ANNEX L - HIGH-FREQUENCY WIRELESS-TOKEN-RING-PROTOCOL (WTRP) REQUIREMENTS

L.1 INTRODUCTION

This Annex specifies a High-Frequency Wireless Token Ring Protocol (WTRP) for enhanced media access control within single-frequency multi-node radio networks using STANAG 5066. The protocol is in two parts: a message design for the management tokens exchanged by nodes in the radio network, and the algorithms used to create, maintain, and repair the radio-transmission sequence (i.e., the virtual ring) of nodes in the network.

There is a token that represents right to transmit, which is transferred using a STANAG 5066 EOW (Engineering Order Wire) message. This means that the token can be transmitted without any protocol overhead and can be repeated for resilience. Additional messages to manage the token ring are sent using STANAG 5066 Extension D_DPUs as specified in S5066-EP10.

The High Frequency Wireless Token Ring Protocol (WTRP) is a self-organizing Medium Access Control (MAC) protocol for HF wireless networks. The MAC protocol by which mobile stations can share a broadcast channel is crucial in wireless networks, especially for HF wireless networks where bandwidth is considerably lower than other types of wireless networks. WTRP is a MAC protocol derived from Ergen et al [1] and tailored for low-speed wireless networks. WTRP provides bounded latency. WTRP is efficient in reducing the number of retransmissions due to collisions and is fair in that each station uses the channel at least once for each ring traversal.

All stations are connected on a single channel with common waveform operating on the same frequency. For waveforms with variable bandwidth a common bandwidth must be used by all stations. The frequency and bandwidth may be fixed or may be negotiated by ALE for all stations. For autobaud waveforms, transmission speed and interleaver may vary during operation or fixed values may be configured.

WTRP requires that stations in a ring take turns to transmit for a specified maximum amount of Time, with an order of transmission that includes each node at least once. WTRP is robust against single node failure. WTRP is different from its parent protocol [1] in that it provides the notion of self-rings, that it supports connected networks with arbitrary connectivity, that there is no ring owner, and that stations can transmit more than once per ring cycle.

This Annex of STANAG 5066 is organized as follows.

- Section L.2 gives an overview of WTRP;
- section L.3 specifies the STANAG 5066 message design for WTRP management messages;
- section L.4 gives the complete WTRP state diagram and description for each state in WTRP;
- section L.5 specifies parameters and timer selection for ring-management and operation;
- section L.6 specifies requirements for token and message transmission.

L.2 OVERVIEW: WIRELESS TOKEN RING PROTOCOL

Token-Ring definitions and management concepts are introduced below prior to detailed specification of the protocol in later sections. This protocol is based on work by Mustafa Ergen, Duke Lee et. al., "Wireless Token Ring Protocol", University of California, Berkeley, CA 94720, USA.

L.2.1 Definitions

The following terms are used in the specification of the High-Frequency Wireless Token Ring Protocol (WTRP).

L.2.1.1 Stations and Nodes

The terms "station" and "node" are used interchangeably to describe the communication entities on the shared transmission medium.

L.2.1.2 Tokens, Messages, Rings and Ring Members

The WTRP protocol is a Medium Access Control (MAC) protocol. The task of this protocol is to schedule the access of two or more stations connected to the same physical medium, nominally an HF wireless network. Messages exchanged between stations for WTRP control are called *messages*.

The WTRP protocol organizes stations in such a manner that they rotate a *right-to-transmit Token (*or just *token)* among stations connected to the same physical medium. Only a *station* that receives the *right-to-transmit token* has the right to transmit user data. This *station* is then the *token-holder*. In normal operation there should only be one *token-holder*.

A set of stations sharing the same *right-to-transmit token* is defined as a *ring*, or *virtual ring.* A station participating in a ring is called a *ring member*.

When a station starts initially it is not a member of any ring. It will only be able to become a *ring member* if there is at least already one station connected to the same physical medium. If the station connected to the medium finds out there is no existing ring, it will attempt to set up a new ring with itself as its only member. Such a ring is called a *self ring*. If the station trying to establish a ring finds another station willing to join the ring, the *self ring* becomes a *ring*.

The *right-to-transmit* will be passed in rotation among the ring members, with a cycle that includes every ring member. A complete transit of this ring is referred to as a *ring cycle*.

L.2.1.3 Successors and Predecessors

If station S_A is passing the right to transmit to station S_B, then station S_B shall be called the *successor* of station S_A. The *predecessor* of station S_B shall be station S_A.

In other words, the *successor* of station X is the station to which X sends the *right-to-transmit token* to and the *predecessor* is the station from which X received the *right-to-transmit token*. Note that these stations are member of the same ring.

L.2.1.4 Ring Transmit Order

The order in which the Right-To-Transmit (RTT) token is passed around in the virtual ring is called the *transmit order*, which also defines the *successor* and *predecessor* relationship among *ring members*. For example, if the RTT token is passed from station S_A to station S_B, from station S_B to station S_C, and back to station S_A, then the transmit order is $\{S_A > S_B > S_C > S_A\}$. In a ring whose transmit order equals $\{S_A > S_B > S_C > S_A\}$, station S_B is the *successor* of station S_A, station S_C is the *successor* of station S_B, and station S_A is the *successor* of station S_C.

The ring transmit order is sent to current and potential ring members as a Transmit-Order-List (TOL) which represents a directed graph of transmission order. The TOL is a graph that connects all of the stations in a ring, which may include a station one or more times. This graph is encoded as a list of nodes. Stations may modify the ring transmit order to accommodate connectivity changes in the network, loss or addition of stations in the network, or to define a more efficient ring transmit order. In the TOL, the last node in the list connects to the first one.

Then *token* is passed directly between each *successor* and *predecessor* in the TOL. In order to accommodate complex ring topologies, a node may appear more than once in the TOL. Every ring member must be included in the TOL at least once.

L.2.1.5 Node States

The following WTRP states are defined in the protocol:

- Floating State (FLT) - the initial (starting) state where a station is not part of a ring looks for an existing ring that it can join.
- Self Ring State (SFR) - in this state a station assumes there is **no** existing ring to join, and will therefore try to setup a new ring and declare itself the ring owner. The new ring will start with this station as the only member and is therefore called a self-ring.
- Seeking State (SEK) - in this state a station considers itself to be in a self-ring and has broadcast an INVITE message as an invitation for other stations to join its ring.
- Solicit Reply State (SRP) - in this state a station tries to join another ring; it intends to respond to a received INVITE message but is waiting for a timeout to avoid congestion before sending its reply.
- Joining State (JON) - In this state a station has replied to the invitation to join (i.e., to the received INVITE message) by sending a JOIN message to the node inviting it to join the net.
- Have-Token State (HVT) - in this state a station is part of a ring. It has received a RTT (right-to-transmit) token from its predecessor and with this token, the right to transmit. From this state a node may optionally transition to SLT in order to invite other nodes, after which it will return to this state. Then it will transmit token, messages and user data and transition to MON.
- Monitoring State (MON) - in this state a station is part of a ring. It passed the RTT token to its successor, but is unsure if the successor has received the right to transmit (i.e., the station is MONitoring the channel for an implicit acknowledgement the RTT was successfully passed);
- Idle State (IDL) - In this state a station is part of a ring and knows it has successfully passed the RTT token its successor; it will process any D_PDU messages received from other stations;

- Soliciting State (SLT) - in this state a station is part of a ring, currently has the right to transmit, and is inviting new stations to join the ring by broadcasting an INVITE message and listening for replies;

L.2.1.6 Primary Timers

Most states have one or more associated timers. The primary Timers defined within the WTRP state machine are:

- Claim Token Timer (TCLT) - Timer used in the *floating-state* (FLT). Controls the time a station waits while in the *floating state* to claim a token before transiting to another state; a station restarts its TCLT timer when it transits to the FLT state.

- Solicit Successor Timer (TSLS) - This timer is used in the *self-ring-state* (SFR). If it times out the station shall transit to the *seeking-state* (SEK). The timer is specifed with a random timeout to reduce the probability of collisions by transmissions from stations attempting to establish different rings at the same time.

- Solicit Reply Timer (TSRP) - Timer used in the *solicit-reply-state* (SRP). A station starts the *solicit-reply timer* when it transits to the SRP state. When it expires the station will send a JOIN message and transition to the JON state.

- Contention Timer (TCON) - Timer used in the *joining-state* (JON). It controls the time a station waits for a response from another station following an attempt to join the network, so-named because failure to receive a response is attributed to contention with other stations attempting to join the network at the same time; a station starts its *contention timer* when it goes to JON state.

- Idle Timer (TIDL) - Timer used in the *idle-state* (IDL). It controls the time a station waits for its *right-to-transmit* before transiting to the *floating-state* (FLT).

- Solicit Wait Timer (TSLW) - This timer is used in the *soliciting state* (SLT) and the *seeking-state* (SEK) by stations inviting new ring members. When it expires, the inviting station will update the Transmit Order List to include all nodes that have sent JOIN messages.

- Token Pass Timer (TPST) - Used in the *monitoring-state* (MON). It controls the time a station waits after passing an RTT (or other) token to another *station* and failing to hear an implicit acknowledgement before considering the *right-to-transmit* as lost.

L.2.1.7 Token Contents

The right-to-transmit-token can be considered as a simple flag that controls the right to transmit. The token is transmitted along with the identification of the “next to solicit” node. This mechanism enables nodes to take it in turns to solicit for new ring members. This is important, as it enables growing the ring with nodes that can only communicate directly with a subset of the current ring members.

L.2.1.8 Promiscuous reception

Promiscuous reception is a receive mode in which a station performs limited processing and information collection on all D_PDUs it receives, whether or not they are not addressed to it (i.e., the station takes in all traffic).

Promiscuous reception is the means by which a station discovers and confirms the existence or loss of links in the HF radio network (or that may be used to construct a network), compiles local node adjacency information, which is used to determine the TOL. This is core to the operation of WTRP.

L.2.1.9 Receive Table, Connectivity Table, Adjacency Matrix and *Next Hop Table*

Each node maintains a *receive table* that records information on all other ring members where transmissions have been recently received. Each ring member will transmit regularly as the token passes around the ring, so it is certain that there will be transmissions from each node in a working ring. For each node from which data has been recently received, the receive table will record that this node can be heard. Where no data has been received from a node in the ring, the receive table will record that no data is being heard.

In addition, for each node where data can be heard, the *receive table* **shall** record a *transmit speed*. This speed reflects the maximum recommended speed for bulk throughput for data transmitted to the node. The model and encoding is aligned to S5066-EP4 "Data Rate Selection in STANAG 5066 for Autobaud Waveforms". This speed will be calculated from the SNR and Frame Error Rate of data received from the node.

Each node **shall** send it's *receive table* to all other nodes, by transmission around the ring whenever the receive table changes. This means that every node will have the *receive table* for all nodes, which provides the node with information on transmission of data to the node from every other node.

The *receive table* data enables the node to build up an internal *connectivity table* for all nodes. For each other node this table will record:

1. The connectivity status of the node. One of:
    a. No direct connectivity
    b. Bidirectional transmission
    c. Transmission only from the node to local node.
    d. Transmission only from the local node to the node.
2. Transmit speed from the node to local node
3. Transmit speed from the local node to the node

Unidirectional transmission can be useful for non-ARQ data. For ARQ data and token transfer, it is essential that transmission is bidirectional. Because of this, primary connectivity calculations in this specification are based on bidirectional transmission, which is safe for all types of transfer. An *adjacency matrix* is built using the receive table information to show for each node pair with bidirectional connectivity the *transmission speed* in each direction.

Token transmission to nodes that are not directly connected is managed by the TOL, which provides a sequence of nodes with bidirectional data transfer between each pair.

The *adjacency matrix* and *connectivity table* can then be used to build a *next hop table* that indicates for each node in the ring one of:

1. The node can be reached directly for all transmissions; or
2. The node can be reached directly for non-ARQ transmission only; or
3. The preferred *data relay node* to be used for other communication.
4. Transmission speed.

The first two can be determined directly from the *connectivity table*, and when there is direct communication this supplies the transmission speed. For nodes that cannot be reached directly, the *adjacency matrix* can be used to determine the best *data relay node*. For each directly connected node, the adjacency matrix can be used to determine paths to nodes directly connected to these nodes. This can be done iteratively to determine the distance to each node, and a set of paths to each node.

Where several directly connected nodes can act as the *data relay node*, one of them needs to be chosen for the *next hop table*. The usual choice of the preferred *data relay node* is the one that is closest in the TOL, which will usually minimize transfer time. The transmission speeds of each link **may** be considered, and it may be preferable to choose a *data relay node* that leads to faster transmissions on each hop. Note that the next hop table must be recalculated whenever the TOL or *receive table* data changes.

The *adjacency matrix* is used to communicate to the DTS and CAS layers of STANAG 5066, which nodes can be accessed directly, and which nodes need to be relayed. This also implicitly identifies nodes that cannot be reached.

To send data to nodes that are not directly connected, STANAG 5066 needs to send data to the *data relay node* for that node. The *data relay node* then needs to relay the data onwards to its final destination or to the next relay hop. The protocol to do this is specified in S5066-EP13 (STANAG 5066 Routing Sublayer).

L.2.1.10 Tour

A *tour* of a graph is a sequence of nodes from the graph such that each node appears at least once and two nodes are adjacent in the sequence only if they are adjacent in the graph. An unconstrained, ordinary tour allows revisits to network nodes.

In the context of the wireless token-ring protocol, the sequence of ownership of the right-to-transmit (RTT) token, i.e., the Transmit-Order-List should be a *closed tour* of the network that starts and ends with nodes that are adjacent in the network, i.e., every node gets an opportunity to transmit, and passing the RTT token along the closed tour is feasible because nodes that are adjacent in the transmission sequence are adjacent in the network. The TOL is always a *closed tour* and may be an *unconstrained closed tour*.

L.2.1.11 Ring-Cycle Length (RCL)

The Ring-Cycle Length (RCL) is the length in hops of the tour through the WTRP network taken by the RTT token as it follows the ring transmit order list (TOL).

L.2.2 Concept of Basic Ring Operation

The basic concept of the Wireless Token Ring Protocol is to provide a mechanism that allows two or more stations operating on the same single-frequency radio channel to share it in such a way that only one station will transmit at a time. Only the station that has the *right-to-transmit* (*token-holder*) is allowed to transmit on the shared channel. The *right-to-transmit* is passed onward to all stations that joined the ring. Figure L-1 shows how the *right-to-transmit* is circulated among the *ring-members*.

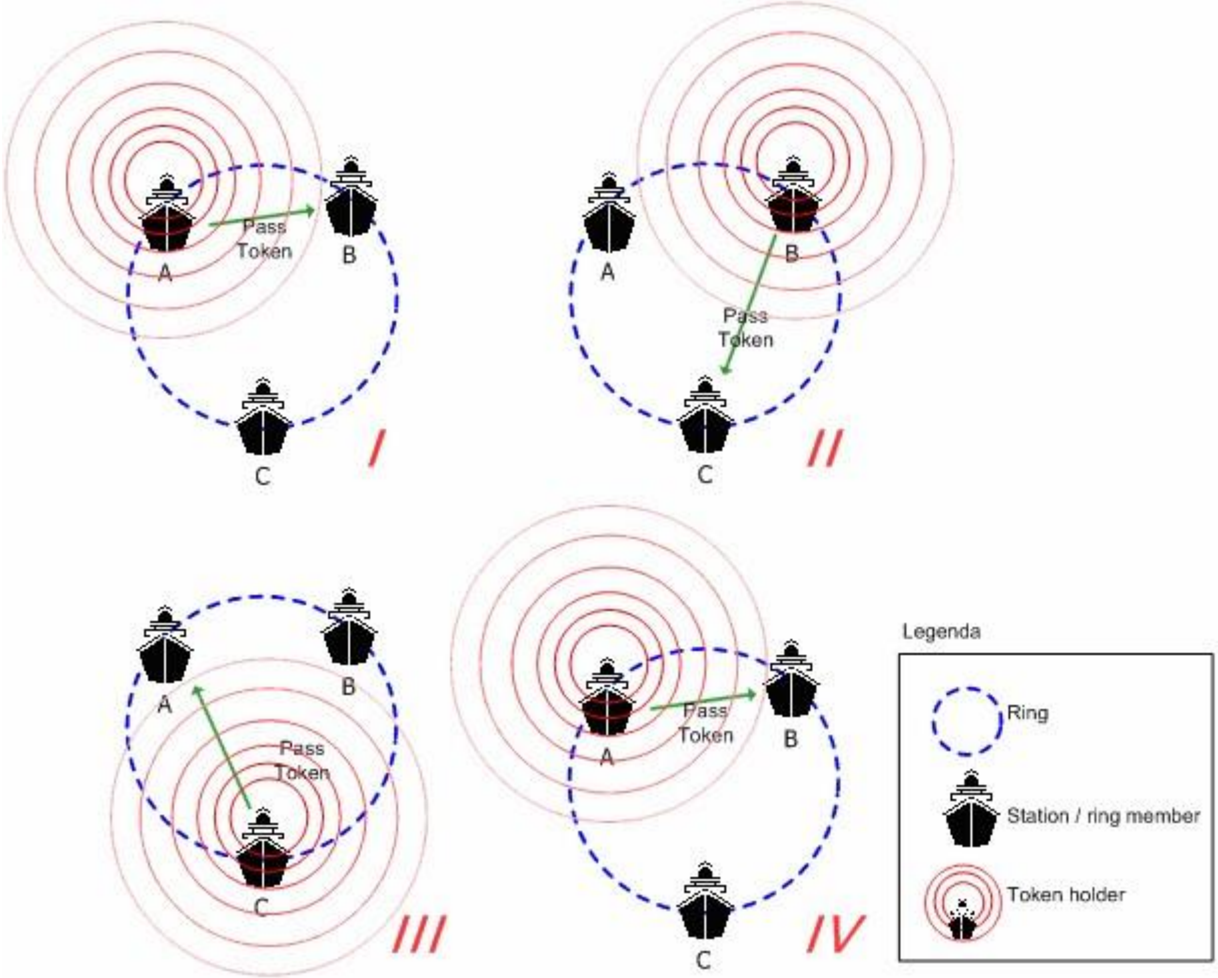

Figure L-1 - Normal Token-Ring Operation

The ring is a closed cycle of stations that transmit each in turn in a prescribed sequence, which will be adapted if new stations join the network or changes in network connectivity force adaptation. When a station receives the *right-to-transmit* from its *predecessor* it will take control of the channel. In addition to the *right-to-transmit token* the station may also receive additional WTRP messages, in particular an updated *transmit-order-list (TOL)* and updated *receive table* messages from one or more nodes. These messages will not be transmitted when the information is stable. The messages are passed around the ring to ensure that all nodes have the latest information. A node will add its own *receive table* message if this information has changed. The *right-to-transmit token* is encoded in a standard STANAG 5066 EOW message and will generally be included multiple times in a transmission to minimize risk of token loss. If the station has no data to transmit, then it shall pass the *right-to-transmit* and associated messages immediately.

The messages sent by a station are transferred in a set of one or more D_PDUs. The D_PDU **shall** count down the *end of transmit time* (EOT) in accordance with the requirements of Annex C Section C.3.2.3. A station **shall** not exceed the maximum transmission time allowed.

Note that though the transmission sequence and ownership of the *token* in the ring is prescribed, a station with the *right-to-transmit* **may** send data to any other station in the network that it determines from the *connectivity table* can receive data and transmit directly back acknowledgements for ARQ data, not just its successor or predecessor in the virtual ring. Any station in the virtual ring may therefore engage in multiple concurrent traffic exchanges with any other station in range. This includes the capability to support concurrent soft-link, physical-link, and ARQ connections in accordance STANAG 5066 Annex A, B, and C, with retransmission timers and other timing parameters selected to allow for the network size.

Possession of the *right-to-transmit* controls access to the radio channel; it does not prescribe the destination(s) to which traffic can be sent. The token-ring protocol controls node access to the transmission medium to preclude collisions and self-interference in the network and thereby increase the efficiency and throughput in the network. To do so, in general, there is no requirement that all nodes in the network be in communications range, only that the network have a closed tour, i.e., that a cyclic sequence of nodes exist where every node is in communications range of its predecessor and successor in the sequence. To support WTRP networks where direct communication is not possible between all nodes, the WTRP layer communicates connectivity to the higher layers of STANAG 5066. The higher layers will only send data to nodes that can be reached directly. Relay to other nodes is provided by S5066-EP13 (STANAG 5066 Routing Sublayer).

Each station will retransmit the *token,* passing the *right-to-transmit* to its *successor* until this has been acknowledged by its *successor.* This acknowledgement is implicit, by observing when the successor transmits data.

L.2.3 Ring formation

Before operating as a ring, a ring must be formed by stations on the same radio channel. The ring is a self-organizing and self-repairing mechanism, subject to a number of requirements:

- As soon as a station starts to operate it will listen to the radio channel in a floating state, unaffiliated with any ring.
- A station may transmit data (i.e., D_PDU other than ring-management messages) only if it is part of a ring that is not a self ring.
- A station shall first listen for an existing ring on the radio-channel; if it hears a ring it will try to join that ring, otherwise it will form a new ring (a self ring).
- If a station receives the right-to-transmit from another node, it has become a part of the ring.
- If a station is unable to pass the right-to-transmit to its successor, it shall recalculate the TOL on the basis that the token cannot be passed to the current successor. This TOL recalculation may lead to one or more nodes being dropped from the TOL. The station will then attempt to pass the token to

the new successor. This process is called iteratively until the token is successfully passed or the node reverts to a self ring.

- A station dropped from a ring must re-join the ring to become again part of it and obtain transmission rights.

There are two possible ways to form a ring:

1. Join an existing one.
2. Start a new ring yourself.

The process of Ring Formation is illustrated below in Figure L-2 - Ring Formation.

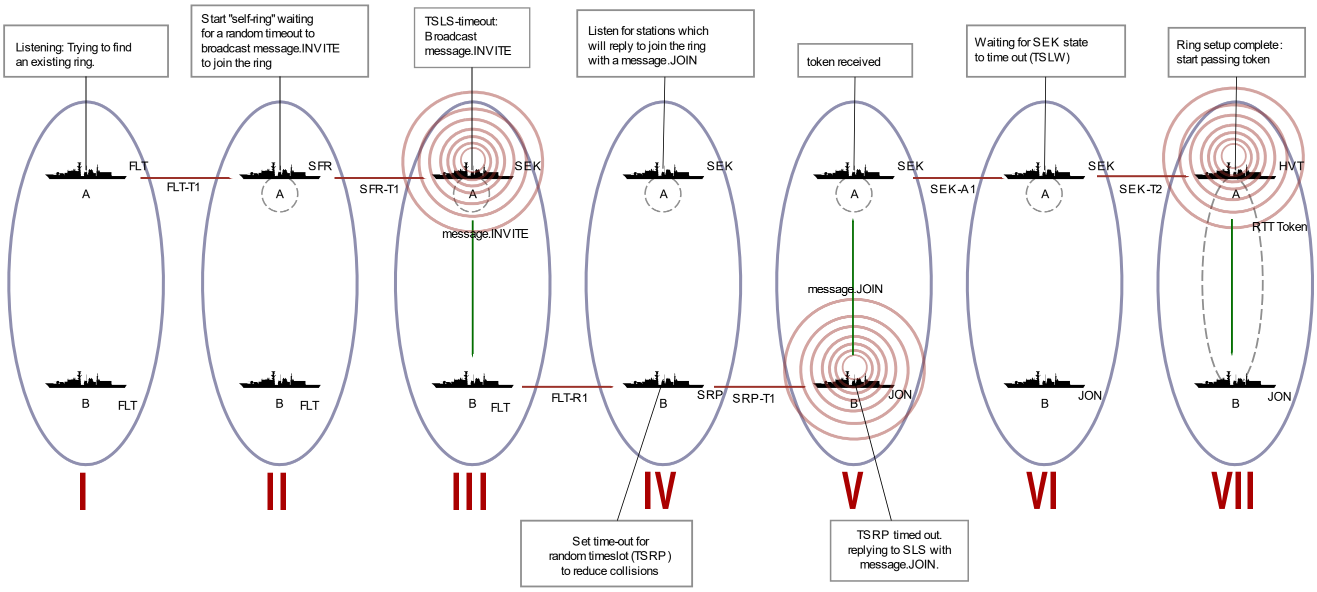

Figure L-2 - Ring Formation

Figure L-2 shows a time-sequence evolution of two stations, station A and station B, as they form a two-node ring.

In situation ***I*** station A is listening for an existing ring. After its TCLT timer times out, station A assumes there is no existing ring (otherwise it would have heard it) and it transits from a FLT state to the SFR state, forming a *self ring* as illustrated in situation ***II***.

The transition moment from the SFR state in situation ***II*** to the SEK state in situation ***III*** is controlled by a random timeout that avoids collision between two stations in the same state. The value of the timeout is calculated based on picking an asynchronous timeslot (See TSLS timer details in section L.5).

In the SEK state of situation ***III*** a station invites new members by sending an INVITE message. Each station that is seeking to join will reply with a JOIN (join ring) message. There could be more than one station waiting to reply to an INVITE message. Therefore, to reduce the probability of collision between replies from multiple joining nodes, the joining node's reply shall be transmitted in a randomly chosen time slot among a set of slots available for replies. The station that sent the INVITE message will add all of the nodes requesting to join to the Ring and pass the token to one of them with a TOL (Transmit Order List) that reflects the ring membership. The ring is created at this time.

Once the ring is created, each node periodically creates an opportunity to invite new stations to join the ring by broadcasting an INVITE message and waiting for JOIN message replies from nodes wishing to join the ring. This mechanism can also be used to merge two rings.

* L.2.4 Ring Entry

The invitation to join shall be made periodically by a station receiving the right-to-transmit, with invitation frequency about once every SLS_INTERVAL seconds. The opportunity to invite new members needs to be shared between ring members, as new nodes may only be able to connect to one existing member. The *token* identifies the next node to invite new members. Once this node has issued an invite, it will pass the invite to another member of the ring, by setting the value in the token. This is done using a mechanism which gives the invite opportunity equally to all ring members. A configurable maximum ring size may be specified, which can be used to stop rings growing too large. This can also be used when the number of nodes is known, to prevent taking time to invite new members when there are no possible new members. Invitations can also be configured so that they are less frequent (lower overhead) for stable rings.

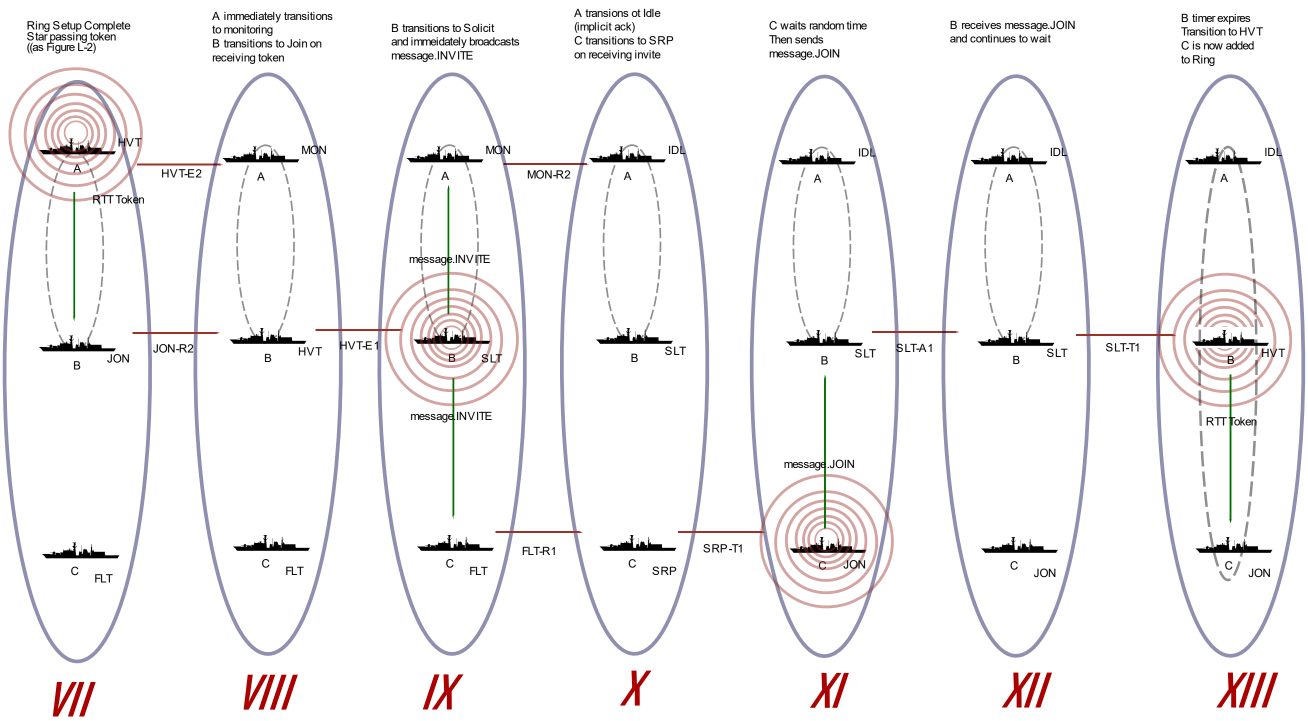

Figure L-3 - Ring Entry

Figure L-3 illustrates adding a station to an existing ring and continues the example of Figure L-2, showing addition of station C to the ring.

In VII, station B has received the token as in HVT state. Station B determines that it needs to issue an invite, so immediately transitions to SLT (IX) and immediately broadcasts a message.INVITE. Station A receives the message.INVITE, which it uses as implicit acknowledgement of token transfer, so it transitions from MON to IDL (X). Node C in FLT plans to accept the invitation and so transitions to SRP (X).

Station C waits for a random time before responding with message.JOIN (XI). Station B receives the message.JOIN (XII) and continues to wait for potential other message.JOIN from other stations. After a timer, station B transitions to HVT and station C is added to the ring. Station B sends message.JOIN to station C (VIII).

L.2.5 Connectivity Update

Each node monitors the connectivity from all other ring members. It **may** also monitor connectivity from other nodes, which is recommended as it will provide useful information in the event of a monitored node joining the ring. This information from other ring members is recorded in the *receive table*, which notes nodes that are heard and an associated recommended maximum transmit speed, determined from SNR, FER and other information.

Nodes will be removed from the table on either reaching a configurable maximum age or a configurable number of ring cycles with no data heard.

L.2.6 Communicating Connectivity

The calculated receive table is shared with all other nodes. This is done by sending an TABLE message that contains the receive table. TABLE messages are sent along with the token and will progress around the ring. A node sending an TABLE message will ensure that it arrives with the token after a complete ring cycle (noting that a node may get the token multiple times in a ring cycle). When this has happened, the node will know that all other nodes in the ring have the current receive table and so there is no need to transmit the receive table again.

Whenever the receive table changes, it is sent around the ring to share this information. In this way, all nodes have full connectivity information.

L.2.7 Ring-Cycle (i.e., TOL) Reconfiguration

There are four scenarios or cases in which the ring-cycle, i.e.,the TOL, **must** or **may** be reconfigured:

- Scenario 1 (Joining Scenario): A node joins the network and thereby **must** be inserted into the TOL;

- Scenario 2 (Leaving Scenario): A node chooses to drop out of the ring, either due to operator request or due to detecting another ring. In the latter case, the whole ring is closed, with the expectation that the nodes will join the new ring.

- Scenario 3 (Topology Change Scenario): Changes in the network may lead to a situation where token transfer to the current success node fails. This requires TOL re-ordering and a new successor. It may lead to nodes being dropped from the ring.

- Scenario 4 (TOL-Optimization Scenario): It is possible that a better TOL can be calculated, either with less hops or making use of better quality links.

These four scenarios are considered in more detail below.

The first three scenarios are known as critical TOL updates. The protocol ensures that these updates propagate robustly, which may lead to delays due to timers. The fourth scenario is a non-critical update, which is not protected by timers and as a consequence the new TOL may need multiple attempts to transit the entire ring.

L.2.7.1 Ring-Reconfiguration in the Joining Scenario

A new node joining the network results in a new network topology, through determination and recognition of the new links between the joining node and other network members. The new node **must** be added to the TOL.

The inviting node sends the current TOL in an INVITE message, so that all joining nodes know the current ring membership. This enables a joining node to send a receive table to the inviting node in the JOIN message. The inviting node will have information on connectivity of all nodes, enabling it to determine a suitable TOL.

L.2.7.2 Ring-Reconfiguration in the Leaving Scenario

A node choosing to leave a ring **shall** do so by waiting until its transmit opportunity and then forwarding the right-to-transmit to its successor, including monitoring for correct handover. The node will remove itself from the TOL and send a TOL message reflecting the updated TOL.

L.2.7.3 Ring-Reconfiguration in the Transient-Topology Scenario

Instances will occur with topology change where a node holding the RTT token cannot communicate with the next node in the TOL. When this happens, the TOL **must** be recalculated, which may lead to nodes being dropped from the TOL.

L.2.7.4 Ring-Reconfiguration in the TOL-Optimization Scenario – Ring Transmit-Order Recomputation

Rings with viable TOL **may** be reconfigured with the generation and promulgation of a new TOL only after they have been stable for one or more ring cycles. Any node may determine an updated TOL and propagate using the TOL message. The TOL message will propagate around the ring. The node making the non-critical

TOL update will check that the TOL message has correctly propagated around the ring. If it has not, it will send it again.

A node **may** choose any algorithm when recomputing the transmit order list, though good, near-optimum algorithms are closely related to those which solve the so-called Travelling-Salesman Problem for a weighted network. In general, the “best” solution cannot be determined, although it is anticipated that good solutions can determined for typical networks in a reasonably straightforward manner.

L.2.8 Service Provision to Higher STANAG 5066 Layers

STANAG 5066 Annex J defines the MAC model which WTRP provides. There is a clean layer model, with M_ service elements communicating between the layers. The details of these service elements are not specified, but the model and associated protocols are clear. WTRP provides some additional services to the generic MAC functions defined in Annex J.

L.2.8.1 Data Rate, Interleaver and Transmission Length

When Annex M is not used, STANAG 5066 DTS layer will control selection of transmission speed, interleaver and transmission length. A modern and flexible approach to achieve this is specified in S5066-EP4 “Data Rate Selection in STANAG 5066 for Autobaud Waveforms”. The DTS layer will also determine which D_PDUs to duplicate, such as sending ACKs multiple times to improve reliability.

WTRP introduces new D_PDUs with rules for transmission and duplication. Selection of data rate, interleaver and transmission length must be cognizant of this information. WTRP also replaces the S5066-EP4 mechanism with a simpler service that provides recommended maximum transmission speed for each directly connected node, using the *connectivity table* defined in L.2.1.9.

L.2.8.2 Node Availability and Routing

WTRP determines which nodes in the ring can accessed directly and for those nodes which cannot be accessed directly it specified a preferred relay node. This information is generated in the *next hop table*, as specified in L.2.1.9. This information is passed upwards as an Annex J M_ service primitive. This information can be used by the CAS layer as specified in S5066-EP13 (STANAG 5066 Routing Sublayer) to:

1. Immediately reject messages to nodes that are not in the ring; and
2. For nodes that are not directly connected, to cause the data to be relayed.

L.2.8.3 Broadcast Retransmission

When a broadcast/multicast PDU is received, information is provided to the higher layers. It is important that broadcast and multicast work correctly for nodes that are not directly connected. This needs to be done in a way that prevents duplicate onward broadcast. Consider four nodes connected in a long trapezoid shape, where the two end nodes cannot communicate directly. If one of the end nodes sends a broadcast message, then one (but not both) of the middle nodes that can communicate with both nodes needs to broadcast the message, so that all nodes receive it.

This is achieved by providing a *Relay Responsible List* to the higher layer for any broadcast or multicast PDU that is received. This tells the higher layers which nodes it needs to re-broadcast to. Handling this is described in S5066-EP13 (STANAG 5066 Routing Sublayer).

To do this calculation, the node needs to handle connectivity data as if it is the node which sent the broadcast. The set of nodes which will have received the message directly can be determined. The additional set of nodes that need to receive the message can be determined from this list and from the membership of the broadcast list or multicast list (noting that the node knows the identity of all nodes). For each of these nodes, the preferred relay node (from the perspective of the sending node) can be determined. The *relay responsible list* is the list of nodes for which the preferred relay node is the local node.

L.3 WTRP TOKEN AND MESSSAGE SPECIFICATION

L.3.1 Node ID

Nodes are identified by a single byte ID (range 0-255). This limits the number of nodes in a ring to 256, which is expected to be higher than the practical limit. As nodes need to be extensively referenced in the protocol, a compact representation is important. Externally, each node is identified by a variable length STANAG 5066 address (typically 3.5 bytes).

The Node ID will usually be the same as the last byte of the node's STANAG 5066 address, unless there is a conflict in the ring.

Each Node will maintain a map between Node ID and STANAG 5066 address. The mapping is derived from the set of receive tables; when distributed the receive table includes this mapping. A node must have a complete set of receive tables, with a receive table for all nodes in the ring. The WTRP protocol enables "missing" receive tables to be requested.

This mapping is assumed in a number of the procedures below, so that nodes can be identified either by Node ID or STANAG 5066 address.

L.3.2 Token Protocol

The token is transmitted using a standard EOW message. This enables the token to be transmitted along with standard data. The EOW must be sent in a message with destination address of the token recipient. The source address will always be that of the token sender.

The single byte value of the EOW is set to the node ID of the node that currently has responsibility to invite new nodes. This ensures robust transfer of this information along with the token.

Two types of EOW are used.

1. Token only. This uses EOW value 13. This message transfers token only. There may be other messages containing updated maps, updated TOL, and new receive tables. However, none of these messages cause impact on the topology. If these messages are lost in transit around the ring, they will be retransmitted. By taking this approach, the potential delays of ensuring robust TOL transfer are avoided.

2. Token and critical TOL. This uses EOW value 14. When there is a critical change in the TOL (adding a node; removing a node; change to address broken link) this message transfers the token and signals that a TOL message must be received. Note that any critical receive table changes will lead to a TOL change. Any critical ID mapping table changes can be inferred from the TOL. If a node receiving this token does not correctly receive a transmitted TOL message, it can request retransmission of the TOL using a RETANS message.

L.3.3 WTRP Messages

WTRP uses a set of messages to communicate information. These use D_PDUs of the format defined in S5066-EP10 "Extension D_PDU". The allowed messages and defined extension D_PDU numbers are specified in Table L-1:

Message	Description	Extension Number
TOL	Transmit Order List	1
TABLE	Receive Table	2
INVITE	Invite other nodes to join ring	3
JOIN	Request to join ring	4
RETRANS	Retransmission Request	5
CLOSE	Close the current ring	6

Table L-1 – WTRP Messages

The syntax of these messages is specified in the following sections, and procedures to use them are specified in Section L.4.

A number of the messages will often be short enough to fit within a STANAG 5066 header, but will not always be. Because of this, the encoding of some messages has data in the header, while allowing for additional data after the header.

L.3.3.1 TOL Message

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 1						LSB
5+m	MSB	RCL (n)						LSB
	MSB	Version						LSB
	TOL (max 22 bytes)							
CRC CRC	CRC on Header							
h+m+1	Extended TOL							
h+m+n-22	MSB							
	CRC							
h+m+n-18								LSB

Table L-2 – TOL Message

The TOL Message communicates a TOL. Each byte in the TOL represents a node identified by its node ID. The TOL is a loop and the start point does not matter. This means that the same TOL encoding, with arbitrary first node, can be used by all nodes. Note that nodes may be repeated in the TOL, but that for each repeat the predecessor of the node will be different.

The TOL Message encoding has the standard S55066-EP10 fields shown without colour. The extended D_PDU type is set to 1 for TOL. RCL (Ring Cycle Length) is the number of nodes in the TOL. If extended TOL is used, a standard STANAG 5066 data checksum is used on the extended data.

The TOL has a version number starting at 0, which is incremented for each change, resetting to zero after 255. This is to ensure that when multiple versions of the TOL are circulating, that a node will ignore a version prior to the one it holds.

L.3.3.2 TABLE Message

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 2						LSB
5+m	MSB			Sending Node Address				
8+m								LSB
9+m	MSB			Sending Node ID				LSB
10+m	MSB			Version				LSB
11+m	MSB			Number Table Entries (n)				LSB
				Table (max 17 bytes)				
CRC CRC	CRC on Header							
h+m+1				Extended Table				
h+m+n*2-22	MSB			CRC				
h+m+n*2-18								LSB

Table L-3 – TABLE Message

The TABLE Message communicates a receive table. The Sending Node Address field is the STANAG 5066 address of the node that generated the receive table. The Sending Node field is the node ID of the node that generated the receive table.

The table has a version number starting at 0, which is incremented for each change, resetting to zero after 255. This is to ensure that when multiple versions of a receive table are circulating, that a node will ignore a version prior to the one it holds.

Each table entry comprises two bytes. The first byte is the node ID of the node that data can be received from. This **must** be a node in the TOL. The second byte is the recommended transmit speed. This uses the same encoding as S5066-EP4 "Data Rate Selection in STANAG 5066 for Autobaud Waveforms", which is repeated here for convenience.

The transmission speed byte is encoded as follows.

MSB 7	6	5	4	3	2	1	LSB 0
Interleaver			Speed				

The Speed field specified transmission speed. This encodes the actual speed for standard HF, and the waveform for WBHF (with the speed dependent on channel size).

Value	**Speed (bps)**
0001-1101	WBHF. Waveform is the recommended Annex D Waveform number (1-13)
10001	75
10010	150
10011	300
10100	600
10101	1200
10110	2400
10111	3200
11000	4800
11001	6400
11010	8000
11011	9600
11100	12800

The Interleaver field specifies the minimum Interleaver recommended for use with this transmission speed.

Longer interleavers give better performance for data, and so in general the longest interleaver option should be selected. However, longer interleaver means longer block size. In some situations, particularly when transmitting smaller amounts of data, it is desirable to reduce the block size. This field gives the minimum interleaver recommended.

The Interleaver field is encoded as follows.

Value	Interleaver
000	No recommendation
001	Ultra Short
010	Very Short
011	Short
100	Medium
101	Long
110	Very Long
111	Reserved

L.3.3.3 INVITE Message

The INVITE message is used by a node with the token to invite other nodes to join. It is sent to the broadcast address, so that any listening node will receive it.

The INVITE message uses the same encoding as the TOL message. The TOL is needed to inform nodes of the list of nodes in the ring.

L.3.3.4 JOIN Message

The JOIN message is used by a node responding to an INVITE message.

The JOIN message uses the same encoding as the TABLE message. This is used to communicate the node's receive table the inviting node, noting which nodes in the ring can be heard by the invited node and the quality of reception for those nodes. This receive table information is then circulated around the ring by the inviting node using TABLE messages.

The joining node will allocate a suggested value for it's own Node ID. This **shall** be different to all other Node IDs in the TOL received in the INVITE. When there is no conflict, this Node ID **shall** be the same as the last byte of the node's STANAG 5066 Address.

The receive table **shall** include the inviting node. The mapping Node ID of the inviting node may be known, because a receive table message from the ring has been received. If this is the case, the correct Node ID is used in the table. If this is not the case, the last byte of the inviting node's STANAG 5066 address is used.

When the inviting node receives a JOIN message with just one table entry, it will treat this table entry as referring to itself, irrespective of the value of the Node ID.

If the Node ID of the inviting node is used, information for other nodes **may** be included where the Node ID of the nodes is known and traffic has recently been received from the node.

L.3.3.5 RETRANS Message

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 5						LSB
5+m	Missing Receive Table List (0-24)							
CRC CRC	CRC on Header							

Table L-5 – RETRANS Message

The RETRANS Message is used when a node receives a token but is missing critical information. There are two scenarios where this can happen:

1. Where the Token is received with an EOW that indicates "TOL Critical" and no TOL message is received. In this case, the message is sent without a Missing Receive Table List. This indicates a request to resend the TOL.

2. Where a TOL is received, but the receiver does not have a receive table entry from one or more of the nodes in the TOL. Each byte of the Missing Receive Table List is the Node ID of a required receive

table. The RETRANS message can encode up to 24 nodes. If more receive tables are needed, multiple RETRANS messages are used.

L.3.3.6 CLOSE Message

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 6						LSB
CRC	CRC on Header							
CRC								

Table L-6 – CLOSE Message

The CLOSE Message is used to signal to close a ring. This will be used when a node has detected another ring, which it intends to join after leaving the ring, with the expectation that nodes in the ring being closed will subsequently join the new ring.

L.4 PROCEDURE OF OPERATION

WTRP operation is defined in terms of the state machine specified here. Specifications and definitions are provided for states and state transitions. This makes use of a number of data structures and procedures defined in this section.

L.4.1 Node Data Structures

A node shall maintain the data structure listed Table L-6. These data structures are all prefixed with the following prefixes to facilitate clarity when they are referenced:

- “Node.”: Generic information associated with the node.
- “TOL.”: Information associated with the TOL.
- “LocalTable”: Information associated with the local receive table.
- “PeerTables.”: Information associated with receive tables from other nodes. .
- “Invite.”: Information associated with control of invitations.

Table L-6 – Node Data Structures

Register / Flag	Type	Initial Value	Description
Node.address	STANAG 5066 Address	n/a	The station's address. The address is expected to be configured.
Node.ID	Integer	Last byte of Node.address	Node ID of the station. May be re-assigned when the node joins a ring.
Node.cycleCount	Integer	0	The number of times that a complete ring cycle has been completed. Note that a cycle is only considered complete when the TOL has not changed for the cycle. This is to help ensure information is propagated to all nodes. Initalized to zero.
Node.Predecessor	STANAG 5066 Address	n/a	Set to sender of Token, when token is received. This is needed when a node appears multiple times in the TOL, to identify the current TOL position.
Node.Sucessor	STANAG 5066 Address	*n/a*	Set to the Token receiver, when token is transmitted This is needed for monitoring after the token has been passed.
Node.TokenTransferred	Boolean	false	Acknowledgement is implicit, by monitoring traffic. This variable is set if traffic arrives that indicates token transfer
Node.CloseRing	Boolean	false	Set if node detects another ring. This will lead to the ring being closed down.
Node.OperatorDrop	Boolean	false	This variable is set by an operator. If set, it will cause the node to be dropped from the ring.
Node.txTokenCounter	Integer	0	Used to record repeats of token transmission.
Node.NextHopTable	Next Hop Table	empty	This is maintained so that higher layers of STANAG 5066 can select best speed for sending to a given node and can send indirectly to nodes that cannot be reached.
Node.NewSuccessor	Boolean	False	Use in MON state to control switching successor
TOL.Changed	Boolean	false	True if TOL has been changed since circulation of the TOL started.

TOL.CriticalChange	Boolean	false	True if the TOL represents a critical change that must be reliably passed to the next node.
TOL.Circulated	Boolean	false	A TOL is being circulated
TOL.CiruclatedCount	Integer	0	The ring cycle count when the TOL was circulated. This is used to ensure circulation to all nodes
TOL.Missing	Boolean	false	Set if TOL is expected but missing
LocalTable.Table	Receive Table	Empty	The value of the local receive table, holding information on receive quality from peer nodes
LocalTable.TableChanged	Boolean	false	True if receive table has been changed since circulation of the receive table started.
LocalTable.Circulated	Boolean	false	A table is being circulated
LocalTable.CiruclatedCount	Integer	0	The ring cycle count when the receive table was circulated. This is used to ensure circulation to all nodes
LocalTable.UpdateTimes	Array of Times, indexed by node ID	empty	For each node in the TOL, the time that the entry in LocalTable.Table was last updated.
PeerTables.Tables	Receive Table List	empty	The active list of receive tables from peers. This list is updated whenever a more recent receive table is received from any node.
PeerTables.ReceivedDirect	Received Table List	empty	List of receive tables from previous node. Need to relay on, only values received directly from predecessor. This ensures that if originator gets back the table it sent, that all nodes have a copy of the table
PeerTables.MissingList	Set of Node IDs	Empty	This is set when a TOL is analysed and there is not a receive table for one or more ring members. This gives a list of nodes for which to request receive table retransmission.
Invite.LastTime	Time	Current time	When the last invite was sent by current node
Invite.LastCycleCount	Integer	0	Ring Cycle Count when invitation last sent.
Invite.Number	Integer	0	Number of invitations issued since joining ring

Invite.Next	Node ID	0	The node that should issue invitations next, and will be sent with each token

L.4.2 Message and Token Notation

The notation specified in this section is used to refer to tokens and messages and their components. Tokens have the following references, which are used to describe both reading and setting the token.

Table L-7 – Token Notation

Notation	Type	Description
token.sender	STANAG 5066 Address	Node sending the token from D_PDU source address
token.receiver	STANAG 5066 Address	Node to which the token is being sent taken from D_PDU destination address
token.criticalTOL	Boolean	Set to true if a MAP message being sent in the same transmission as the token is critical and retransmission **must** be requested if it is not received. This value is set from the EOW type.
token.inviterID	NodeID	The node ID of the node which is responsible for sending the next invite and for updating this field to set the subsequent inviter.

Messages are reference by notation of the form message.<message name>, for example message.TOL. The following references to components of messages are used.

Table L-8 – Message Notation

Notation	Type	Description
message.<message name>.sender	STANAG 5066 Address	Node sending the message from D_PDU source address
message.<message name>.receiver	STANAG 5066 Address	Node to which the token is being sent taken from D_PDU destination address
message.TOL.TOL	Transmit Order List	TOL in a TOL message. When set, the RCL (Ring Cycle Length) element is set from the TOL length

message.TABLE.Table	Receive Table	Includes sender node ID and length
message.INVITE.TOL	Transmit Order List	As for TOL
message.JOIN.Table	Receive Table	As for TABLE
message.RETRANS.TOL	Boolean	Request to retransmit TOL
message.RETRANS.MAP	Boolean	Request to retransmit MAP

L.4.3 Node Procedures

The following functions are used in the state machine to specify node behaviour. Note that while this functional notation specified behaviour, this specification imposes no requirement on an implementation to use these functions.

InitializeNode()

Return Type: None

Description: Initializes node. The following values are set:

- TOL is initialized to a single entry for the local node
- Other values are set to the default values as specified in Table L-6

---

removeNode(<node-address>)

Return Type: None

Description: This procedure removes a node from the ring..

The receive table for <node-address> is removed from PeerTables.Tables.

TOL.Changed and TOL.CriticalChange are both set to true.

All entries of the node ID associated with <node-address> are removed from the TOL.

Then the procedure in L.4.7 is followed to generate a valid optimized TOL.

---

getSuccessor()

Return type: STANAG 5066 Node Address
Description: Returns the address of the local nodes's current *successor*.

The local node location is identified in the TOL. If local node is present multiple times in the TOL, Node.Predecessor is used to determine which of these locations is the current one. The successor is the node after this one in the TOL.

This procedure is only valid if RCL() > 1

---

isMember(<node address>)
Return type: Boolean

Description: Returns true if the given *node address* is a ring member, otherwise false.

Inspect the TOL to determine if the specified node is present.

RCL()

Return type: Integer

Description: Returns the number of unique nodes in the TOL (Ring Cycle Length).

BreakLink(<node address>)
Return type: None
Description: When a node cannot be reached, this is used to change network topology. This is used to recalculate the TOL to determine a new successor.

This is achieved by finding the entry for <node address> in PeerTables.Tables and removing the entry for the local node in the identified receive table. This changes the connectivity record, to indicate that the node in question cannot be reached from the local node. The TOL can then be optimized to identify a node that may be reached.

RetransmitMessages(<message.RETRANS>)

Return Type: None

Description: Used when message.RETRANS is received

If Missing Receive Table List in message.RETRANS.TOL is empty, retransmit the TOL in message.TOL.

If Missing Receive Table List in message.RETRANS.TOL is not empty, retransmit each of the requested receive tables in a message.TABLE message for each receive table.

The token **must** also be transmitted following L.4.8. Other WTRP messages **may** be retransmitted following L.4.8. which is recommended. It is recommended to transmit at conservative speed and to send the requests messages multiple times in the transmission. User data **must not** be sent.

L.4.4 Processing Inbound Transmissions

This section, in conjunction with the following sections (L.4.5 – L.4.7) describes the process for handling inbound transmissions. This functionality, is driven from the state machine by a single call of Receive(). This will listen for a call and continue processing until a full transmission has been received. At the end of transmission it will return EOT and optionally one of the following as events to the state machine:

1. EOT Event. To indicate end of transmission This is always returned, and the following associated Booleans are set.
    a. D_PDUs received. Set to true if any valid D_PDUs received.
    b. Joins received. Set to true if Message.JOIN directed to another node is received
2. One of the following may also be returned with the EOT event:
    a. Token. If a token directed to the local node has been received and none of the following messages.
    b. Message.INVITE. If this has been received, it will be the only message and no Token will be received.
    c. Message.JOIN. If this has been received, it will be the only message and no Token will be received.
    d. Messsage.RETRANS with Token. If this message been received, Token will always be received.

Processing of Message.TABLE and Message.TOL received is handled by this procedure and transparent to the state machine.

When a transmission is received it is fully processed until the EOT, prior to any actions being taken in the state machine. D_PDUs other than WTRP messages are processed following the rules of STANAG 5066. If no other event is returned, EOT is returned at end of transmission.

WTRP Message.INVITE is always sent to the broadcast address. A transmission with a message.INVITE may contain multiple copies of this message. TOL is updated to the TOL value in message.INVITE, if the version number is more recent. Message.INVITE is returned

WRTP Message.RETRANS is sent to a single address and will not have any other messages, but may have duplicates and will include the token. If this is received, Message.RETRANS with Token is returned.

WRTP Message.JOIN is sent to a single address. A single transmission may contain multiple copies of this message.

If Message.JOIN.receiver is not the local node, it reflects a node that is responding to an invitation from a different node. This node may or may not be added to the ring. This message may be cached, for use in the (likely) event that the node is added to the ring and possible loss of transmitted receive table. It may be ignored.

If Message.JOIN.receiver is the local node, this is the response to an invitation. This is processed by this procedure. If there is only one entry in Message.JOIN.table, change the Node ID in the table entry to Node.ID. This entry must refer to the local node, and the node sending the message may not know the correct Node ID value.

The receive table in <receive table> is added to PeerTables.Tables

TOL.Changed and TOL.CriticalChange are both set to true.

The node ID of the new node is taken from <receive-table> and is added into the TOL immediately after the local node. Then the procedure in L.4.7 is then followed to generate a valid optimized TOL.

A transmission containing WTRP **shall** contain the token in each WTRP message. All WTRP messages and tokens in a transmission will have the same source and destination address. Other D_PDUs directed to the successor **may** contain the token. When token is transferred, it must be put into at least on D_PDU. If necessary, a Padding D_PDU can be used for this. The token will generally be repeated many times.

The sender of the transmission can be identified from any D_PDU. Use of isRingMember(sender) can determine if the sender is in the current ring. If the sender is not a ring member, the procedure of L.4.10 is followed.

Once a transmission has been processed, it will be possible to determine either the target destination node for the WTRP information, or that the message is a broadcast message.INVITE. message.INVITE is processed according to the state machine and TOL is updated by this procedure.

If the target destination node is the local node, the Section L.4.6 is followed. If the target destination is another node then Section L.4.5 is followed.

The transmission is always analysed to determine recommended transmission speed from this node to the local node. This transmission speed value is compared with the value in LocalTable.Table for the sending. If the value has changed, the entry for LocalTable.Table is updated and Table.Changed is set to true. . Update the entry for the node in LocalTable.UpdateTimes to be the current time. If the sender is not in the TOL, the transmission speed value may be cached in order to optimize performance if the node joins the ring in the future.

L.4.5 Handing Transmissions Directed to Other Nodes

Transmissions directed to other nodes are not handled by the state machine. However, WTRP must listen for these transmissions (promiscuous mode). Information in these transmissions is used to update local information.

Message.JOIN, Message.INVITE and Message.RETRANS are special transmissions with no user data. Handling these is covered in L.4.5.

A message containing a token directed to another node indicates explicitly that the transmission is directed at another node. WTRP messages will always contain a token, so it will be always be possible to determine where a transmission is directed when it contains WTRP messages.

Arriving tokens may be noted. Token.sender and token.receiver may be useful to provide as operator information to monitor progress of the token around the ring.

If the transmission sender of any D_PDU is Node.Successor, set Node.TokenTransferred to true.

If an arriving D_PDU contains the WTRP token and isMember(token.Sender) is true, set Node.TokenTransferred to true. This setting is used to change out of MON state.

Handling of received message as follows:

1. Messsage.TOL. If isMember(Message.TOL.sender), compare with message.TOL with the current TOL. If they are different and version of message.TOL is more recent, update the TOL with the value in message.TOL and set TOL.Changed to true.

2. Message.TABLE. If the message.TABLE sender node ID is the local node, ignore. If the receive table is from the current ring, update the table in PeerTables.Tables with message.TABLE if the version is more recent.

TOL and TABLE have version numbers encoded as a single byte. To compare version numbers of current and new, Mod(new – current, 256) < 127 will be true if new is more recent than current.

L.4.6 Procedure for Receiving the Token

This section describes how to handle a message with a token that is directed to the local node and therefore needs processing by the local node.

Invite.Next is set to token.inviterID.

If token.critical is true and there is no Message.TOL received, process any message.TABLE as for a transmission directed to another node. Then set TOL.Missing to true. TOL retransmission will then be requested using message.RETRANS and no further processing is done. Other WTRP messages received are retained for processing once the TOL is received.

Next, consider any Messsage.TOL in the transmsission. Compare with message.TOL with the TOL. If message.TOL has higher version, update the TOL to message.TOL and set TOL.Changed to true.

The next step is to update PeerTables, and in the event that any entries are missing, to request them.

PeerTables.ReceivedDirect is cleared, unless a retransmission of receive tables has been requested.

Message.TABLE messages for nodes other than the local node are considered. If the Message.TABLE node ID is not in the TOL, message ignored. For receive tables in the TOL both of the following actions are taken for each Message.TABLE:

The following actions are taken on WTRP messages in the transmission.

1. Compare message.TABLE with the value for the node in PeerTables.Tables. If the version in message.TABLE is more recent, update PeerTables.Tables with this receive table.
2. Add the most recent version of this receive table to PeerTables.ReceivedDirect. This enables passing on the list of messages received, so that updated receive tables circulate back to the node that generated the update.

For each node in the TOL, check that there is a receive table entry in PeerTables.Tables. If there are any missing, set PeerTables.MissingList to the Node IDs of the missing entries. This will lead to requesting the missing receive tables using Message.RETRANS. No further processing is done. Retain any Message.TABLE for the current node for processing once the missing receive tables are received.

If message.CLOSE is received, set Node.CloseRing to true.

Node.Predecessor is set to token.sender. This enables correct successor to be determined when a node appears more than once in the TOL.

The next action is to determine if a complete ring cycle has been made with no changes to TOL. A complete ring cycle is one that is certain to have reached all nodes. The following conditions must be true:

1. That TOL.Changed is false; and
2. That Node.Predcessor is the node before the first occurrence the local node in the TOL. This forces the ring cycle to belong to the first occurrence of the node in the ring.

If a complete ring cycle has been performed, increment Node.cycleCount.

For each entry in LocalTable.Table look at the associated update timestamp in Local.TableUpdateTimes. If it is older than RECEIVE_TABLE_EXPIRY_AGE, remove the entry from LocalTable.Table and set LocalTable.TableChanged to true.

If LocalTable.TableChanged is true, clear LocalTable.Ciruclated. As a new receive table will be sent, there is no need to check if circulation of the previous one completed.

Messages must be circulated around the ring, and two checks need to be made for each the two types of message. If the message has correctly circulated around the ring, then mark so that the check is turned off. If the message has not correctly circulated around the ring, force it to be circulated again. The checks are:

1. If TOL.Circulated is true and no message.TOL received, set TOL.Changed to true to force repeat circulation.
2. If TOL.Circulated is true and TOL.CirculateLCount less than Node.CycleCount and message.TOL received, set TOL.Circulated to false. Also set Tol.CriticalChange to false, as any critical change has been circulated.
3. If LocalTable.Circulated is true and no message.TABLE received for the local node, set LocalTable.TableChanged to true to force repeat circulation.
4. If LocalTable.Circulated is true and LocalTable.CirculatedCount less than Node.CycleCount, set LocalTable.Circulated to false.

Next, Node.NextHopTable is calculated from the receive tables stored in PeerTables.Tables following the procedure specified in L.2.1.10. This information **shall** be passed up to the higher layers of STANAG 5066.

The TOL is now optionally updated, following the procedure in the next section.

The Receive() procedure returns Token.

L.4.7 Calculating the TOL

Calculation of the best TOL is not in general possible. This problem is analogous to the well known travelling salesman problem.

This process invoked from the state machine as the procedure OptimizeTOL(). It is also used directly as a part of L.4.6 when a node has received the token.

This procedure is always called when a node is added or removed from the TOL. When this is done, TOL.CriticalChange is set to true.

The key inputs to calculating a TOL are the receive tables, stored in LocalTable.Table and PeerTables.Tables.

L.2.1.10 specifies calculation of an adjacency matrix from this information, which can be used to determine which nodes are directly connected.

The first check that must be made on the current TOL is to ensure that each step in the TOL has bidirectional connectivity using information in the adjacency matrix. If it is determined that any step is broken the calculation described in the rest of this section must be followed. TOL.CriticalChange is set to true, as the updated TOL needs to be robustly propagated.

A working TOL **shall** only be updated if it is stable. If TOL.Cirulated is false, then the TOL is not being actively circulated and so is stable and shared with all node. The TOL **may** be optimized as described below. It is recommended to optimize the TOL. The TOL **shall not** be changed unless the calculation leads to a new TOL with anticipated improved performance.

The adjacency matrix lists bidirectional links, which are essential for token transfer and recommended maximum transmission speed in each direction.

The following baseline algorithms are suggested:

1. For a node with direct connectivity to only one other node, place this node in the TOL between two instances of the node to which it connects.
2. If the “outer” node on this list only connects to one other node, place this third node on both ends of the TOL being built. This process is repeated to handle nodes connected in a line. Such a line of nodes, with one end connected, is treated as a single node in subsequent calculation.
3. Where a node has only two direct connections, the natural TOL fragment is built.

4. If there are multiple fragments, they are joined to form a ring. If end points do not connect directly, L.2.1.10 specifies how to form a route between a pair of nodes. Preference should be given to nodes not in the TOL and to fast links.
5. Other nodes can be joined into the TOL, seeking first pairs of adjacent nodes to which each remaining node connects to.

There is scope for optimization, to minimize RCL and to use fastest links, noting that links may have different speeds in each direction.

NOTE: It is anticipated that implementation experience will provide input to extend and refine the algorithm specified here, to be updated in a future version of this specification.

If there are nodes that cannot be reached from the current node, they are dropped from the TOL.

L.4.8 Sending Token, WTRP Messages and User Data

A node will transmit data using the Transmit() procedure, which invokes the process described in this section. This is called from the state machine.

When a node has the token, it will make a transmission that includes the at least one copy of the token and may include user data. The token is encoded as an EOW and will usually be repeated many times in the transmission. Mechanisms to facilitate this are set out in L.6. If TOL.CriticalChange is true “Token and Critical TOL” EOW is used. Otherwise “Token Only” EOW is used.

token.inviterID is set to Invite.Next.

If Node.CloseRing is true or Node.OperatorDrop is true, set TOL.CriticalChange to true and follow the procedure RemoveNode(Node.address) and do not send any user data.

The following WTRP messages **must** also be transmitted. These messages may be repeated.

1. If TOL.Changed is true, message.TOL is sent containing the TOL. If TOL.CriticalChange is true it is recommended to send this message several times. TOL.Circulated is set to true and TOL.CirculatedCount is set to Node.cycleCount.
2. If LocalTable.TableChanged is true, message.TABLE is sent containing LocalTable.Table. LocalTable.Circulated is set to true and LocalTable.CirculatedCount is set to Node.cycleCount.
3. For each received table in PeerTables.ReceivedDirect a message.TABLE is sent.
4. If Node.CloseRing is true, message.CLOSE is sent.

User data, if available, **may** be sent in this transmission in addition to the messages above **which** must be sent.

Node.TokenTransferred is set to false. The token has been sent, but it is not considered to be transferred until transfer is (implicitly) acknowledged.

Node.Sucessor is set to the STANAG 5066 address of the node to which the token is being sent.

L.4.9 Controlling Invitations

The token indicates which node is next due to make an invitation. This section specifies the algorithm for the node to determine whether or not to issue an invitation and setting the next inviter

These algorithms are specified in the context of two procedures that can be called form the state machine.

ReadyToSendInvite ()

Return Type: Boolean

Description: Returns true if it is the local node's turn to invite and if the criteria here are met

If Invite.Next is not the local node, return false.

There is a configurable maximum ring size (MAX_NET_SIZE). If the number of notes in the ring is equal to or greater than this size, no invitation is issued. Procedure returns false.

The following parameters are used to control issuing and invitation:

1. Time since Invite.LastTime. If greater than MIN_INVITE_INTERVAL (configurable), an invitation should issued, and procedure returns true.
2. Number of ring circuits since last invite by this node, determined by Node.CycleCount – Invite.LastCycleCount. If greater than or equal to MIN_INVITE_CYCLES (configurable), invite should be issued and procedure returns true
3. InviteNumber. If this is less than or equal to EARLY_INVITE_COUNT (configurable) number and ring circuits less than EARLY_INVITE_CYCLES (configurable), an invite should be issued and procedure returns true. This option **may** be omitted. It is designed to give a higher invitation rate in a new ring.

If none of the above conditions are met, no invite should be issued and procedure returns false.

SendInvite()

Return Type: None

Description: Send an Invite

Send a message.INVITE with:

- message.INVITE.destination = broadcast address
- message.INVITE.TOL = TOL

Update variables as follows:

- Invite.LastInviteTime set to current time
- Invite.LastCycleCount set to Node.CycleCount
- increment Invite.Number
- Invite.Next set using getSuccessor()

L.4.10 Handling Transmissions from Nodes not in the TOL

Where a transmission and WTRP messages are received from a node not in the ring and not simply joining the ring, there are three possible scenarios identified.

1. A node forming a self ring that has not yet heard this ring.   Strategy is to just let it find the current ring and join.

2. A node joining elsewhere in the current ring.   This will sort out without any action.

3. Another formed ring. The approach is to close the current ring, which will enable a merged ring to form, based on the other ring.   Care needs to be taken with rings which are on the edge of communication, because of potential instability due to poor links.   This algorithm requires repeat hearing.

The definitive indication of another ring is token transfer.   If this is not detected, the other transmission is ignored.   If token transfer is detected, this will be recorded.   If NUM_OTHER_RING_HEARD (configurable) of token transmissions are heard within OTHER_RING_TIME (configurable), this is considered definitive detection of another ring, which is within range.

When another ring is definitively detected, set Node.CloseRing to true.

L.4.11 Overall State Diagram

Figure L-9 below shows the complete state diagram of WTRP. Each state is described in detail in the following sections. The set of states are divided into three subsets that correspond to a station unaffiliated with any token ring, a station soliciting membership in a ring that has invited it to join, and a station in operation within an active token ring.

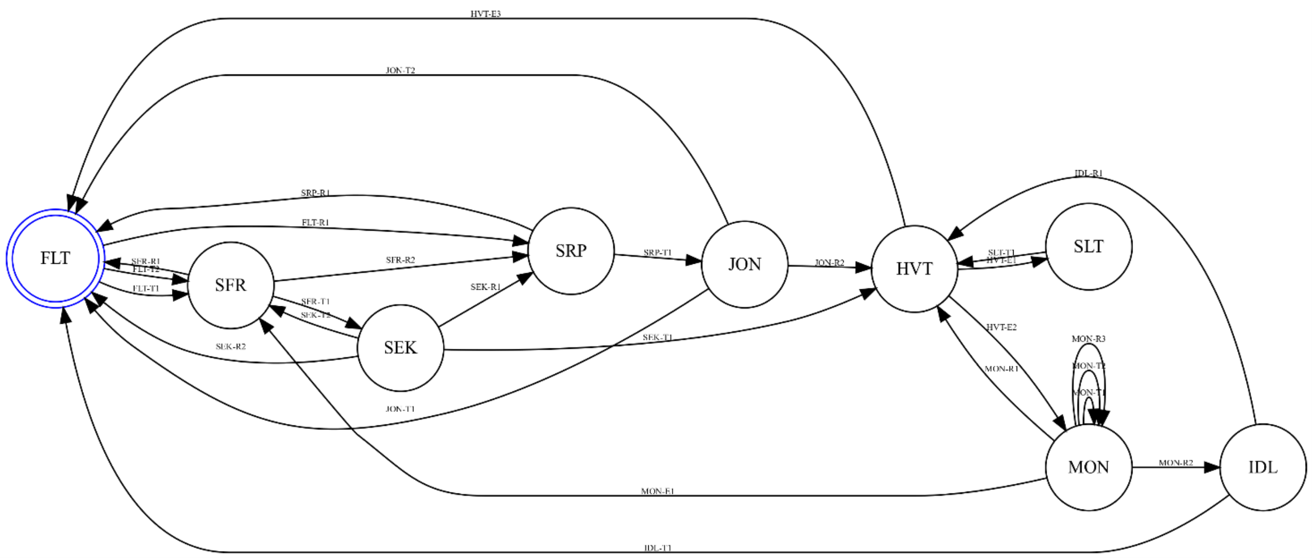

Table L-9 – Overall State Machine

L.4.12 State-Machine Specification

This section and its subsections specify the actions of the WTRP state machine. For every state there are state-entry actions and an outbound-transition table defined. When entering a state, a station first **shall** execute the state-entry actions and then it **shall** wait for an event to occur which triggers one of the transitions to the next state defined in the outbound-transition table. There are two types of event that trigger state transitions:

- Events caused by timeouts; a timeout event is prefixed with the label “Exp” (i.e., for expiry or expired), followed with the name of the timer causing the timeout.

- Events caused by received data; this event is prefixed with the label "Rcv" (i.e., for 'received'), followed with information to clearly identify what is received.

Only one transition rule **shall** be executed after an event: this **shall** be the first and only the first transition for which the condition is met as the state-machine logic examines the outbound-transitions in the order in which they are listed in the table.

When an action causes data to be transmitted, transition to the next state is expected to happen after the data has been transmitted and before any data is received.

L.4.12.1 Floating State (FLT)

The *Floating State* is the WTRP start-up state and the state in which a station is not part of a ring and waits to join a ring. The floating state is a *listening-only* state. A station **shall** stay in the FLT state until there is a joining opportunity (i.e., a message.INVITE is received inviting the station to solicit membership in the ring) or the TCLT timer expires.

The TCLT timer is used to determine when a station **will** assume that there is no existing ring present. If this timer times out (i.e., expires) the station **shall** proceed to *Self Ring State* (SFR).
If the station receives a message.INVITE it **shall** transit to the *Joining State* (JON) via the *Solicit Reply* (SRP) state,

L.4.12.1.1 Floating State entry actions

On this state entry the station **shall** execute the following actions:

- Execute the IntializeNode() function
- Start the TCLT timer.
- Start Receive()

L.4.12.1.2 Floating State outbound transitions

Outbound transitions from the FLT state are shown in the table below. The most significant outbound transition is to the Solicit-Reply (SRP) State, which occurs when a node receives an invitation to join a ring. For other transitions, the node either remains in the FLT state waiting to receive invitations or transits to the SFR state where it will wait before deciding to send its own invitations for nodes to join its ring.

If any transmission is heard, it will cause the node to wait longer using the configurable TCLT_TRANS_HEARD timer, as there may be an active ring.

Table L-10 - FLT outbound transitions

transition	event	Condition	Action	next state
FLT-R1	EOT	Message.INVITE	Save message.INVITE for processing in SRP state	SRP
FLT-R2	EOT		Start TCLT_TRANS_HEARD Timer Receive()	N/A
FLT-T1	Exp: TCLT			SFR
FTL-T2	Exp: TCLT_TRANS_HEARD			SFR

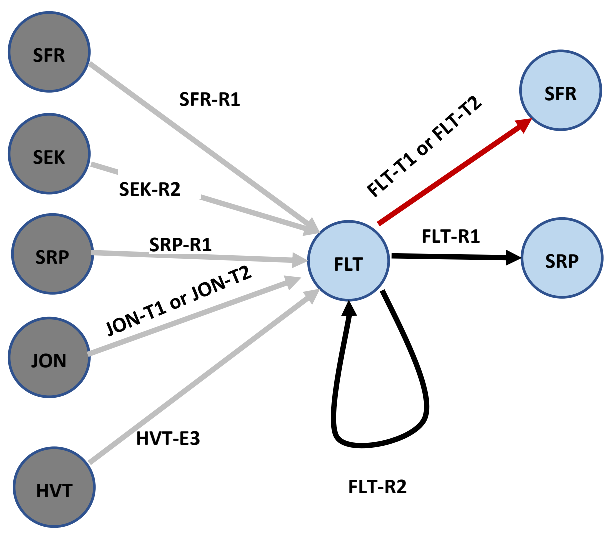

Figure L-11 - FLT Outbound transitions

L.4.12.2 Self Ring State (SFR)

In this state a station has concluded there is no ring to join and therefore will setup a new ring by itself. This condition is called the *self ring*. This state is like the Floating State (FLT) in that it is a listening-only state. On state entry the TSLS timer **shall** be set to a random timeout value, which is used to avoid collisions with any other station trying to setup a ring. If this timer (i.e., if TSLS) times out the station **shall** transit to the *Seeking State* (SEK), sending an invitation to nodes to join its network as an outbound action during the transition.

While waiting for the TSLS timeout, a message.INVITE **might** be received from another station in SEK state; in this case the station **shall** transit to the *Solicit Reply State* (SRP), where it will respond to the invitation.

If any other transmission is received the station **shall** go to the *Floating State* (FLT).

L.4.12.2.1 Self Ring State (SFR) entry actions

On SFR-state entry a station **shall** start its TSLS Timer with a random time-out value over a configurable range.   Receive() procedure.

L.4.12.2.2 Self Ring State (SFR) outbound transitions

Outbound transitions for the SFR state are shown in the table below. Transitions from the SFR state are triggered: when the station receives an invitation from another node and then will reply; when the station receives any other D_PDU from another station and thus should wait for an invitation; or when the station hears nothing for some time and then sends its own invitation.

Table L-11 - SFR-State Outbound-Transition Table

transition	event	condition	action	next state
SFR-R1	EOT			FLT
SFR-R2	EOT	Message.INVITE	Save message.INVITE to be processed in SRP state	SRP
SFR-T1	Exp: TSLS		SendInvite() See L.4.9	SEK

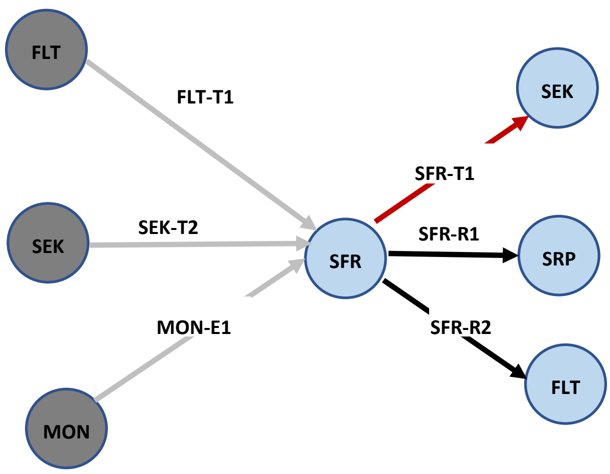

Figure L-12 - Outbound transitions from SFR state

L.4.12.3 Seeking State (SEK)

The *Seeking State* is a state in which a station in a *self ring*, the seeking node, broadcasts a message.INVITE inviting new ring members and listens for replies from solicitors until the TSLW timer times out. The initial TOL for the self ring will have a single entry, which is included in the message.INVITE. Any solicitor will reply by sending a message.JOIN which includes a receive table as message.JOIN.Table. A solicitor may send multiple message.JOIN messages when merging rings. Each message.JOIN message received is processed, which will lead to an updated TOL.

When the TSLS timer expires and no message.JOIN has been received, the node reverts to FLT state. If one or more message.JOIN has been received, the TOL will be updated to reflect all of the new ring members. The station will then proceed to the RCT state.

If a message.INVITE is received before the TLSW timer expires, then another ring (possibly a self-ring) is active within radio range of the station. The station in this case **shall** cease its invitations to other nodes to join its self-ring and **shall** transit to the SRP state, with intent to join the ring it has detected.

Reception by a station when it is in the SEK state of any other transmission shall force the station into the FLT state, as reception of such tokens is an indication there is an active ring within radio range, the self-ring condition no longer applies and the station should not be soliciting to form its own ring.

L.4.12.3.1 Seeking State entry actions

On state exit from the SFR state the station has broadcast a message.INVITE, and shall start the TSLW timer waiting for replies. Start Receive() procedure.

L.4.13.3.2 Seeking state outbound transitions

Outbound transitions from the SEK state are shown in the table below. As noted above, reception of message.JOIN does not force an outbound transition; the message.JOIN messages are stored for later processing and the station remains in the SEK state. Reception of a message.INVITE forces a transition to the SRP solicit reply state and reception of any other transmission forces a transition to the FLT floating state; both of these triggering events invalidate the station's assumption that it can form its own ring but with different responses by the station. As long as the station receives only message.JOIN messages, it will continue to handle them until the TLSW timer expires, at which point the station proceeds to the RCT state and subsequent operation in a multi-node network. If no responses of any kind are heard after waiting for replies, the station returns to the SFR self-ring state.

Table L-12 - SEK state outbound transitions

transition	event	condition	action	next state

SEK-R1	EOT	Message.INVITE	InitializeNode() Save message.INVITE for processing in SRP state	SRP
SEK-R2	EOT	D_PDUs Received set to true		FLT
SEK-R3	EOT	Message.JOIN	Receive() Note that Message.JOIN is processed within Receive() following L.4.4	N/A
SEK-R4	EOT	D_PDUs Received set to false	Receive()	N/A
SEK-T1	Exp: TSLW timer	At least one message.JOIN received		HVT
SEK-T2	Exp: TSLW timer	No message.JOIN received		SFR

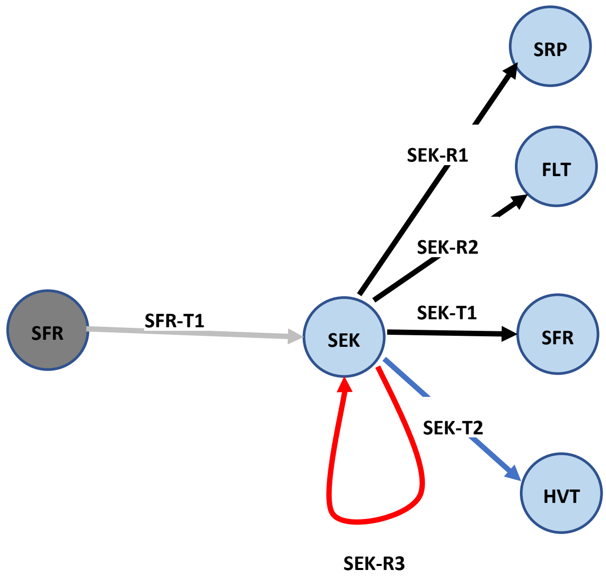

Figure L-13 - Outbound transitions from SEK state

L.4.12.4 Solicit Reply State (SRP)

In the *Solicit Reply State* a station will reply to an invitation to join the network (i.e., to a message.INVITE received from another node) and attempt to join the ring.

The TSRP timer **shall** be started on state entry to determine the moment to reply.

While waiting in the *Solicit Reply State* for the TSRP time to expire, other message.JOIN messages might be received and should be ignored, as they could be expected as replies by other stations to the message.INVITE.

Receipt of other message traffic indicates that the channel is not clear during the protocol's invite-and-solicit dialog for new ring members (an error condition in the protocol). The error condition is best handled by having all solicitors return to a known state (the FLT state) in which they do not transmit.

The soliciting station **shall** reply by sending a message.JOIN to the inviting station, the message.INVITE originator. The message.JOIN contains a receive table, which will enable the inviting node to add the local node to the TOL.

L.4.12.4.1 State entry actions:

Set the TSRP timer following the algorithm defined in Section L.5. Start Receive() procedure

L.4.12.4.2 *Solicit Reply* State Outbound Transitions

Outbound transitions from the solicit transition can be categorized as error recovery or incremental success in joining the network. If any message other than a message.JOIN is heard, the station effectively declares error and transitions to the FLT state, where it will wait for another invitation to join. If the channel remains clear of unexpected traffic, the node sends its solicitation (a message.JOIN) to join the network and transits to the JON state where it will wait to see if its solicitation succeeded.

Table L-13 - SRP outbound transition table

transition	Event	condition	action	next state
SRP-R1	EOT	Joins Received set to false and D_PDUSs Receives set to true (Message.JOINs being sent by other nodes in response to invite are ignored)		FLT

SRP-R2	EOT	(other than SRP-R1)	Receive()	N/A
SRP-T1	Exp: TSRP		Send message.JOIN where message.JOIN.sender = Node.address message.JOIN.receiver = address of inviter message.JOIN.Table = LocalTable.Table	JON

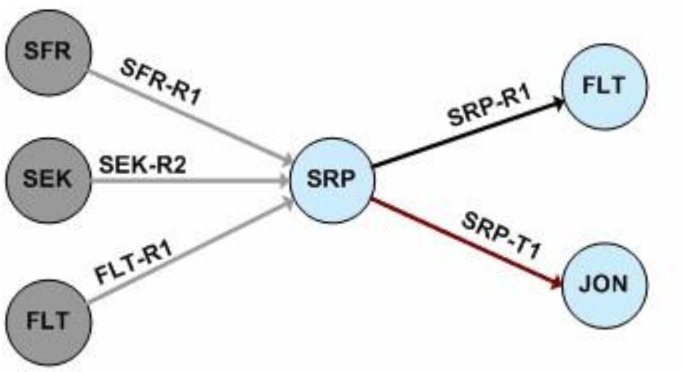

Figure L-14 - Outbound transitions from SRP state

L.4.12.5 Joining State (JON)

In the *Joining State* a station has replied to solicitation opportunity by sending a message.JOIN and is now waiting for the *right-to-transmit token*.

The test that the soliciting station has successfully joined the ring is straightforward; the soliciting station either receives an RTT token or receives an updated TOL sent to another station that includes the local node. The TOL is determined by the inviting station and the local (soliciting) station does not know where it will be placed in the TOL.

If the node determines that it is in the TOL, a longer TCON_IN_RING timer is set, as the node can confidently wait for the token to arrive.

L.4.12.5.1 Joining State entry actions

Start the TCON timer.   Start Receive().

.

L.4.12.5.2 Joining State outbound transitions

Outbound transitions for the JON state are shown in the table below. A station either succeeds with its solicitation to join, and transits to the HVT state as a ring member and able to transmit, or fails and transits to the FLT state where it will wait for the next invitation from a ring member (or its own declaration that there is no ring present and eventual transition to the SFR self-ring state).

Table L-14 - JON-State Outbound-Transition Table

transition	event	condition	action	next state
JON-R1	EOT	isMember(Node.address) is true	Set TCON_IN_RING timer Stop TCON timer Receive()	N/A
JON-R2	EOT	Token		HVT
JON-R3	EOT	isMember(Node.address) is false	Receive()	N/A
JON-T1	Exp: TCON timer			FLT
JON-T2	Exp: TCON_IN_RING timer			FLT

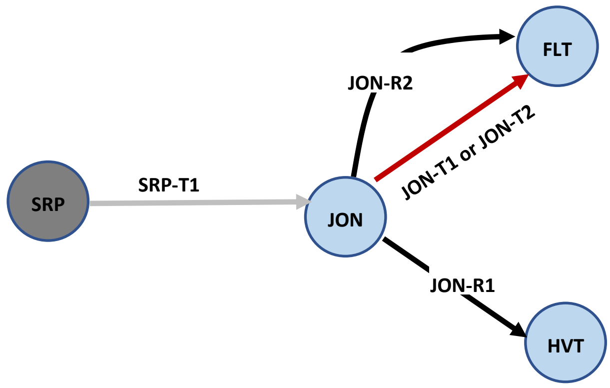

Figure L-15 - Outbound transitions from JON state

L.4.12.6 Have Token State (HVT)

In the *Have Token State* a station has received the *right to transmit token* (RTT) and as part of the state exit action the channel "owner", is allowed to send D_PDUs as long as the length of its transmission does not exceed the *maximum time to transmit*.

L.4.12.6.1 Have Token State entry actions

The node will have received the token in the previous state, which causes the node to enter HVT. The received transmission shall be processed according to L.4.6.

If map or receive table information is needed, but not received, token is sent to predecessor with message.RETRANS and node enters MON state.

The node will verify whether or not an invitation should be sent, following the procedure in L.4.9. If it is determined that an invitation should be sent, and invitation is sent and the node transitions to SLT state. After SLT, the node will return to HVT.

After any invitation has been sent, the token, optional messages and optional user data will be sent following L.4.8, and the node transitions to MON state. L.4.8 will also handle three special situations:

1. If it has been identified that the ring needs to be closed, due to another ring, node will remove itself from TOL and send message.CLOSE to successor along with any messages, but no user data.
2. The HVT state is where the node processes operator or other local requests for the node to drop out of the ring. The node will remove itself from the TOL and send messages but no user data.
3. Where retransmission of Map or TOL is needed.

L.4.12.6.2 Have Token State outbound transitions

Outbound transitions from the HVT state are shown in the table below.

Transitions from the HVT state to the MON state happens after the transmission has completed.

Table L-15 - HVT-State outbound transition table

transaction	event	condition	action	next state
HVT-E1	State entry	ReadyToSendInvite () == true	SendInvite()	SLT
HVT-E2	State entry	ReadyToSendInvite () == false && RCL() > 1	Transmit() following L.4.8	MON
HVT-E3	State entry	ReadyToSendInvite () == false && RCL() <= 1		FLT

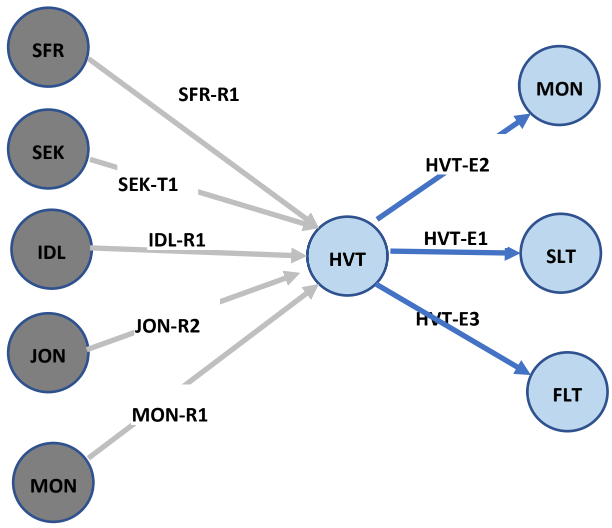

Figure L-16 - Outbound transitions from HVT state

L.4.12.7 Monitoring State (MON)

The *Monitoring State* is a state in which a station has finished transmitting data, passed the RTT token to its successor and is waiting for an acknowledgement. An implicit acknowledgement mechanism is used. Any D_PDU received in the MON state with source address equal to the station's successor are taken as implicit acknowledgement that the successor received the RTT token and has taken control of the radio channel. Also, any token transfer on the current ring indicates that the token is circulating in the ring. This second mechanism addresses when the transmission by the successor is not heard. This monitoring is handled by L.4.5 and reflected in the variable Node.TokenTransferred.

In a two node ring, the token will be passed directly back.

If a critical TOL or has been lost, a message.RETRANS will be received with the token. The token and all messages, but no user data, shall be transmitted following L.4.8.

At expiration of the TPST timer the token transfer is considered to have failed. The station shall resend it for a number of times until the MAX_TOKEN_PASS count has been reached. It is recommended that retransmissions are done at progressively slower speeds.

If the MAX_TOKEN_PASS count has been reached the station declares the next-hop node unreachable and reflects this in connectivity by using the breaklink() procedure. The TOL is then optimized which will give a new next hop, and may remove a node or nodes from the TOL. This process continues until the token is transferred or all other nodes are removed from the ring. If all other nodes are eliminated, the node transitions to SFR.

L.4.12.7.1 Monitoring State entry actions

On state entry the station **shall** start the TPST timer, setting the time the station shall wait for receipt of an explicit or implicit acknowledgement of the *right-to-transmit*.

For any entry other than a self-transition, i.e, if the preceding state is not the MON state, Node.txTokenCounter is set to 1.

Start Receive()

L.4.12.7.2 Monitoring State outbound transition table

Outbound transitions from the MON state are shown in the table below.

Transitions representing a successful pass of the *right-to-transmit* token are to HVT state (token comes back) or to IDL state.

Other transitions will cycle through the MON state to transmit to nodes in turn.

Table L-16 - MON-State Outbound-Transition Table

transition	event	Condition	Action / Comments	next state

MON-E1	State entry	RCL() < 2	If TOL has been reduced to 1, move to self ring	SFR
MON-E2	State entry	Node.NewSucessor == true (set in MON-T2)	Set Node.NewSuccessor to false Transmit token TOL and other messages, and optionally user data following L.4.8	N/A
MON-R1	EOT	Token		HVT
MON-R2	EOT	Node.TokenTransferred == true	Implicit Ack	IDL
MON-R3	EOT	Message.RETRANS	RetransmitMessage(message.RETRANS)	MON
MON-R3	EOT	Node.TokenTransferred == false	Receive()	N/A
MON-T1	Exp: TPST	Node.txTokenCounter < MAX_TOKEN_PASS	Increment Node..txTokenCounter Transmit token TOL and other messages, and optionally user data following L.4.8	MON
MON-T2	Exp: TPST	Node.txTokenCounter >= MAX_TOKEN_PASS	Set Node.txTokenCounter to 1 BreakLink(Node.Successor) OptimizeTOL() Set Node.NewSuccessor to true MON-E1 will check validity of TOL MON-E2 will do the transmission	MON

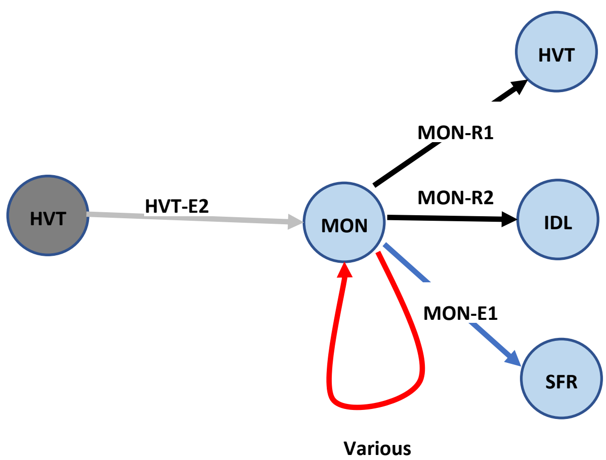

Figure L-17 - MON Outbound transitions from monitoring state

L.4.12.8 Idle State (IDL)

In the *Idle State* the station waits for the *right-to-transmit token* until its TIDL timer expires. When it receives an RTT token, it transits directly to the HVT state. In the IDL state, the node listens for traffic following L.4.4, L.4.5, L.4.6 and L.4.10.

If the TIDL timer expires the station assumes an error-condition in the protocol and transits to the FLT state to recover.

L.4.12.8.1 <u>Idle State entry Actions</u>

The station **shall** start its TIDL timer on state entry. Receive()

L.4.12.8.2 <u>Idle State (IDL) outbound transitions</u>

Outbound transitions from the IDL state are shown in the table below.

Table L-17 - IDL State Outbound-Transition Table

transition	event	condition	action	next state
IDL-R1	EOT	Token		HVT
IDL-R2	EOT		Receive()	N/A
IDL-T1	Exp: TIDL			FLT

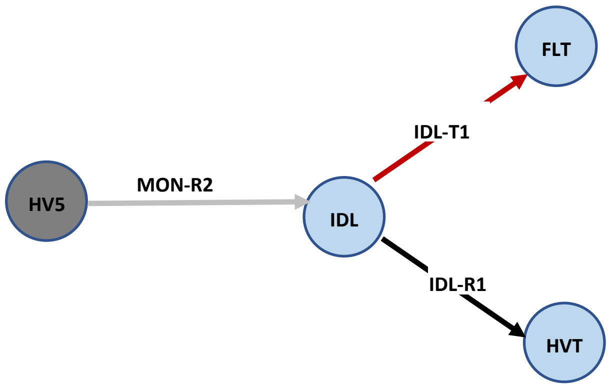

Figure L-18 - Idle State outbound transitions

L.4.12.9 Soliciting State (SLT)

The purpose of the SLT state is to allow new stations to join the ring. A station in the *Soliciting State* is a ring member and has received the *right-to-transmit* from the previous HVT state. A message.INVITE was sent on transition to SLT.

The station waits until TSLW timer times out and then returns to the HVT state. Every message.JOIN is processed, leading to update of the TOL, which will be used to determine transmission in HVT state.

L.4.12.9.1 Soliciting State (SLT) entry actions

On state entry the station **shall** start the TSLW timer and listen for message.JOIN.

Start Receive()

L.4.12.9.2 Soliciting State (SLT) outbound transitions

Outbound transitions from the SLT state are shown in the table below.

The normal transition (SLT-T1) to the HVT state occurs when the TSLW time expires, and is made whether or not new stations have solicited to join.

Table L-18 - SLT-State Outbound-Transition Table

transition	event	condition	action	next state
SLT-R1	EOT	Message.JOIN	Receive()	N/A

			Note that Message.JOIN is processed within Receive() following L.4.4	
SLT-R2	EOT		Receive()	N/A
SLT-T1	Exp: TSLW			HVT

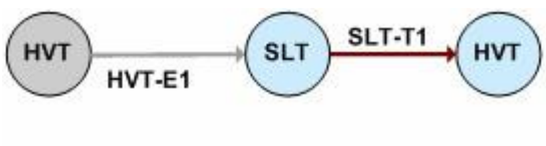

Figure L-19 - Outbound transitions from soliciting state

L.5 TIMERS AND PARAMETERS

This section reviews all of the parameters and timers. The core of this section is a table with all of the specified timers and parameters. The rationale for the parameter and considerations for optimal settings are described. Each of these parameters and timers **shall** be modifiable by an operator to change the protocol responsiveness or behaviour in response to different operational requirements or tradeoffs (e.g., increased collision probability for newly joining nodes versus reduced solicitation overhead).

L.5.1 HF Considerations

HF provides a difficult channel with highly variable speed, with short and extended period where no communication is possible. The nature of this channel is a primary consideration in choosing parameter and timer values.

L.5.2 Surface Wave Considerations

A major target for WTRP is surface wave, particularly for communication between naval vessels where surface communication is good and extends a long way beyond line of sight. WTRP is less suitable for HF Sky Wave, where it will generally be preferable to use ALE to optimize point to point links, rather than attempting to use a single channel for many nodes. For this reason, default parameters are chosen on the basis of using surface wave to communicate between a group of ships. Where WTRP is used in other scenarios, it is likely to be desirable to change defaults.

With surface wave, SNR will systematically decrease with distance. This means that SNR will change as nodes move and will be better for nodes that are closer together. For this reason, it is anticipated that the variable speed operation provided by this specification will be highly beneficial to good performance.

At the limit of communication distance, communications is expected to be very intermittent.   Care needs to be taken with this, as there is possibility to spend resource trying to maintain a link between a pair of nodes where they can hear each other, but the channel is too poor for robust transfer.

L.5.3 Ring Forming Strategy

A node that is not in a ring will attempt to join or form a ring.   A node cannot belong to more than one ring, and having two "intersecting" rings is highly undesirable and should be avoided.   If a node hears an existing ring, the strategy is to wait patiently to join the ring.

If a node is not hearing anything, it is hard to distinguish between the following scenarios:

1. No ring, and potentially other nodes that could form a ring.
2. A large ring with long transit time.
3. An existing ring that is hidden by poor HF conditions

L.5.4 Ring Maintenance Strategy

Once a node is in a ring, efforts should be made to maintain the ring.   Node failure must be allowed for, but is unlikely.   The most common failure is expected to be due to poor HF conditions, and retry is the best approach here.   It is also possible that surface wave distance is increasing between the nodes, leading to the link becoming not being viable

L.5.5 Parameter and Timer Table

Some Table L-19 sets out parameters and timers.   Some timers specify use of jitter to vary the time to prevent nodes deadlocking with repeat cycle.

Table L-19 – Parameters and Timers

Parameter Name	Default Value	States	Notes
TCLT	2 minutes	FLT	Claim Token Timer (TCLT) - Timer used in the floating-state (FLT). Controls the time a station waits while in the floating state to claim a token before transiting to another state; a station restarts its TCLT timer when it transits to the FLT state. A node in this state does not know if there are other rings. Primary strategy is to find another ring. If this timer is set too short, risk is that two nodes will form a ring while there is another ring. Set too long, and it will delay setting a ring. Making this value slightly longer than an expected typical ring cycle time is sensible.

Parameter Name	Default Value	States	Notes
TCLT_TRANS_HEARD	6 minutes	FLT	This is similar to TCL, except that a transmission has been heard (that is not message.INVITE) and it can be inferred that a ring exists. In this situation, it makes sense to simply wait for an invitation to be issued. This is likely to take a number of ring cycles and a loner timer is appropriate. It is likely that the ring will continue to be heard before an invite is issued, so this timer will generally get reset.
TSLS	n/a	SFR	Solicit Successor Timer (TSLS) - This timer is used in the self-ring-state (SFR). If it times out the station shall transit to the seeking-state (SEK). The timer shall be set with a random timeout to reduce the probability of collisions by transmissions from stations attempting to establish different rings at the same time. This timer is derived from the following algorithm Random Number in range 0 to NUM_SEK_SLOTS multiplied by SEK_SLOT_TIME
NUM_SEK_SLOTS	3	SFR	Number of asynchronous seeking slots; a parameter to calculate the TSLS timer. Collisions are not expected to be common, so a fairly small number is reasonable.
SEK_SLOT_TIME	5 seconds	SFR	Time of the slots for TSLS timer. There is no slot co-ordination, so the slot should be somewhat longer than the expected message.INVITE transmission time
TSRP	n/a	SRP	Solicit Reply Timer (TSRP) - Timer used in the solicit-reply-state (SRP). A station starts its solicit-reply timer when it transits to the SRP state. If it expires a station will transit to the JON state where it replies to one of the received SLS tokens, if any, with a SET token. This timer is randomized, as it is possible that multiple nodes will hear message.INVITE and send message.JOIN in response This timer is derived from the following algorithm Random Number in range 0 to NUM_SLS_SLOTS multiplied by SEK_INTERVAL
SLS_INTERVAL	5 seconds	SRP	Time of the slots for TSRP timer. There is no slot co-ordination, so the slot should be somewhat longer than the expected message.INVITE transmission time

Parameter Name	Default Value	States	Notes
NUM_SLS_SLOTS	5	SRP	Number of asynchronous seeking slots; a parameter to calculate the TSRP timer. Collisions are not expected to be common, so a fairly small number is reasonable.
TCON	n/a	JON	Contention Timer (TCON) - Timer used in the joining-state (JON). It controls the time a station waits for a response from another station following an attempt to join the network, so-named because failure to receive a response is attributed to contention with other stations attempting to join the network at the same time; a station restarts its contention timer when it goes to JON state. This needs to be tied to the TSRP timer, as the time to wait is linked to the time which the inviting node will wait before it sends. Then the length of a transmission needs to be waited. The transmission will either advertise the TOL (enabling detection of ring joing) or will pass the token to the local node. The following algorithm is used: (NUM_SLS_SLOTS +1) * SLS_INTERVAL + MAX_TX_TIME This gives time for all nodes to send responses. It assumes that all nodes on the network have same values for the two parameters used here.
TCON_IN_RING	n/a	JON	The node has determined that node is in ring and needs to wait for its turn. This allows the maximum time around the entire ring, with a factor that allows for other nodes being added and retransmissions. MAX_TX_TIME * RCL() * TCON_FACTOR
TCON_FACTOR	1.2	JON	Factor to adjust TCON_IN_RING timer for retransmissions and TOL changes.
TIDL	n/a	IDL	Idle Timer (TIDL) - Timer used in the idle-state (IDL). It controls the time a station waits for its right-to-transmit before transiting to the floating-state (FLT). The following algorithm is based on max time for token to propagate around the ring. MAX_TX_TIME * RCL() * TIDL_FACTOR
TIDL_FACTOR	1.2	JON	Factor to adjust TIDL timer for retransmissions and TOL changes.

Parameter Name	Default Value	States	Notes
TSLW	n/a	SLT, SEK	Solicit Wait Timer (TSLW) - This timer is used in the soliciting state (SLT) and the seeking-state (SEK) by stations inviting new ring members. When it expires, the inviting station it will update the Transmit Order List. The following algorithm is used, which reflects the algorithm used is SRP state to transmit message.INVITE: (NUM_SLS_SLOTS +1) * SLS_INTERVAL
TPST	3 seconds	MON	Token Pass Timer (TPST) - Used in the monitoring-state (MON). It controls the time a station waits after passing an RTT (or other) token to another station and failing to hear an implicit acknowledgement before considering the right-to-transmit as lost. It is expected that the this will happen very quickly in normal conditions. If this fails, either the token was not received or the transmission from next node is not being heard. For the former condition a short value is best. Operational experience is expected to give insight on the best choice for this setting and others.
MAX_TOKEN_PASS	3	MON	Specifies the number retransmissions of a *right-totransmit* after which a station shall stop trying the current node and update the TOL to give a different node.
MAX_NET_SIZE	255	HVT	The maximum number of nodes allowed in the ring. If there is a fixed number of nodes in operation, setting this value will save the overhead of issuing invitations when the ring is “complete”. Setting a low limit will improve ring performance, but is unfair to excluded nodes. The recommended default is to use the upper bound, which in practice means no limit.
MIN_INVITE_INTERVAL	15 minutes	HVT	Minimum interval between which a node will issues invites. Setting this value high will improve performance. Setting it low will enable new nodes to join more quickly.
MIN_INVITE_CYCLES	4	HVT	Minimum number of complete stable ring cycles between a node issuing invites. . Setting this value high will improve performance. Setting it low will enable new nodes to join more quickly.
EARLY_INVITE_COUNT	2	HVT	If a node has issued this number of invites or less, use EARLY_INVITE_CYCLES

Parameter Name	Default Value	States	Notes
EARLY_INVITE_CYCLES	1	HVT	Minimum number of complete stable ring cycles between a node issuing invites when EARLY_INVITE_COUNT is used
RECEIVE_TABLE_EXPIRY_AGE	n/a	HVT	This sets the time when entries should be removed from the receive table. A node is expected to transmit at least once per ring cycle. This timer is tied to the maximum time for a ring cycle and a factor, which allows for a number of cycles and that cycle time may be much shorter than the maximum. The following algorithm is used MAX_TX_TIME * RCL() * RTEA_FACTOR
RTEA_FACTOR	3	HVT	Variable to control RECEIVE_TABLE_EXPIRY_AGE
MAX_TX_TIME	127.5 seconds	HVT, MON	Maximum Transmit Time. This is a general STANAG 5066 control parameter that can be up to 127.5 seconds. By limiting this time, ring transit times will be reduced and latency reduced. However, for bulk transfers, longer times will significantly improve throughput, particularly where only a small proportion of the nodes are sending data.
NUM_OTHER_RING_HEARD	3	Active Ring	Number of transmissions that must be heard from another ring before closing current ring. Setting this too high will delay ring merging and may lead to two nodes transmitting together. Setting it too low may lead to ring tear down when it is not viable to establish a merged ring.
OTHER_RING_TIME	10 minutes	Active Ring	Period within which NUM_OTHER_RING_HEARD transmissions from another ring must be heard before closing current ring. Considerations for setting this parameter are similar.

L.6 TOKEN AND MESSAGE TRANSMISSION

L.6.1 Transmitting Tokens

Tokens are carried in EOWs. Every D_PDU carries exactly one EOW. A token can be carried in any D_PDU which is being sent to the successor, which will include all WTRP messages being sent. STANAG 5066 may require to use EOWs for other purposes. Subject to this, it is desirable to send multiple copies of the token for reliability.

Additional EOWs may be sent using the ACK_PDU or the padding D_PDU specified in S5066-EP2. This can enable extra token copies to be sent, which may be particularly useful if most D_PDUs being sent are to different destinations.

Where possible, the Tokens should be spread out over the length of the transmission, to minimize the effects of fading and data loss.

L.6.2 Transmitting Messages

L.4.8 specifies which messages are required to be sent. Messages are small and it may be beneficial to repeat them, particularly if a higher transmission speed is chosen. Where Map and Table messages are critical, it is particularly desirable to duplicate.

Where messages are duplicated, it is best to have them at different parts of the transmission, rather than close together.

All messages are processed after the transmission is received, so there is no particular benefit to placing them in a particular part of the transmission.

L.6.3 Speed and Interleaver Selection

WTRP can be operated at fixed speed or at variable speed. This specification provides a recommended maximum transmission speed for each node. When making a transmission, the speed associated with the "slowest" node needs to be considered as maximum speed for the whole transmission. Where speed is highly variable, this may impact the choice of which D_PDUs to send.

When there is a lot of data to send, it is recommended to use the maximum transmission speed with a long or very long interleaver. This is expected to maximize throughput and reliability.

Messages will often be WTRP messages only, which are generally small. Where there is a smaller amount of data a slower speed and shorter interleaver is likely to give better performance.

When sending message.INVITE to a broadcast address, it is recommended to use a conservatively slow speed. Similarly with message.JOIN. Once communication is established, both ends can use SNR to determine an appropriate operational speed.