UNCUYO UNIVERSIDAD NACIONAL DE CUYO | FACULTAD DE INGENIERÍA Licenciatura en Ciencias de la Computación

Redes de Computadoras
Trabajo práctico Nº1 - Año 2023
Capa Física y de enlace. Manejo de Tramas

Objetivos

- Repasar los conocimientos adquiridos en clases mediante el análisis de tramas de diferentes protocolos de capa de enlace de datos.
- Introducción a herramientas de análisis de tramas.

Metodología

Trabajo individual o grupal. 2 estudiantes por grupo máximo.
Tiempo de realización: 1-2 clases.

Aprobación

- Mostrar en clase el programa de computación implementado para resolver la actividad 1 y subir código fuente a través de la plataforma Moodle.
- Escribir un breve informe (2 páginas máximo) del programa realizado. El informe debe incluir:
    - Instrucciones necesarias para que otra persona (con conocimientos de computación) pueda ejecutar el programa.
    - Resultado de la ejecución.
- Responder cuestionario “Trabajo Práctico N°1 - Cuestionario Actividad 2” que encontrará en la plataforma Moodle. Las preguntas están referidas a la actividad 2 indicada en esta guía, por lo que deberá realizar primero dicha actividad.

La entrega es individual. Si el trabajo se hizo en grupo, cada integrante debe realizar su entrega. Indicar en el informe los nombres de los integrantes.

Materiales necesarios

- Computadoras con acceso a Internet (provistas por la facultad de Ingeniería).
    - Se sugiere utilizar sistema operativo Linux.
- Analizador de tráfico de red Wireshark (Disponible en los repositorios de Linux (*sudo apt install wireshark*). Se puede instalar desde el centro de Software de Ubuntu. Disponible en www.wireshark.org/download.html para Linux, Windows y macOS).

Introducción teórica

El estándar IEEE802.15.4 es un estándar de IEEE para la implementación de las capas físicas y de enlace para redes de baja velocidad y muy bajo consumo de

energía. Es utilizado por computadoras destinadas a trabajar durante semanas o meses alimentadas solo con baterías.
Las tramas IEEE 802.15.4 están formadas por bytes que se representan mediante dos números hexadecimales cada byte. Por ejemplo: la trama 7E0012920013A200403A3BF8000001010013000003F3 posee 22 bytes, de los cuales los primeros 4 son:

- Primer byte: 0x7E (01111110 binario o 126 decimal), delimitador de trama.
- Segundo byte: 0x00 (00000000 binario o 0 decimal)
- Tercer byte: 0x12 (00010010 binario o 18 decimal)
- Cuarto byte: 0x92 (10010010 binario o 146 decimal)

Trama IEEE 802.15.4:

bandera (1 byte)	Longitud (2 bytes)	Tipo trama (1 byte)	Identificador de trama (1 byte)	Carga útil (variable)	Checksum (1 byte)

Bandera o delimitador de trama: indicador de comienzo de trama. Su valor es siempre 0x7E. En caso de aparecer la bandera en los datos (carga útil), se agrega adelante la secuencia de escape 0x7D.
Longitud: longitud de la trama sin incluir los campos bandera, longitud y checksum
Tipo de trama (algunos de los más utilizados son):

0x17: Petición de comando remoto.
0x97: Respuesta a comando remoto.
0x88: Respuesta a un comando.
0x90: Paquete Zigbee recibido (los nodos envían periódicamente paquetes).
0x92: Paquete de muestreo de datos.

Identificador de trama: número de secuencia.
Checksum: se calcula como: 0xFF – ((suma de todos los bytes sin incluir el bit bandera y el campo longitud) & 0xFF).
La operación & es un AND bit a bit.

Protocolo PPP:
El protocolo PPP (visto en clase, unidad 2) está formado por varios protocolos (o subprotocolos) entre ellos:

- CHAP (Challenge-handshake authentication protocol) o PAP (Password Authentication Protocol): protocolos de autenticación (se verán en la unidad 7) utilizados en la primera etapa de establecimiento de una conexión PPP (ver máquina de estado PPP).

- LCP (Link Control Protocol): Protocolo de negociación de opciones (autentificación, control de errores a utilizar, compresión, etc.) con la capa de enlace del dispositivo remoto. Un extremo propone opciones, el otro las acepta o rechaza. Si las rechaza, puede sugerir otras opciones, o esperar una nueva oferta de opciones.
- IPCP (IP Control Protocol): negociación de opciones a utilizar para transferir paquetes provenientes del protocolo de capa de red IP. Un extremo propone opciones, el otro las acepta o rechaza. Si las rechaza, puede sugerir otras opciones.

Actividades

Actividad 1: Tramas IEEE 802.15.4

Implementar un programa de computación que analice tramas de datos de la capa de enlace del protocolo IEEE 802.15.4.
Las tramas que debe analizar se encuentran en el archivo **tramas_802-15-4.log** que podrán encontrarse en la carpeta “capturas de tráfico” del trabajo práctico N°1 en aula virtual de la cátedra (plataforma Moodle). Las mismas son flujos de datos reales obtenidos a partir de la transmisión de datos realizada por un dispositivos Digi XBee module (https://www.digi.com/xbee). Podrá utilizar el lenguaje de computación que crea conveniente. Se sugiere **C++** o **Python** (en el apéndice al final de esta guía se sugieren instrucciones de C++ y Python que pueden ser de utilidad).

El programa debe realizar lo siguiente:

1. Indicar el número total de tramas recibidas.
2. Indicar el número de tramas con longitud correcta y con longitud incorrecta.
3. Indicar el número de tramas con suma de verificación es correcta y con suma de verificación incorrecta (nota, una trama con longitud incorrecta no podrá tener suma de verificación correcta).

Actividad 2: Análisis de capturas de tráfico

Utilizar la herramienta de análisis de tráfico de red Wireshark para analizar las capturas de tráfico “captura1.cap”, “captura2.pcapng”, “captura3.cap” y “captura4.pcapng”. que encontrará en la carpeta “Capturas de tráfico” del trabajo práctico N°1. Dichas capturas contienen tráfico real capturado con Wireshark.
Conteste las preguntas del cuestionario “Trabajo Práctico N°2 - Cuestionario y entrega archivos”.

Anexo

Funciones de utilidad de C++

1) Imprimir por pantalla:
cout << “texto a imprimir 1 ” << variable1 << “más texto a imprimir”;
cout << std::hex << variable1 << std::dec << variable2 << “texto”;
std::hex indica que la siguiente variable debe imprimirse en hexadecimal. Std::dec indica que la siguiente variable debe imprimirse en decimal.
También puede utilizarse printf(), pero cout es más simple e intuitivo.
Para usar cout necesita agregar las librerías #include<iostream> y #include <string>

2) Manejo de archivos:
Abrir un archivo en modo lectura:
ifstream mi_archivo("tramas_802-15-4.log");
El archivo debe estar en el mismo directorio que el programa. “mi_archivo” es una variable de tipo ifstream que se utilizará para hacer referencia al archivo. Para abrir el archivo en modo escritura:
ofstream mi_archivo("tramas_802-15-4.log");

Verificar si un archivo llegó a su fin:
while(!mi_archivo.eof())
{
Código a ejecutar si el archivo no ha llegado a su fin.
}

Leer una línea del archivo
mi_archivo >> cadena;
Donde cadena es una variable tipo String (se declara mediante String cadena;) donde se guardará la primera línea. Las variables tipo String son matrices de tamaño nx1 (vectores) formadas por elementos tipo char (caracteres);

2) Manejo de cadenas de caracteres:

Para buscar una cadena de caracteres dentro de otra cadena de caracteres de mayor tamaño:
posicion=cadena.find(cadena1);
Donde “posicion” es una variable tipo entero (int) que indica la posición de la cadena “cadena1” dentro de la cadena “cadena” (se indica la posición del primer caracter de “cadena1” dentro de “cadena”). “cadena1” es una cadena de menor tamaño que previamente debe inicializarse (ejemplo: tring cadena1 = "7E";). Si no se encuentra la cadena, la función devuelve -1.

Para copiar parte de una cadena de caracteres dentro de otra cadena de menor tamaño:
cadena1 = cadena2.substr(posición inicial,longitud);
Donde cadena1 es la cadena de menor tamaño y cadena2 es la cadena de mayor tamaño. Por ejemplo, si cadena2 tiene almacenado “hola_mundo”, y ejecutamos: cadena1 = cadena2.substr(2,5); en cadena1 se almacenará “la_mu”.

Para transformar una cadena de caracteres en un número entero:
longitud = stoi(cadena,NULL,base); donde longitud es una variable tipo int y cadena es una variable tipo String. Por ejemplo, si cadena vale A0 y ejecutamos
longitud = stoi(cadena,NULL,16); longitud valdrá A0 hexadecimal o 10 decimal.
Nota, para utilizar stoi, debe utilizar el modificador -std=c++11 al compilar.

Para conocer la longitud de una cadena: cadena.length() o cadena.size();

Funciones de utilidad de Python
1) Manejo de archivos:
Abrir un archivo:
archivo=open("tramas_802-15-4.log")
El archivo debe estar en el mismo directorio que el programa. “archivo” es una variable que se utilizará para hacer referencia al archivo. Para abrir el archivo en modo escritura:
archivo=open("tramas_802-15-4.log",”w”)

Leer una línea del archivo
contenido=archivo.readlines() o contenido=archivo.read()
Dónde contenido es una variable tipo String donde se guardará el texto leído.

2) Manejo de cadenas de caracteres:

Para buscar una cadena de caracteres dentro de otra cadena de caracteres de mayor tamaño:
posicion=cadena.find("7E")
Dónde posición es una variable tipo entero. Si no se encuentra la cadena buscada, la función devuelve -1.

Para conocer la longitud de una cadena: len(cadena)

Para copiar parte de una cadena de caracteres dentro de otra cadena de menor tamaño:
cadena1 = cadena2[posición inicial:posición final]
Donde cadena1 es la cadena de menor tamaño y cadena2 es la cadena de mayor tamaño. Por ejemplo, si cadena2 tiene almacenado “hola_mundo”, y ejecutamos: cadena1 = cadena2[2,5], en cadena1 se almacenará “la_”.
Para copiar desde una posición inicial hasta el final de la cadena:
cadena1 = cadena2[posición inicial:]

Para transformar una cadena de caracteres en un número entero: int(cadena,base), donde base es un número entero que expresa la base del número contenido en cadena (ejemplo: para binario base debe valer 2, para decimal 10, para hexadecimal 16).

Para conocer la longitud de una cadena: cadena.length() o cadena.size();

Declarar un list vacío: mi_list=[]
Agregar elementos a un list: mi_list.append(elemento a agregar)
Recorrer un list:

```
for elemento in mi_list:
    Acciones a realizar con cada elemento
    ……..
```