```
MODULE P2PNetwork
EXTENDS $Integers, Sequences, TLC, Naturals, FiniteSets$

CONSTANTS $ControllerNode, AuthorityNode, Payloads, MessageType$
$Nodes \triangleq \{ControllerNode, AuthorityNode\}$
VARIABLES $messages, alerts$
$RemoveElement(s, e) \triangleq$
    IF $\exists i \in 1 \,..\, Len(s) : s[i] = e$
     THEN
        LET $index \triangleq$ CHOOSE $i \in 1 \,..\, Len(s) : s[i] = e$
        IN  $SubSeq(s, 1, index - 1) \circ SubSeq(s, index + 1, Len(s))$
     ELSE  $s$  If element not found, return original sequence



 Action
$SendAlert(payload) \triangleq$
  LET $newMsg \triangleq [src \mapsto ControllerNode, dest \mapsto AuthorityNode, type \mapsto$ "alert", $data \mapsto payload]$
  IN
     $\land \neg(\exists i \in 1 \,..\, Len(messages) : messages[i] = newMsg)$
     $\land messages' = Append(messages, newMsg)$
     $\land$ UNCHANGED $alerts$

$Deliver \triangleq$
  $\exists i \in 1 \,..\, Len(messages) :$
     $\land messages[i].dest = AuthorityNode$
     $\land alerts' = alerts \cup \{messages[i]\}$
     $\land messages' = RemoveElement(messages, messages[i])$


$Init \triangleq$
   $\land messages = \langle\rangle$
   $\land alerts = \{\}$

$Next \triangleq \exists payload \in Payloads :$ $\lor SendAlert(payload)$
                                      $\lor Deliver$

$vars \triangleq \langle messages, alerts\rangle$
$Spec \triangleq Init \land \Box[Next]_{vars}$

 Invariants
$NoDuplicateMessages \triangleq$
  $\forall i, j \in 1 \,..\, Len(messages) :$
     $i \neq j \Rightarrow messages[i] \neq messages[j]$

$TypeOK \triangleq$
   $\land messages \in Seq([src : Nodes, dest : Nodes, type : MessageType, data : Payloads])$
   $\land alerts \subseteq [src$          $: Nodes, dest : Nodes, type : MessageType, data : Payloads]$
```

```
(* $RouteMessage \triangleq$
  $\exists\, msg \in messages$:
    $\exists\, n \in Nodes$:
      $\land\ RoutingTable(n,\ msg.dest) = NextHop$
      $\land\ messages' = (messages \setminus \{msg\}) \cup \{msg \setminus \{src \mapsto NextHop\}\}$
      $\land$ routes' = routes $\cup\, \{[msg \mapsto NextHop]\}$
      $\land\ alerts' = alerts$ *)

(* $MessageDeliveryInvariant \triangleq$
  $\forall\, msg \in \text{DOMAIN}$ routes:
    $\exists\, n \in Nodes:\ n = msg.src \land msg.dest = AuthorityNode$ *)
(* For $TLC$ to explore, $Next$ needs to allow $SendAlert$ with some $payload$ or $Deliver$ *)
(* This will be refined when integrating with $AnomalyAlertModel$ *)
```