Towards a reliability analysis of the design space for the communication subsystem of FT4FTT

David Gessner, Julián Proenza, and Manuel Barranco
Departament de Matemàtiques i Informàtica
Universitat de les Illes Balears, Spain
davidges@gmail.com, {julian.proenza, manuel.barranco}@uib.es

Paulo Portugal
INESC-TEC, Faculty of Engineering
University of Porto, Portugal
pportugal@fe.up.pt

***Abstract*—Fault Tolerance for Flexible Time-Triggered Ethernet-based systems (FT4FTT) is a project to devise an architecture for distributed embedded systems that provides both flexibility to changing real-time requirements and high reliability through fault tolerance. One of the key parts of such an architecture is the communication subsystem. When designing such a subsystem many decisions have to be made. To understand how such decisions impact the reliability of the final design, in this paper we present a framework to evaluate the reliability of a large number of potential designs. The approach is based on storing a finite subset of the design space for the communication subsystem of FT4FTT in an undirected graph and then generating a continuous-time Markov chain from the graph to evaluate the reliability of each design belonging to the subset.**

I. Introduction

When designing a system to solve a problem there are usually many different design variables, i.e., decisions that need to be made. Depending on the value assigned to each of them, we may come up with one design or another. If there are n design variables, the set of possible designs can be thought of as an n-dimensional space, where each dimension corresponds to one particular variable. The values for a dimension are the possible values for the corresponding variable and each point of the space is a possible design that solves the problem. This space is known as the *design space* for the given problem [1].

In this paper we focus on the design space that describes the main design alternatives for the communication subsystem of FT4FTT, a project whose goal is to provide high reliability through fault tolerance for real-time distributed embedded systems that must be able to adapt to changing real-time requirements. The main criteria we use to compare the designs is the reliability of each one of them. In particular, we are interested in how the Flexible Time-Triggered Replicated Star (FTTRS) [2], which is the current design for the communication subsystem of FT4FTT, compares with other possible designs. For this purpose, we are currently developing a framework to evaluate the reliability of a finite subset of the FT4FTT design space.

This paper presents the basics of this approach under the simplifying assumptions that only permanent faults may occur and that there is perfect fault coverage, i.e., the system always successfully isolates faulty components and, if there are sufficient components left for a proper operation, continues providing its service correctly. The main idea is to use a single mathematical model to store all points of a finite subset of the design space and then to use this model to generate a Markov chain to evaluate the reliability of each point of the subset. Specifically, we model a whole design space subset with an undirected graph [3]. From this graph we then generate subgraphs that are mapped onto states of a Markov chain, which represent faulty or non-faulty states, as determined by a boolean-valued function expressed in terms of graph properties. Using this Markov chain we can evaluate the reliability of all points of the design space subset and aspire to find the optimal design in terms of reliability of the subset.

The remainder of the paper proceeds as follows. Section II introduces the design space for the communication subsystem of FT4FTT. Section III describes how we model a finite subset of the design space by means of graphs. Section IV introduces two functions to check if a graph represents a valid input to our Markov chain generating algorithm and a non-faulty design, respectively. Section V explains how we generate a Markov chain and use it to evaluate the reliability of the points of a design space subset. Finally, Section VI concludes the paper.

II. The FT4FTT communication subsystem design space

FT4FTT is based on a switched ethernet implementation of the Flexible Time-Triggered (FTT) communication paradigm [4], a master/slave communication paradigm that allows a real-time distributed embedded system to adapt to changing real-time requirements. The potential designs for the communication subsystem of FT4FTT will therefore be master/slave and based on the use of ethernet switches. Thus, all designs have the following types of components as building blocks: ethernet switches, ethernet links, FTT masters, and FTT slaves. Moreover, we can distinguish between FTT masters embedded within a switch, like in the HaRTES implementation of FTT [5], and external masters connected to one or more switches by means of ethernet links, like in the FTT-SE implementation [6]. Different designs for the communication subsystem of FT4FTT may then differ in the number of components of each type and how these components are interconnected, i.e., in their topology. This leads us to the following discrete dimensions for the design space: (D1) number of slaves; (D2) number of switches; (D3) number of masters; and (D4) the topology of the network, which not only tells us how the components are interconnected by ethernet links, but also which masters are embedded within switches. Note that the first dimension, the number of slaves, is the only one not related to the redundancy of the communication subsystem of FT4FTT. In fact, its values will generally be determined

©2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

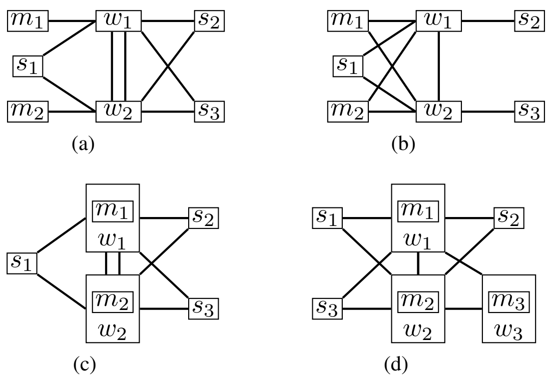

Fig. 1. Four example designs of the design space.

by application requirements other than reliability. We may therefore not have complete freedom in setting a value for it. Indeed, we will generally lower bound it to a value $N > 0$ to take into account that a given application requires N slaves. Furthermore, the second and third dimensions have a lower bound of one, i.e., a design must have at least one switch and one master. Moreover, note that the last dimension, the topology, is not orthogonal to the others, i.e., its values are conditioned to a certain extent by the other dimensions. Finally, since FT4FTT is based on master/slave communication, the topology of a design must be such that it allows the slaves to communicate with each other and the master.

Regarding the cardinality of the dimensions, they are all infinite as in principle there is no upper bound on the number of switches, masters, and slaves — and thus topologies — that a design may have. Therefore, the design space for FT4FTT is also infinite and it is thus impossible to evaluate the reliability of all its points. Nevertheless, we can upper bound each dimension, i.e., limit the maximum number of switches, masters, slaves, and topologies we consider, and evaluate the reliability of the points of the corresponding finite subset of the FT4FTT design space. However, even if the dimensions are upper bounded, the number of points can be very high as the cardinality of this subset is given by the Cartesian product of each dimension, meaning that it increases geometrically with the dimension. Moreover, analyzing the reliability of each design point is NP-hard [7]. Thus, it is necessary to efficiently store and process the points of a design space subset.

Figure 1 shows a few example designs, i.e., points of the FT4FTT design space. In each of the subfigures arcs represent ethernet links and rectangles represent components. The components are labeled by their type: s_i indicates an FTT slave, w_i an ethernet switch, and m_i an FTT master. For instance, (a) is a design with two switches interconnected by two ethernet links, with slaves that are connected to both switches, and with two external masters, each connected to a separate switch; and (d) is a design with three switches, each with its own embedded FTT master, that are interconnected forming a ring.

III. The modeling approach

Let S be the set of all slaves, W the set of all switches, M the set of all masters, and L the set of all ethernet links. In our approach, we model each point d_i of the FT4FTT design space by an undirected graph $G_i = (V_i, E_i)$, such that the set of vertices $V_i \subseteq S \cup W \cup M \cup L$ and the set of edges $E_i = \{\{u, v\} \mid u, v \in V_i$ and u is attached to v in the design $d_i\}$. We write $V(G_i)$ and $E(G_i)$ to indicate the vertex set V_i and edge set E_i of a graph G_i, respectively. Figure 2 shows two example designs with their corresponding graphs.

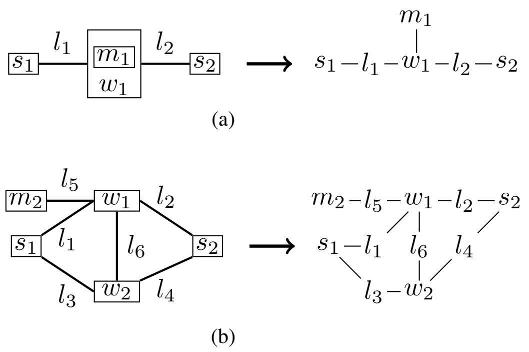

Fig. 2. Two example designs with their graphs.

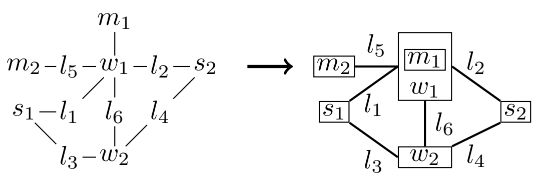

Fig. 3. Union of the graphs from Figure 2.

The *union* of two graphs $G_i = (V_i, E_i)$ and $G_j = (V_j, E_j)$ is a graph $G_k = (V_i \cup V_j, E_i \cup E_j)$ and we write $G_k = G_i \cup G_j$. Such a union graph G_k is also the graph of a point d_k of the design space. For example, Figure 3 shows the union graph of the two graphs from Figure 2 and the design corresponding to the union.

A graph $G_j = (V_j, E_j)$ is a *subgraph* of a graph $G_i = (V_i, E_i)$ if $V_j \subseteq V_i$ and $E_j \subseteq E_i$ such that $E_j = \{\{u, v\} \mid u, v \in V_j\}$. We write $G_j \subseteq G_i$. Moreover, $G_j \subseteq G_i$ is an *induced subgraph* of G_i if $V_j \subseteq V_i$ and $E_j = \{\{u, v\} \mid u, v \in V_j$ and $\{u, v\} \in E(G_i)\}$. In other words, G_j is an induced subgraph of G_i if it can be obtained by deleting vertices from G_i, but preserving the edges between all undeleted vertices. We write $G_j = G_i[V_j]$ and say that G_j is the subgraph induced by the vertex set V_j. Note that if we have a union graph $G_k = G_i \cup G_j$, then $G_i = G_k[V(G_i)]$ and $G_j = G_k[V(G_j)]$. For instance, the two graphs from Figure 2 are induced subgraphs of the union graph from Figure 3.

With these notions, we can store all points of a design space in a single graph. Specifically, let D be the set of points of the design space. The union graph of the whole design space is the graph $G_D = \cup_{n \in D} G_n$. Now, if $\mathrm{Subi}(G_D) = \{G_k \mid V(G_k) \subseteq V(G_D)$ and $G_k = G_D[V(G_k)]\}$ is the set of all induced subgraphs of G_D, then $G_i \in \mathrm{Subi}(G_D)$ for all $d_i \in D$. In other words, from G_D we can generate the graphs of all points of the design space D by generating all induced subgraphs of G_D. However, note that this also generates additional graphs that do not correspond to a point of the design space D, i.e., $\{G_i \mid d_i \in D\} \subset \mathrm{Subi}(G_D)$. For instance, the null graph $(\{\}, \{\})$ and the trivial graph $(\{w_1\}, \{\})$ are in $\mathrm{Subi}(G_D)$, but they do not correspond to acceptable designs for FT4FTT. We therefore say that $\mathrm{Subi}(G_D)$ contains the graphs of an *extended design space* D', such that $D \subset D'$.

IV. The validity and correctness function

As explained above, the extended design space D', when compared with the FT4FTT design space D, contains additional non-acceptable designs. To distinguish between graphs that model acceptable designs and graphs that do not, we use two boolean-valued functions: a validity function and a correctness function. The first tells us if a graph models a *valid* design, i.e., it tells us whether a graph models what we consider possible ways of connecting slaves, switches, masters, and ethernet links. The second function tells us whether a graph models a *correct*, i.e., non-faulty system. An acceptable design should be valid and correct. Next we define these functions.

The *validity function* for the FT4FTT communication subsystem is a function $v\colon \gamma \to \{0,1\}$, where γ is the set of all undirected graphs that have a vertex set $V \subseteq S \cup W \cup M \cup L$. Specifically, we define it as

$$v(G) = \begin{cases} 1, \text{if } G \text{ is valid} \\ 0, \text{otherwise,} \end{cases}$$

where *valid* means that G satisfies the following conditions:

- two vertices $u_1, u_2 \in S \cup W, u_1 \neq u_2$, cannot be adjacent, i.e., two slaves or switches cannot be attached directly to each other (they must be connected by an ethernet link);
- similarly, two vertices $u_1, u_2 \in S \cup M, u_1 \neq u_2$, cannot be adjacent;
- a vertex $l \in L$ can only be adjacent to a vertex $u \in S \cup W \cup M$, i.e., ethernet links can only be attached to slaves, switches, and masters, but not to other links;
- any vertex $l \in L$ has at most degree 2, i.e., an ethernet link cannot interconnect more than two components;
- if a vertex $l \in L$ is adjacent to vertices u_1 and u_2, then $u_1 \in W$ or $u_2 \in W$, i.e., if a link interconnects two components, then at least one of them must be a switch.

Note that in a valid graph a vertex $m \in M$ and a vertex $w \in W$ may be adjacent. If they are, this models the master m being embedded in the switch w. Moreover, note that if $v(G) = 1$, then $v(G_i) = 1$ for each induced subgraph $G_i \in \mathrm{Subi}(G)$. This is so because our validity function only takes into account the adjacency among vertices, and subgraphs of a graph G do not make any vertices adjacent that were not already adjacent in G. We say that validity is *inherited* by the subgraphs.

Regarding the *correctness function*, it is a function $c\colon \gamma \to \{0,1\}$ defined as

$$c(G) = \begin{cases} 1, \text{if } G \text{ is correct} \\ 0, \text{otherwise,} \end{cases}$$

where *correct* means that G satisfies a certain graph property. For instance, since FT4FTT is based on master/slave communication, the graph property could be that G has a connected component, i.e., connected subgraph, that spans a set of slaves S_c and at least one master. By saying this we are saying that in the modeled system all slaves in S_c can communicate with each other and with at least one master. This graph property

```
1:  procedure MCGenerate($G, v, c$)
2:      if $v(G) = 0$ or $c(G) = 0$ then
3:          return Error
4:      end if
5:      $MC.addState(G)$                              ▷ root state
6:      $MC.addState(FAIL)$                           ▷ failure state
7:      Explore($G, c$)
8:      return MC
9:      function Explore($H, c$)
10:         for all vertex $u \in H$ do
11:             $H' \leftarrow deleteVertex(H, u)$
12:             if $c(H') = 1$ then
13:                 $MC.addState(H')$
14:                 $MC.addRate(H, H', u)$
15:                 Explore($H', c$)
16:             else
17:                 $MC.addRate(H, FAIL, failRate(u))$
18:             end if
19:         end for
20:     end function
21: end procedure
```

Fig. 4. Algorithm to generate a CTMC from a design space union graph.

allows us to evaluate the reliability of a design where S_c is the set of slaves that are critical for the operation of the system.

V. Evaluating the reliability of a design space subset

To evaluate the reliability of all points of a subset of the FT4FTT design space we assume that all components of a design have known constant failure rates and fail independently of each other, and no more than one component fails at each instant. These are common assumptions for electronic based components [8]. Under these assumptions we can use a Continuous-Time Markov Chain (CTMC) for the reliability evaluation [9]. Specifically, we will generate a CTMC from the union graph corresponding to the design space subset we want to evaluate. Since from the union graph we can obtain the graphs for all points of the subset, we can generate the CTMC such that it has a state for each point of the design space subset. This will allow us to evaluate the reliability of each point, as explained at the end of this section.

The procedure to generate the CTMC can be implemented by a recursive algorithm. The pseudocode for this procedure is shown in Figure 4. The initial parameters of the procedure are a union graph G for the design space subset for which we want to evaluate the reliability, a validity function, and a correctness function (line 1). The procedure begins by checking whether G is valid and correct (line 2) and exits with an error if it is not (line 3). Otherwise it initializes the Markov chain, MC, with two states (lines 5-6). The first state corresponds to the union graph G. We call it the *root state*. The second state corresponds to a faulty system and we call it the *failure state*. Subsequent states of the CTMC are obtained by calling recursively the function Explore (line 7). This function has as parameters a graph H and a correctness function (line 9). When the function is called, a state has already been added to the CTMC for the graph H. We call this state the *parent state*. Note that the first time the function is called the parent state is the root

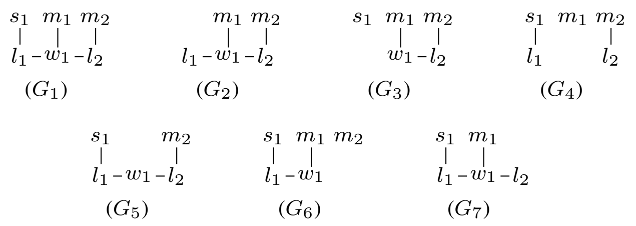

Fig. 5. Graphs corresponding to a subset of the states of the example CTMC.

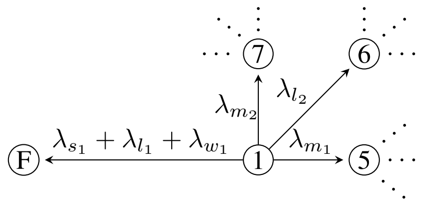

Fig. 6. Partial CTMC built after one recursion.

state. The function starts by exploring all induced subgraphs of H' that can be obtained from H by deleting single vertices from it (lines 10-11). If the resulting subgraph H' is correct, a state corresponding to H' is added to the CTMC (line 13) and this state is made accessible through a transition from the parent state (line 14) – validity is not checked as we know that it is inherited from the parent state's graph. The rate for this transition gets assigned the failure rate of the vertex that has been deleted (line 14). Next, the function is invoked again for the newly added state (line 15), which becomes the parent state in the recursive call. If the subgraph H' was not correct (line 12), then no state is added for the subgraph and instead a transition is added from the parent state to the failure state (line 17). Note that after reaching the failure state, the function stops exploring new states. This allows to reduce both the number of states of the CTMC and the execution time of the procedure.

To illustrate the procedure, consider the subset of the FT4FTT design space that is given by limiting the number of slaves, switches, external masters, and embedded masters to one. The union graph G_1 for this design space subset is shown in Figure 5. We will assume that the failure of the single slave is not tolerated. When the procedure is invoked, the state corresponding to the union graph G_1 becomes the root state of the CTMC. We have labeled this state as 1 in Figure 6. Through single vertex deletions we get the graphs G_2 through G_7 in Figure 5. Of these graphs, G_2, G_3, and G_4 are considered faulty because they do not allow the slave to communicate with a master. They therefore fail the correctness function and, thus, transitions are added from the root state to the failure state, as shown in Figure 6, with the corresponding failure rate λ_u, where u is s_1, l_1, and w_1 for G_2, G_3, and G_4 respectively. Graphs G_5, G_6, and G_7, on the other hand, allow communication between the slave and a master and are therefore correct. The EXPLORE function of Algorithm 4 therefore adds new states for them to the CTMC, as well as appropriate transitions from the root state. These new states are labeled 5, 6, and 7 in Figure 6 for G_5, G_6, and G_7 respectively. Successive recursive calls of EXPLORE complete the CTMC.

Once the CTMC is built for a union graph G, it can be used to compute the reliability of each point of the design space subset corresponding to G. Let d_i be a point of the design space subset and G_i its corresponding graph. The generated CTMC will contain a state i corresponding to G_i and the reliability of the design point d_i is simply obtained by calculating the probability of not reaching the failure state from state i. For example, the probability of not reaching the failure state from state 5 in Figure 6 would give us the reliability of the design corresponding to graph G_5 in Figure 5.

VI. Conclusions and Future Work

In this paper we presented a framework based on graphs and Markov chains to evaluate the reliability of a large number of designs for a communication subsystem for FT4FTT. This is a first step towards determining which design decisions have the greatest impact on the final reliability in FT4FTT. Future work includes further reducing the state space by merging non-faulty states that can be considered equivalent (e.g., in Figure 6, states 6 and 7 could be merged); taking into account imperfect fault coverage; using design of experiments techniques to find out whether there are any interaction effects between the different dimensions of the design space (e.g., maybe for a low number of switches a certain network topology is the most reliable, but with more switches a different one becomes more reliable); automatically generating design space union graphs from design space specifications based on the validity and correctness function; and adapting our approach to consider transient faults instead of permanent faults.

Acknowledgements

This work was supported by project DPI2011-22992 and grant BES-2012-052040 (Spanish *Ministerio de economía y competividad*), and by FEDER funding.

References

[1] J. Arora, *Introduction to Optimum Design*, 3rd ed. Academic Press, 2011.

[2] D. Gessner, J. Proenza, M. Barranco, and L. Almeida, "Towards a Flexible Time-Triggered Replicated Star for Ethernet," in *18th IEEE Conf. on Emerging Technologies & Factory Automation (ETFA)*, Cagliari, Italy, Sep. 2013.

[3] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*, 3rd ed. Springer, 2005.

[4] P. Pedreiras and L. Almeida, "The Flexible Time-Triggered (FTT) paradigm: an approach to QoS management in distributed real-time systems," in *Proc. Int. Parallel and Distributed Processing Symposium*. IEEE Comput. Soc, 2001.

[5] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida, "Designing a Costumized Ethernet Switch for Safe Hard Real-Time Communication," in *Proc. 7th IEEE Int. Workshop on Factory Communication Systems (WFCS)*. Dresden, Germany: IEEE, 2008.

[6] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over COTS Ethernet switches," in *Proc. 6th IEEE Int. Workshop on Factory Communication Systems (WFCS)*. Torino, Italy: IEEE, 2006, pp. 295–302.

[7] M. Ball, "Computational complexity of network reliability analysis: An overview," *Reliability, IEEE Transactions on*, vol. 35, no. 3, pp. 230–239, Aug 1986.

[8] *MIL-HDBK 217F : Reliability Prediction of Electronic Equipment*. U.S. Dept. of Defense, 1995.

[9] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. Chichester, UK: John Wiley and Sons Ltd., 2002.