The Analytical Model of Distributed Interrupt Mechanism in SpaceWire Network

Liudmila Koblyakova
SUAI
Saint-Petersburg, Russia
liudmila.koblyakova@guap.ru

***Abstract*—The distributed interrupt mechanism is intended for reliable and low-latency transmission of system signals with acknowledge in SpaceWire networks. Important tasks are rigorous proof of the distributed interrupt mechanism properties and time characteristics calculation. For these purpose the analytical model has built, which describes the distributed interrupt mechanism in terms of graph theory. The article shows basic data, assumptions and notations for graph theory based model building, describes distributed interrupt propagation algorithm in terms of graph. Here is proved the correctness of the algorithm, i.e.: distributed interrupts/acknowledges wave propagation time is finitely; the distributed interrupts/acknowledges wave propagates to all nodes of a graph. The assertions are proved: acknowledge wave to the interrupt does not cross in time with distributed interrupt wave if the interrupt processing time is greater than the maximum interrupt propagation time; the next interrupt wave does not cross with acknowledge wave to the previous interrupt if the time interval between acknowledge receiving and next interrupt generation is greater than the maximum acknowledge wave propagation time. Here are provides algorithm consequence that Interrupt-code (Acknowledge-code) from source node (handler node) to all other graph nodes propagates by the shortest path and forms oriented covering tree. The necessity of the distributed Interrupt mechanism parameters restrictions determined looping problem in networks with loops and provides a solution to this problem, which makes changes to the algorithm and prove the correctness of the changes.**

I. INTRODUCTION

Standard SpaceWire [1] for onboard communication networks integrates the data and control information transmission. According to consolidated set of requirements for SpaceWire-RT from European and Russian industry [2] the important task for onboard distributed real-time systems is single a signals transmission with acknowledge to inform devices about system critical events in real-time, such as equipment failure or readiness to some action. For this purpose, the distributed interrupt mechanism is included to the second edition of SpaceWire standard. Hard real-time signalling imposes strict signal delivery constraints and requires high reliability of signal delivery [3], so the strict proof of the distributed interrupt mechanism properties is also important task.

The Distributed Interrupt mechanism uses broadcast distribution of hard real-time signals providing ultra-low delivery latency and high reliability.

A Distributed Interrupt code consists of the 4-bit SpaceWire Escape character followed by a 10-bit SpaceWire data character; the total size of the distributed Interrupt code is 14 bits. Distributed Interrupt codes take priority over SpaceWire FCT characters, data characters and NULL control codes. Therefore, the transmission of Interrupt signals is not affected by data packets flowing through the same links. As a SpaceWire control code, each 14-bit Interrupt code carries 8-bit data field which, in turn, contain 3-bit code identifier and 5-bit Interrupt identifier. The 3-bit code identifier is used to distinguish Interrupt codes from other SpaceWire control codes (e.g. Time-codes) as well as to determine the type of Interrupt code. There are two types of Distributed Interrupt codes. Each Interrupt request (Interrupt code) has a particular 5-bit Interrupt identifier that is used to distinguish this request from other Interrupt requests in the network. Therefore, in a network there may be up to 32 different Interrupt requests with identifiers from 0 to 31. It is assumed that for any Interrupt request in the network there is at least one node that is assigned to receive and process the code. Such node is called an Interrupt handler. When an Interrupt handler receives an Interrupt request which this handler is assigned to process, it may issue a confirmation code that is called an Interrupt acknowledgment, which is another type of Interrupt code. Each Interrupt acknowledgment has the same Interrupt identifier as the correspondent Interrupt request.

Broadcast distribution of Interrupt codes allows simple configuration of the network that does not require routing tables in switches. However, broadcast distribution in networks with circular connections may lead to repeated propagation of Interrupt codes. So as to overcome the problem each SpaceWire switch or node has a 32-bit Interrupt Source Register (ISR). Each i-th ISR bit corresponds to the Interrupt identifier with the same number. When a node issues an Interrupt request or a node or a switch receives an Interrupt request, the correspondent

bit of the ISR must be checked. If the bit is already set to '1', it means that the incoming Interrupt request is invalid and must not be forwarded or processed. Otherwise, if the bit is '0', it is switched to '1' and the correspondent Interrupt request is considered to be valid for processing in the node and forwarding to all the output ports of the switch. On the contrary, an incoming Interrupt acknowledgment code is assumed to be valid if the correspondent ISR bit is '1' and invalid otherwise. The example of Interrupts and Acknowledge codes is shown in Fig. 1.

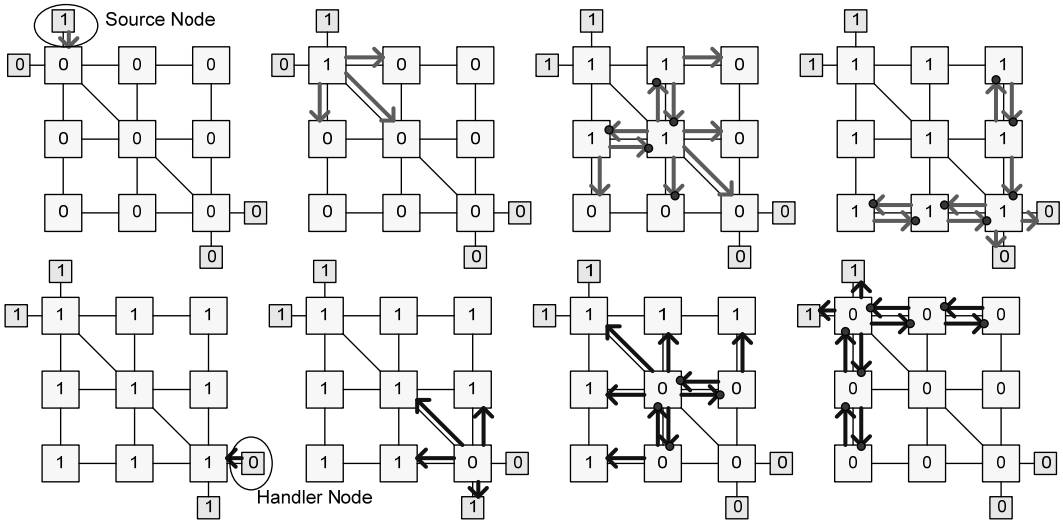

Fig. 1. Example of the Interrupt and Acknowledge codes propagation

To prove the correctness of work of the distributed interrupt mechanism and to calculate a time characteristics the analytical model was built, it describes the distributed interrupt propagation mechanism in terms of graph theory. The distributed interrupt mechanism in detail with time characteristics, timeouts and errors recovery described in previous papers [4-7], which use the results of the analytical model, but the following proofs and details of analytical model was out of their scope. So, in this paper, the description of this analytical model provided and their properties have proved.

II. INPUT DATA FOR DISTRIBUTED INTERRUPT ANALYTICAL MODEL BUILDING

A. Network model description

For description of the distributed interrupt mechanism a graph theory is used,[8, 9]. Let represent a network in a form of finite graph *G(V, E)*. Graph's edges are network links. Graph's nodes are network nodes and routers.

The aim is to build such analytical model and derive such formulas, which will be valid for any network topology with known following parameters (Table I):

Let make the following assumptions:

- the graph is connected
- terminal nodes have only one input/output port, so end nodes of a graph correspond to terminal network nodes. The nodes with degree greater than one correspond to routers

TABLE I. NETWORK PARAMETERS

Designation	Description
D	Number of edges in a shortest path between two most distant nodes.
P_{Len}	Number of edges in a longest simple path between two most distant nodes (graph diameter)
T_{bit}	Single bit transmission time over channel
N_{CC}	Number of bit in control-code symbol
T_{wtc}	Code propagation time throw a router without delay to wait for the previous code transmission

- network consist of minimum two nodes, one of this correspond to network node, i.e. $D \geq 1$.

Definition. Wave propagation control code will be call the process of code's propagation from the source code to all the other nodes in the network.

In the distributed interrupt mechanism could be up to $2^{N_{ICB}}$ different interrupt types, where N_{ICB} – number of bit for Interrupt/Acknowledge code (in symbol on symbol level) is a system parameter. For every interrupts type there is only one source, and every interrupt source could send their code independent from other sources (in parallel). Similarly, there is the same number of interrupt handlers (acknowledges source), that also work separately from each other. Therefore, in the system at the same time could propagate up to $2^{N_{ICB}}$ interrupt/acknowledge wave.

Every Acknowledge-code propagates with priority next after time-code, and Interrupt-codes propagate with next after Acknowledge-code priority.

Description of a every such code propagation process separately is similar to the shortest path finding algorithm Dijkstra, and traverse the graph in width.

B. Notation system for describing the distributed interrupt mechanism model using graph theory

Marking graph:

For each graph node associate pair of numbers $v(x, y)$, where $x \in \{0,1\}$, and it is a flag of permission/prohibition of distributed interrupt code or acknowledge code propagation; y is a value of real (astronomical) time when the last interrupt/acknowledge code was accepted. Time is measured in user-defined units, for example in ns). For designation of undefined time, we use symbol «∞».

For each edge associate a number – weight $c(v_i, v_j)$, which depend on link speed and means real (astronomical) control code transmitting time over link between two nodes v_i and v_j.

Introduce notations and parameters in Table II.

TABLE II. NOTATIONS AND PARAMETERS FOR DISTRIBUTED INTERRUPT MECHANISM DESCRIPTION

Designation	Description
F_{CI}/F_{CH}	Current interrupt/acknowledge wave front, consist of nodes pairs $(v_i(x, y), v_j(x, y))$. Each pair shows that interrupt/acknowledge-code propagates to node $v_j(x, y)$ from node $v_i(x, y)$. If v_j is end node (interrupt source or handler), than instead first node use «-»
Ft	Auxiliary set, elements similar to F_{CI}/F_{CH} set elements and shows that interrupt/acknowledge-code has come to node $v_j(x, y)$ from node $v_i(x, y)$, but not checked yet, i.e. unknown whether this code will pass further or not
F_{NI} / F_{NH}	New interrupt/acknowledge wave front. Elements similar to F_{CI}/F_{CH} set elements. Every pair shows that interrupt/acknowledge-code come to node $v_j(x, y)$ from $v_i(x, y)$, checked and wait further transmission.
D_I / D_H	Distance sets from source/handler node. Set's element $d(s, (v_i, v_j))$ shows weighted distance (time, edge's weight sum in path) from interrupt/acknowledge source node s to node v_j, and A path from node s to v_j contain the edge (v_i, v_j). For short writes $d(v_i, v_j)$
T_{SI} / T_{SH}	Interrupt/acknowledge sending time. If time undefined the symbol « ∞ » in used
t_H	Interrupt processing time (time from interrupt receiving to acknowledge sending). It's distributed interrupt mechanism parameter.
t_G	Time period from acknowledge-code receiving time by source node to new interrupt-code send time to a network. It's distributed interrupt mechanism parameter.

III. THE ALGORITHM OF INTERRUPT AND ACKNOWLEDGE CODES PROPAGATION

A. The algorithm in terms of graph theory

Describe the algorithm in terms of graph theory using notations.

Algorithm 1 The algorithm of the distributed Interrupts mechanism

Step 1. Initialization

All nodes mark as $(0, \infty)$. Sets F_{CI}, F_{CH}, F_{NI}, F_{NH}, F_t are empty, $F_{CI} = \{\}$, $F_{CH} = \{\}$, $F_{NI} = \{\}$, $F_{NH} = \{\}$, $F_t = \{\}$. System time T=0. $T_{SI} = 0$. $T_{SH} = \infty$

Step 2. Start wave propagation Interrupt. Source node - $v_s(0, \infty)$, change its mark on $v_s(1,T)$. Source node include to set $F_{NI} = \{(-, v_s(1,T))\}$.

Step 2.1.

1) For every pair $(v_i(x, y), v_j(x, y))$ from the set F_{NI} review all adjacent to node $v_j(x, y)$ nodes excluding node $v_i(x, y)$.
2) For every adjacent node $v_k(x, y)$ calculate distance to it $d(v_j, v_k) = (y_j - y_s) + c(v_j, v_k)$ from the source node and add it to set D_I.
3) Add node v_k to set F_{CI} in pair with node $v_j(x, y)$. All nodes are removed from F_{NI}.

Step 2.2.

1) Find in set F_{CI} node which has minimal distance $d(v_j, v_k)$ and first part of mark is zero ($x_k = 0$), and move to set F_t all nodes with distance less or equal to distance to found node. If the set F_{CI} does not contain nodes with zero mark of first part than all nodes move to set F_t.
2) All D_I set's elements with distance less and equal to distance to found node are deleted.
3) All nodes from F_t set with zero mark of first part add to F_{NI} set.
4) System time is $T = y_s + d(v_j, v_k)$, if there is node with zero mark first part, or $T = y_s + max\{D_I\}$ (maximal distance in D_I set.
5) For each node's pair $(v_i(x, y), v_j(x, y))$ from F_t set the mark of v_j node change to $(1, T)$.
6) All nodes from F_t set are removed.

Step 2.3.

Check the F_{CI} set and mark handler node $v_h(x, y)$. The following warrants are possible:

1) The F_{CI} set is not empty (this means that interrupt wave has not finish its propagation),

Algorithm 1 The algorithm of the distributed Interrupts mechanism

node $v_h(x, y)$ has in label $x = 0$, so the interrupt code has not reach to handler node yet. Go to Step 2.1

2) The F_{CI} set is not empty (this means that interrupt wave has not finish its propagation), node $v_h(x, y)$ has in label $x = 1$, so interrupt code has reached handler node. If $T_{SH} = -1$, than define acknowledge code sending time as $T_{SH} = y_h + t_{H.}$ Go to Step 2.1.

3) The F_{CI} set is empty (this means that interrupt wave has finished its propagation), node $v_h(x, y)$ has in label $x = 1$, so interrupt code has reached handler node. If $T_{SH} = \infty$, than define acknowledge code sending time as $T_{SH} = y_h + t_{H.}$ Go to Step 3.

4) The F_{CI} set is empty, node $v_h(x, y)$ has in label $x = 0$. This means that there is no handler node in the system or graph is disconnected. Go to Step 6.

Step 3. $T_{SI} = \infty$. Interrupt code processing finish waiting: $T = T_{SH}$

Step 4. Start wave propagation acknowledges. Handler node (source of acknowledges) - $v_h(1,T)$, change its label to $v_s(0,T)$. Include handler node to $F_{NH} = \{(-, v_h(0,T))\}$ set.

Step 4.1.

1) For every pair $(v_i(x, y), v_j(x, y))$ from the F_{NH} set review all nodes adjacent to node $v_j(x, y)$ excluding node $v_i(x, y)$.

2) For every adjacent node $v_k(x, y)$ calculate the distance to it from the handler node: $d(v_j, v_k) = (y_j - y_s) + c(v_j, v_k)$, and add it to D_H set.

3) Add v_k node to F_{CH} set in pair with $v_j(x, y)$ node. all nodes removed from the F_{NI} set.

Step 4.2.

1) Find in F_{CH} set node with minimal distance $d(v_j, v_k)$, which has first label part equal to one and move to F_t set all nodes with distance less or equal to distance to found node. If the F_{CH} set does not contain nodes with first label part equal to one, than all nodes move to the F_t set.

2) All elements with distance less and equal to distance to found node are removed from the D_H set.

3) All elements from F_t with first label part equal to one are added to F_{NH} set.

4) System time $T = y_s + d(v_j, v_k)$, if there is node with label equal to one, or $T = y_s + max(D_H)$ (maximal distance from D_H set).

5) For every pair of nodes $(v_i(x, y), v_j(x, y))$ from the F_t set for node v_j change the label to $(0, T)$.

6) All nodes from the F_t set are removed.

Step 4.3.

Check the F_{CH} set and source node label $v_s(x, y)$. The following variants are possible:

1) The F_{CH} set is not empty (this means that acknowledge code wave has not finish its propagation), node $v_s(x, y)$ has in label $x = 1$, so acknowledge code has not reached source node. Go to Step 4.1.

2) The F_{CH} set is not empty (this means that acknowledge code wave has not finish its propagation), node $v_s(x, y)$ has in label $x_s = 0$, so acknowledge code has reached source node. If $T_{SI} = \infty$, than define interrupt code sending time as $T_{SI} = T + t_{g.}$. Go to Step 4.1.

3) The F_{CH} set is empty (this means that acknowledge code wave has finished its propagation), node $v_s(x, y)$ has in label $x = 0$, so acknowledge code has reached source node. If $T_{SI} = \infty$, than define next interrupt code sending time as $T_{SI} = T + t_{g.}$ Go to Step 5.

4) The F_{CH} set is empty, node $v_s(x, y)$ has in label $x = 0$. This is unreachable state. Go to Step 6.

Step 5. Acknowledge code propagation wave has finished. $T_{SH} = \infty$. The generation of the next interrupt code with the same type is waiting $T = T_{SI}$. Go to Step 2.

Step 6. Exit

B. Proof the algorithm correctness

Subsequent execution of Step2-Step5 will call main iteration. It corresponds to interrupt code has been: generated, sent to network, received by all nodes, processed, after than acknowledge has been generated, send to network, received by all nodes including source. Cycled execution of Step 2 – Step2.3 corresponds to interrupt code wave propagation from the source to all other nodes. Cycled execution of Step 4 – Step 4.3 corresponds to acknowledge code wave propagation from handler to all other nodes. Prove that:

1) Step 3 reachable, Step 5 reachable, i.e. interrupt/acknowledge wave propagation time is finitely.

2) Step 2.3.4 and Step 4.3.4 unreachable, i.e. interrupt/acknowledge code wave propagates to all graph nodes.

Proof. In the beginning of Step 2 all nodes have the first part label value $x = 0$, that correspond to interrupt code is allowed to be passed throw any node. Further, at the every repeat of Step 2.2.5 interrupt code reaches to one or several nodes, so at every step the number of nodes to which the code has not yet reached, reduced. Due to the finiteness of the graph (see 2.A) for finishing the interrupt code wave propagation required the finite number of steps. The acknowledge wave propagates similarly, except label value changes from 1 to 0 and the reducing the number of nodes, to which acknowledge code has not reached yet, takes place in Step 4.2.4. So the first part is proved.

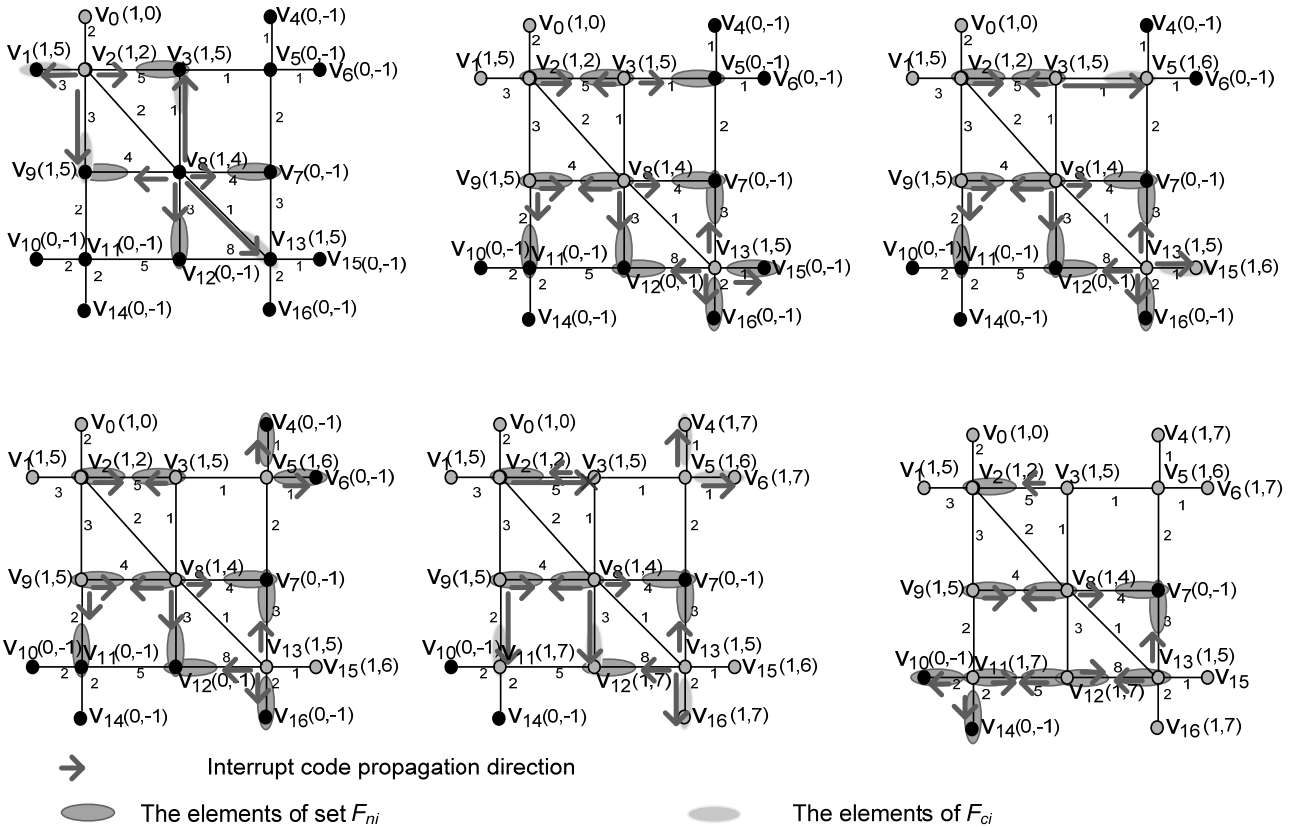

Fig. 2. Fragment of algorithm work

In Step 2.1.1 (4.1.1) all adjacent nodes for nodes, to which interrupt/acknowledge code has already reached, are reviewed, and to F_{CI}/F_{CH} set the nodes added, which interrupt/acknowledge code has already sent to.

In step 2.2.2. (4.2.2) from the F_{CI}/F_{CH} set the nodes are removed, which code reached to and its first part label changed to 1 (0). Since the number of steps is finitely, than for finite number of steps will occur situation when the graph will not contain the nodes, which will be possible to add to F_{CI}/F_{CH} set. Next, as at every repeat of Step 2.2.2 (4.2.2), the number of nodes in set F_{CI}/F_{CH} is reduced, then for the finite number of steps F_{CI}/F_{CH} set becomes empty, that means that interrupt/acknowledge code has reached to all nodes. If some node will have first part label equal to 0 (1) that will mean that graph is disconnected, which contradict to par.2.A. So the second part is proved.

C. Proposition about interrupt and acknowledge wave cross

As was said earlier, after interrupt wave follows acknowledge wave, and after could follow interrupt wave again, and so on. Let prove the next proposition.

Proposition:

1) Acknowledge wave to the interrupt does not cross with the interrupt wave in time, if time of interrupt processing tH will be more than maximum interrupt processing time.

2) The next interrupt wave does not cross with wave of acknowledge codes to previous interrupt, if time interval between acknowledge received and next interrupt generated tg will be more than maximum acknowledge code wave propagation time.

Proof. Consider two points of real time. First point, in the beginning of Step 3, when interrupt wave has finished its propagation. The value of system time (1)

$$T = y_s + d(v_j, v_k) \quad (1)$$

Where v_k is the last node, which the code of current wave reached to, that is time $d(v_j, v_k)$ correspond to interrupt wave propagation time. Second point of time correspond to acknowledge code generation time (2):

$$T_{SH} = y_h + t_H, \quad (2)$$

Where y_h is time when handler node has received interrupt code, obvious that

$$y_s < y_h \le y_k \quad (3)$$

Waves does not cross and time value is increasing, so the following is true:

$$y_s + d(v_j, v_k) < y_h + t_H \quad (4)$$

$$y_s + d(v_j, v_k) < y_s + d(v_i, v_h) + t_H \quad (5)$$

$$d(v_i, v_h) + t_H > d(v_j, v_k) \quad (6)$$

$$t_H > d(v_j, v_k) - d(v_i, v_h) \quad (7)$$

Based on Step 2.2.4-Step 2.2.4:

$$d(v_j, v_k) = y_k - y_s, \quad (8)$$

Thus, have

$$t_H > (y_k - y_s) - (y_h - y_s) \quad (9)$$

$$t_H > y_k - y_h \quad (10)$$

Due to inequality 3, get:

$$t_H \ge y_k - y_h, \quad (11)$$

That is to guarantee that waves does not cross, interrupt processing time t_H shell be more than interrupt wave propagation time, which required in first part. The proof of the second part is similar, because the process of interrupt and acknowledge waves is equal exclude the first part label value, which is not involved in the proof. So the proposition is proved.

D. Consequence of the algorithm

Let prove the consequence. Interrupt (acknowledge) code from the source node v_s (handler node v_h) to all other network nodes propagates by the shortest path and forms the oriented covering tree.

The proof. Consider the next iteration of interrupt/acknowledge codes wave propagation. Every times at the performing of Step 2.2.1- Step 2.2.3 (Step 4.2.1 – Step 4.2.3), to the F_{NI} (F_{NH}) set the pair of nodes are added, which the minimal distanced path has found for, that is the nodes is defined, which the interrupt (acknowledge) code has reached to. The source node $v_s(v_h)$, which ascribe to the minimal distance, is added to this set first. Further, for this node all adjacent nodes (the weight of incident to node v_s (v_h) edges) are reviewed. So the following is added to the pair of node (v_s (v_h), v_k), where v_k has minimal distance (weight) (Step 2.2.1, Step 4.2.1) from the source (handler), which is equal to edge weight $c(v_s(v_h), v_k)$ over which code has come (Step 2.1.2, Step 4.1.2). By induction, at every Step 2.2 (4.2) will be defined and added to the F_{NI} (F_{NH}) set the new edge (v_k, v_n) by which the interrupt (acknowledge) code has reached the node v_n. And, the distance to v_k had been found at the previous step and it is minimal, also the node is chosen based on a fact that from all nodes which interrupt/acknowledge code has not reached yet, to this node the distance is minimal. So the code has reached to node v_n by the shortest path. If there are several paths to the node, then each other will not be shortest (that is if other path will be shorter than founded path, that will be contradicted to Step 2.2.1 (4.2.1)). The interrupt (acknowledge) code, reached to the node by the not shortest path, will not propagate further, that is: at Step 2.2.5 (4.2.5) the first part label value has changed when the interrupt (acknowledge) code reached to the node in first time by the shortest path; and by condition at Step 2.2.1 (4.2.1) the node with the first part label value is not equaled to zero, has not choose to the further transmission, that is repeatedly received code is ignored. So the node v_n was chosen arbitrarily, so we could conclude that interrupt (acknowledge) code to any node comes by the shortest path and repeatedly comes codes are ignored. That is required to prove in the first part.

The edges (nodes pairs included to F_{NI} and F_{NH} sets) forms the orienting shorter paths trees with root v_s for interrupt codes and v_h for interrupt acknowledges, that is at every repeating of Step 2.2 and 4.2 the new edge and node are added, which correspond to minimal distance, so the algorithm's work at every iteration can be regarded as continuous building shorter paths trees of the interrupts propagation in the first part iteration and acknowledges propagation at the second part. The Step 2 (4) will be finished when all nodes will be added to the F_{NI} (F_{NH}) set and built oriented tree will contain all graph nodes, so we get the oriented covering tree that is required to prove in the second part of consequence.

Oriented covering tree or the shorter paths tree shows all shorter paths of interrupt/acknowledge codes propagation from the source node (handler node) which is the root of tree, to all other graph nodes. The every path length, which corresponds to interrupt/acknowledge code time propagation, is a sum of edge weight contained in the path. Source node (tree root) and leaves trees correspond to

network terminal nodes. The number of tree levels is equal to parameter *D*.

IV. The necessity of the distributed interrupt mechanism parameters restrictions

A. The looping problem determining in the network with cycles

Earlier the proposition was proved that for interrupt and acknowledge waves does not cross, the parameters value t_G and t_H should be more than interrupt/acknowledge propagation time. Let us justify the need of that the waves of the interrupts and acknowledges should not cross in time.

Obvious that in networks without cycles (trees) there is only one path for code propagation from the source to every other node, so the looping is not possible. In the cycled network there are several path between nodes, so interrupt and acknowledges wave imposition may cause the looping problem. Lets consider the next example. Chouse the graph as in the first example and look at another part of time. Take $t_H = 1$. In the Fig. 3 the fragment of algorithm work is shown before the problem situation is appear. In the Fig.4 the looping problem is shown.

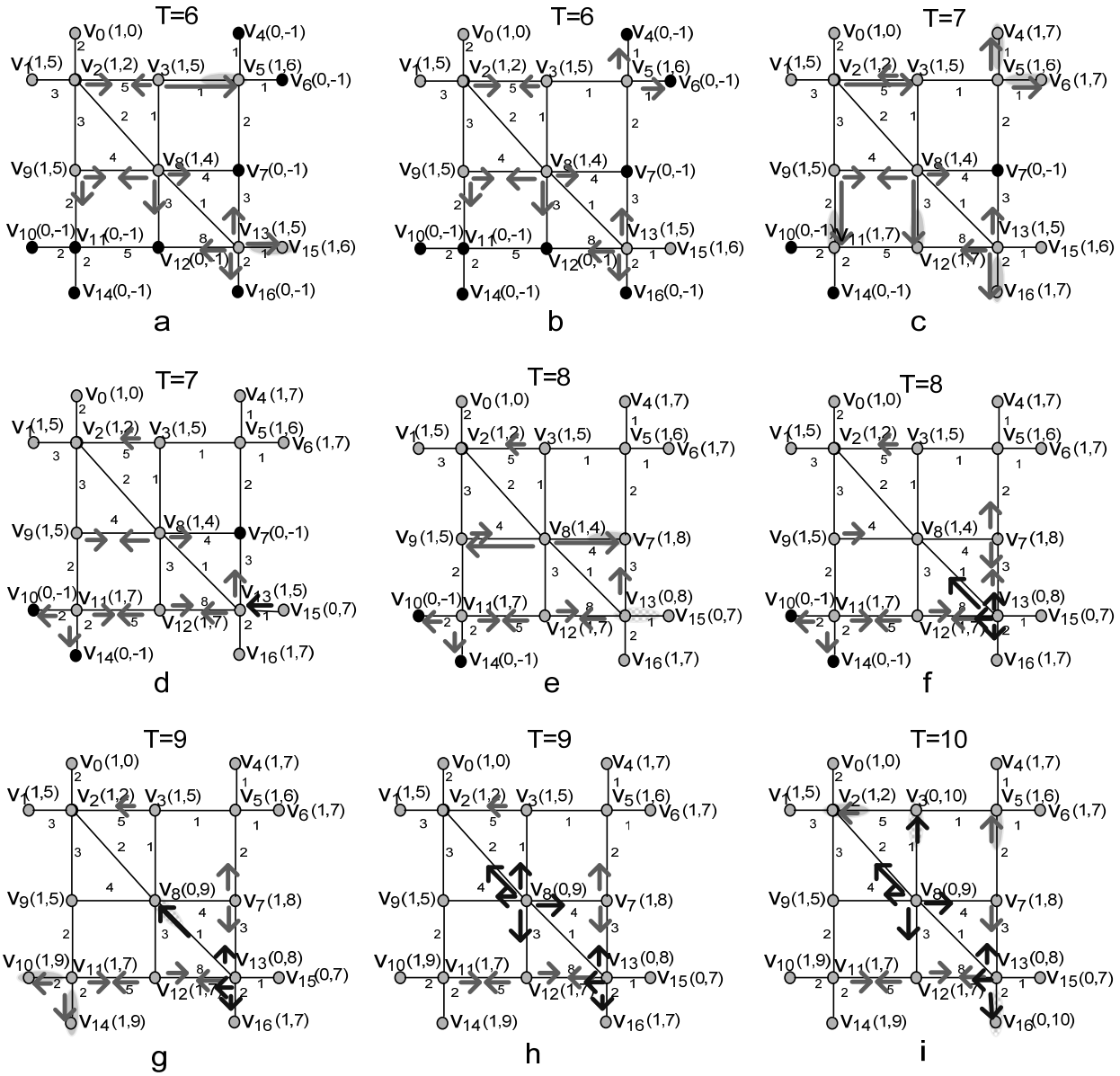

Fig. 3. Algorithm work before looping problem

The picture shows that in time T = 6 into the handler node v_{15} (Fig.3.a) interrupt code come, and in that handle time t_H = 1, then in time T = 7 (Fig.3.d) handler sends acknowledge code to network which start propagating in network. From this times point the wave of interrupt codes and the wave of acknowledge codes propagate in the network simultaneously. The Fig. 3.c-g show that interrupt code comes into some nodes again in does not propagate further because the condition in Step 2.2 of the algorithm is not fulfilled. This condition is fulfilled only after the acknowledge code will pass throw this nodes and the first part label value will be changed. That's interrupt code will not be pass to the link again.

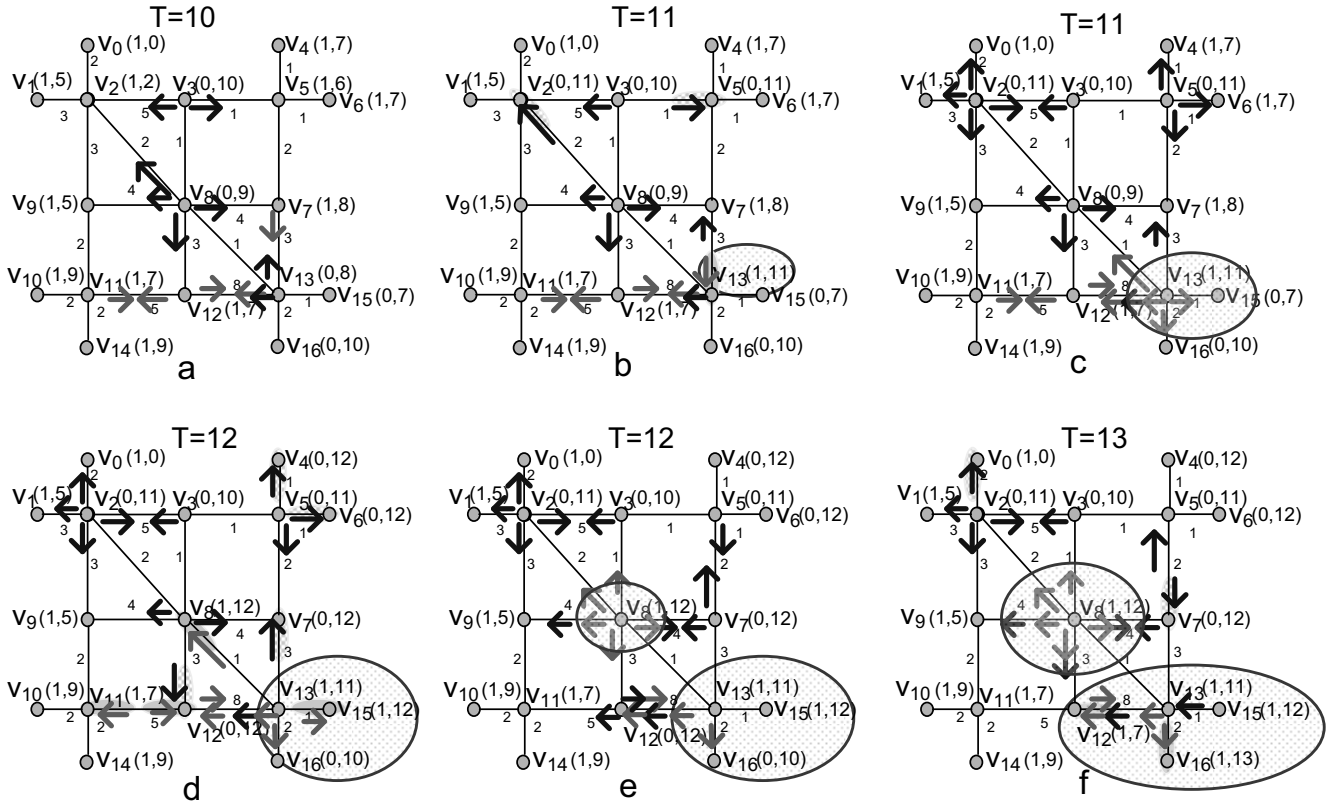

Fig. 4. Looping problem example

At the time point T = 11 (Fig.4.b) interrupt code came again to the node v_{13}, throw which the interrupt code has already been passed and the label value was changed, as a result the erroneous situation occur, scilicet the label value has changed again and the interrupt code will be passed again to the link, which it has already passed throw. Further (Fig.4.c-f) at the each subsequent times point interrupt codes will be pass erroneously throw the nodes, which they reached again, and thereby they will generate new interrupt codes which should not be in the system. So at the times point T = 12 the interrupt code repeatedly reached handler node and will interpreter as new interrupt code, which it generate acknowledge for. So, in such network the number of interrupt and acknowledge codes will increase and interrupt and acknowledge waves may never finish their propagation in a certain situation. For example, in Fig.4.f at the lowest speed link between the v_{12}, v_{13} nodes after each other propagates interrupt and acknowledge codes that is also erroneously. Therefore, it is obvious that interrupt and acknowledge wave crossing caused the erroneous situation and disrupts the network work, i.e. interrupt and acknowledge wave uncrossing in time is necessary condition for the correct algorithm and network operation. Earlier was proved that interrupt and acknowledge waves will not cross in time if the parameters value t_G and t_H will be more than full time of interrupt and acknowledge waves propagation. Hence, on the input parameters impose following restrictions:

- Interrupt code processing time t_H should be more than maximally possible code propagation time from the source node to the farthest distanced node (not necessary handler-node) in the network by the longest path in a worst case;
- Minimal time interval t_G between source node receiving code and next interrupt sending should be more than maximally possible code propagation time from the handler node to the farthest distanced node

(not necessary source node) in the network by the longest path in a worst case.

- So, the t_H and t_G are system parameters required for crossing wave protection in networks without errors.

B. Looping problem solving

For protection from such type of errors, specifically interrupt and acknowledge wave crossing, we offer to do restriction to minimally possible time interval $T_{ISRchange}$ between every ISR's bit changes of every router (which correspond to the first part label value changing). I.e. the ISR's bit value (first part label value) could not be changed earlier than time $T_{ISRchange}$ has elapsed. This guarantee that if a network router receives the unexpected interrupt or acknowledge code that it will be not propagate further and will not cause the looping problem. The time $T_{ISRchange}$ is a system parameter, which should be chosen more than maximally possible propagation time of interrupt and acknowledge codes waves.

Add to the interrupt and acknowledge propagation algorithm the following changes:

- In the Step 1 add the initialization of $T_{lc} = \infty$, which will hold a time value of last first part label value changes;
- Step 2.2.5 change to: "if the condition $T>T_{lc}+T_{ISRchange}$ is true, than for every node pair $(v_i(x, y), v_j(x, y))$ from F_t set in node v_j change the label to $(1, T)$ and $T_{lc} = T$. If the condition is not true the label is not changed".
- Step 4.2.5 change as: "If the condition $T>T_{lc}+T_{ISRchange}$ is true, than for every node pair $(v_i(x, y), v_j(x, y))$ from the F_t set for every node v_j change the label to $(0, T)$, $T_{lc} = T$. If the condition is not true there is nothing to change".

Prove that using of addition parameter $T_{ISRchange}$ protects network from a looping. Consider again the example with loping situation from Fig.3. Assume that handle time t_H chosen correct and $t_H = 30$. However, in times point $T = 7$ in edge (v_{15}, v_{13}) as a result of error acknowledge code has appeared. Then in point time $T = 8$ by this acknowledge code propagation the first part label value was changed in node v_{13} because in Step 4.2.5 the condition is true. After the algorithm was changed, specifically the check $T>T_{lc}+T_{ISRchange}$ is added, this condition will not be true and the label will not be change. So in times point $T = 11$, interrupt code which was come repeatedly by longer length path, will be not propagated further like in example with error illustration. That is right because the reason causing looping is eliminated. Therefore, the adding of $T_{ISRchange}$ parameters protects the network to interrupt and acknowledge waves looping that is required to prove.

In the network without cycles, this timeout will be not used, and in case of unexpected code appearance during timeout, this code will be not propagated further.

IV. Conclusion

SpaceWire is standard for onboard communication networks. In the paper analytical model for distributed interrupt mechanism is considered. The algorithm correctness work and the algorithms properties are proved. Based on this model in other papers the distributed interrupt time characteristics are derived. The model allow to derive equations which fit to any networks topology with known parameters D and P_{Len}, they define the number of edges in the shortest an longest path between two most distanced nodes and also T_{bit}, N_C and T_{wt} which are network time parameters. So, this paper in couple with the other papers [4-7] gives to users all necessary information for Distributed Interrupt mechanism using in SpaceWire onboard networks.

References

[1] ECSS-E-50-12C. SpaceWire - Links, nodes, routers and networks. - European Cooperation for Space Standardization (ECSS), 31 July 2008

[2] S, Parkes D 1.1 Consolidated set of Requirements for SpaceWire-RT, SpaceWire-RT Consortium

[3] S, Parkes, "D2.1 SpaceWire-RT Outline Specification," SpaceWire-RT Consortium, September 2012.

[4] Yuriy Sheynin, Sergey Gorbatchev, Liudmila Onishchenko, "Real-Time Signalling in SpaceWire Networks". *International SpaceWire Conference*, Dundee 2007. Conference Proceedings. ISBN: 978-0-9557196-0-8, 4pg.

[5] Liudmila Onishchenko, Artur Eganyan, Irina Lavrovskaya, "Distributed interrupts mechanism verification and investigation by modeling on SDL and SystemC". *International SpaceWire Conference*, Nara 2008. Conference Proceedings. ISBN: 978-0-9557196-1-5

[6] Liudmila Koblyakova, Yuriy Sheynin, Dmitry Raszhivin, "Real-time signalling in networked embedded systems". *International SpaceWire Conference*, St.Petersburg 2010. Conference Proceedings. ISBN: 978-0-9557196-2-2, p. 385-388

[7] Sergey Gorbachev, Liudmila Koblyakova, Yuriy Sheynin, Alexander Stepanov, Elena Suvorova, Martin Suess, "Distributed Interrupt Signalling for SpaceWire Networks". *Proceedings of the 5th International SpaceWire Conference*. Gothnburg 2013. ISBN: 978-0-9557196-4-6.

[8] Reinhard Diestel, *Graph Theory*, New York: Springer-Verlag Heidelberg, 2005.

[9] James A. Anderson, *Discrete Mathematics with Combinatorics*. Moscow: Williams, 2004