

see presentation for concept

Important:

Not every DatagramSocket 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
 - o Creates MULTICAST Receiver Thread (PC2_PC1)
 - o Creates UNICAST Sender Thread (PC2_PC3)
 - o Creates UNICAST Receiver Thread (PC2_PC3)
 - o Sets the different properties of these Threads
 - o Starts the Threads and
 - o checks every 2nd second if there are messages to send from the multicast network
 - o loop
 - checking messages
 - starting threads if they are stopped (timeOuts)
- MULTICAST Receiver Thread (PC2_PC1)
 - o Receives bytes
 - o Creates PDUs from received bytes
 - o Checks if ID is already received (than message will be skipped)
 - o If test above failed -> adds ID to archive and message to sendList
 - o Loop
- UNICAST Sender Thread (PC_PC3)
 - o Locks the sendList-variable
 - o Pulls the first PDU
 - o Unlocks sendList-variable
 - o Sends PDU via DatagramSocket (as Datagrammpackage) via UNICAST to PC3
- UNICAST Receiver Thread (PC2_PC3)
 - o Receives UNICAST messages from PC3 (Timeout if not: 30.000ms)
 - o Prints statement with message in system.out

PC3

- UNICAST Receiver AND MULTICAST Sender Thread (PC3_PC2)
 - o Receives message from PC2 via UNICAST
 - o Sends confirmation via UNICAST to PC2
 - o 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