) What is the value of the *opcode* field within the ARP-payload part of the Ethernet frame in which an ARP request is made?
 - c) Does the ARP message contain the IP address of the sender?

- d) Where in the ARP request does the "question" appear the Ethernet address of the machine whose corresponding IP address is being queried?
- 15. Now find the ARP reply that was sent in response to the ARP request.
 - a) How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?
 - b) What is the value of the *opcode* field within the ARP-payload part of the Ethernet frame in which an ARP response is made?
 - c) Where in the ARP message does the "answer" to the earlier ARP request appear the IP address of the machine having the Ethernet address whose corresponding IP address is being queried?
- 16. What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?
- 17. Open the *ethernet-ethereal-trace-1* trace file in
 - http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip. The first and second ARP packets in this trace correspond to an ARP request sent by the computer running Wireshark, and the ARP reply sent to the computer running Wireshark by the computer with the ARP-requested Ethernet address. But there is yet another computer on this network, as indiated by packet 6 another ARP request. Why is there no ARP reply (sent in response to the ARP request in packet 6) in the packet trace?

Extra Credit

EX-1. The *arp* command:

arp -s InetAddr EtherAddr

allows you to manually add an entry to the ARP cache that resolves the IP address *InetAddr* to the physical address *EtherAddr*. What would happen if, when you manually added an entry, you entered the correct IP address, but the wrong Ethernet address for that remote interface?

EX-2. What is the default amount of time that an entry remains in your ARP cache before being removed. You can determine this empirically (by monitoring the cache contents) or by looking this up in your operation system documentation. Indicate how/where you determined this value.