|  |  |
| --- | --- |
| **C:\Users\Fer\Desktop\Mario\Diseños\ipn.png** | **INSTITUTO POLITÉCNICO NACIONAL**  **ESCUELA SUPERIOR DE CÓMPUTO**  **INGENIERÍA EN SISTEMAS COMPUTACIONALES**  Redes de Computadoras |
|  | Resultado de imagen para redes |
| Grupo: 2CM9 | Práctica 01: Análisis de checksum |
|  | **Integrantes:**   * Díaz Medina Jesús Kaimorts. * Esquivel Valdez Alberto. * Vargas Romero Erick Efraín.   **Profesor:** Moreno Cervantes Axel Ernesto  **Fecha de realización:**  **Fecha de entrega:** 12 de septiembre del 2017 |

**REDES DE COMPUTADORAS**

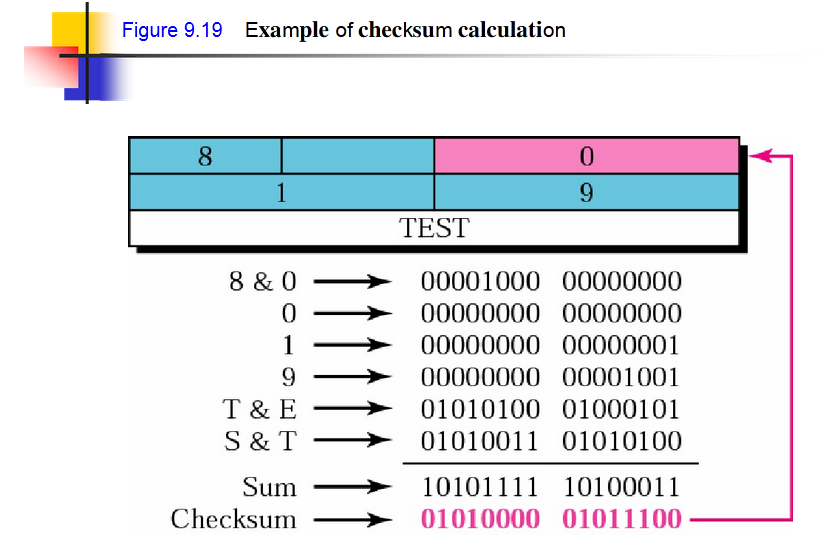
**PRÁCTICA 01: Análisis de Checksum**

**Introducción**

Checksum:

Es una suma de verificación, también conocida como checksum o suma de chequeo que permite la verificación de los datos que se envían a través de las tramas de TCP/IP. Si el 'checksum' que se genera al leer los datos no coincide con el que hay grabado al final, normalmente significa que ha habido un error de lectura. El 'checksum' es rápido de calcular, es transparente a la pérdida de bits (rotación de octetos aparentes de datos) y es efectivo ante los grupos de bits a ceros o a unos.

En ocasiones, los paquetes de datos al moverse de un nodo a otro pueden alterarse, generando errores en la información que transportan. Este tipo de errores deben evitarse para que la información que transporte la red sea confiable. Sin esta confiabilidad las personas (los usuarios) no utilizarían las redes.



*Ejemplo de cómo se calcula el checksum.*

La idea básica detrás de los esquemas de detección de errores en redes es adicionar información redundante al paquete, de tal forma que permita determinar si un error ha sido introducido mientras se llevaba de un nodo a otro. Un esquema de detección de errores que podemos imaginar es la transmisión completa de dos copias del mismo paquete. Si las dos copias son idénticas cuando lleguen al receptor, es muy probable que la información esté correcta. Si no son iguales, un error fue introducido en una de las copias (o en ambas) y deben ser descartadas. Este esquema de detección de errores es deficiente: se requieren n bits de correción de errores para un mensaje de n bits y además, si por alguna razón, los errores ocurren en el mismo lugar en los dos paquetes, no serán detectados.

Para la realización de la práctica tomamos como referencia el encabezado de IP:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1-4 | Versión | IHL | Serv. Def. | Longitud | | | |
| 5-8 | ID | | | 0 | X | Y | Offset |
| 9-12 | TTL | Protocolo | | **checksum** | | | |
| 13-16 | IP origen | | | | | | |
| 17-20 | IP destino | | | | | | |
|  | Opciones | | | | | | |

Los números que se encuentran de lado izquierdo son los bytes que le corresponden a cada parte del encabezado.

**Desarrollo**

1. Descargar e instalar la biblioteca jnetpcap desde:

148.204.58.221/axel/redesnp/sniffer/jnetpcap/

* jnetpcap-1.3.0-1 win 64....zip
* jnetpcap-javadoc-1.3.0....zip
* jnetpcap-src-1.3.0...zip

(Quizá haya que descargar e instalar también la biblioteca winpcap de www.winpcap.com)

\*Para la instalación leer el archivo: instalacion\_netbeans.pdf

1. Descargar y probar el programa Captura.java

|  |
| --- |
| 1. **import** java**.**util**.**ArrayList**;** 2. **import** java**.**util**.**Date**;** 3. **import** java**.**util**.**List**;** 4. **import** java**.**io**.\*;** 5. **import** org**.**jnetpcap**.**Pcap**;** 6. **import** org**.**jnetpcap**.**PcapIf**;** 7. **import** org**.**jnetpcap**.**packet**.**PcapPacket**;** 8. **import** org**.**jnetpcap**.**packet**.**PcapPacketHandler**;** 9. **import** org**.**jnetpcap**.**PcapBpfProgram**;** 10. **import** org**.**jnetpcap**.**protocol**.**lan**.**Ethernet**;** 11. **import** org**.**jnetpcap**.**protocol**.**tcpip**.\*;** 12. **import** org**.**jnetpcap**.**protocol**.**network**.\*;** 13. **import** org**.**jnetpcap**.**nio**.**JBuffer**;** 14. **import** org**.**jnetpcap**.**packet**.**Payload**;** 15. **import** org**.**jnetpcap**.**protocol**.**network**.**Arp**;** 16. **import** org**.**jnetpcap**.**protocol**.**lan**.**IEEE802dot2**;** 17. **import** org**.**jnetpcap**.**protocol**.**lan**.**IEEE802dot3**;** 18. public class Captura **{** 19. /\*\* 20. Main startup method 21. \* 22. **@param** args 23. ignored 24. \*/ 25. private static String asString**(**final byte**[]** mac**)** **{** 26. final StringBuilder buf **=** **new** StringBuilder**();** 27. **for** **(**byte b **:** mac**)** **{** 28. **if** **(**buf**.**length**()** **!=** 0**)** **{** 29. buf**.**append**(**':'**);** 30. **}** 31. **if** **(**b **>=** 0 **&&** b **<** 16**)** **{** 32. buf**.**append**(**'0'**);** 33. **}** 34. buf**.**append**(**Integer**.**toHexString**((**b **<** 0**)** **?** b **+** 256 **:** b**).**toUpperCase**());** 35. **}** 36. **return** buf**.**toString**();** 37. **}** 38. public static void main**(**String**[]** args**)** **{** 39. List**<**PcapIf**>** alldevs **=** **new** ArrayList**<**PcapIf**>();** // Will be filled with NICs 40. StringBuilder errbuf **=** **new** StringBuilder**();** // For any error msgs 41. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 42. First get a list of devices on this system 43. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 44. int r **=** Pcap**.**findAllDevs**(**alldevs**,** errbuf**);** 45. **if** **(**r **==** Pcap**.**NOT\_OK **||** alldevs**.**isEmpty**())** **{** 46. System**.**err**.**printf**(**"Can't read list of devices, error is %s"**,** errbuf 47. **.**toString**());** 48. **return;** 49. **}** 50. System**.**out**.**println**(**"Network devices found:"**);** 51. int i **=** 0**;** 52. **try{** 53. **for** **(**PcapIf device **:** alldevs**)** **{** 54. String description **=** 55. **(**device**.**getDescription**()** **!=** **null)** **?** device**.**getDescription**()** 56. **:** "No description available"**;** 57. final byte**[]** mac **=** device**.**getHardwareAddress**();** 58. String dir\_mac **=** **(**mac**==null)?**"No tiene direccion MAC"**:**asString**(**mac**);** 59. System**.**out**.**printf**(**"#%d: %s [%s] MAC:[%s]\n"**,** i**++,** device**.**getName**(),** description**,** dir\_mac**);** 60. **}**//for 61. PcapIf device **=** alldevs**.**get**(**0**);** // We know we have atleast 1 device 62. System**.**out 63. **.**printf**(**"\nChoosing '%s' on your behalf:\n"**,** 64. **(**device**.**getDescription**()** **!=** **null)** **?** device**.**getDescription**()** 65. **:** device**.**getName**());** 66. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 67. Second we open up the selected device 68. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 69. /\*"snaplen" is short for 'snapshot length', as it refers to the amount of actual data captured from each packet passing through the specified network interface. 70. 64\*1024 = 65536 bytes; campo len en Ethernet(16 bits) tam mÃ¡x de trama \*/ 71. int snaplen **=** 64 **\*** 1024**;** // Capture all packets, no trucation 72. int flags **=** Pcap**.**MODE\_PROMISCUOUS**;** // capture all packets 73. int timeout **=** 10 **\*** 1000**;** // 10 seconds in millis 74. Pcap pcap **=** 75. Pcap**.**openLive**(**device**.**getName**(),** snaplen**,** flags**,** timeout**,** errbuf**);** 76. **if** **(**pcap **==** **null)** **{** 77. System**.**err**.**printf**(**"Error while opening device for capture: " 78. **+** errbuf**.**toString**());** 79. **return;** 80. **}**//if 81. /\*\*\*\*\*\*\*\*F I L T R O\*\*\*\*\*\*\*\*/ 82. PcapBpfProgram filter **=** **new** PcapBpfProgram**();** 83. String expression **=**""**;** // "port 80"; 84. int optimize **=** 0**;** // 1 means true, 0 means false 85. int netmask **=** 0**;** 86. int r2 **=** pcap**.**compile**(**filter**,** expression**,** optimize**,** netmask**);** 87. **if** **(**r2 **!=** Pcap**.**OK**)** **{** 88. System**.**out**.**println**(**"Filter error: " **+** pcap**.**getErr**());** 89. **}**//if 90. pcap**.**setFilter**(**filter**);** 91. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 92. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 93. Third we create a packet handler which will receive packets from the 94. libpcap loop. 95. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 96. PcapPacketHandler**<**String**>** jpacketHandler **=** **new** PcapPacketHandler**<**String**>()** **{** 97. public void nextPacket**(**PcapPacket packet**,** String user**)** **{** 98. System**.**out**.**printf**(**"Received packet at %s caplen=%-4d len=%-4d %s\n"**,** 99. **new** Date**(**packet**.**getCaptureHeader**().**timestampInMillis**()),** 100. packet**.**getCaptureHeader**().**caplen**(),** // Length actually captured 101. packet**.**getCaptureHeader**().**wirelen**(),** // Original length 102. user // User supplied object 103. **);** 104. /\*\*\*\*\*\*Desencapsulado\*\*\*\*\*\*\*\*/ 105. **for(**int i**=**0**;**i**<**packet**.**size**();**i**++){** 106. System**.**out**.**printf**(**"%02X "**,**packet**.**getUByte**(**i**));** 107. **if(**i**%**16**==**15**)** 108. System**.**out**.**println**(**""**);** 109. **}** 110. System**.**out**.**println**(**"\n\nEncabezado: "**+** packet**.**toHexdump**());** 111. **}** 112. **};** 113. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 114. Fourth we enter the loop and tell it to capture 10 packets. The loop 115. method does a mapping of pcap.datalink() DLT value to JProtocol ID, which 116. is needed by JScanner. The scanner scans the packet buffer and decodes 117. the headers. The mapping is done automatically, although a variation on 118. the loop method exists that allows the programmer to sepecify exactly 119. which protocol ID to use as the data link type for this pcap interface. 120. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 121. pcap**.**loop**(**10**,** jpacketHandler**,** "jNetPcap rocks!"**);** 122. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 123. Last thing to do is close the pcap handle 124. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 125. pcap**.**close**();** 126. **}catch(**IOException e**){**e**.**printStackTrace**();}** 127. **}** 128. **}** |

*Código del programa Captura.java*

|  |
| --- |
|  |

*Prueba del programa Captura.java*

1. Descargar el archivo Checksum.zip y probar el programa

|  |
| --- |
| 1. package practica\_01**;** 2. public class Checksum **{** 3. /\*\* 4. \* Calculate the Internet Checksum of a buffer (RFC 1071 - http://www.faqs.org/rfcs/rfc1071.html) 5. \* Algorithm is 6. \* 1) apply a 16-bit 1's complement sum over all octets (adjacent 8-bit pairs [A,B], final odd length is [A,0]) 7. \* 2) apply 1's complement to this final sum 8. \* 9. \* Notes: 10. \* 1's complement is bitwise NOT of positive value. 11. \* Ensure that any carry bits are added back to avoid off-by-one errors 12. \* 13. \* 14. \* **@param** buf The message 15. \* **@return** The checksum 16. \*/ 17. public static long calculateChecksum**(**byte**[]** buf**)** **{** 18. int length **=** buf**.**length**;** 19. int i **=** 0**;** 20. long sum **=** 0**;** 21. long data**;** 22. // Handle all pairs 23. **while** **(**length **>** 1**)** **{** 24. // Corrected to include @Andy's edits and various comments on Stack Overflow 25. data **=** **(((**buf**[**i**]** **<<** 8**)** **&** 0xFF00**)** **|** **((**buf**[**i **+** 1**])** **&** 0xFF**));** 26. sum **+=** data**;** 27. // 1's complement carry bit correction in 16-bits (detecting sign extension) 28. **if** **((**sum **&** 0xFFFF0000**)** **>** 0**)** **{** 29. sum **=** sum **&** 0xFFFF**;** 30. sum **+=** 1**;** 31. **}** 32. i **+=** 2**;** 33. length **-=** 2**;** 34. **}** 35. // Handle remaining byte in odd length buffers 36. **if** **(**length **>** 0**)** **{** 37. // Corrected to include @Andy's edits and various comments on Stack Overflow 38. sum **+=** **(**buf**[**i**]** **<<** 8 **&** 0xFF00**);** 39. // 1's complement carry bit correction in 16-bits (detecting sign extension) 40. **if** **((**sum **&** 0xFFFF0000**)** **>** 0**)** **{** 41. sum **=** sum **&** 0xFFFF**;** 42. sum **+=** 1**;** 43. **}** 44. **}** 45. // Final 1's complement value correction to 16-bits 46. sum **=** **~**sum**;** 47. sum **=** sum **&** 0xFFFF**;** 48. **return** sum**;** 49. **}** 51. public static void main**(**String**[]** args**){** 52. byte**[]** buf **=** **{(**byte**)** 0x45**,(**byte**)** 0x00**,(**byte**)** 0x00**,** **(**byte**)** 0x3c**,** **(**byte**)** 0x0a**,** **(**byte**)** 0x1c**,** **(**byte**)** 0x40**,** **(**byte**)** 0x00**,(**byte**)** 0xff**,** **(**byte**)**0x06**,** **(**byte**)**0x00**,** **(**byte**)** 0x00**,** **(**byte**)**0xa8**,** **(**byte**)**0xb0**,** **(**byte**)** 0x03**,** **(**byte**)** 0x19**,** **(**byte**)** 0xa8**,** **(**byte**)** 0xb0**,** **(**byte**)** 0x03**,** **(**byte**)** 0x6c**};** 53. long resultado **=** Checksum**.**calculateChecksum**(**buf**);** 54. System**.**out**.**printf**(**"Valos del check: %02X\n"**,**resultado**);** 55. **}**//main 56. **}** |

*Código checksum.java*

1. Modificar el programa Captura.java para que verifique el campo checksum de IP,TCP/UDP de las tramas capturadas.

UDP

|  |  |
| --- | --- |
| # Puerto origen | # Puerto destino |
| longitud | Checksum |

Comenzamos modificando el código de Captura.java

|  |
| --- |
| 1. package practica\_01**;** 2. /\*@author Home 3. \*/ 4. **import** java**.**util**.**ArrayList**;** 5. **import** java**.**util**.**Date**;** 6. **import** java**.**util**.**List**;** 7. **import** java**.**io**.\*;** 8. **import** java**.**nio**.**BufferUnderflowException**;** 9. **import** java**.**util**.**Arrays**;** 10. **import** java**.**util**.**jar**.**Pack200**;** 11. **import** practica\_01**.\*;** 12. **import** org**.**jnetpcap**.**Pcap**;** 13. **import** org**.**jnetpcap**.**PcapIf**;** 14. **import** org**.**jnetpcap**.**packet**.**PcapPacket**;** 15. **import** org**.**jnetpcap**.**packet**.**PcapPacketHandler**;** 16. **import** org**.**jnetpcap**.**PcapBpfProgram**;** 17. **import** org**.**jnetpcap**.**protocol**.**lan**.**Ethernet**;** 18. **import** org**.**jnetpcap**.**protocol**.**tcpip**.\*;** 19. **import** org**.**jnetpcap**.**protocol**.**network**.\*;** 20. **import** org**.**jnetpcap**.**nio**.**JBuffer**;** 21. **import** org**.**jnetpcap**.**packet**.**Payload**;** 22. **import** org**.**jnetpcap**.**protocol**.**network**.**Arp**;** 23. **import** org**.**jnetpcap**.**protocol**.**lan**.**IEEE802dot2**;** 24. **import** org**.**jnetpcap**.**protocol**.**lan**.**IEEE802dot3**;** 25. public class Captura **{** 26. /\*\* 27. Main startup method 28. \* 29. **@param** args ignored 30. \*/ 31. private static String asString**(**final byte**[]** mac**)** **{** 32. final StringBuilder buf **=** **new** StringBuilder**();** 33. **for** **(**byte b **:** mac**)** **{** 34. **if** **(**buf**.**length**()** **!=** 0**)** **{** 35. buf**.**append**(**':'**);** 36. **}** 37. **if** **(**b **>=** 0 **&&** b **<** 16**)** **{** 38. buf**.**append**(**'0'**);** 39. **}** 40. buf**.**append**(**Integer**.**toHexString**((**b **<** 0**)** **?** b **+** 256 **:** b**).**toUpperCase**());** 41. **}** 42. **return** buf**.**toString**();** 43. **}** 44. public static void main**(**String**[]** args**)** **{** 45. // Checksum trama = new Checksum(); 46. List**<**PcapIf**>** alldevs **=** **new** ArrayList**<**PcapIf**>();** // Will be filled with NICs 47. StringBuilder errbuf **=** **new** StringBuilder**();** // For any error msgs 48. /\*\* 49. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 50. First get a list of devices on this system 51. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 52. \*/ 53. int r **=** Pcap**.**findAllDevs**(**alldevs**,** errbuf**);** 54. **if** **(**r **==** Pcap**.**NOT\_OK **||** alldevs**.**isEmpty**())** **{** 55. System**.**err**.**printf**(**"Can't read list of devices, error is %s"**,** errbuf 56. **.**toString**());** 57. **return;** 58. **}** 59. System**.**out**.**println**(**"Network devices found:"**);** 60. int i **=** 0**;** 61. **try** **{** 62. **for** **(**PcapIf device **:** alldevs**)** **{** 63. String description 64. **=** **(**device**.**getDescription**()** **!=** **null)** **?** device**.**getDescription**()** 65. **:** "No description available"**;** 66. final byte**[]** mac **=** device**.**getHardwareAddress**();** 67. String dir\_mac **=** **(**mac **==** **null)** **?** "No tiene direccion MAC" **:** asString**(**mac**);** 68. System**.**out**.**printf**(**"#%d: %s [%s] MAC:[%s]\n"**,** i**++,** device**.**getName**(),** description**,** dir\_mac**);** 69. **}**//for 70. PcapIf device **=** alldevs**.**get**(**1**);** // We know we have atleast 1 device 71. System**.**out 72. **.**printf**(**"\nChoosing '%s' on your behalf:\n"**,** 73. **(**device**.**getDescription**()** **!=** **null)** **?** device**.**getDescription**()** 74. **:** device**.**getName**());** 75. /\*\* 76. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 77. Second we open up the selected device 78. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 79. \*/ 80. /\*"snaplen" is short for 'snapshot length', as it refers to the amount of actual data captured from each packet passing through the specified network interface. 81. 64\*1024 = 65536 bytes; campo len en Ethernet(16 bits) tam mÃ¡x de trama \*/ 82. int snaplen **=** 64 **\*** 1024**;** // Capture all packets, no trucation 83. int flags **=** Pcap**.**MODE\_PROMISCUOUS**;** // capture all packets 84. int timeout **=** 10 **\*** 1000**;** // 10 seconds in millis 85. Pcap pcap 86. **=** Pcap**.**openLive**(**device**.**getName**(),** snaplen**,** flags**,** timeout**,** errbuf**);** 87. **if** **(**pcap **==** **null)** **{** 88. System**.**err**.**printf**(**"Error while opening device for capture: " 89. **+** errbuf**.**toString**());** 90. **return;** 91. **}**//if 92. /\*\* 93. \*\*\*\*\*\*F I L T R O\*\*\*\*\*\*\* 94. \*/ 95. PcapBpfProgram filter **=** **new** PcapBpfProgram**();** 96. String expression **=** ""**;** // "port 80"; 97. int optimize **=** 0**;** // 1 means true, 0 means false 98. int netmask **=** 0**;** 99. int r2 **=** pcap**.**compile**(**filter**,** expression**,** optimize**,** netmask**);** 100. **if** **(**r2 **!=** Pcap**.**OK**)** **{** 101. System**.**out**.**println**(**"Filter error: " **+** pcap**.**getErr**());** 102. **}**//if 103. pcap**.**setFilter**(**filter**);** 104. /\*\* 105. \*\*\*\*\*\*\*\*\*\*\*\*\* 106. \*/ 107. /\*\* 108. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 109. Third we create a packet handler which will receive packets from 110. the libpcap loop. 111. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 112. \*/ 113. PcapPacketHandler**<**String**>** jpacketHandler **=** **new** PcapPacketHandler**<**String**>()** **{** 114. public void nextPacket**(**PcapPacket packet**,** String user**)** **{** 115. System**.**out**.**printf**(**"Received packet at %s caplen=%-4d len=%-4d %s\n"**,** 116. **new** Date**(**packet**.**getCaptureHeader**().**timestampInMillis**()),** 117. packet**.**getCaptureHeader**().**caplen**(),** // Length actually captured 118. packet**.**getCaptureHeader**().**wirelen**(),** // Original length 119. user // User supplied object 120. **);** 121. /\*\* 122. \*\*\*\*Desencapsulado\*\*\*\*\*\*\* 123. \*/ 124. **for** **(**int i **=** 0**;** i **<** packet**.**size**();** i**++)** **{** 125. System**.**out**.**printf**(**"%02X "**,** packet**.**getUByte**(**i**));** 126. **if** **(**i **%** 16 **==** 15**)** **{** 127. System**.**out**.**println**(**""**);** 128. **}** 129. **}** 130. System**.**out**.**println**(**"\n\nEncabezado: " **+** packet**.**toHexdump**());** 131. /\*-------------------------Verificacion de ETHERNET-------------------------\*/ 132. int tipoIP**;** 133. //tipoIP = (int) ((packet.getUByte(12)\*(Math.pow(2, 8))) + packet.getUByte(13)); 134. tipoIP **=** **(**packet**.**getUByte**(**12**)** **<<** 8 **|** packet**.**getUByte**(**13**));** 135. System**.**out**.**printf**(**"IP = %02X \n"**,(**byte**)**tipoIP**);** 136. /\*--------------------------------------------------------------------------\*/ 137. /\*-------------LONGITUD DEL PAQUETE (PDU DE IP)-----\*/ 138. byte**[]** longitudTrama **=** **new** byte**[**2**];** 139. longitudTrama**[**0**]** **=** **(**byte**)** **(**packet**.**getUByte**(**16**));** 140. longitudTrama**[**1**]** **=** **(**byte**)** **(**packet**.**getUByte**(**17**));** 141. /\*--------------------------------------------------\*/ 142. /\*------------------------CHECKSUM DE LA TRAMA--------------------------\*/ 143. byte**[]** checksum **=** **{(**byte**)**packet**.**getUByte**(**24**),(**byte**)**packet**.**getUByte**(**25**)};** 144. /\*---------------------------------------------------------------------\*/ 145. **if** **(**tipoIP **==** 0x086DD**)** **{** //IPV6 146. System**.**out**.**println**(**"IPV6: No contiene Checksum"**);** 147. **}** **else** **if** **(**tipoIP **==** 0x0800**)** **{** //IPV4 148. System**.**out**.**println**(**"IPV4"**);** 149. /\*Capturar bytes del encabezado IP. 150. Va del byte 14-33, son 20 bytes------------\*/ 151. byte**[]** IP **=** **new** byte**[**20**];** 152. **for** **(**int j **=** 14**,**i**=**0**;** j **<** 34**;**j**++)** 153. **{** 154. IP**[**i**]=(**byte**)**packet**.**getUByte**(**j**);** 155. **if(**j**==**24 **||** j**==**25**){** //Lugar del Checksum 156. IP**[**i**]** **=** **(**byte**)**0x00**;** 157. **}** 158. i**++;** 159. **}** 160. /\*Obtener Checksum\*/ 161. long checksumIP **=** Checksum**.**calculateChecksum**(**IP**);** 162. System**.**out**.**printf**(**"|-- Valor de Checksum IP: %02X --|\n"**,**checksumIP**);** 163. /\*------------------------------\*/ 164. int protocol **=** **(**packet**.**getUByte**(**23**));** 165. System**.**out**.**printf**(**"\tProtocolo: %02X\n"**,(**byte**)**protocol**);** 166. **if** **((**byte**)**protocol **==** 0x06**)** //PROTOCOLO TCP 167. **{** 168. System**.**out**.**printf**(**"PRTOCOLO TCP: %02X\n"**,(**byte**)**protocol**);** 169. System**.**out**.**println**(**"---PROTOCOLO TCP/IP---"**);** 170. /\*---------------------IHL: 45---------------------\*/ 171. byte IHL\_limpio **=** **(**byte**)** **(**packet**.**getUByte**(**14**)** **&** 0x0000000F**);** 172. int IHL **=** **(**IHL\_limpio **\*** 4**);** //Longitud del encabezado 173. /\*Solamente se requiere la longitud del encabezado: 5 174. La otra parte especifica la versión: 4\*/ 175. /\*---------------------------------------------\*/ 176. /\*---------------------PDU de IP---------------------\*/ 177. int longitud **=** **(**int**)** **((**packet**.**getUByte**(**16**)** **\*** **(**Math**.**pow**(**2**,** 8**)))** **+** **(**packet**.**getUByte**(**17**)));** 178. /\*---------------------------------------------------\*/ 179. /\*------------Longitud que irá al Pseudo Encabezado---------\*/ 180. int longitudTotal **=** longitud **-** IHL**;** 181. /\*----------------------------------------------------------\*/ 182. //Reservado para la longitud de encabezado (TCP O UDP) 183. byte**[]** pseudoEncabezado **=** **new** byte**[**IHL**];** 184. **for** **(**int j **=** 0**;** j **<** 4**;** j**++)** **{** 185. /\*----------------------IP ORIGEN----------------------\*/ 186. pseudoEncabezado**[**j**]** **=** **(**byte**)** **(**packet**.**getUByte**(**26 **+** j**));** 187. /\*-----------------------------------------------------\*/ 188. /\*---------------------IP DESTINO---------------------\*/ 189. pseudoEncabezado**[**j **+** 4**]** **=** **(**byte**)** **(**packet**.**getUByte**(**30 **+** j**));** 190. /\*----------------------------------------------------\*/ 191. **}** 192. pseudoEncabezado**[**8**]** **=** 0x00**;** 193. /\*-------------------PROTOCOLO (0x06)------------------\*/ 194. pseudoEncabezado**[**9**]** **=** **(**byte**)** **(**packet**.**getUByte**(**23**));** 195. /\*----------------------------------------------------\*/ 196. /\*-------------------LONGITUD-------------------------\*/ 197. pseudoEncabezado**[**10**]** **=** **(**byte**)**packet**.**getUByte**(**16**);** 198. pseudoEncabezado**[**11**]** **=** **(**byte**)(**packet**.**getUByte**(**17**)-**IHL**);** 199. /\*---------------------------------------------------\*/ 200. /\*El pseudoEncabezado representa la primer mitad para obtener el Checksum\*/ 201. /\*---La segunda mitad es representada por el PDU\_Trans---\*/ 202. byte**[]** PDU\_Trans **=** **new** byte**[**longitudTotal**];** 203. **for** **(**int k **=** 34**,**q**=**0**;** k **<** 34**+**longitudTotal**;** k**++,**q**++)** 204. **{** 205. PDU\_Trans**[**q**]=(**byte**)**packet**.**getUByte**(**k**);** 206. **}** 207. /\*------------------------------------------------------\*/ 208. /\*-----------UNION de PDU\_Trans con pseudoEncabezado----\*/ 209. int length\_pseudoE **=** pseudoEncabezado**.**length**;** 210. int length\_PDU\_Trans **=** PDU\_Trans**.**length**;** 211. byte**[]** encabezadoFinal**;** 212. encabezadoFinal**=** **new** byte**[**length\_pseudoE **+** length\_PDU\_Trans**];** 213. System**.**arraycopy**(**pseudoEncabezado**,** 0**,** encabezadoFinal**,** 0**,** length\_pseudoE**);** 214. System**.**arraycopy**(**PDU\_Trans**,** 0**,** encabezadoFinal**,** length\_pseudoE**,** length\_PDU\_Trans**);** 215. long checksumTCP **=** Checksum**.**calculateChecksum**(**encabezadoFinal**);** 216. System**.**out**.**printf**(**"Checksum del TCP: %04X \n"**,**checksumTCP**);** 217. /\*------------------------------------------------------\*/ 218. **}else** **if** **((**byte**)**protocol **==** 0x11**){** //PROTOCOLO UDP 219. System**.**out**.**printf**(**"\tPROTOCOLO UDP: %02X\n"**,(**byte**)**protocol**);** 220. /\*---------------------IHL: 45---------------------\*/ 221. byte IHL\_limpio **=** **(**byte**)** **(**packet**.**getUByte**(**14**)** **&** 0x0000000F**);** 222. int IHL **=** **(**IHL\_limpio **\*** 4**);** //Longitud del encabezado 223. /\*---------------------------------------------\*/ 224. /\*---------------------PDU de IP---------------------\*/ 225. int longitud **=** **(**int**)** **((**packet**.**getUByte**(**16**)** **\*** **(**Math**.**pow**(**2**,** 8**)))** **+** **(**packet**.**getUByte**(**17**)));** 226. /\*---------------------------------------------------\*/ 227. /\*------------Longitud que irá al Pseudo Encabezado---------\*/ 228. int longitudTotal **=** longitud **-** IHL**;** 229. /\*----------------------------------------------------------\*/ 230. //Reservado para la longitud de encabezado (TCP O UDP) 231. byte**[]** pseudoEncabezado **=** **new** byte**[**IHL**];** 232. **for** **(**int j **=** 0**;** j **<** 4**;** j**++)** **{** 233. /\*----------------------IP ORIGEN----------------------\*/ 234. pseudoEncabezado**[**j**]** **=** **(**byte**)** **(**packet**.**getUByte**(**26 **+** j**));** 235. /\*-----------------------------------------------------\*/ 236. /\*---------------------IP DESTINO---------------------\*/ 237. pseudoEncabezado**[**j **+** 4**]** **=** **(**byte**)** **(**packet**.**getUByte**(**30 **+** j**));** 238. /\*----------------------------------------------------\*/ 239. **}** 240. pseudoEncabezado**[**8**]** **=** 0x00**;** 241. /\*-------------------PROTOCOLO (0x11)------------------\*/ 242. pseudoEncabezado**[**9**]** **=** **(**byte**)** **(**packet**.**getUByte**(**23**));** 243. /\*----------------------------------------------------\*/ 244. /\*-------------------LONGITUD-------------------------\*/ 245. pseudoEncabezado**[**10**]** **=** **(**byte**)**packet**.**getUByte**(**16**);** 246. pseudoEncabezado**[**11**]** **=** **(**byte**)(**packet**.**getUByte**(**17**)-**IHL**);** 247. /\*---------------------------------------------------\*/ 248. /\*El pseudoEncabezado representa la primer mitad para obtener el Checksum\*/ 249. /\*---La segunda mitad es representada por el PDU\_Trans---\*/ 250. byte**[]** PDU\_Trans **=** **new** byte**[**longitudTotal**];** 251. **for** **(**int k **=** 34**,**q**=**0**;** k **<** 34**+**longitudTotal**;** k**++,**q**++)** 252. **{** 253. PDU\_Trans**[**q**]=(**byte**)**packet**.**getUByte**(**k**);** 254. **}** 255. /\*------------------------------------------------------\*/ 256. /\*-----------UNION de PDU\_Trans con pseudoEncabezado----\*/ 257. int length\_pseudoE **=** pseudoEncabezado**.**length**;** 258. int length\_PDU\_Trans **=** PDU\_Trans**.**length**;** 259. byte**[]** encabezadoFinal**;** 260. encabezadoFinal**=** **new** byte**[**length\_pseudoE **+** length\_PDU\_Trans**];** 261. System**.**arraycopy**(**pseudoEncabezado**,** 0**,** encabezadoFinal**,** 0**,** length\_pseudoE**);** 262. System**.**arraycopy**(**PDU\_Trans**,** 0**,** encabezadoFinal**,** length\_pseudoE**,** length\_PDU\_Trans**);** 263. long checksumUDP **=** Checksum**.**calculateChecksum**(**encabezadoFinal**);** 264. System**.**out**.**printf**(**"Checksum del TCP: %04X \n"**,**checksumUDP**);** 265. /\*------------------------------------------------------\*/ 266. **}** 267. **}** 268. **}** 269. **};** 270. /\*\* 271. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 272. Fourth we enter the loop and tell it to capture 10 packets. The 273. loop method does a mapping of pcap.datalink() DLT value to 274. JProtocol ID, which is needed by JScanner. The scanner scans the 275. packet buffer and decodes the headers. The mapping is done 276. automatically, although a variation on the loop method exists 277. that allows the programmer to sepecify exactly which protocol ID 278. to use as the data link type for this pcap interface. 279. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 280. \*/ 281. pcap**.**loop**(**10**,** jpacketHandler**,** "jNetPcap rocks!"**);** 282. /\*\* 283. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 284. Last thing to do is close the pcap handle 285. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 286. \*/ 287. pcap**.**close**();** 288. **}** **catch** **(**IOException e**)** **{** 289. e**.**printStackTrace**();** 290. **}** 291. **}** 292. **}** |

*Código Captura.java modificado*

La modificación consiste en lo siguiente:

1. Para la verificación de Ethernet obtuvimos el byte 12 y 13 de la trama y lo guardamos en una variable de tipo entero.

(En el byte 12 hicimos un corrimiento de 8 bits para que el bit de mayor y menor peso estuvieran en la posición correcta).

|  |
| --- |
| int tipoIP**;**  tipoIP **=** **(**packet**.**getUByte**(**12**)** **<<** 8 **|** packet**.**getUByte**(**13**));**  System**.**out**.**printf**(**"IP = %02X \n"**,(**byte**)**tipoIP**);** |

1. Para la longitud de la trama obtuvimos el byte 16 y 17 y lo guardamos en un arreglo de bytes.

|  |
| --- |
| byte**[]** longitudTrama **=** **new** byte**[**2**];**  longitudTrama**[**0**]** **=** **(**byte**)** **(**packet**.**getUByte**(**16**));**  longitudTrama**[**1**]** **=** **(**byte**)** **(**packet**.**getUByte**(**17**));** |

1. Detectamos la posición del checksum, que es el byte 24 y 25 de la trama y lo guardamos en otro arreglo de bytes.

|  |
| --- |
| byte**[]** checksum **=** **{(**byte**)**packet**.**getUByte**(**24**),(**byte**)**packet**.**getUByte**(**25**)};** |

1. Posteriormente comparamos la versión de IP para saber si tiene checksum o no, ya que en la versión 6 de IP no contamos con un checksum.

|  |
| --- |
| **if** **(**tipoIP **==** 0x086DD**)** **{** //IPV6  System**.**out**.**println**(**"IPV6: No contiene Checksum"**);**  **}** **else** **if** **(**tipoIP **==** 0x0800**)** **{** //IPV4 System**.**out**.**println**(**"IPV4"**);** |

1. Luego en otro arreglo almacenamos los bytes correspondientes al encabezado IP que van desde el byte 14 hasta el 23 siendo un total de 20 bytes.

|  |
| --- |
| byte**[]** IP **=** **new** byte**[**20**];**  **for** **(**int j **=** 14**,**i**=**0**;** j **<** 34**;**j**++)**  **{**  IP**[**i**]=(**byte**)**packet**.**getUByte**(**j**);**  **if(**j**==**24 **||** j**==**25**){** //Lugar del Checksum  IP**[**i**]** **=** **(**byte**)**0x00**;**  **}**  i**++;**  **}** |

1. Para el checksum hacemos referencia al programa Checksum y le enviamos como parámetro el arreglo que contiene la IP

|  |
| --- |
| long checksumIP **=** Checksum**.**calculateChecksum**(**IP**);**  System**.**out**.**printf**(**"|-- Valor de Checksum IP: %02X --|\n"**,**checksumIP**);** |

1. Para saber si corresponde al protocolo TCP o UDP consultamos el RFC-768, en el caso de TCP observamos que es el 06 así que lo comparamos con el byte 23.

|  |
| --- |
| **if** **((**byte**)**protocol **==** 0x06**)** //PROTOCOLO TCP  **{**  System**.**out**.**printf**(**"PRTOCOLO TCP: %02X\n"**,(**byte**)**protocol**);**  System**.**out**.**println**(**"---PROTOCOLO TCP/IP---"**);** |

También de ahí pudimos detectar el resto de bytes que pertenecen a:

* 1. IHL

|  |
| --- |
| byte IHL\_limpio **=** **(**byte**)** **(**packet**.**getUByte**(**14**)** **&** 0x0000000F**);**  int IHL **=** **(**IHL\_limpio **\*** 4**);** //Longitud del encabezado |

* 1. PDU de IP

|  |
| --- |
| int longitud **=** **(**int**)** **((**packet**.**getUByte**(**16**)** **\*** **(**Math**.**pow**(**2**,** 8**)))** **+** **(**packet**.**getUByte**(**17**)));** |

Cabe mencionar que aquí también hicimos el corrimiento de 8 bits para el byte 16.

* 1. IP origen e IP destino

|  |
| --- |
| **for** **(**int j **=** 0**;** j **<** 4**;** j**++)** **{**  /\*----------------------IP ORIGEN----------------------\*/  pseudoEncabezado**[**j**]** **=** **(**byte**)** **(**packet**.**getUByte**(**26 **+** j**));**  /\*---------------------IP DESTINO---------------------\*/  pseudoEncabezado**[**j **+** 4**]** **=** **(**byte**)** **(**packet**.**getUByte**(**30 **+** j**));**  **}**  pseudoEncabezado**[**8**]** **=** 0x00**;** |

Para ello, antes declaramos un arreglo de bytes con el tamaño del IHL

* 1. Protocolo

|  |
| --- |
| pseudoEncabezado**[**9**]** **=** **(**byte**)** **(**packet**.**getUByte**(**23**));** |

* 1. Longitud

|  |
| --- |
| pseudoEncabezado**[**10**]** **=** **(**byte**)**packet**.**getUByte**(**16**);**  pseudoEncabezado**[**11**]** **=** **(**byte**)(**packet**.**getUByte**(**17**)-**IHL**);** |

Obtenemos el PDU de transporte y lo almacenamos en un arreglo de bytes para posteriormente unirlo con el pseudoencabezado:

|  |
| --- |
| byte**[]** PDU\_Trans **=** **new** byte**[**longitudTotal**];**  **for** **(**int k **=** 34**,**q**=**0**;** k **<** 34**+**longitudTotal**;** k**++,**q**++)**  **{**  PDU\_Trans**[**q**]=(**byte**)**packet**.**getUByte**(**k**);**  **}** |

*Obtención del PDU de transporte*

|  |
| --- |
| int length\_pseudoE **=** pseudoEncabezado**.**length**;**  int length\_PDU\_Trans **=** PDU\_Trans**.**length**;**  byte**[]** encabezadoFinal**;**  encabezadoFinal**=** **new** byte**[**length\_pseudoE **+** length\_PDU\_Trans**];**  System**.**arraycopy**(**pseudoEncabezado**,** 0**,** encabezadoFinal**,** 0**,** length\_pseudoE**);**  System**.**arraycopy**(**PDU\_Trans**,** 0**,** encabezadoFinal**,** length\_pseudoE**,** length\_PDU\_Trans**);**  long checksumTCP **=** Checksum**.**calculateChecksum**(**encabezadoFinal**);**  System**.**out**.**printf**(**"Checksum del TCP: %04X \n"**,**checksumTCP**);** |

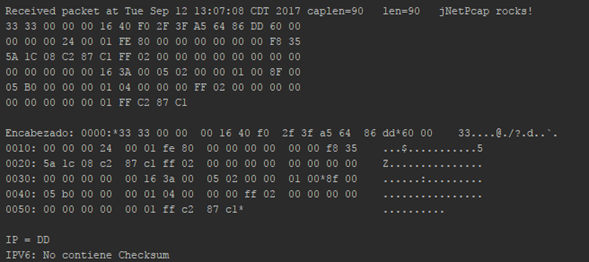
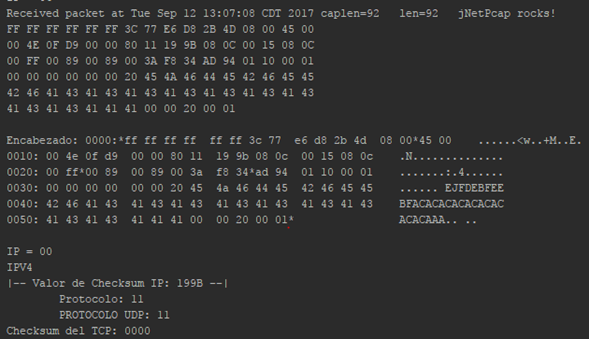
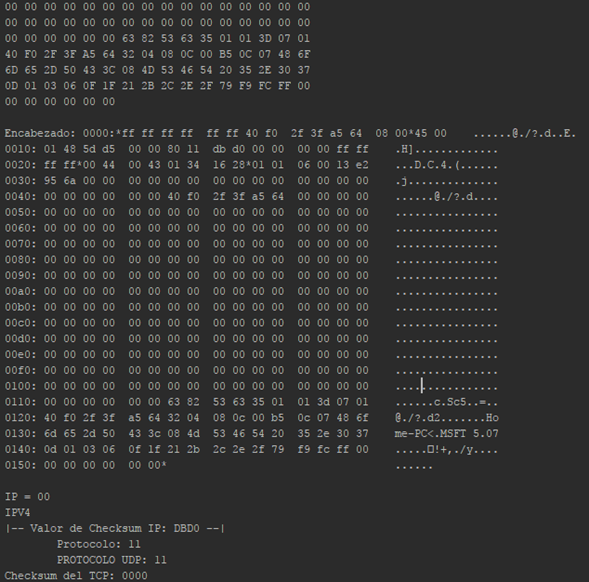
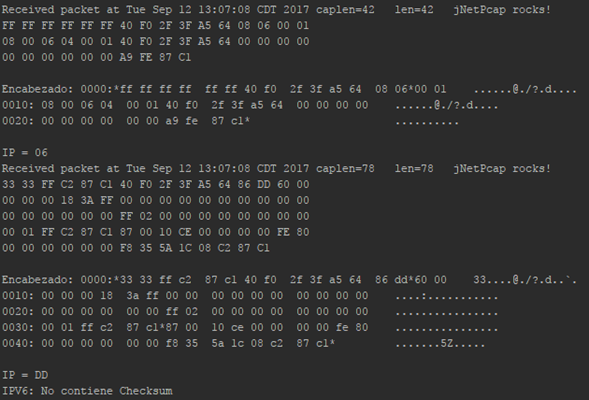
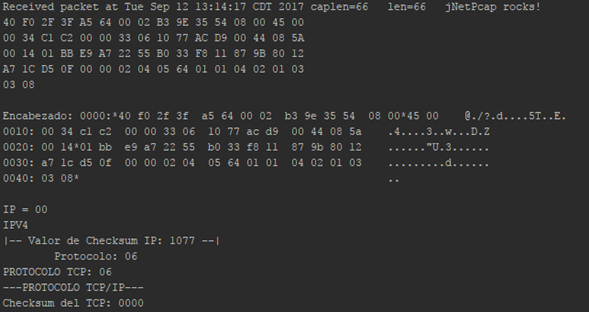
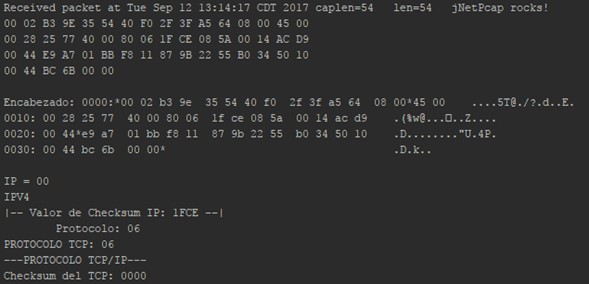
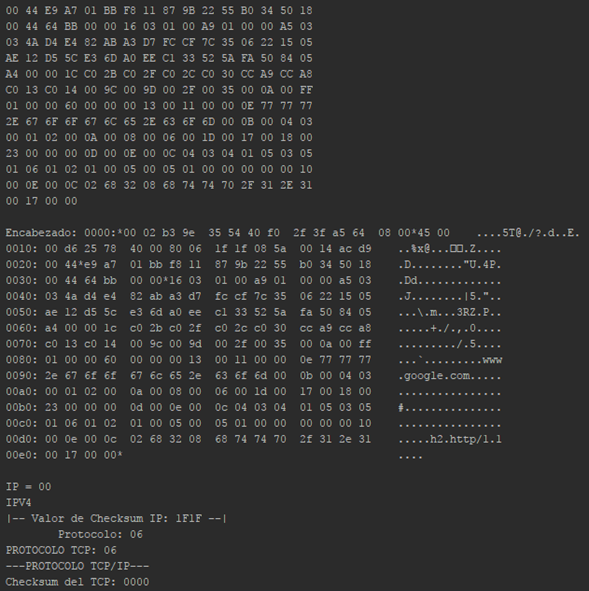
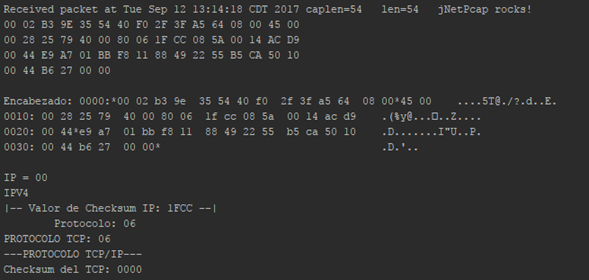
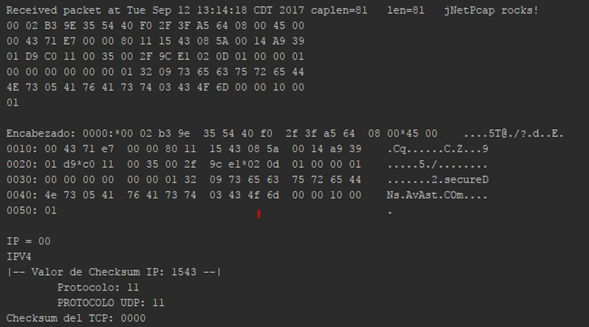
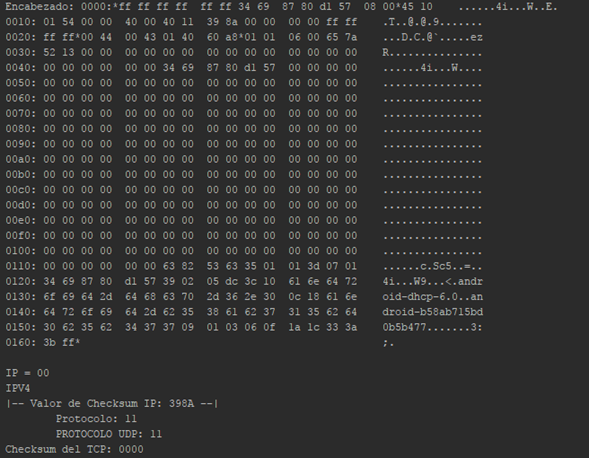
*Unión del PDU de transporte con el pseudoencabezado*

En el caso de UDP observamos que es el 11 así que lo comparamos con el byte 23 y de la misma forma que con el TCP obtenemos el IHL, PDU, longitud del pseudoencabezado, IP origen, IP destino y el PDU de transporte lo concatenamos con el pseudoencabezado:

|  |
| --- |
| int length\_pseudoE **=** pseudoEncabezado**.**length**;**  int length\_PDU\_Trans **=** PDU\_Trans**.**length**;**  byte**[]** encabezadoFinal**;**  encabezadoFinal**=** **new** byte**[**length\_pseudoE **+** length\_PDU\_Trans**];**  System**.**arraycopy**(**pseudoEncabezado**,** 0**,** encabezadoFinal**,** 0**,** length\_pseudoE**);**  System**.**arraycopy**(**PDU\_Trans**,** 0**,** encabezadoFinal**,** length\_pseudoE**,** length\_PDU\_Trans**);**  long checksumUDP **=** Checksum**.**calculateChecksum**(**encabezadoFinal**);**  System**.**out**.**printf**(**"Checksum del TCP: %04X \n"**,**checksumUDP**);** |

**Pruebas de funcionamiento**

A continuación se muestran las capturas de pantalla del funcionamiento del programa Captura.java con las modificaciones mencionadas anteriormente, se muestra la trama, el encabezado, la versión de IP y si es el caso el checksum y protocolo, así como la verificación del checksum.

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 

**Conclusiones personales**

* Díaz Medina Jesús Kaimorts:
* Esquivel Valdez Alberto:

Mediante la realización de esta práctica pude observar el funcionamiento y utilidad del cálculo del checksum, para lo cual creamos un pseudoencabezado con ciertos datos de la trama como el IHL, IP origen, IP destino, PDU, longitud y PDU de transporte, permitiéndonos ver a detalle cómo se compone la trama y verificando que el checksum sea correcto para esa trama, ya que en casos pueden llegar tramas con errores y de esta forma podemos detectar esas tramas erróneas.

* Vargas Romero Erick Efraín:

En esta práctica de ha verificado el funcionamiento del checksum visto en clase previamente. Primeramente creando un pseudoencabezado

**Bibliografía**

* J. Postel , “*User Datagram Protocol”* RFC 768. Agosto 1980. Disponible en línea en: <https://www.rfc-editor.org/pdfrfc/rfc768.txt.pdf>
* “*Algoritmo de checksum de Internet*”. Disponible en línea en: <http://www.arcesio.net/checksum/checksuminternet.html>
* R. Braden, “Computing the Internet Checksum”. Laboratorios BBN. Septiembre 1988. Disponible en línea en: <https://tools.ietf.org/pdf/rfc1071.pdf>