```
──────────────────── MODULE net_pcal ────────────────────
EXTENDS Naturals, TLC, Bags

CONSTANTS
  Messages,
  MaxSamePackets,
  MessagesToSend

ASSUME
  MessagesToSend ⊆ Messages

   --algorithm net_pcal
variables
  network = EmptyBag,
  outbox = MessagesToSend,
  processed = {}

define
    Type definitions
  IdReq ≜ "req"
  IdRep ≜ "rep"
  ReqPackets ≜ [type : {IdReq}, msg : Messages]
  RepPackets ≜ [type : {IdRep}, msg : Messages]
  Packets ≜ ReqPackets ∪ RepPackets

  TypeInvariants ≜ ∧ IsABag(network)
                   ∧ BagToSet(network) ⊆ Packets
                   ∧ outbox ⊆ Messages
                   ∧ processed ⊆ Messages

    Utility
  Req(m) ≜ [type ↦ IdReq, msg ↦ m]
  Rep(m) ≜ [type ↦ IdRep, msg ↦ m]
  Sent(type) ≜ {p ∈ BagToSet(network) : p ∈ type}

    Completion
  Completed ≜ ∧ processed = MessagesToSend
              ∧ outbox = {}

  EventuallyCompleted ≜ ◇□Completed
end define

  Network communication
macro Comm(in, out)
begin
  network := LET LimitPackets(net) ≜
                [p ∈ BagToSet(net) ↦ IF CopiesIn(p, net) > MaxSamePackets
                                      THEN MaxSamePackets
```

```
                                                  ELSE  $CopiesIn(p, net)]$
               IN   $LimitPackets(network \ominus SetToBag(in) \oplus SetToBag(out))$
end macro

fair process $send\_request \in Messages$
begin
  $send\_request$:
  while TRUE do
    await $self \in outbox$ ;
    $Comm(\{\}, \{Req(self)\})$
  end while
end process

fair +process $recv\_request \in ReqPackets$
begin
  $recv\_request$:
  while TRUE do
    await $self \in Sent(ReqPackets)$ ;
    $Comm(\{self\}, \{Rep(self.msg)\})$ ;
    $processed := processed \cup \{self.msg\}$
  end while
end process

fair +process $recv\_reply \in RepPackets$
begin
  $recv\_reply$:
  while TRUE do
    await $self \in Sent(RepPackets)$ ;
    $Comm(\{self\}, \{\})$ ;
    $outbox := outbox \setminus \{self.msg\}$
  end while
end process

process $lose\_packet =$ "lose_packet"
begin
  $lose\_packet$:
  while TRUE do
    with $lost\_p \in Sent(Packets)$ do
      $Comm(\{lost\_p\}, \{\})$
    end with
  end while
end process

end algorithm
```

BEGIN TRANSLATION

Label *send_request* of process *send_request* at line 56 col 3 changed to *send_request_*

Label *recv_request* of process *recv_request* at line 65 col 3 changed to *recv_request_*

Label *recv_reply* of process *recv_reply* at line 75 col 3 changed to *recv_reply_*
Label *lose_packet* of process *lose_packet* at line 85 col 3 changed to *lose_packet_*

```
VARIABLES $network, outbox, processed$

define statement
$IdReq \triangleq$ "req"
$IdRep \triangleq$ "rep"
$ReqPackets \triangleq [type : \{IdReq\}, msg : Messages]$
$RepPackets \triangleq [type : \{IdRep\}, msg : Messages]$
$Packets \triangleq ReqPackets \cup RepPackets$

$TypeInvariants \triangleq$ $\land IsABag(network)$
                  $\land BagToSet(network) \subseteq Packets$
                  $\land outbox \subseteq Messages$
                  $\land processed \subseteq Messages$

$Req(m) \triangleq [type \mapsto IdReq, msg \mapsto m]$
$Rep(m) \triangleq [type \mapsto IdRep, msg \mapsto m]$
$Sent(type) \triangleq \{p \in BagToSet(network) : p \in type\}$

$Completed \triangleq$ $\land processed = MessagesToSend$
             $\land outbox = \{\}$

$EventuallyCompleted \triangleq \Diamond\Box Completed$

$vars \triangleq \langle network, outbox, processed\rangle$

$ProcSet \triangleq (Messages) \cup (ReqPackets) \cup (RepPackets) \cup \{$"lose_packet"$\}$

$Init \triangleq$ Global variables
          $\land network = EmptyBag$
          $\land outbox = MessagesToSend$
          $\land processed = \{\}$

$send\_request(self) \triangleq$ $\land self \in outbox$
                   $\land network' = ($LET $LimitPackets(net) \triangleq$
                                $[p \in BagToSet(net) \mapsto$ IF $CopiesIn(p, net) > MaxSamePackets$
                                                  THEN $MaxSamePackets$
                                                  ELSE $CopiesIn(p, net)]$
                            IN $LimitPackets(network \ominus SetToBag((\{\})) \oplus SetToBag((\{Req(self)\})$
                   $\land$ UNCHANGED $\langle outbox, processed\rangle$

$recv\_request(self) \triangleq$ $\land self \in Sent(ReqPackets)$
                   $\land network' = ($LET $LimitPackets(net) \triangleq$
                                $[p \in BagToSet(net) \mapsto$ IF $CopiesIn(p, net) > MaxSamePackets$
                                                  THEN $MaxSamePackets$
```

```
                                                          ELSE  CopiesIn(p, net)]
                                 IN   LimitPackets(network ⊖ SetToBag(({self})) ⊕ SetToBag(({Rep(sel
                      ∧ processed′ = (processed ∪ {self.msg})
                      ∧ UNCHANGED outbox

recv_reply(self) ≜ ∧ self ∈ Sent(RepPackets)
                   ∧ network′ = (LET LimitPackets(net) ≜
                                    [p ∈ BagToSet(net) ↦ IF CopiesIn(p, net) > MaxSamePackets
                                                          THEN MaxSamePackets
                                                          ELSE  CopiesIn(p, net)]
                                 IN   LimitPackets(network ⊖ SetToBag(({self})) ⊕ SetToBag(({}))))
                   ∧ outbox′ = outbox \ {self.msg}
                   ∧ UNCHANGED processed

lose_packet ≜ ∧ ∃ lost_p ∈ Sent(Packets) :
                   network′ = (LET LimitPackets(net) ≜
                                  [p ∈ BagToSet(net) ↦ IF CopiesIn(p, net) > MaxSamePackets
                                                        THEN MaxSamePackets
                                                        ELSE  CopiesIn(p, net)]
                               IN   LimitPackets(network ⊖ SetToBag(({lost_p})) ⊕ SetToBag(({}))))
              ∧ UNCHANGED ⟨outbox, processed⟩

Next ≜ lose_packet
          ∨ (∃ self ∈ Messages : send_request(self))
          ∨ (∃ self ∈ ReqPackets : recv_request(self))
          ∨ (∃ self ∈ RepPackets : recv_reply(self))

Spec ≜ ∧ Init ∧ □[Next]_vars
       ∧ ∀ self ∈ Messages : WF_vars(send_request(self))
       ∧ ∀ self ∈ ReqPackets : SF_vars(recv_request(self))
       ∧ ∀ self ∈ RepPackets : SF_vars(recv_reply(self))

END TRANSLATION
```