Contrôle de TD - module RSX

1h30 - polycopié de cours autorisé

Exercice 1 : *Bus ethernet*

On considère un bus Ethernet (protocole anti-collision CSMA-CD) à 100 Mbits/s de 1500m de long.

Q 1 . *Expliquez comment est régulé l'accès au bus ?*

Q 2 . *Quelles sont les contraintes de taille de paquets de ce bus (vitesse du signal dans le cuivre = 200000 km/s) ?*

Q 3 . *Pour amplifier le signal, on ajoute tous les 500m un répéteur qui engendre un retard de 5 μs. Quelles sont désormais les contraintes de taille de paquets de ce bus ?*

Exercice 2 : *ADSL*

(*source : `http://fr.wikipedia.org/wiki/Asymmetric_Digital_Subscriber_Line`*)

L'ADSL (Asymmetric Digital Subscriber Line) est un schéma de modulation qui fait appel à la notion de sous-porteuses : la bande de fréquences comprise entre 0 Hz et environ $1,1$ MHz est divisée en 255 intervalles de $4,3125$ kHz. A chaque frontière d'intervalle est associée une sous-porteuse, qui est un signal modulé. La n-ième sous-porteuse est donc matérialisée sous la forme d'un signal dont la fréquence de base vaut [$n \times 4,3125$ kHz]. Un modem ADSL peut donc être considéré comme la mise en parallèle d'un grand nombre de modems analogiques, chacun transmettant sur une fréquence différente : 4,3125 kHz pour le premier, 8,625 kHz pour le second, 12,9375 kHz pour le troisième, etc ...

La sous-porteuse d'indice 0 n'est pas utilisée, car elle correspond à un signal de fréquence nulle. Les sous-porteuses d'indice 1 à 255 sont théoriquement utilisables pour transmettre des données. Toutefois, les sous-porteuses d'indice 1 à 15 ne sont en général pas exploitées en raison de la présence possible de signaux téléphoniques dans une gamme de fréquences proche des fréquences utilisées par ces sous-porteuses. Dans la pratique, lorsque l'ADSL est mis en oeuvre sur une ligne téléphonique classique (analogique), les sous-porteuses d'indice 16 à 255 sont donc disponibles pour la communication ADSL proprement dite. On utilise en général les sous-porteuses d'indice 16 à 31 pour la voie montante, et celles d'indices au moins égal à 32 pour la voie descendante.

Q 1 . *Quel est le nombre de sous-porteuses utilisées pour les voies montantes/descendantes ?*

Q 2 . *Quel est le spectre des fréquences des voies montantes/descendantes ?*

Q 3 . *En supposant que chaque sous-porteuse soit à 8000 bauds, et que chaque signal associé corresponde à 16bits d'information, quels seront les débits théoriques montants et descendants du modem ADSL ?*

(*source : `http://en.wikipedia.org/wiki/Quadrature_amplitude_modulation`*)

On s'intéresse désormais à la modulation d'une seule de ces porteuses : celle-ci est réalisée à l'aide d'une méthode particulière appelée *Quadrature Amplitude Modulation* (QAM) :

Le principe est simple : une sous-porteuse d'indice i est en fait composée de deux sinusoïdes déphasées de 90 degrés l'une de l'autre. Ces deux sinusoïdes sont *indépendamment modulées en amplitude* par deux fonctions $I(t)$ et $J(t)$. La forme du signal de la sous-porteuse d'indice i (et fréquence f_i) est ainsi donné par la somme des deux sinusoïdes déphasée et modulées :

$$I(t) \cdot cos(2 \cdot \pi \cdot f_i \cdot t) + J(t) \cdot sin(2 \cdot \pi \cdot f_i \cdot t)$$

Les fonctions $I(t)$ et $J(t)$ donnent l'amplitude des deux sinusoïdes, leurs valeurs respectives changent 8000 fois/s.

Q 4 . *Donnez la forme du signal sur une seule période pour les 3 cas suivants (utilisez une même figure pour vous aider)*

- $I(t) = x > 0, J(t) = 0$
- $I(t) = 0, J(t) = x > 0$
- $I(t) = x > 0, J(t) = x > 0$

Q 5 . *Dans le dernier cas, que constatez vous sur la fréquence et la phase du signal résultat ? Cela change t'il dans le cas général (sommation de deux sinusoïdes de même fréquence et de phases/d'amplitudes différentes) ? En quoi cela est important dans le cadre de sous-porteuses du modem ADSL ?*

Enfin, nous terminons cet exercice en faisant remarquer qu'il est possible de "retrouver", à l'aide d'un démodulateur produit (calcul d'une corrélation et extraction d'une composante continue), les valeurs de $I(t)$ et de $J(t)$, sans qu'il y aie de collisions entre les deux signaux.

Q 6 . *En supposant que seule l'amplitude soit modulable, sur combien de niveaux d'amplitudes doit on moduler chacune des deux sinusoïdes pour transmettre 16bits/signal et ainsi obtenir le débit théorique de la question 3 ?*

Exercice 3 : *Routage d'un réseau global*

Un réseau global de classe B et d'identifiant 0x06CE est composé de 3 bus Ethernet notés $ET1$,$ET2$,$ET3$: chaque bus comporte 4 machines. Deux passerelles **P1** et **P2** qui disposent chacune de deux cartes ethernet sont respectivement connectées aux bus $ET1$ et $ET2$/ aux bus $ET2$ et $ET3$. La passerelle **P1** dispose d'un modem ADSL interne noté $AD1$.

Q 1 . *Quelle est la plage d'adresses possibles pour les machines de ce réseau.*

Q 2 . *Donnez un schéma du réseau.* **Q 3 .** *Proposez un masque de sous réseau pour identifier les sous réseaux associés à $ET1$, $ET2$, $ET3$. Proposez une adresse de sous réseau pour chaque bus ethernet et finalement, une adresse IP pour chaque machine du réseau (et deux adresses IP pour chaque passerelle).*

Q 4 . *Donnez les tables de routage des passerelles* **P1** *et* **P2***. Chaque table contiendra 4 champs : adresse de Réseau, masque de Réseau, adresse Passerelle, $Liaison$.* **Q 5 .** *Donnez les commandes à exécuter pour configurer "intelligemment" une machine située sur le lien $ET2$, en expliquant les différences que cela implique en terme de paquets transmis par rapport à une configuration "basique".*

Exercice 4 : *Chat*

En utilisant vos connaissances et en vous aidant des codes fournis dans le cours, proposez un client de chat en UDP (Unicast) qui permette simultanément d'envoyer (vers un serveur donné $server$ à un port donné $portA$) des message saisis sur l'inputstream `System.in`, et de recevoir (sur le port local utilisé par l'émission) des messages pour les afficher sur l'outputstream `System.out`. Vous ne tiendrez pas compte des exceptions.

```
----------------
BufferedReader :
----------------

Constructor Summary
 BufferedReader(InputStreamReader in)
          Create a buffering character-input stream that uses a default-sized input buffer.

Method Summary
 int  read()
          Read a single character.
 int  read(char[] cbuf, int off, int len)
          Read characters into a portion of an array.
 String readLine()
          Read a line of text.

-------------------
InputStreamReader :
-------------------

Constructor Summary
 InputStreamReader(InputStream in)
          Create an InputStreamReader that uses the default charset.

Method Summary
 int  read()
          Read a single character.
 int  read(char[] cbuf, int offset, int length)
          Read characters into a portion of an array.

-------------
PrintStream :
-------------

Constructor Summary
 PrintStream(OutputStream out)
          Create a new print stream.

Method Summary
 void  flush()
          Flush the stream.
 void  print(xxxx value)
          Print any boolean, char, int, double, floag, String
          ... value in the outputstream.
 void  println(xxxx value)
          Print any boolean, char, int, double, floag, String
          ... value in the outputstream and then terminate the line.
 void  write(byte[] buf, int off, int len)
          Write len bytes from the specified byte array starting at offset off to this stream.
 void  write(int b)
          Write the specified byte to this stream.



----------------
DatagramSocket :
----------------
```

```
Constructor Summary
   DatagramSocket()
   DatagramSocket(int port)
          Constructs a datagram socket and binds it to the specified port on the local host machine.
   DatagramSocket(int port, InetAddress laddr)
          Creates a datagram socket, bound to the specified local address.

Method Summary
 void  bind(SocketAddress addr)
          Binds this DatagramSocket to a specific address & port.
 void  close()
          Closes this datagram socket.
 void  connect(InetAddress address, int port)
          Connects the socket to a remote address for this socket.
 void  connect(SocketAddress addr)
          Connects this socket to a remote socket address (IP address + port number).
 void  disconnect()
          Disconnects the socket.
 InetAddress  getInetAddress()
          Returns the address to which this socket is connected.
 InetAddress  getLocalAddress()
          Gets the local address to which the socket is bound.
 int  getLocalPort()
          Returns the port number on the local host to which this socket is bound.
 int  getPort()
          Returns the port for this socket.
 void  receive(DatagramPacket p)
          Receives a datagram packet from this socket.
 void  send(DatagramPacket p)
          Sends a datagram packet from this socket.

----------------
DatagramPacket :
----------------

Constructor Summary
DatagramPacket(byte[] buf, int length)
          Constructs a DatagramPacket for receiving packets of length length.
DatagramPacket(byte[] buf, int length, InetAddress address, int port)
          Constructs a datagram packet for sending packets of length length
          to the specified port number on the specified host.

Method Summary
 InetAddress  getAddress()
          Returns the IP address of the machine to which this datagram is being sent
          or from which the datagram was received.
 byte[]  getData()
          Returns the data buffer.
 int  getLength()
          Returns the length of the data to be sent or the length of the data received.
 int  getPort()
          Returns the port number on the remote host to which this datagram is being sent
          or from which the datagram was received.
 void  setAddress(InetAddress iaddr)
          Sets the IP address of the machine to which this datagram is being sent.
 void  setData(byte[] buf)
          Set the data buffer for this packet.
 void  setPort(int iport)
          Sets the port number on the remote host to which this datagram is being sent.

-------------
InetAddress :
-------------

Method Summary
static InetAddress[]  getAllByName(String host)
          Given the name of a host, returns an array of its IP addresses,
          based on the configured name service on the system.
static InetAddress  getByAddress(byte[] addr)
          Returns an InetAddress object given the raw IP address.
static InetAddress  getByName(String host)
          Determines the IP address of a host, given the host's name.
 String  getCanonicalHostName()
          Gets the fully qualified domain name for this IP address.
 String  getHostAddress()
          Returns the IP address string in textual presentation.
 String  getHostName()
          Gets the host name for this IP address.
static InetAddress  getLocalHost()
          Returns the local host.
 String  toString()
          Converts this IP address to a String.
```