ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA DE TELECOMUNICACIÓ
DEPARTAMENT D'ENGINYERIA TELEMÀTICA
APLICACIONS I SERVEIS TELEMÀTICS
Examen final. Primavera 15.

Notes Prov.: Dimecres 17 Juny. Revisió Presencial: Divendres 26 Juny, 10:00h Lab AST.

Problema 1 (2.5 punts).

Es vol realitzar un mecanisme de pas de valors des d'un procés productor a N processos consumidors **MITJANÇANT MONITORS**. El procés productor (**només n'hi ha un**) invoca `putValue(v)` per indicar el valor a transmetre als consumidors, i els consumidors invoquen `getValue(id)` per obtenir el valor del productor (el paràmetre `id` és l'identificador del consumidor de 0 a $N-1$). Si el valor del productor encara no ha sigut consumit per tots els consumidors aleshores la següent invocació a `putValue` haurà d'aturar el productor (podeu considerar el mecanisme com un *buffer* de capacitat 1).

Suggeriment - Fer servir el següent esquema:

```
public class BufferBroadcastUn {
    protected Object espai;

    protected boolean[] disponible;
    // disponible[i] indica que el i-essim consumidor
    // no ha consumit l'objecte actual

    ...

    altres atributs, constructor i mètodes
}
```

Problema 2 (2.5 punts).

Implementar l'arquitectura distribuïda (pas de missatges) i servidor multifil de l'exercici anterior. Implementar tant la part del client com la del servidor.

Problema 3 (2 punts). Mida màxima de segment.

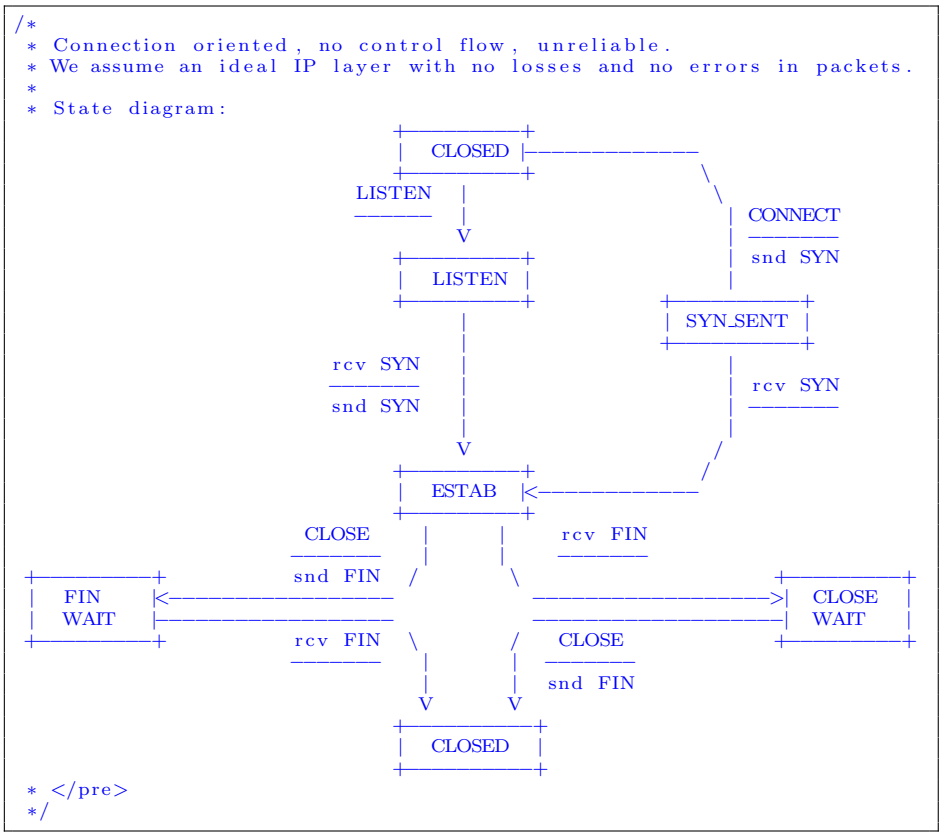
El següent mètode `sendData` de la classe `TCBlockUnreliable` construeix segments de mida arbitrària.

```
public void sendData(byte[] data, int data_off, int data_len) {
    TCPSegment segment = new TCPSegment();
    segment.setSourcePort(localPort);
    segment.setDestinationPort(remotePort);
    segment.setFlags(TCPSegment.PSH);
    segment.setData(data, data_off, data_len);
    sendSegment(segment);
}
```

L'enviament de segments de mida molt gran pot arribar a suposar fragmentació en capes inferiors amb la conseqüent pèrdua de rendiment. Suposant que l'atribut `sndMSS` manté la mida màxima de segment per a què no hi hagi fragmentació en capes inferiors, implementa el mètode `sendData(byte[] data, int data_off, int data_len)` de manera que no es transmetin segments que superin aquesta mida.

Problema 4 (2 punts).

Suposant que la classe `TCBlockUnreliable` disposa d'una variable de condició `appCV` i tenint en compte el següent diagrama d'estats d'un protocol de transport, completa els punts suspensius en els següents mètodes.



```
public void listen(int backlog) throws IOException {
  lk.lock();
  try {
    if (state != CLOSED) {
        throw new IOException("connection already exists");
    }
    acceptQueue = new CircularQueue<TCBlockUnreliable1>(backlog);

    state = . . . . .

    addListenTCB(this);
    logDebugState();
  } finally {
    lk.unlock();
  }
}
```

```
public TCBlock accept() throws IOException {
  lk.lock();
  try {
        // Pista: acceptQueue


          . . . . .


    } catch (InterruptedException ie) {
        log.warn("Interrupted Exception at accept()");
    }
    TCBlockUnreliable1 r = acceptQueue.get();
    return r;
  } finally {
    lk.unlock();
  }
}
```

```
public void connect(int addr, int port) throws IOException {
  lk.lock();
  try {
    initActive(addr, port & 0xffff, SYN_SENT);
    TCPSegment sseg = new TCPSegment();
    sseg.setSourcePort(localPort);
    sseg.setDestinationPort(remotePort);
    sseg.setFlags(TCPSegment.SYN);
    sendSegment(sseg);
    logDebugState();
    try {
        // Pista: state


          . . . . .


    } catch (InterruptedException ie) {
        log.warn("Interrupted Exception in connect()");
    }
  } finally {
    lk.unlock();
  }
}
```

```
protected void processReceivedSegment(int sourceAddr, TCPSegment rseg) {
  lk.lock();
  try{
    switch (state) {
      case LISTEN: {
        if (rseg.isSyn()) {
          if (acceptQueue.full()) {
                return;
          }
          // create TCB for new connection
          TCBlockUnreliable1 ntcb = (TCBlockUnreliable1) newTCB();

          ntcb.initActive(sourceAddr, rseg.getSourcePort(), . . . . . );

          //prepare the created connection for accept
          acceptQueue.put(ntcb);

          . . . . .

          // send SYN segment for new connection
          TCPSegment sseg = new TCPSegment();
          sseg.setSourcePort(ntcb.localPort);
          sseg.setDestinationPort(ntcb.remotePort);
          sseg.setFlags(TCPSegment.SYN);
          ntcb.sendSegment(sseg);
       }
      break;
      }

      case SYN_SENT: {
        if (rseg.isSyn()) {
          // Pista: state


          . . . . . .

        }
        break;
      }

      // Other cases

      default:
        // Segment is ignored
      }
   } finally {
     lk.unlock();
   }
}
```

```
// initialize for new connection
protected void initActive(int remAddr, int remPort, int state) {
    remoteAddr = remAddr;
    remotePort = remPort;
    this.state = state;
    rcvQueue = new ByteQueue(1000);
    addActiveTCB(this);
}
```

Problema 5 (1 punt). Explica el control de flux en TCP.