7 Programación de un protocolo de aplicación basado en sockets (ALCOPEN)

7.1 Objetivo de la Práctica

El alumno debe implementar el protocolo de aplicación ALCOPEN que se detalla a continuación para comunicar dos PCs. El protocolo ALCOPEN es un protocolo ficticio que se define a continuación con el fin de ayudar al alumno a familiarizarse y a entender los niveles OSI y su empleo.

7.2 Teoría de la Práctica

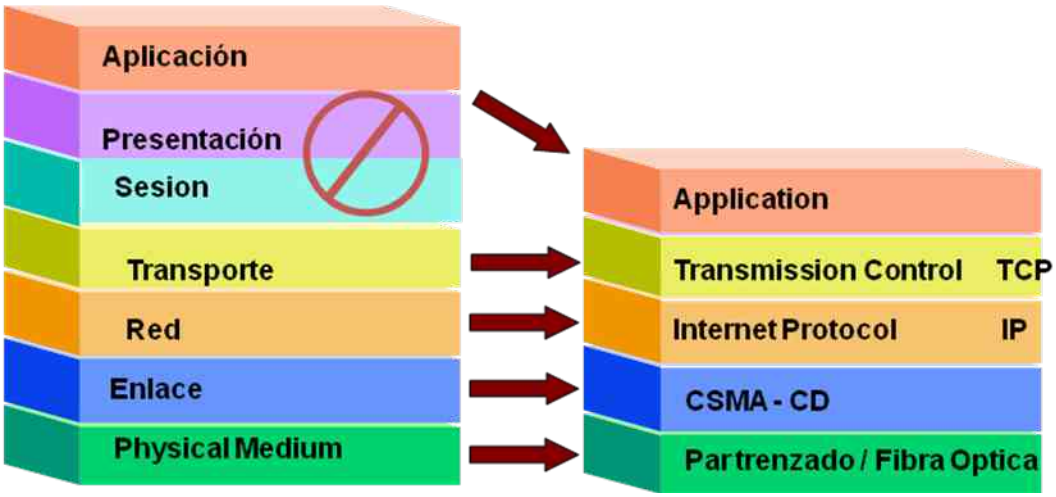


Figura 1 Capa del protocolo ALCOPEN

Como se vé en la Figura el procolo ALCOPEN es un protocolo de comunicación que va emplear otros protocolos por debajo. En este caso el protocolo de aplicación debe hacer uso de la capa TCP que no es otra cosa que emplear la librería de sockets TCP de JAVA. El alumno es libre de implementar el programa en lenguje JAVA, C++ o .NET.

7.2.1 Entorno del Sistema

Se considera que el sistema para el cúal se desea establecer un protocolo de comunicaciones consta de un nodo maestro y uno o varios nodos esclavos. El nodo maestro simula por ejemplo una unidad de control principal de automoción y los nodos esclavos simulan otras unidades de control auxiliares que se comunican con el nodo maestro.

Todos los nodos esclavos se consideran que pueden estar en tres estados OPERACIONAL, PREOPERACIONAL o STOP. El nodo maestro se asume que siempre está en estado OPERACIONAL.

La máquina de estados del nodo esclavo debe ser la siguiente:

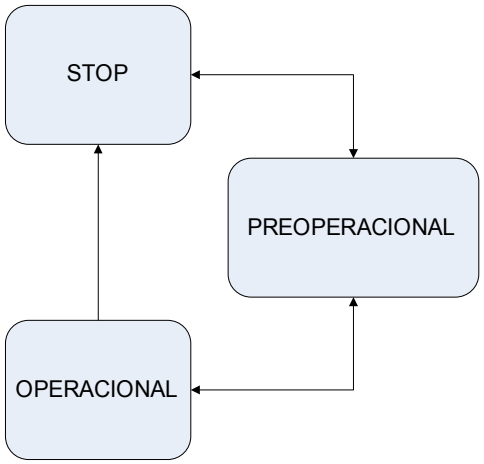


Figura 2 Máquina de estados protocolos ALCOPEN

Cada nodo esclavo tiene diez parámetros que el nodo maestro puede leer o escribir mediante el protocolo a implementar. En dichos parámetros el nodo esclavo contiene valores de sensores o ajustes que modifican su funcionamiento.

NUMERO PAR	TIPO DE DATO	DESCRIPCIÓN
1	INT32	Parámetro a transmitir en el PDO primer dato (ver apartados siguientes) (3-8) Es decir aquí se indica por ejemplo con un '4' que se debe enviar el par
2	INT32	Parámetro a transmitir en el PDO segundo dato (ver apartados siguientes) (3-8)
3	FLOAT	Presión de colector de admisión (0.0-10.0)
4	FLOAT	Temperatura de gas (0.0-50.0)
5	FLOAT	Avance de encendido (0-100.0)
6	FLOAT	Presión de colector de escape (0.0-10.0)
7	FLOAT	Posición de mariposa (0-100.0)
8	FLOAT	Velocidad de motor (0-5000.0rpm)
9	FLOAT	Periodo de transmisión del mensaje PDO por parte del esclavo (0.0 segundos a 10000.0 segundos)
10	FLOAT	Periodo de transmisión del mensaje NMT por parte del esclavo (0.0 segundos a 10000.0 segundos)

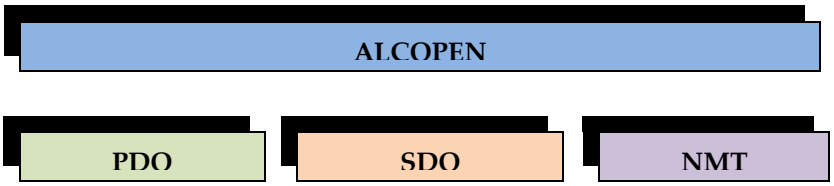
El protocolo se basará en interpretar una serie de "tramas de datos" de 10bytes. Es decir, el alumno debe intercambiar diez bytes e interpretar o establecer valores en los mismos de acuerdo a la siguiente especificación del protocolo.

El protocolo define tres tipos de tramas de datos:

- **Tramas NMT (Network Management):** Es una trama de datos empleada por el nodo maestro para establecer el estado del nodo esclavo y por el nodo esclavo para indicar al nodo maestro el estado en el que se encuentra.

- **Trama SDO (Spurious Data)**: Es una trama asíncrona que el maestro envía al esclavo para leer o escribir un parámetro de dicho nodo. El nodo esclavo responde con la información. Se suele emplear en la realidad para permitir al nodo maestro configurar que datos quiere que el nodo esclavo le envíe en los PDOs y para leer o escribir datos que no es necesario recibirlos de una manera muy rápida.

- **Trama PDO (Peridic Data)**: Es una trama de datos periódica enviada por el nodo esclavo al maestro. Se suele emplear en la realidad para enviar datos de manera rápida (cada periodo se reenvian los datos).



A continuación, se muestran las tramas que debe implementar el protocolo y el contenido de las mismas:

7.2.1.1 NMT trama (Network Management)

7.2.1.1.1 Trama NMT de nodo maestro a esclavo

Esta es una trama de datos que el nodo maestro intercambia con los nodos esclavos. Los nodos esclavos inicialmente cuando se inician entran en el estado PREOPERACIONAL y se quedan a la espera de recibir el mensaje de NMT Start del nodo maestro.

BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7	BYTE8	BYTE9
0x0000		COMMAND	NODE -ID	0X00	0X00	0X00	0X00	0X00	0X00

COMMAND**:**

0x01: El nodo maestro para hacer el START (paso a OPERACIONAL) debe escribir 0x01 en dicho campo.

0x00: Paso del nodo esclavo seleccionado en NODE-ID de OPERACIONAL o STOP a PREOPERACIONAL del nodo.

0x02: Paso del nodo esclavo seleccionado en NODE-ID de OPERACIONAL o PREOPERACIONAL a STOP del nodo.

NODE-ID (Valor de 0-128, red de máximo 128 nodos):

Identificador del nodo al que va dirigido el mensaje. En caso de que el NODE-ID sea 0x00 todos los nodos que reciben el mensaje deben entender que es para ellos y procesar el comando (tipo de broadcast).

7.2.1.1.2 Trama NMT de nodo esclavo a maestro

BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7	BYTE8	BYTE9
0x0000+NODEID		STATE	0x00	0X00	0X00	0X00	0X00	0X00	0X00

Los nodos esclavos cuando arrancan entran en el estado PREOPERACIONAL y deben enviar cada 10 segundos la siguiente trama al nodo maestro para identificar en el estado que se encuentran. STATE toma los valores: '0' PREOPERACIONAL. '1' OPERACIONAL, '2' STOP

Si el nodo está en el estado OPERACIONAL, el nodo esclavo debe enviar el mensaje con un periodo dado por el valor de parámetro número 10 del nodo esclavo en segundos. En caso de tener un valor de 0.0 el nodo esclavo no debe enviar el estado.

Si el nodo está en estado STOP no debe enviar ningún tipo de mensaje pero sí procesar los dirigidos a él.

7.2.1.2 SDO trama (Spurios Data)

El nodo maestro hará uso de estas tramas para leer o escribir datos de manera asíncrona a el nodo seleccionado.

BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7	BYTE8	BYTE9
0x800+NODEID		RW	NUMPAR	DATA0	DATA1	DATA2	DATA3	0X00	0X00

El nodo esclavo cuyo identificador coincide con el NODE-ID de la trama debe procesar el mensaje.

RW: '1' Escritura, '0' Lectura.
NUMPAR: Número del parámetro a leer.
DATA0-3: Valor a escribir en el parámetro en formato flotante de 32bits.

El nodo esclavo debe responder con la siguiente trama.

BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7	BYTE8	BYTE9
0x1000+NODEID		RW	NUMPAR	DATA0	DATA1	DATA2	DATA3	0X00	0X00

DATA0-3: Es donde el nodo esclavo debe escribir en formato flotante de 32bits el valor del número de parámetro solicitado por el maestro.
El campo de datos consta de 8 bytes de datos.

7.2.1.3 PDO trama (Peridic Data)

El nodo esclavo debe enviar periódicamente el mensaje PDO:

BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7	BYTE8	BYTE9
0x400+NODEID		PAR1_B0	PAR1_B1	PAR1_B2	PAR1_B3	PAR2_B0	PAR2_B1	PAR2_B3	PAR2_B4

PAR1_B0-PAR1_B4: Es donde el nodo esclavo debe escribir en formato flotante de 32bits el valor del parámetro mapeado en el parámetro número 1 del esclavo.

PAR2_B0-PAR2_B4: Es donde el nodo esclavo debe escribir en formato flotante de 32bits el valor del parámetro mapeado en el parámetro número 2 del esclavo.

El nodo esclavo debe transmitir estos mensajes con el periodo dado por el contenido del parámetro número 9.

7.3 Tareas a realizar

1. El alumno debe implementar el protocolo descrito.
2. El alumno debe detallar la capa de aplicación del protocolo programado. Es decir, detallar las clases o métodos implementados al estilo de como se detallan en la realidad (ver ejemplo Clase socket)

OBSERVACIÓN: Las funciones o clases que implementen la comunicación NMT, PDO, SDO deben empezar por NMT_*NombreDeFuncion_ó_Clase*, PDO_*NombreDeFuncion_ó_Clase*, SDO_*NombreDeFuncion_ó_Clase* respectivamente.

java.net

Class Socket

java.lang.Object
  └ **java.net.Socket**

Direct Known Subclasses:
SSLSocket

public class **Socket**
extends Object

This class implements client sockets (also called just "sockets"). A socket is an endpoint for communication between two machines.

The actual work of the socket is performed by an instance of the `SocketImpl` class. An application, by changing the socket factory that creates the socket implementation, can configure itself to create sockets appropriate to the local firewall.

Since:
JDK1.0

See Also:
`setSocketImplFactory(java.net.SocketImplFactory)`, `SocketImpl`, `SocketChannel`

Constructor Summary	
	Socket() Creates an unconnected socket, with the system-default type of SocketImpl.
	Socket(InetAddress address, int port) Creates a stream socket and connects it to the specified port number at the specified IP address.
protected	**Socket**(SocketImpl impl) Creates an unconnected Socket with a user-specified SocketImpl.

	Socket(String host, int port) Creates a stream socket and connects it to the specified port number on the named host.

Method Summary	
void	**bind**(SocketAddress bindpoint) Binds the socket to a local address.
void	**close**() Closes this socket.
void	**connect**(SocketAddress endpoint) Connects this socket to the server.

.....