Markovian Reliability Analysis of Non-Coherent Non-repairable Fault-Tolerant Interconnection Networks

Cristian-Iulian Rincu[1], Juan A. Carrasco[2] and Victor Suñé[2]

[1]Military Technical Academy
G.Cosbuc 81–83, Bucharest, 75275 Romania
email: r_iulian@mta.ro
[2]Electronics Engineering Department, Universitat Politècnica de Catalunya
Diagonal 647, plta. 9, 08028 Barcelona, Spain
email: carrasco,sunyc@ccl.upc.es

Abstract

This paper targets the reliability analysis of a fault-tolerant MIN network (the ASEN-Max network) in a non-repairable context. We show that, under certain conditions concerning how the operation of the network is controlled, the network is non-coherent, preventing the use of combinatorial techniques. Finally, we examine the limitations of continuous-time Markov chain (CTMC) techniques.

1. Introduction

The design and analysis of Multistage Interconnection Networks (MINs) is a central problem in computer and communication systems [6]. Many such applications have stringent reliability requirements, making it important that the interconnection network has a high reliability. Fault-tolerance can be used to achieve such a high reliability. Fault-tolerance can be achieved by using hardware redundancy and time redundancy. Using hardware redundancy, extra elements are added to the network to provide alternative paths between inputs and outputs and tolerate faults. Using time redundancy, requests which cannot be served from the originating input because there are not paths available from that input are redirected to other inputs. Time redundancy increases the reliability of the MIN but degrades its performance [8].

A number of hardware redundancy schemes have been proposed to increase the reliability of Multistage Interconnection Networks: network replication, addition of extra stages, and addition of extra links. The addition of links tends to have a more profound impact on the reliability than the addition of extra stages [3]. The ASEN-Max network [7] is a fault-tolerant MIN with moderate hardware redundancy which can tolerate any single fault.

2. Network description and fault tolerance

An ASEN-Max network with $2^n = N$ inputs has N *2x1* multiplexers, N *1x2* demultiplexers and $log_2 N - 1$ stages of $N/2$ switches. Figure 1 depicts a 16-input ASEN-Max network. The switches in the last stage are 2x2 switches; switches in the remaining stages are 3x3 switches. In each stage except the last one, switches can be grouped into conjugate pairs, switches in the same conjugated pair being connected to the same switches of the next stage. Conjugate pairs can be grouped into conjugate subsets, where a conjugate subset includes all switches in a particular stage that lead to the same subset of outputs. Switches in the same conjugate subset that do not belong to the same conjugate pair communicate through auxiliary links and build up loops. Loops are formed in such a way that all switches in a loop have their conjugate switches in another loop. Such pairs of loops are called conjugate loops. The network can tolerate the failure of all switches in a loop. As illustration, in the 16-input ASEN-Max network, switches $S_{0,0}, S_{0,1}, \ldots, S_{0,7}$ form a conjugate subset; within that subset, switches $S_{0,0}, S_{0,4}$ are a conjugate pair; and switches $S_{0,0}, S_{0,1}, S_{0,2}, S_{0,3}$ form a loop. That loop is a conjugate loop of the loop built by switches $S_{0,4}, S_{0,5}, S_{0,6}, S_{0,7}$.

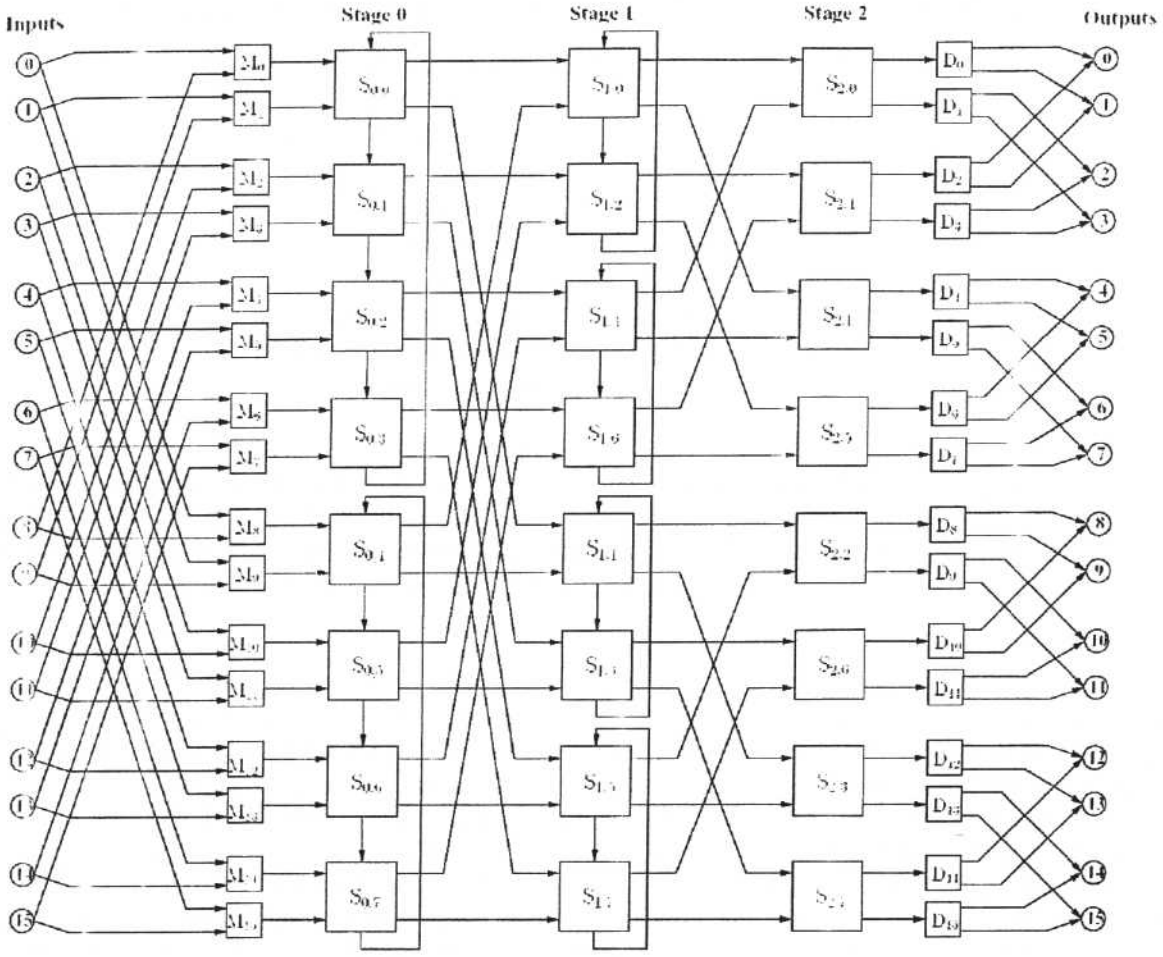

Fig.1. 16-input ASEN-Max network.

An important issue in fault-tolerant MINs is the manner in which routing is performed in the presence of faults. Routing can be done using either backtracking or non-backtracking algorithms. Backtracking algorithms are expensive and difficult to implement because they need bi-directional links and, using packet switching techniques, reverse queues. Non-backtracking algorithms are easier to implement. In the presence of faults, non-backtracking algorithms may base routing decisions on local or global information about the failed state of the elements of the network. When routing decisions are based on local information, a non-backtracking algorithm may not find an existing path from some input to some output.

A non-backtracking routing algorithm based on local fault information for the ASEN-Max network has been proposed in [7]. Specifically, the routing algorithm assumes that: 1) each input knows whether each multiplexer to which it is connected or the switch to which the multiplexer is connected is failed, 2) each switch of any stage except the last two ones knows whether each switch to which it is connected is failed, and 3) each switch of the stage previous to the last one knows whether each switch to which it is connected or some demultiplexer to which that switch is connected is failed. That local knowledge assumes a fault model in which links do not fail. A more general fault model including link faults could however be easily accommodated by collapsing link faults into the multiplexer/demultiplexer or switch to which the link is connected. The routing algorithm uses the binary codification $d_0\, d_1 \ldots d_{n-1}$ of the address of the output, where d_0 is the most significant bit and d_{n-1} is the least significant bit. In the description of the algorithm, we will make reference to the primary link and secondary link emanating of an input: primary links are drawn above secondary links in Figure 1. We will also make reference to the 0-link and the 1-link emanating of a switch or a demultiplexer: 0-links are drawn above 1-links in Figure 1. The network is assumed to operate by processing synchronously pendent requests from the inputs in

rounds using circuit-switching techniques. Conflicts may arise in multiplexers and switches if requests which should be routed through the same output link arrive simultaneously to the switch. It is assumed that some mechanism is implemented to solve those conflicts. Requests loosing the conflict are dropped. To simplify the description of the routing algorithm, we will not make reference explicitly to that conflict solution mechanism. The routing algorithm is as follows.

- For each input: if the multiplexer to which the primary link is connected is not failed and the switch to which that multiplexer is connected is not failed submit the request through that link: otherwise, if the multiplexer to which the secondary link is connected is not failed and the switching element to which that multiplexer is connected is not failed submit the request through the secondary link; otherwise, drop the request.
- For each switch in stage i, $i < n-3$: use bit i of the address of the output to choose between submitting the request through the 0-link (if bit i is 0) or through the 1-link (if bit i is 1); if the chosen 0-link or 1-link is not busy and the switch connected to it is not failed submit the request through that link; otherwise, if the auxiliary link is not busy and the switch connected to it is not failed submit the request through the auxiliary link; otherwise, drop the request.
- For each switch in stage $n-3$: use bit $n-3$ of the address of the output to choose between submitting the request through the 0-link (if bit $n-3$ is 0) or the 1-link (if bit $n-3$ is 1): if the chosen 0-link or 1-link is not busy, the switch connected to it is not failed and no demultiplexer connected to that switch is failed submit the request through that link; otherwise, if the auxiliary link is not busy and the switch connected to it is not failed submit the request through the auxiliary link; otherwise, drop the request.
- For each switch in stage $n-2$: use bit $n-2$ of the address of the output to choose between submitting the request through the 0-link (if bit $n-2$ is 0) or through the 1-link (if bit $n-2$ is 1); if the chosen 0-link or 1-link is not busy, submit the request through that link; otherwise, drop the request.
- For each demultiplexer: use bit $n-1$ of the address of the output to choose between submitting the request through the 0-link (if bit $n-1$ is 0) or through the 1-link (if bit $n-1$ is 1) and submit the request through that link.

3. *Non-coherence of the ASEN-Max network*

The presence of several requests may prevent the routing of a request which would be served if the request were isolated. Assume, for instance, that the only failed components are switches $S_{2,0}$ and $S_{1,2}$ and that there is a pending request from input 0 to output 0. If the request pattern is such that, besides that request, there are indefinitely requests from input 2 to output 0, those requests will be indefinitely routed through switch $S_{1,0}$ and when the request from input 0 to output 0 arrives to switch $S_{1,0}$, the request will be indefinitely dropped. We will, therefore, assume that when a request is dropped many times, other pending requests are held and the network tries to serve that request in isolation. Therefore, the network will be considered up if isolated requests can be routed from any input to any output. That assumption is guaranteed to be pessimistic and makes feasible a (pessimistic) request-pattern independent evaluation of the reliability of the network. We will prove the non-coherence of the ASEN-Max network under that criterium.

A system is coherent if, as components fail, it is impossible for the system to go from a down to an up state [2]. Therefore, to prove the non-coherence of the network it is enough to find a subset of components C_1 and a subset of components $C_2 \supset C_1$ such that when the set of failed components is C_1 some isolated request from some input to some output is not served and when the set of failed components is C_2 an isolated request from any input to any output is served. The existence of such a pair of subsets of elements will be illustrated by considering a 16-input ASEN-Max network (see Figure 2). When the set of failed components is $C_1 = \{S_{1,2}, S_{2,0}\}$, isolated requests from input 0 to outputs 0, 1, 2 and will not be served because they will be routed through multiplexer M_0, switch $S_{0,0}$ and switch $S_{1,0}$ and will be dropped in $S_{1,0}$. However, when the subset of failed components is $C_2 = \{S_{1,0}, S_{1,2}, S_{2,0}\}$, those requests will be routed through multiplexer M_0 and switches $S_{0,0}$, $S_{0,1}$, $S_{0,2}$, $S_{1,4}$, $S_{1,6}$, and $S_{2,4}$ and will reach

their destination. It can easily be checked that requests for other input-output pairs are served when the set of failed components is C_2. Given the regularity of the ASEN-Max networks subsets C_1 and $C_2 \supset C_1$, similar to the previously discussed subsets, such that the system is down when the set of failed components is C_1 and the system is up when the set of failed components is C_2 exist for ASEN-Max networks of at least 8 inputs.

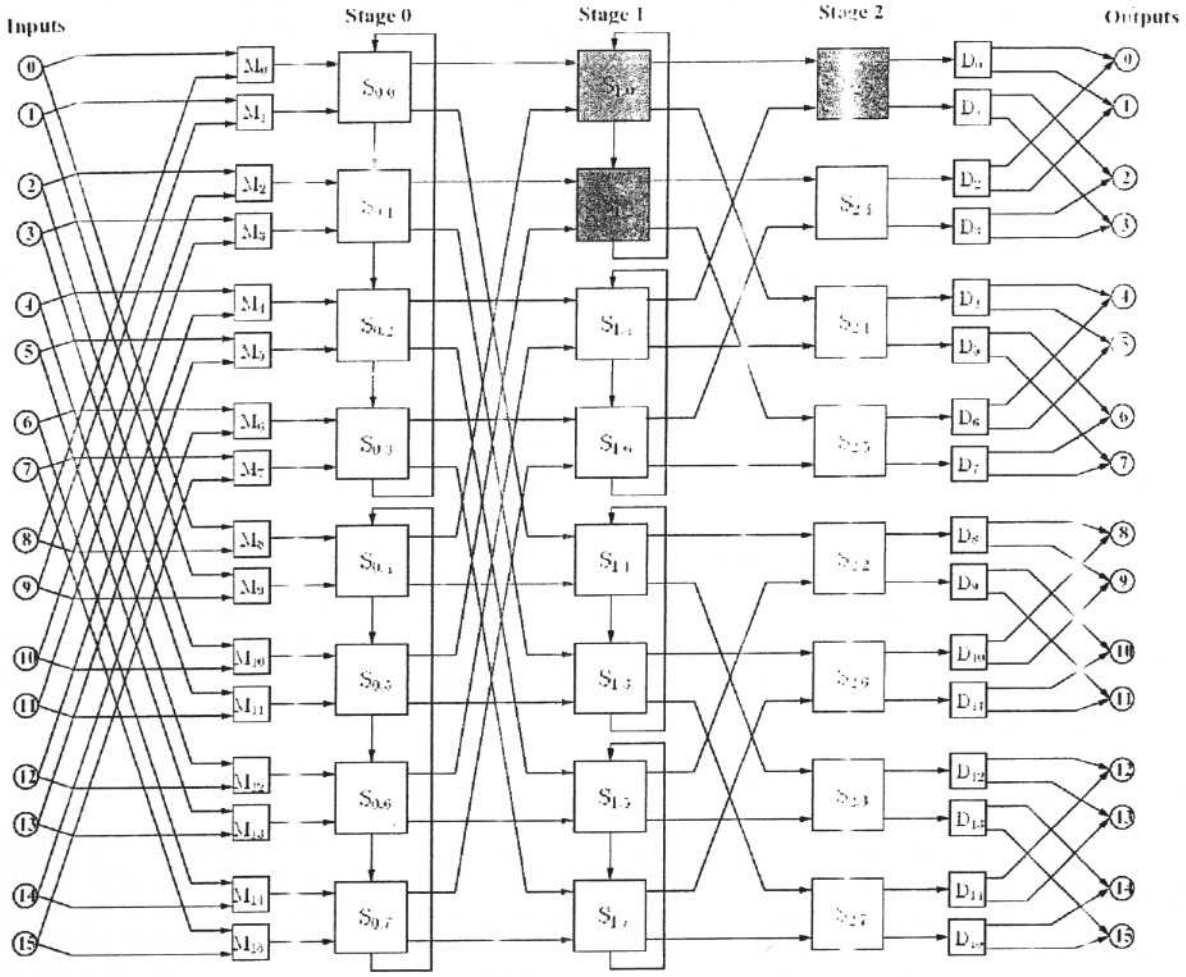

Fig.2. 16-input ASEN-Max network indicating sets of failed components which prove the non coherence of the network.

4. Reliability analysis

A realistic evaluation of the reliability of the ASEN-Max network should take into account coverage faults. If the ASEN-Max network were coherent, existing combinatorial techniques able to cope with coverage faults [1,5] could be used to evaluate the reliability of the ASEN-Max network. However, being the network non-coherent, those techniques cannot be used since they truly compute the probability that the system is up at time t, which does not coincide with the reliability if the system is non-coherent. CTMC models are an alternative under the assumption that the components have constant failure rates. In this section we will explore the limitations of that alternative. We will assume that links do not fail.

Exact CTMC reliability models of the ASEN-Max network grow fast with the size of the network. Using a workstation equipped with 128 MB of memory, the largest ASEN-Max network which can be analyzed using exact CTMC models is an 8-input ASEN-Max network. For that network the exact CTMC model has 45,506 states and 319,553 transitions. Larger ASEN-Max networks can be analyzed using bounding techniques [4]. A lower bound, $\left[ur(t)\right]_{lb}$, for the unreliability $ur(t)$ can be computed using a

lower bounding CTMC model, X_{lb}, having state space $U_K \cup \{f, a\}$, where U_K includes the up states with up to K failed components reachable from the all-components-up state through up states, f is an absorbing state into which X_{lb} enters when the system fails and a is an absorbing state into which X_{lb} enters when the system enters through up states an up state with more than K failed components. We have $[ur(t)]_{lb} = P[X_{lb}(t) = f]$. An upper bound, $[ur(t)]_{ub}$, for $ur(t)$ can be computed using an upper bounding CTMC, X_{ub}, having state space $U_K \cup \{f\}$, where U_K is as for X_{lb} and f is an absorbing state into which X_{ub} enters when either the system fails or the system enters through up states an up state with more than K failed components. We have $[ur(t)]_{ub} = P[X_{ub}(t) = f]$. We will examine how the size of the bounding models and the quality of the bounds depend on K. The experiments will be performed assuming a failure rate $5 \times 10^{-7}\,h^{-1}$ for multiplexers and demultiplexers, a failure rate $10^{-6}\,h^{-1}$ for switches, a coverage 0.99 for multiplexers and demultiplexers and a coverage 0.98 for switches. The quality of the bounds degrades as t increases and we will take a large value for t, specifically, $t = 50{,}000$ h (5,7 years). Table 1 gives the sizes of the bounding models and the CPU times consumed in a 167 MHz, 128 MB UltraSPARC workstation to compute the bounds for a 16-input ASEN-Max network. The models were specified and solved using the METFAC2.1 tool [4]. As solution method we used randomization, which is very efficient for the models under consideration. Table 2 gives the bounds. We can note that K=4 is required to obtain bounds of good quality. That value of K is the largest value for which the bounding models fit into the available memory. The CPU times are very high (more than 4 hours for K=4). Those high values are due to the cost of determining while the CTMC is generated whether transition rates lead to up or down states. Since the model has a large number of components, the number of transitions from a given state is high and, in order to generate the bounding models it is necessary to check a very large number of times whether transitions lead to up or down states. That check is performed making a call to a function implemented using dynamic programming techniques to determine the existence of paths under the non-backtracking protocol from every input to every output.

Table 1. Sizes of the bounding CTMC's and consumed CPU times for a 16-input ASEN-Max network.

K	lower bound			upper bound		
	States	Transitions	CPU time	States	Transitions	CPU time
1	59	169	4.5	58	113	0.0782
2	1,479	5.793	110	1,478	4,373	109
3	23,271	113.269	1,628	23,270	91,477	1,628
4	251,535	1,458.341	16,698	251,534	1.230,077	16,683

Table 2. Bounds obtained for a 16-input ASEN-Max network.

K	lower bound	upper bound
1	7.8218517430E-02	5.9292781590E-01
2	1.3145864420E-01	3.5151441310E-01
3	1.5640051650E-01	2.2460293100E-01
4	1.6422968250E-01	1.8001505740E-01