* Initialize()
  + Set tc = 0
  + Set tc = tc + (length+H)/C
  + Set SN = 0
  + Next\_expected AcK = (SN+1) mod 2
  + Next\_expected frame = 0
  + Insert timeout in Es at time tc+delta
* Pass to forward channel tc, sn, length
* Forward chan()
  + Generate random probability
  + For each bit in the frame and check if
    - Prob < BER -> bit error
    - Prob > BER -> error free
  + Counter of bits in error
    - If counter >= 5 -> lost
    - 0 < counter < 5 -> error
    - counter = 0 -> error frame
  + set tc += tao

Pass to Reciever tc, sn, flag length

Receiver()

* + check flag
  + if lost
    - do nothing
  + elif error
    - send ack with rn = next expected frame
    - tc = tc + H/C
  + elif error free
    - increment next expected frame
    - set rn = next expected frame
    - send ack
  + end if
* Pass tc, rn, length to reverse channel
  + Same func as forward
  + Tc += tao
* Send tc, rn, length flag to es

Is flag is error free or in error

* + No -> do nothing
  + Yes -> insert ack in es at tc, flag, rn
    - Es-event, time, flag, sequence
* Es processor
  + While es is not empty, do
    - If event = ack and flag = error free and sequence = next expected frame
      * Increment sn
      * Set next expected ack = (sn+1) mod 2
      * Purge old time out
      * Insert time out at tc = tc + delta
      * Send new frame at tc = time + (length+H)/C to send()
      * counter of received frames if = 10000->exit
    - Elif event == ack and (flag == error or sn != next expected ack)
      * Do nothing (max BM)
    - Elif event == timeout
      * Purge timeout
      * Tc = tc + (l+h)/c
      * Send();
      * Insert time\_out at tc = tc + delta