MODULE *service*12

Follow up from "*service*11", with verification $[V]$ conditions added

Changes: Add $P9$ - $P14$

EXTENDS *Naturals*, *Sequences*, *TLC*, *FiniteSets*

CONSTANTS $Node,\ NoNode,$
$UserTask,\ UserTaskL(\_),$
$KernelTask,\ KernelTaskL(\_),$
$TxTask,\ TxTaskL(\_),\ HLinkTx(\_),$
$RxTask,\ RxTaskL(\_),\ RxHLink(\_),$ $HLinkRx(\_),$
$HLink,\ NodeConnect(\_),$
$RouterHL(\_,\ \_),\ RouterTx(\_,\ \_),\ RouterRx(\_,\ \_),$
$Port,\ PortL(\_),$
$Message,\ NoMessage,$
$EmptyPacket,$
$RxAdr,\ RxAdrL(\_)$

DEFINITIONS

$L0ServiceID \triangleq \{$"start", "stop", "suspend", "resume", "acknw", "send", "receive"$\}$ $\backslash *$,"acksnd","ackrcv"}

$L0Status \triangleq \{$"ok", "fail", "failTO", "blank"$\}$

$TaskState \triangleq \{$"reserved", "inactive", "started"$\}$

$Task \triangleq UserTask \cup KernelTask \cup TxTask \cup RxTask$

$Packet \triangleq [service : L0ServiceID,$
$requestingtask : UserTask,$
$destination : Port \cup UserTask,$
$status : L0Status,$
$data : Message \cup \{NoMessage\},$
$memoryaddress : UserTask \cup RxAdr$ The so-called "OWNER" field!!!
$]$

$TaskCB \triangleq [taskstate : TaskState,$
$issuspended : \{0,\ 1\}$ $0 = NotSuspended$; $1 =$ Suspended
$]$

$InitTaskCB \triangleq$ CHOOSE $tcb \in TaskCB : \land\ tcb.taskstate =$ "inactive"
$\land\ tcb.issuspended = 0$

$StartedTaskCB \triangleq$ CHOOSE $tcb \in TaskCB : \land\ tcb.taskstate =$ "started"
$\land\ tcb.issuspended = 0$

VARIABLES $TaskController,$
$UserTaskMem,$
$KernelInputPort,\ TxInputPort,$
$ReadyList,$
$Error,$
$InitSetup,$
$RxMem,\ RxInputPort,$
$WireMem,$
$PortInputPort$

$vars \triangleq \langle TaskController,\ UserTaskMem,\ KernelInputPort,\ ReadyList,\ Error,\ InitSetup,\ TxInputPort,\ RxMem,\ RxInputPor$

$TypeInvariant \triangleq$
$\land\ TaskController \in [Task \rightarrow TaskCB]$

$\land\ UserTaskMem \in [UserTask \rightarrow Packet \cup \{EmptyPacket\}]$

$\land\ KernelInputPort \in [KernelTask \rightarrow Seq(UserTask \cup RxAdr)]$

$\land\ TxInputPort \in [TxTask \rightarrow Seq(UserTask \cup RxAdr)]$

```
    $\land ReadyList \subseteq (Task)$

    $\land Error \in \{0, 1\}$

    $\land InitSetup \in \{0, 1\}$

    $\land RxMem \in [RxAdr \rightarrow Packet \cup \{EmptyPacket\}]$

    $\land RxInputPort \in [RxTask \rightarrow Seq(RxAdr)]$

    $\land WireMem \in [HLink \rightarrow [Pac\ : Packet \cup \{EmptyPacket\},$
                                  $Dest : Node \cup \{NoNode\}]]$

    $\land PortInputPort \in [Port \rightarrow Seq(UserTask \cup RxAdr)]$

$Init \triangleq$
    $\land TaskController = [t \in Task \mapsto InitTaskCB]$
    $\land UserTaskMem = [ut \in UserTask \mapsto EmptyPacket]$
    $\land KernelInputPort = [kt \in KernelTask \mapsto \langle\rangle]$
    $\land TxInputPort = [txt \in TxTask \mapsto \langle\rangle]$
    $\land ReadyList = \{\}$
    $\land Error = 0$
    $\land InitSetup = 0$
    $\land RxMem = [ra \in RxAdr \mapsto EmptyPacket]$
    $\land RxInputPort\ = [rxt \in RxTask \mapsto \langle\rangle]$
    $\land WireMem = [hl \in HLink \mapsto [Pac \mapsto EmptyPacket,$
                                       $Dest \mapsto NoNode]]$
    $\land PortInputPort\ = [p \in Port \mapsto \langle\rangle]$
--------------------------------------------------------------------------------
 Functions

$NotEmpty(seq) \triangleq$ IF $Len(seq) \geq 1$ THEN TRUE ELSE FALSE
$IsEmpty(seq) \triangleq$ IF $Len(seq)\ = 0$ THEN TRUE ELSE FALSE

--------------------------------------------------------------------------------
 "Services"

$ServeStartTask(requestPacket) \triangleq$
    LET
        $reqSource \triangleq requestPacket.requestingtask$
        $requestStatus \triangleq requestPacket.status$
        $destTask \triangleq requestPacket.destination$
        $destTaskState \triangleq TaskController[destTask].taskstate$

        $reqnode \triangleq UserTaskL(reqSource)$
        $destnode \triangleq UserTaskL(destTask)$

    IN
      IF $reqnode = destnode$ THEN   (local,local)

          IF $destTaskState =$ "inactive" THEN
               activate the $Task$
              $\land TaskController' = [TaskController$ EXCEPT $![destTask].taskstate =$ "started"$]$
               make task ready
              $\land ReadyList' = ReadyList \cup \{destTask\} \cup \{reqSource\}$
               "ok"'
              $\land UserTaskMem' = [UserTaskMem$ EXCEPT $![reqSource].status =$ "ok"$]$
              $\land$ UNCHANGED $\langle Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort\rangle$

          ELSE  packet was already "started"
             $\land UserTaskMem' = [UserTaskMem$ EXCEPT $![reqSource].status =$ "fail"$]$
             $\land ReadyList' = ReadyList \cup \{reqSource\}$
             $\land$ UNCHANGED $\langle TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPo$

        ELSE  (remote, local)
          LET
```

```
            RxEntry  ≜ requestPacket.memoryaddress
            TxToUse  ≜ RouterTx(destnode, reqnode)
        IN
          IF destTaskState = “inactive” THEN

               activate the Task
               ∧ TaskController′ = [TaskController EXCEPT ![destTask].taskstate = “started”]
               make task ready
               ∧ ReadyList′ = ReadyList ∪ {destTask} ∪ {TxToUse}

           Change the packet header to send acknowledgment!

               ∧ RxMem′ = [RxMem EXCEPT ![RxEntry].service = “acknw”,
                                        ![RxEntry].destination = reqSource,
                                        ![RxEntry].status = “ok”]

               ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToUse] = Append(@, RxEntry)]
               ∧ UNCHANGED ⟨UserTaskMem, Error, InitSetup, RxInputPort, WireMem, PortInputPort⟩

           ELSE   packet was already “started”

               ∧ ReadyList′ = ReadyList ∪ {TxToUse}

               ∧ RxMem′ = [RxMem EXCEPT ![RxEntry].service = “acknw”,
                                        ![RxEntry].destination = reqSource,
                                        ![RxEntry].status = “fail”]

              ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToUse] = Append(@, RxEntry)]

              ∧ UNCHANGED ⟨TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem, PortInputPort⟩


├────────────────────────────────────────────────────────────────────────┤


ServeAcknowledgment(requestPacket) ≜
    LET
       reqSource ≜ requestPacket.requestingtask
       destTask  ≜ requestPacket.destination
       reqowner  ≜ requestPacket.memoryaddress
    IN
      IF reqSource = destTask THEN   This is just a sanity check! Remove afterwards!


          IF reqowner ∈ UserTask THEN   (local,local)

              Set the task ready and ’clean’ the packet allocated to the usertask
              ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![reqSource] = EmptyPacket]
              ∧ ReadyList′ = ReadyList ∪ {reqSource}
              ∧ Error′ = 2
              ∧ UNCHANGED ⟨TaskController, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort⟩

          ELSE   (remote,local)

              Set the task ready, Clean the packet allocated to the usertask, and clean RxMem

              ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![reqSource] = EmptyPacket]
              ∧ ReadyList′ = ReadyList ∪ {reqSource}

              ∧ RxMem′ = [RxMem EXCEPT ![reqowner] = EmptyPacket]

              ∧ UNCHANGED ⟨TaskController, Error, InitSetup, TxInputPort, RxInputPort, WireMem, PortInputPort⟩


      ELSE   (remote, local)
         ∧ Error′ = 2
         ∧ UNCHANGED ⟨TaskController, UserTaskMem, ReadyList, InitSetup, TxInputPort, RxMem, RxInputPort, WireM
```

```
$ServeSend(requestPacket) \triangleq$
    LET
        $reqSource \triangleq requestPacket.memoryaddress$   $requestingtask$
!!!! TRADEOFF DECISION HERE!!!!: If $reqSource$ is read from the "owner field", then there's no difference between (local,local) and (remote,local). If 'owner field' wouldn't exist then we would have to make an IF $reqnode = destnode$ THEN ELSE and split the code. !!!!$BTW$!!!!: How would you know the $RxAdr$??? $ok$, $i$ know, the distinction would have to made in the $RxLoop$.
!!Same question for $SERVERECEIVE()$!!
        $destin \triangleq requestPacket.destination$
    IN

      IF $Len(PortInputPort[destin]) > 0$ THEN   Port is not empty
        LET
          $storedReq \triangleq Head(PortInputPort[destin])$
          $storedPac \triangleq$ IF $storedReq \in UserTask$ THEN $UserTaskMem[storedReq]$ ELSE $RxMem[storedReq]$
          $storedReqType \triangleq storedPac.service$
        IN
          IF $storedReqType =$ "receive" THEN   AND There's a complementary request waiting $\rightarrow$ $SYNCHRONIZE$!

              CASE $((reqSource \in UserTask) \wedge (storedReq \in UserTask)) \rightarrow$   (local,local)
1. Set the status of both requests to "ok" and COPY THE DATA FROM ONE PACKET TO THE OTHER!
2. Add both tasks to the $ReadyList$
3. Remove the stored request from the $PortInputPort$
                $\wedge\ UserTaskMem' = [UserTaskMem$ EXCEPT $![reqSource].status =$ "ok",
                                                           $![storedReq].status =$ "ok",
                                                           $![storedReq].data = UserTaskMem[reqSource].data]$
                $\wedge\ ReadyList' = ReadyList \cup \{reqSource\} \cup \{storedReq\}$
                $\wedge\ PortInputPort' = [PortInputPort$ EXCEPT $![destin] = Tail(@)]$
                $\wedge$ UNCHANGED $\langle TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem\rangle$

              $\Box\ ((reqSource \in UserTask) \wedge (storedReq \in RxAdr)) \quad \rightarrow$   (local,remote)
1. The requesting task is local $\rightarrow$ simply set its status "ok".
2. The stored task is remote $\rightarrow$ Update the packet (With the Data COPIED) and send the Acknowledgment through $Tx$
3. Add the requesting task and $Tx$ to the $ReadyList$
4. Append the request in $Tx$
5. Remove the stored request from the $PortInputPort$
                LET
                  $localnode \triangleq PortL(destin)$
                  $remnode \triangleq UserTaskL(RxMem[storedReq].requestingtask)$
                  $TxToUse \triangleq RouterTx(localnode, remnode)$
                IN
                  $\wedge\ UserTaskMem' = [UserTaskMem$ EXCEPT $![reqSource].status =$ "ok"$]$
                  $\wedge\ RxMem' = [RxMem$ EXCEPT $![storedReq].service =$ "acknw",
                                              $![storedReq].destination = RxMem[storedReq].requestingtask,$
                                              $![storedReq].status =$ "ok",
                                              $![storedReq].data = UserTaskMem[reqSource].data]$
                  $\wedge\ ReadyList' = ReadyList \cup \{reqSource\} \cup \{TxToUse\}$
                  $\wedge\ TxInputPort' = [TxInputPort$ EXCEPT $![TxToUse] = Append(@, storedReq)]$
                  $\wedge\ PortInputPort' = [PortInputPort$ EXCEPT $![destin] = Tail(@)]$
                  $\wedge$ UNCHANGED $\langle TaskController, Error, InitSetup, RxInputPort, WireMem\rangle$

              $\Box\ ((reqSource \in RxAdr) \wedge (storedReq \in UserTask)) \quad \rightarrow$   (remote,local)
1. The requesting task is remote $\rightarrow$ Update the packet (With the Data ERASED!!) and send the Acknowledgment through $Tx$
2. The stored task is local $\rightarrow$ $set$ its status "ok" and COPY the DATA
3. Add the stored task and $Tx$ to the $ReadyList$
4. Append the request in $Tx$
5. Remove the stored request from the $PortInputPort$
                LET
                  $localnode \triangleq PortL(destin)$
                  $remnode \triangleq UserTaskL(RxMem[reqSource].requestingtask)$
                TxToUse ≜ RouterTx(localnode, remnode)
            IN
              ∧ RxMem′ = [RxMem EXCEPT ![reqSource].service = "acknw",
                                       ![reqSource].destination = RxMem[reqSource].requestingtask,
                                       ![reqSource].status = "ok",
                                       ![reqSource].data = NoMessage]

              ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![storedReq].status = "ok",
                                                   ![storedReq].data = RxMem[reqSource].data]

              ∧ ReadyList′ = ReadyList ∪ {storedReq} ∪ {TxToUse}
              ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToUse] = Append(@, reqSource)]
              ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Tail(@)]
              ∧ UNCHANGED ⟨TaskController, Error, InitSetup, RxInputPort, WireMem⟩


          □ ((reqSource ∈ RxAdr) ∧ (storedReq ∈ RxAdr))     →  (remote,remote)
  1. The requesting task is remote → Update the packet (With the Data ERASED!!), send the Acknowledgment through Tx1
  2. The stored task is remote → Update the packet(With the Data COPIED!!), send the Acknowledgment through Tx2
  3. Remove the stored request from the PortInputPort
  4. Add Tx1 and Tx2 to the ReadyList (They may or not be the same!!!)
  5. Append the requests in Tx1 and Tx2
            LET
              localnode ≜ PortL(destin)
              remnodeReq ≜ UserTaskL(RxMem[reqSource].requestingtask)
              remnodeSto ≜ UserTaskL(RxMem[storedReq].requestingtask)
              TxToReq ≜ RouterTx(localnode, remnodeReq)
              TxToSto ≜ RouterTx(localnode, remnodeSto)
            IN
              ∧ RxMem′ = [RxMem EXCEPT ![reqSource].service = "acknw",
                                       ![reqSource].destination = RxMem[reqSource].requestingtask,
                                       ![reqSource].status = "ok",
                                       ![reqSource].data = NoMessage,

                                       ![storedReq].service = "acknw",
                                       ![storedReq].destination = RxMem[storedReq].requestingtask,
                                       ![storedReq].status = "ok",
                                       ![storedReq].data = RxMem[reqSource].data]

              ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Tail(@)]

              ∧ IF TxToReq = TxToSto THEN
                    ∧ ReadyList′ = ReadyList ∪ {TxToReq}
                    ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToReq] = Append(Append(@, reqSource), storedReq
                    ∧ UNCHANGED ⟨TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem⟩
                 ELSE
                    ∧ ReadyList′ = ReadyList ∪ {TxToReq} ∪ {TxToSto}
                    ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToReq] = Append(@, reqSource),
                                                         ![TxToSto] = Append(@, storedReq)]
                    ∧ UNCHANGED ⟨TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem⟩

  ———————— End of SYNCHRONIZATION ————————

        ELSE   There are requests in the queue, but NOT complementary → Store the request in the queue

          (local ∨ remote , local ∨ remote) (There's no difference)
          ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
          ∧ UNCHANGED ⟨TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInpu

    ELSE   The Port is Empty → Store the request in the queue (No difference from the previous!)

      (local ∨ remote , local ∨ remote) (There's no difference)
      ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
      ∧ UNCHANGED ⟨TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort,
├──────────────────────────────────────────────────────────────────────────┤

ServeReceive(requestPacket) ≜

    LET
       reqSource ≜ requestPacket.memoryaddress
       destin ≜ requestPacket.destination          ID of the port being used
    IN

      IF Len(PortInputPort[destin]) > 0 THEN   Port is not empty
         LET
            storedReq ≜ Head(PortInputPort[destin])
            storedPac ≜ IF storedReq ∈ UserTask THEN UserTaskMem[storedReq] ELSE RxMem[storedReq]
            storedReqType ≜ storedPac.service
         IN
           IF storedReqType = "send" THEN    THERE's a complementary request waiting → SYNCHRONIZE


               CASE ((reqSource ∈ UserTask) ∧ (storedReq ∈ UserTask)) →   (local,local)
  1. Set the status of both requests to "ok" and COPY THE DATA FROM ONE PACKET TO THE OTHER!
  2. Add both tasks to the ReadyList
  3. Remove the stored request from the PortInputPort
                  ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![reqSource].status = "ok",
                                                       ![reqSource].data = UserTaskMem[storedReq].data,
                                                       ![storedReq].status = "ok"]
                  ∧ ReadyList′ = ReadyList ∪ {reqSource} ∪ {storedReq}
                  ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Tail(@)]
                  ∧ UNCHANGED ⟨TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem⟩


               □ ((reqSource ∈ UserTask) ∧ (storedReq ∈ RxAdr))    →   (local,remote)
  1. The requesting task is local → set its status "ok" and COPY THE DATA FROM the PACKET !!!IN  RxMem!!!
  2. The stored task is remote → Update the packet and send the Acknowledgment through Tx
  3. Add the requesting task and Tx to the ReadyList
  4. Append the request in Tx
  5. Remove the stored request from the PortInputPort
                  LET
                     localnode ≜ PortL(destin)
                     remnode  ≜ UserTaskL(RxMem[storedReq].requestingtask)
                     TxToUse ≜ RouterTx(localnode, remnode)
                  IN
                    ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![reqSource].status = "ok",
                                                         ![reqSource].data = RxMem[storedReq].data]
                    ∧ RxMem′ = [RxMem EXCEPT ![storedReq].service = "acknw",
                                             ![storedReq].destination = RxMem[storedReq].requestingtask,
                                             ![storedReq].status = "ok",
                                             ![storedReq].data = NoMessage]
                           !!! Delete the data so that it doesn't have to be sent back!!!IS this worth??? It could also be
                           used to make a distinction between acknowledgments: WHENEVER There is Data Copy IT!
                    ∧ ReadyList′ = ReadyList ∪ {reqSource} ∪ {TxToUse}
                    ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToUse] = Append(@, storedReq)]
                    ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Tail(@)]
                    ∧ UNCHANGED ⟨TaskController, Error, InitSetup, RxInputPort, WireMem⟩


               □ ((reqSource ∈ RxAdr) ∧ (storedReq ∈ UserTask))    →   (remote,local)
  1. The requesting task is remote → Update the packet (WITH THE DATA COPIED!!) and send the Acknowledgment through
  Tx
  2. The stored task is local → simply set its status "ok"
  3. Add the stored task and Tx to the ReadyList
  4. Append the request in Tx
  5. Remove the stored request from the PortInputPort
                  LET
                     localnode ≜ PortL(destin)
```

```
                remnode  ≜  UserTaskL(RxMem[reqSource].requestingtask)
                TxToUse  ≜  RouterTx(localnode, remnode)
            IN
                ∧ RxMem′ = [RxMem EXCEPT ![reqSource].service = "acknw",
                                         ![reqSource].destination = RxMem[reqSource].requestingtask,
                                         ![reqSource].status = "ok",
                                         ![reqSource].data = UserTaskMem[storedReq].data]
                ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![storedReq].status = "ok"]
                ∧ ReadyList′ = ReadyList ∪ {storedReq} ∪ {TxToUse}
                ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToUse] = Append(@, reqSource)]
                ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Tail(@)]
                ∧ UNCHANGED ⟨TaskController, Error, InitSetup, RxInputPort, WireMem⟩

          □ ((reqSource ∈ RxAdr) ∧ (storedReq ∈ RxAdr))      →  (remote,remote)
  1. The requesting task is remote → Update the packet (WITH THE DATA COPIED!!), send the Acknowledgment through Tx1
  2. The stored task is remote → Update the packet(With the Data Erased(worth ?? )), send the Acknowledgment through Tx2
  3. Remove the stored request from the PortInputPort
  4. Add Tx1 and Tx2 to the ReadyList (They may or not be the same!!!)
  5. Append the requests in Tx1 and Tx2
            LET
                localnode  ≜  PortL(destin)
                remnodeReq  ≜  UserTaskL(RxMem[reqSource].requestingtask)
                remnodeSto  ≜  UserTaskL(RxMem[storedReq].requestingtask)
                TxToReq  ≜  RouterTx(localnode, remnodeReq)
                TxToSto  ≜  RouterTx(localnode, remnodeSto)
            IN
                ∧ RxMem′ = [RxMem EXCEPT ![reqSource].service = "acknw",
                                         ![reqSource].destination = RxMem[reqSource].requestingtask,
                                         ![reqSource].status = "ok",
                                         ![reqSource].data = RxMem[storedReq].data,

                                         ![storedReq].service = "acknw",
                                         ![storedReq].destination = RxMem[storedReq].requestingtask,
                                         ![storedReq].status = "ok",
                                         ![storedReq].data = NoMessage]

                ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Tail(@)]

                ∧ IF TxToReq = TxToSto THEN
                        ∧ ReadyList′ = ReadyList ∪ {TxToReq}
                        ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToReq] = Append(Append(@, reqSource), storedReq
                        ∧ UNCHANGED ⟨TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem⟩
                    ELSE
                        ∧ ReadyList′ = ReadyList ∪ {TxToReq} ∪ {TxToSto}
                        ∧ TxInputPort′ = [TxInputPort EXCEPT ![TxToReq] = Append(@, reqSource),
                                                             ![TxToSto] = Append(@, storedReq)]
                        ∧ UNCHANGED ⟨TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem⟩

  ————— End of SYNCHRONIZATION —————

        ELSE   There are requests in the queue, but NOT complementary → Store the request in the queue

          (local ∨ remote , local ∨ remote) (There's no difference)
          ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
          ∧ UNCHANGED ⟨TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInpu

    ELSE   The Port is Empty → Store the request in the queue (No difference from the previous!)

      (local ∨ remote , local ∨ remote) (There's no difference)
      ∧ PortInputPort′ = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
      ∧ UNCHANGED ⟨TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort,

├──────────────────────────────────────────────────────────────────────────┤
```

```
Actions

StartSystem ≜
 Wakes up the KernelTask, TxTask, RxTask and (in this example!!!) a UserTask
    ∧ InitSetup = 0
    ∧ LET
         firsttask ≜ CHOOSE ut ∈ UserTask : TRUE

         AllTasksToStart ≜ KernelTask ∪ TxTask ∪ RxTask ∪ {firsttask}

         SetToSeq(set) ≜
            LET TC[S ∈ SUBSET set] ≜ LET elt ≜ CHOOSE e ∈ S : TRUE
                                     IN  IF S = {} THEN ⟨⟩
                                                   ELSE Append(TC[S \ {elt}], elt)
            IN   TC[set]

         SeqTasks ≜ SetToSeq(AllTasksToStart)

      IN

         ∧ TaskController′ = [t ∈ Task ↦ IF t ∈ AllTasksToStart THEN StartedTaskCB ELSE InitTaskCB]

  !!! TxTask is not put in the ReadyList, the KernelTask will do it when necessary !!!
         ∧ ReadyList′ = KernelTask ∪ RxTask ∪ {firsttask}
         ∧ InitSetup′ = 1
         ∧ UNCHANGED ⟨UserTaskMem, KernelInputPort, Error, TxInputPort, RxMem, RxInputPort, WireMem, PortInpu

──────────────────────────────────────────────────────────────────────────────

CreateStartTaskRequest(reqtask, desttask) ≜
    ∧ InitSetup = 1
  ∧ TaskController[reqtask].taskstate = “started”    !!! Pay attention to this! In this case the kernel could indefenetly accumulate requests, overv
    ∧ reqtask ∈ ReadyList
    ∧ LET
         requestPacket ≜ CHOOSE p ∈ Packet : ∧ p.service = “start”
                                             ∧ p.requestingtask = reqtask
                                             ∧ p.destination = desttask
                                             ∧ p.status = “blank”
                                             ∧ p.data = NoMessage
                                             ∧ p.memoryaddress = reqtask

         localkernel ≜ CHOOSE k ∈ KernelTask : KernelTaskL(k) = UserTaskL(reqtask)
      IN
        ∧ UserTaskMem′ = [UserTaskMem EXCEPT ![reqtask] = requestPacket]
        ∧ KernelInputPort′ = [KernelInputPort EXCEPT ![localkernel] = Append(@, reqtask)]
        ∧ ReadyList′ = ReadyList \ {reqtask}
        ∧ UNCHANGED ⟨TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort⟩

──────────────────────────────────────────────────────────────────────────────

AllStartedUserTasks ≜ {ut ∈ UserTask : TaskController[ut].taskstate = “started”}
AllReadyUserTasks ≜ {ut ∈ UserTask : ut ∈ ReadyList}


CreateSendRequest(reqtask, port, message) ≜
    ∧ InitSetup = 1
!!!NEED TO CHECK THIS!!!THERE CAN NOT BE ONLY A STARTED TASK, OTHERWISE NO COMPLEMENTARY
REQUEST WILL EVER BE CREATED AND THE TASK IS BLOCKED FOR EVER!
    ∧ Cardinality(AllStartedUserTasks) > 1
    ∧ Cardinality(AllReadyUserTasks) > 1   !!NOT NECESSARILY!! If there would be complementary requests already generated we wou

    ∧ reqtask ∈ ReadyList
    ∧ LET
         requestPacket ≜ CHOOSE p ∈ Packet : ∧ p.service = “send”
                                             ∧ p.requestingtask = reqtask
                                             ∧ p.destination = port
```

```
                                                        ∧ p.status = "blank"
                                                        ∧ p.data = message
                                                        ∧ p.memoryaddress = reqtask

              localkernel ≜ CHOOSE k ∈ KernelTask : KernelTaskL(k) = UserTaskL(reqtask)
           IN
              ∧ UserTaskMem' = [UserTaskMem EXCEPT ![reqtask] = requestPacket]
              ∧ KernelInputPort' = [KernelInputPort EXCEPT ![localkernel] = Append(@, reqtask)]
              ∧ ReadyList' = ReadyList \ {reqtask}
              ∧ UNCHANGED ⟨TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort⟩


CreateReceiveRequest(reqtask, port) ≜
      ∧ InitSetup = 1
!!!NEED TO CHECK THIS!!!THERE CAN NOT BE ONLY A STARTED TASK, OTHERWISE NO COMPLEMENTARY
REQUEST WILL EVER BE CREATED AND THE TASK IS BLOCKED FOR EVER!
      ∧ Cardinality(AllStartedUserTasks) > 1
      ∧ Cardinality(AllReadyUserTasks) > 1   !!NOT NECESSARILY!! If there would be complementary requests already generated we wo

      ∧ reqtask ∈ ReadyList
      ∧ LET
              requestPacket ≜ CHOOSE p ∈ Packet : ∧ p.service = "receive"
                                                        ∧ p.requestingtask = reqtask
                                                        ∧ p.destination = port
                                                        ∧ p.status = "blank"
                                                        ∧ p.data = NoMessage
                                                        ∧ p.memoryaddress = reqtask

              localkernel ≜ CHOOSE k ∈ KernelTask : KernelTaskL(k) = UserTaskL(reqtask)
           IN
              ∧ UserTaskMem' = [UserTaskMem EXCEPT ![reqtask] = requestPacket]
              ∧ KernelInputPort' = [KernelInputPort EXCEPT ![localkernel] = Append(@, reqtask)]
              ∧ ReadyList' = ReadyList \ {reqtask}
              ∧ UNCHANGED ⟨TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort⟩

├──────────────────────────────────────────────────────────────────────────────┤

KernelLoop(localnode) ≜
                  ∧ InitSetup = 1
       ∧ IF  IsEmpty(KernelInputPort) THEN
           UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireM

         ELSE  \* Serve the pending request
                  ∧ LET
                        localkernel ≜ CHOOSE k ∈ KernelTask : KernelTaskL(k) = localnode
                     IN
                        ∧ Len(KernelInputPort[localkernel]) > 0

                        ∧ KernelInputPort' = [KernelInputPort EXCEPT ![localkernel] = Tail(@)]
                        ∧ LET
                              requestEntry ≜ Head(KernelInputPort[localkernel])
                              requestPacket ≜ IF requestEntry ∈ UserTask THEN UserTaskMem[requestEntry] ELSE  RxMem[request
                              destin ≜ requestPacket.destination
                              destnode ≜ IF destin ∈ UserTask THEN UserTaskL(destin) ELSE  PortL(destin)
                           IN
                              IF destnode ≠ localnode THEN   The request is for a remote usertask
                                 LET
                                    TxToUse ≜ RouterTx(localnode, destnode)
                                 IN
                                    ∧ ReadyList' = ReadyList ∪ {TxToUse}
                                    ∧ TxInputPort' = [TxInputPort EXCEPT ![TxToUse] = Append(@, requestEntry)]
                                    ∧ UNCHANGED ⟨TaskController, UserTaskMem, Error, InitSetup, RxMem, RxInputPort, WireMen

                              ELSE   The request is for a local usertask
```

```
                          LET
                             requestService ≜ requestPacket.service
                          IN
                             ∧ CASE requestService = "start"    → ServeStartTask(requestPacket)
                                 □requestService   = "acknw"    → ServeAcknowledgment(requestPacket)
                                 □requestService   = "send"     → ServeSend(requestPacket)
                                 □requestService   = "receive"  → ServeReceive(requestPacket)
                             Add the other services . . .


TxSendsPacketAway(txtask) ≜
    ∧ txtask ∈ ReadyList
    ∧ LET
          reqSource ≜ Head(TxInputPort[txtask])
          HLinkToUse ≜ HLinkTx(txtask)
      IN

        ∧ WireMem[HLinkToUse].Pac = EmptyPacket   !!! See how to REMOVE THIS!!!!!

      ∧ Read packet "through the pointer" and copy it directly to the Wire.

        ∧ TxInputPort′ = [TxInputPort EXCEPT ![txtask] = Tail(@)]

        ∧ IF reqSource  ∈ UserTask THEN   The request is from a local task: simply put in on wire!
                LET
 !!!!Isto devia SAIR daqui!Sabemos Tx ⇒ Sabemos Destination!REVER 'Modelo de Rede′!!!!
                    reqPacket ≜ UserTaskMem[reqSource]
                    destin ≜ reqPacket.destination
                    destnode ≜ IF destin ∈ UserTask THEN UserTaskL(destin) ELSE PortL(destin)
                IN
                    ∧ WireMem′ = [WireMem EXCEPT ![HLinkToUse].Pac = reqPacket,
                                                 ![HLinkToUse].Dest = destnode]

              ∧ IF  IsEmpty (TxInputPort[txtask]′)  THEN
                    ∧ IF Len(TxInputPort[txtask]) = 1 THEN
                          ∧ ReadyList′ = ReadyList \ {txtask}
                          ∧ UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, Error, InitSetup, RxMem, RxInp
                      ELSE
                          UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, RxMem

             ELSE    reqSource ∈ RxAdr ⇒  The request is remote: !!!!!!!Make RxMem free AND Update Request Packet!!!!!!!

                LET
                    reqPacket ≜ RxMem[reqSource]
                    destin ≜ reqPacket.destination
                    destnode ≜ IF destin ∈ UserTask THEN UserTaskL(destin) ELSE PortL(destin)
                IN
                    ∧ WireMem′ = [WireMem EXCEPT ![HLinkToUse].Pac = reqPacket,
                                                 ![HLinkToUse].Dest = destnode]
                    ∧ RxMem′ = [RxMem EXCEPT ![reqSource] = EmptyPacket]
              ∧ IF  IsEmpty (TxInputPort′[txtask])  THEN
                    ∧ IF Len(TxInputPort[txtask]) = 1 THEN
                          ∧ ReadyList′ = ReadyList \ {txtask}
                          ∧ UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, Error, InitSetup, RxInputPort, P
                      ELSE
                          UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, RxInpu


HLinkPutsPacketInRx(hlink) ≜
    ∧ InitSetup = 1

    ∧ WireMem[hlink].Pac ∈ Packet   Not an EmptyPacket

    ∧ LET
```

```
          reqPacket ≜ WireMem[hlink].Pac
          destnode ≜ WireMem[hlink].Dest
          RxToUse ≜ CHOOSE rx ∈ RxTask : (rx ∈ RxHLink(hlink) ∧ RxTaskL(rx) = destnode)
        IN
  !!! Like this it's always guaranteed that a packet will only arrive when there's memory available for it!!! What to do otherwise???
          ∧ ∃ ra ∈ RxAdr : (RxMem[ra] = EmptyPacket  ∧ RxAdrL(ra) = destnode)
          ∧ LET
                RxAdrToFill ≜ CHOOSE ra ∈ RxAdr : (RxMem[ra] = EmptyPacket ∧ RxAdrL(ra) = destnode)
            IN
              ∧ RxMem′ = [RxMem EXCEPT ![RxAdrToFill] = reqPacket]
              ∧ RxInputPort′ = [RxInputPort EXCEPT ![RxToUse] = Append(@, RxAdrToFill)]
              ∧ WireMem′ = [WireMem EXCEPT ![hlink].Pac = EmptyPacket,
                                           ![hlink].Dest = NoNode]
              ∧ UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, P


RxLoop(rxtask) ≜
    ∧ InitSetup = 1
      ∧ IF  IsEmpty(RxInputPort) THEN
         UNCHANGED ⟨TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireM

        ELSE  \* Send the requests to the KernelInputPort
    ∧ LET
          localnode ≜ RxTaskL(rxtask)
          localkernel ≜ CHOOSE k ∈ KernelTask : KernelTaskL(k) = localnode
      IN
        ∧ Len(RxInputPort[rxtask]) > 0
        ∧     LET
                RxMemEntry ≜ Head(RxInputPort[rxtask])
                reqPacket ≜ RxMem[RxMemEntry]
                UpdatedReqPacket ≜ [reqPacket EXCEPT !.memoryaddress = RxMemEntry]
              IN
  !!!!!!!!!!!!!!Update the Packet in RxMem!!!!!!!!!!!!
                ∧ RxMem′ = [RxMem EXCEPT ![RxMemEntry] = UpdatedReqPacket]
  The pointer to the packet is moved from the RxInputPort to the KernelInputport
                ∧ KernelInputPort′ = [KernelInputPort EXCEPT ![localkernel] = Append(@, RxMemEntry)]
                ∧ RxInputPort′ = [RxInputPort EXCEPT ![rxtask] = Tail(@)]
                ∧ UNCHANGED ⟨TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, WireMem, Por

├──────────────────────────────────────────────────────────────────────┤
  Summary

Next ≜
    ∨ StartSystem

    ∨ ∃ n ∈ Node : KernelLoop(n)

    ∨ ∃ txt ∈ TxTask : TxSendsPacketAway(txt)

    ∨ ∃ rxt ∈ RxTask : RxLoop(rxt)

    ∨ ∃ hl ∈ HLink : HLinkPutsPacketInRx(hl)

    ∨ ∃ uts ∈ UserTask :
        ∨ ∃ utd ∈ UserTask :
              CreateStartTaskRequest(uts, utd)

        ∨ ∃ p ∈ Port :
              ∨ ∃ d ∈ Message :
                    CreateSendRequest(uts, p, d)

              ∨ CreateReceiveRequest(uts, p)

├──────────────────────────────────────────────────────────────────────┤
```

```
LivenessW ≜
     ∧ WF⟨vars⟩(StartSystem)
     ∧ ∀ n ∈ Node : WF⟨vars⟩(KernelLoop(n))
     ∧ ∀ txt ∈ TxTask : WF⟨vars⟩(TxSendsPacketAway(txt))
     ∧ ∀ rxt ∈ RxTask : WF⟨vars⟩(RxLoop(rxt))
     ∧ ∀ hl ∈ HLink : WF⟨vars⟩(HLinkPutsPacketInRx(hl))
     ∧ ∀ uts ∈ UserTask :
          ∧ ∀ utd ∈ UserTask :
                WF⟨vars⟩(CreateStartTaskRequest(uts, utd))
          ∧ ∀ p ∈ Port :
               ∧ ∀ d ∈ Message :
                    WF⟨vars⟩(CreateSendRequest(uts, p, d))
               ∧ WF⟨vars⟩(CreateReceiveRequest(uts, p))


Spec ≜ Init ∧ □[Next]_vars ∧ LivenessW

├──────────────────────────────────────────────────────────────────────┤
  Correctness conditions


  All tasks in the ReadyList have to be "started"
P2 ≜ ∀ t ∈ Task : t ∈ ReadyList ⇒ TaskController[t].taskstate = "started"


  TxTask is in the ReadyList if and only if it has requests to serve
P4 ≜ ∀ txt ∈ TxTask : txt ∈ ReadyList ≡ Len(TxInputPort[txt]) > 0

  RxTask is always in the ReadyList (after initialization)
P5 ≜ InitSetup = 1 ⇒ ∀ rxt ∈ RxTask : rxt ∈ ReadyList


  A PortInputPort never has complementary requests
P9 ≜ LET
          pacsto(x) ≜ IF x ∈ UserTask THEN UserTaskMem[x] ELSE RxMem[x]
        IN
         ∀ p ∈ Port : Len(PortInputPort[p]) > 1 ⇒ ∀ i, j ∈ 1 . . Len(PortInputPort[p]) :
                 pacsto(PortInputPort[p][i]).service = pacsto(PortInputPort[p][j]).service

  A Port only gets "send" and "receive" requests
P10 ≜ LET
          pacsto(x) ≜ IF x ∈ UserTask THEN UserTaskMem[x] ELSE RxMem[x]
        IN
          ∀ p ∈ Port : Len(PortInputPort[p]) > 0 ⇒ ∀ i ∈ 1 . . Len(PortInputPort[p]) :
                 pacsto(PortInputPort[p][i]).service ∈ {"send", "receive"}



  Queues only have distinct elements

P11 ≜ ∧ ∀ p ∈ Port : Len(PortInputPort[p]) > 1 ⇒ ∀ i, j ∈ 1 . . Len(PortInputPort[p]) :
                 PortInputPort[p][i] = PortInputPort[p][j] ⇒ i = j


P12 ≜ ∧ ∀ rx ∈ RxTask : Len(RxInputPort[rx]) > 1 ⇒ ∀ i, j ∈ 1 . . Len(RxInputPort[rx]) :
                 RxInputPort[rx][i] = RxInputPort[rx][j] ⇒ i = j

  Queues only have distinct elements —— A request in never repeated in any two places (InputPorts) at the same time
P13 ≜ LET
           SepSeq(seq) ≜
              LET YY[F ∈ SUBSET DOMAIN seq] ≜
                   LET ind ≜ CHOOSE nr ∈ F : TRUEIN
                       IF F = {} THEN ⟨⟩
                                 ELSE Append(YY[F \ {ind}], seq[ind])
              IN YY[DOMAIN seq]

           TotalQ(set, inputport) ≜
```

```
                  LET XX[S ∈ SUBSET set] ≜
                         LET elt ≜ CHOOSE e ∈ S : TRUEIN
                                IF S = {} THEN ⟨⟩
                                              ELSE
                                                  IF Len(inputport[elt]) > 0 THEN
                                                       Append(XX[S \ {elt}], SepSeq(inputport[elt]))
                                                   ELSE  XX[S \ {elt}]
                  IN    XX[set]

             AllKerQ   ≜ TotalQ(KernelTask, KernelInputPort)

             AllRxQ    ≜ TotalQ(RxTask, RxInputPort)

             AllTxQ    ≜ TotalQ(TxTask, TxInputPort)

             AllPortQ  ≜ TotalQ(Port, PortInputPort)

             GeneralQ  ≜ AllKerQ ∘ AllRxQ ∘ AllTxQ ∘ AllPortQ

        IN
           Len(GeneralQ) > 1 ⇒ ∀ i, j ∈ 1 .. Len(GeneralQ) :
                        GeneralQ[i] = GeneralQ[j] ⇒ i = j


  "Every XInputPort can contain only local pointers"
P14 ≜ LET
             location(a) ≜ IF a ∈ UserTask THEN UserTaskL(a) ELSE  RxAdrL(a)
        IN
            ∧ ∀ p ∈ Port : Len(PortInputPort[p]) > 0 ⇒ ∀ i ∈ 1 .. Len(PortInputPort[p]) :
                                      location(PortInputPort[p][i]) = PortL(p)

            ∧ ∀ k ∈ KernelTask : Len(KernelInputPort[k]) > 0 ⇒ ∀ i ∈ 1 .. Len(KernelInputPort[k]) :
                                        location(KernelInputPort[k][i]) = KernelTaskL(k)

            ∧ ∀ txt ∈ TxTask : Len(TxInputPort[txt]) > 0 ⇒ ∀ i ∈ 1 .. Len(TxInputPort[txt]) :
                                        location(TxInputPort[txt][i]) = TxTaskL(txt)

            ∧ ∀ rxt ∈ RxTask : Len(RxInputPort[rxt]) > 0 ⇒ ∀ i ∈ 1 .. Len(RxInputPort[rxt]) :
                                        location(RxInputPort[rxt][i]) = RxTaskL(rxt)
```