```
┌──────────────────────────── MODULE net ────────────────────────────┐
EXTENDS Naturals, Bags

CONSTANTS
  Messages,
  MaxSamePackets,
  MessagesToSend

ASSUME
  MessagesToSend ⊆ Messages

VARIABLES
  network,
  outbox,
  processed

vars ≜ ⟨network, outbox, processed⟩

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

Init ≜ ∧ network = EmptyBag
       ∧ outbox = MessagesToSend
       ∧ processed = {}

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

Req(m) ≜ [type ↦ IdReq, msg ↦ m]
Rep(m) ≜ [type ↦ IdRep, msg ↦ m]

Comm(in, out) ≜ LET LimitPackets(net) ≜
                      [p ∈ BagToSet(net) ↦ IF CopiesIn(p, net) > MaxSamePackets
                                            THEN MaxSamePackets
                                            ELSE CopiesIn(p, net)]
                IN   network' = LimitPackets(network ⊖ SetToBag(in) ⊕ SetToBag(out))

Sent(type) ≜ {p ∈ BagToSet(network) : p ∈ type}

SendRequest(m) ≜ ∧ m ∈ outbox
                 ∧ Comm({}, {Req(m)})
                 ∧ UNCHANGED ⟨outbox, processed⟩

RecvRequest(p) ≜ ∧ p ∈ Sent(ReqPackets)
```

$$\begin{array}{l}
\land\ \mathit{Comm}(\{p\},\ \{\mathit{Rep}(p.\mathit{msg})\}) \\
\land\ \mathit{processed}' = \mathit{processed} \cup \{p.\mathit{msg}\} \\
\land\ \textsc{unchanged}\ \langle \mathit{outbox} \rangle
\end{array}$$

$$\mathit{RecvReply}(p) \triangleq \begin{array}{l}
\land\ p \in \mathit{Sent}(\mathit{RepPackets}) \\
\land\ \mathit{Comm}(\{p\},\ \{\}) \\
\land\ \mathit{outbox}' = \mathit{outbox} \setminus \{p.\mathit{msg}\} \\
\land\ \textsc{unchanged}\ \langle \mathit{processed} \rangle
\end{array}$$

$$\begin{array}{l}
\mathit{LosePacket} \triangleq \exists\, p \in \mathit{Sent}(\mathit{Packets}) : \\
\qquad\qquad\quad \land\ \mathit{Comm}(\{p\},\ \{\}) \\
\qquad\qquad\quad \land\ \textsc{unchanged}\ \langle \mathit{outbox},\ \mathit{processed} \rangle
\end{array}$$

$$\mathit{Next} \triangleq \begin{array}{l}
\lor\ \exists\, m \in \mathit{Messages} : \mathit{SendRequest}(m) \\
\lor\ \exists\, p\ \ \in \mathit{ReqPackets} : \mathit{RecvRequest}(p) \\
\lor\ \exists\, p\ \ \in \mathit{RepPackets} : \mathit{RecvReply}(p) \\
\lor\ \mathit{LosePacket}
\end{array}$$

$$\mathit{Spec} \triangleq \begin{array}{l}
\land\ \mathit{Init} \\
\land\ \Box[\mathit{Next}]_{\mathit{vars}} \\
\land\ \forall\, m \in \mathit{Messages} : \mathrm{WF}_{\mathit{vars}}(\mathit{SendRequest}(m)) \\
\land\ \forall\, p\ \ \in \mathit{ReqPackets} : \mathrm{SF}_{\mathit{vars}}(\mathit{RecvRequest}(p)) \\
\land\ \forall\, p\ \ \in \mathit{RepPackets} : \mathrm{SF}_{\mathit{vars}}(\mathit{RecvReply}(p))
\end{array}$$

$$\mathit{Completed} \triangleq \begin{array}{l}
\land\ \mathit{processed} = \mathit{MessagesToSend} \\
\land\ \mathit{outbox} = \{\}
\end{array}$$

$$\mathit{EventuallyCompleted} \triangleq \Diamond\Box\mathit{Completed}$$