Final Project Brennenstuhl, Knobeloch, McCann

see presentation for concept

Important:

Not every DatagrammSocket is configured with a timeout for receiving! (waiting infin. -> no exception will be thrown)

PC1

* creates COMMENTPdu
* sets Timestamp ID
* sends PDU via MULTICAST 5 times
* increments ID
* loop

PC 2

* Controller Thread
  + Creates MULTICAST Receiver Thread (PC2\_PC1)
  + Creates UNICAST Sender Thread (PC2\_PC3)
  + Creates UNICAST Receiver Thread (PC2\_PC3)
  + Sets the different properties of these Threats
  + Starts the Threads and
  + checks every 2nd second if there are messages to send from the multicast network
  + loop
    - checking messages
    - starting threads if they are stopped (timeOuts)
* MULTICAST Receiver Thread (PC2\_PC1)
  + Receives bytes
  + Creates PDUs from received bytes
  + Checks if ID is already received (than message will be skipped)
  + If test above failed -> adds ID to archive and message to sendList
  + Loop
* UNICAST Sender Thread (PC\_PC3)
  + Locks the sendList-variable
  + Pulls the first PDU
  + Unlocks sendList-variable
  + Sends PDU via DatagramSocket (as Datagrammpackage) via UNICAST to PC3
* UNICAST Receiver Thread (PC2\_PC3)
  + Receives UNICAST messages from PC3 (Timeout if not: 30.000ms)
  + Prints statement with message in system.out

PC3

* UNICAST Reveiver AND MULTICAST Sender Thread (PC3\_PC2)
  + Receives message from PC2 via UNICAST
  + Sends confirmation via UNICAST to PC2
  + Sends received message via MULTICAST over port 3000

Simulation can be used to simulate scenario like:

* PC1 symbolize many multicast sender within a network (Pentagon)
* PC2 is Relay between Pentagon and Base in Afghanistan
* PC2 is also filter, that Base in AFG only receives a multicast message one time
* PC3 is Relay at base AFG which communicates directly with Pentagon Relay and sends the messages via multicast within the base-network
* By adjusting the thread.sleeps you can also simulate a delay when sending via satellite