On the availability of networks

Wenzhu Zou [1], Milena Janic[2], Robert Kooij[1,2], Fernando Kuipers[1]
[1] Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, f.a.kuipers@tudelft.nl
[2] TNO Information and Communication Technology, Delft, The Netherlands, {milena.janic, robert.kooij}@ tno.nl

Abstract

In all networks that provide a service to the consumer, one of the main performance indicators is availability. The consumer, the user of the service, wants to be able to use the service for at least X% of the time. In order to be able to make such guarantees and commit to them in Service Level Agreements, network operators need to know their network availability. In this paper, we discuss how network availability can be algorithmically computed and we derive analytical expressions for several different network topologies. Finally we show how these results can be used to compute availability of real-life networks, such as SURFnet – a high-speed Dutch national network.

Introduction

Businesses today depend more on network communications than a few years ago. As business applications become more critical to a company's success, so does the availability of the underlying network.

Two main factors determine the availability of the underlying network:

A first factor is the availability of the individual network elements. During the lifetime of a network element, it may endure periods in which it is out of service either because of a malfunctioning, maintenance or repair work.

If we denote the mean time to failure by *MTBF* and the mean repair time by *MTTR*, the availability of a network element (or more formally, the probability that the element is working properly) is defined as $A = \frac{MTBF}{MTBF + MTTR}$ and the unavailability as $U = 1 - A$.

A second factor is the topology of the network. Obviously, higher redundancy in the network (e.g. more links connecting network switches) will lead to higher availability, but also to higher investment costs.

For the traditional telephony service often a five nines (99.999%) availability is guaranteed. Network operators that offer other services need to be able to make similar claims. In order to make such claims, an unambiguous definition of network availability is needed. In this paper we propose a definition based upon connectivity of the network.

The remainder of the paper is organized as follows. In the following section we discuss several network availability definitions. The algorithmic computation of network availability is discussed subsequently. Then we determine analytic expressions for a number of simple network topologies. Finally, we apply these results to obtain the availability for the real-world SURFnet network.

Network availability definitions

We consider a network represented as a graph $G(V,E)$ consisting of a set of nodes V and a set of links E. $|V| = N$ denotes the number of nodes while the number of links is $|E| = L$. Nodes represent routers or switches and links represent communication links (e.g., optical fibers). Both links and nodes have a certain availability. In our study we assume that the nodes are always available, i.e. only the links can fail. This assumption is based on the fact that node failures occur much less frequent than link failures, which occur e.g. when fibers are unintentionally broken by means of shovels. In addition we assume that failures occur independently.

Path availability

Let us first examine network availability in its strictest form. Assume that each user has precisely one path over which (s)he can communicate with another user. If this path fails the communication is precluded. Since a path consists of a serial concatenation of nodes and links, its availability is simply computed as the product of the availabilities of the nodes and links that constitute that path. Network availability can then be derived as the minimum path availability over all node pairs. Since all paths are known (or can be computed via a simple shortest paths algorithm), the network availability is easily computed.

Let p_{ij} denote the availability probability of link (i,j), which connects nodes i and j. To gain the highest possible network availability (defined in its strictest form), the minimum path availability over all node pairs should be maximized. If we assign the weight $-log\ (p_{ij})$ to each link $(i,j) \in E$, then the same goal is achieved by using the shortest paths (given the new weights) between the source-destination (s-d) pairs. The shortest path with the highest weight has the lowest path availability and determines the optimal network availability that can be gained.

Unloaded network availability

The other extreme case is where communication can take place over all possible paths. This closely resembles routing in the Internet where, in case a failure takes place, routing protocols automatically reconfigure the routing tables to direct traffic over alternative working paths towards their destinations. Network unavailability is in this case determined by the probability that between a particular source-destination pair, no path is available (i.e., the network is disconnected).

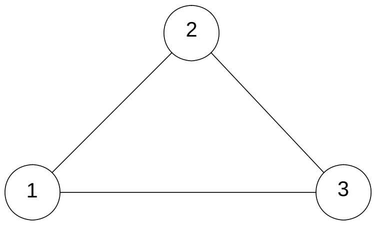
The maximum probability over all s-d pairs determines the network unavailability and hence also the network availability.

Loaded network availability

The computation of network availability in the previous subsection is actually too optimistic in practice, since it assumes that when paths fail the alternate paths have enough resources available to handle the traffic of the failed path(s). Since resources are limited, this may not always be the case. If we consider the example in Figure 1, half of the subgraphs are disconnected - let us call this unloaded network unavailability. However, the three subgraphs (l_1,l_2), (l_2,l_3) and (l_1,l_3) may also not be able to take over the traffic of the failed link - we refer to such case as loaded network unavailability. We will not consider loaded network availability in this paper.

Computation of network availability

When looking at a network, one could say that links are either available (*a*) or unavailable (*u*). By pruning the unavailable links from the graph, we are left with a subgraph which may either be connected or disconnected. In the latter case, for at least one s-d pair, the network is unavailable. By computing the connectivity of all possible subgraphs, one can precisely determine the network (un)availability. We will illustrate this for the three-node network in Figure 1.



(1,2)	(1,3)	(2,3)	A/U
a	a	a	A
u	a	a	A
a	u	a	A
a	a	u	A
u	u	a	U
u	a	u	U
a	u	u	U
u	u	u	U

Figure 1: Example of a three-node network. Links l_1(1,2), l_2 (1,3), and l_3 (2,3) are available (a) or not (u) and the network is available (A) or not (U)

Given the individual link availabilities, we can compute for each subgraph the probability of occurrence and consequently also the network availability. For instance, assume that each link has the same availability *p*. The network availability *NA* is then computed as $NA = p^3 + \binom{3}{1} p^2 (1-p)$ and the network unavailability $NU = (1-p)^3 + \binom{3}{1}(1-p)^2 p = 1 - NA$.

Obviously, in general, computing the connectivity of all 2^L possible subgraphs is too complex. Therefore we suggest a number of heuristics.

Heuristic 1. The probability that many links fail simultaneously is considered to be very small, therefore a heuristic approach to the method described above would be to only consider the subgraphs in which at most *k* links have failed. The number of connectivity computations now reduces from 2^L to $\sum_{i=0}^{k} \binom{L}{i}$. Since *k* is a given constant, this complexity is polynomial ($O(L^k)$), but nonetheless still considered excessive, for large *k*.

Heuristic 2. A cut of a network identifies a set of links whose removal disconnects the network. In this paper, a network cut is considered from the perspective of an s-d pair. In this case a cut disconnects the network into two subgraphs, with the source in one subgraph and the destination in the other. A minimum cut is the cut which has the lowest weight over all possible cuts between an s-d pair. Finding the min-cut is solvable in polynomial time.

By using a min-cut algorithm (e.g., see [1]) we can compute a lower bound on the network unavailability. Let q_{ij} denote the unavailability of link (i,j). Thus, the higher q_{ij}, the more likely that the link becomes unavailable. We assign to each link $(i,j) \in E$ the weight $-log\ (q_{ij})$. The minimum cut based on these weights gives the highest (but not only) probability to disconnect the source and destination. The set of all min-cuts for all s-d pairs (excluding duplicates) provides us with a lower bound on the network unavailability (and an upper bound on the network availability). One can improve on this bound by considering multiple cuts. Nagamochi *et al.* [3] present an algorithm for computing all cuts for which their weights do not exceed $kw(C^*)$, where $w(C^*)$ is the weight of the minimum cut and $k \geq 1$. The complexity of this algorithm is $O(L^2N + N^{2k}L)$ and $O(N^{2k})$ is an upper bound on the number of cuts with weights not exceeding $kw(C^*)$. This complexity is polynomial, but nonetheless still considered excessive for large *k*.

Heuristic 3. Heuristics 1 and 2 gave upper bounds on the network availability, whereas lower bounds are more insightful. One lower bound has been discussed for path availability, when only one path between source and destination is available. By computing link-disjoint paths we can improve upon this lower bound. For instance, consider two link-disjoint paths P_1 and P_2. Their individual path availability is easily computed. Since both paths are disjoint, their joint availability is computed as 1-(1-$A(P_1)$)((1-$A(P_2)$), where $A(P)$ denotes the availability of path *P*. The more disjoint paths we take into account, the higher the computed availability will be between a pair of nodes. Note that there cannot exists more disjoint paths than links present in the minimum cut (where minimum in this case refers to minimum number of links, i.e. all links have equal weight). Bhandari [2] presents an algorithm for computing *k* link-disjoint paths, which has a complexity of $O(kN^2)$ - *k* executions of a modified Dijkstra algorithm. The complexity is much lower than that of the previous heuristics, but the price is paid in tightness of the bound.

Network availability for simple network topologies

As stated in the previous section, network availability depends on both availability of network elements and network topology. In this section we derive analytical expressions for the unloaded network availability of some simple networks. Throughout this section we assume that the availability per link is independent from other links and that all links have the same availability *p*.

Tree

A tree is a connected graph with no cycles. A tree on *N* nodes has *N-1* links and is only connected when all links are available.

Assuming independence, the network availability of a tree has the following form:

$$NA_{tree} = p^{N-1},$$

where *N* is the number of nodes, and *p* is the link availability.

Unicyclic graph

A unicyclic graph is a connected graph which contains only one cycle, as illustrated in Figure 2. A unicyclic graph of *N* nodes has *N* links.

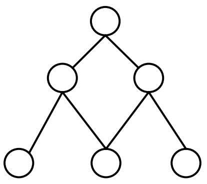


Figure 2 Unicyclic graph (with 4-node cycle)

The network availability of the unicyclic graph has the following form:

$$NA_{uni} = p^N + mp^{N-1}(1-p),$$

where *N* is the number of nodes, *m* is the number of links on the cycle, and *p* is the link availability.

Explanation: When all links are available, the network is obviously connected. The probability of this case is p^N. When one link that belongs to the cycle is unavailable, the network remains connected. The corresponding probability is $\binom{m}{1} p^{N-1}(1-p)$. For all remaining cases the network becomes disconnected.

Double star

A double star is a connected graph that consists of two central nodes that both are connected to all other nodes in the network, but not to each other, as illustrated in Figure 3. The double star occurs very often in telecommunication networks, because it offers a high level of redundancy. For example, the Amsterdam Internet Exchange (AMS-IX), one of the largest public Internet exchanges in the world, uses the double star topology to connect its four locations in Amsterdam to two high-density Ethernet switches[1].

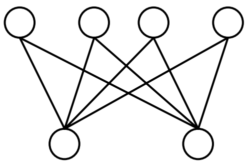


Figure 3 Double star

The network availability of a double star has the following form:

$$NA_{DS} = p^m\left((2-p)^m - 2^m(1-p)^m\right),$$

where *m* is the number of nodes connected to the two central nodes, and *p* is the link availability.

Explanation: Assume exactly *i* links from central node N_1 are unavailable, with $0 \le i \le m$. Then, the remaining *m-i* links from N_1 are available. In order to prevent the *i* nodes that are not connected to N_1 to become isolated, they have to be connected to the central node N_2 , see Figure 4.

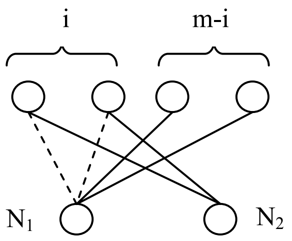


Figure 4 Double star with *i* unavailable links from N_1

The probability that the configuration depicted in Figure 4 occurs is $\binom{m}{i}(1-p)^i p^{m-i} p^i$. In order to make this network connected, of the remaining *m-i* links from N_2, at least one link should be available. The corresponding probability is $1-(1-p)^{m-i}$. Summing over all possible values of *i* we obtain

$$\begin{aligned}
NA_{DS} &= \sum_{i=0}^{m} \binom{m}{i}(1-p)^i p^{m-i} p^i (1-(1-p)^{m-i}) \\
&= p^m \sum_{i=0}^{m} \binom{m}{i}(1-p)^i (1-(1-p)^{m-i}) \\
&\underbrace{=}_{1\text{-p=q}} p^m \sum_{i=0}^{m} \binom{m}{i} q^i (1-q^{m-i}) \\
&= p^m \left(\sum_{i=0}^{m} \binom{m}{i} q^i - q^m \sum_{i=0}^{m} \binom{m}{i} \right) \\
&= p^m\left((1+q)^m - q^m 2^m\right) = p^m\left((2-p)^m - 2^m(1-p)^m\right).
\end{aligned}$$

[1] http://www.ams-ix.net

Crown

A crown is a connected graph that consists of two central nodes that both are connected to all other nodes in the network, and to each other, see Figure 5.

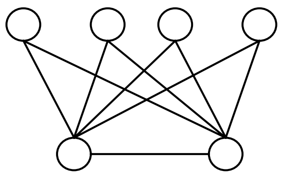


Figure 5 Crown

The network availability of a crown has the following form:

$$NA_{crown} = p^m\left((2-p)^m - 2^m(1-p)^{m+1}\right),$$

The formula can be explained in the same way as for the double star. The only difference is that there are $m\text{-}i\text{+}1$ links remaining from N_2, because of the existence of a link between N_1 and N_2.

Triple star

A triple star is a connected graph that consists of three central nodes that are connected to all other nodes in the network, but not to each other, see Figure 6.

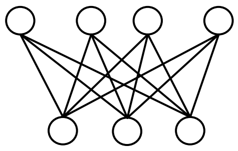


Figure 6 Triple star

The network availability of a triple star satisfies:

$$NA_{TS} = p^m((3-3p+p^2)^m$$
$$-3(1-p)^m(3-2p)^m + 2\cdot 3^m(1-p)^{2m}),$$

where m is the number of nodes connected to the three central nodes, and p is the link availability.

The formula is derived with the same reasoning as in the case of the double star.

Figure 7 shows the comparison of the network availability for the double star and triple star for 8 non-central nodes. Clearly, the triple star provides a significantly higher network availability than the double star.

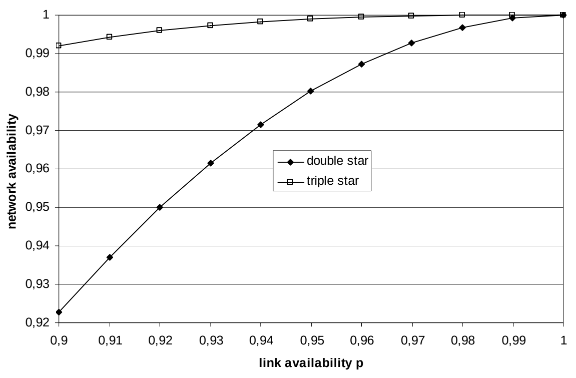


Figure 7 Network availability for double star and triple star with 8 non-central nodes

Availability of the SURFnet network

In this section we determine the network availability of a real-life network, using results from the previous section. SURFnet[2] is a high speed computer network interconnecting higher research institutes and universities in The Netherlands. Staff and students of connected organizations can communicate through SURFnet with other Internet users all over the world. SURFnet has linked its network to the Amsterdam Internet Exchange (AMS-IX) for 'regular' Internet traffic. SURFnet also has links to GÉANT2, the coordinating European research network, and the research networks of, for instance, the United States, Canada, Japan, etc.

Figure 8 depicts the SURFnet topology, Release 5 (currently the SURFnet topology already has been enhanced to Release 6).

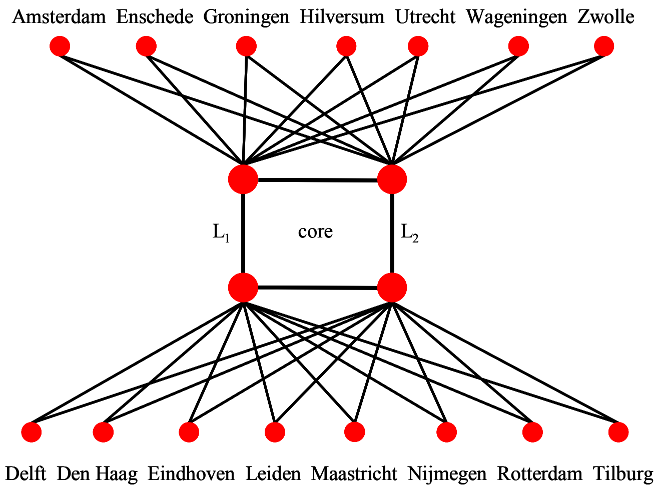


Figure 8 The SURFnet topology, Release 5

From Figure 8 we observe that the SURFnet topology consists of two crown networks, which are connected to each other by two links L_1 and L_2. The upper-crown

[2] http://www.surfnet.nl

contains 7+2 nodes, while the lower-crown consists of 8+2 nodes.

It is easy to see that the SURFnet network is connected in either one of the following three cases:

Case 1: Both crowns are connected and at least one of the two connecting links is available. Because we derived the network availability of a crown already, we can derive the probability P_{case1} that Case 1 occurs:

$$P_{case1} = p^7\left((2-p)^7 - 2^7(1-p)^8\right)\left(1-(1-p)^2\right)\cdot \\ p^8\left((2-p)^8 - 2^8(1-p)^9\right).$$

Case 2: The upper-crown is disconnected but all locations connect to exactly one of the two central nodes in the upper-crown. If the two connecting links are available and the lower-crown is connected, then also the SURFnet network is connected, as depicted in Figure 9.

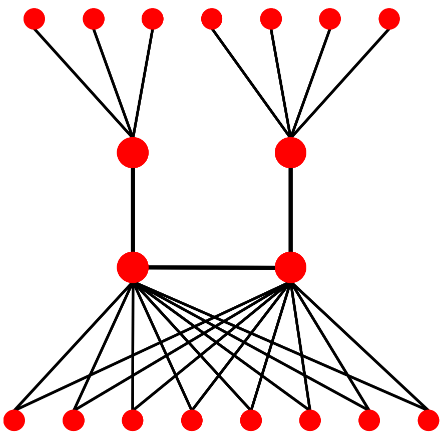


Figure 9 Connected SURFnet with disconnected upper-crown

Denoting the probability that Case 2 occurs by P_{case2}, it is straightforward to show that:

$$P_{case2} = \left(2p(1-p)\right)^7(1-p)p^2\cdot \\ p^8\left((2-p)^8 - 2^8(1-p)^9\right).$$

Case 3: The difference between Case 3 and Case 2 is that in Case 3 the upper-crown is connected while the lower-crown is not. The probability P_{case3} that Case 3 occurs is given by:

$$P_{case3} = \left(2p(1-p)\right)^8(1-p)p^2\cdot \\ p^7\left((2-p)^7 - 2^7(1-p)^8\right).$$

The network availability for SURFnet is obtained by summing the probabilities corresponding to the three cases, i.e.

$$NA_{SURF} = P_{case1} + P_{case2} + P_{case3}.$$

Figure 10 shows the network availability for SURFnet as a function of the link availability p.

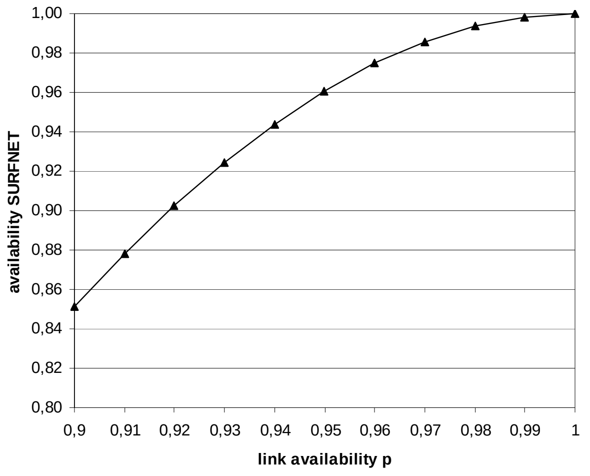


Figure 10 Availability for the SURFnet network

It should be noted that the contribution to the SURFnet network availability due to Cases 2 and 3, is negligible. For instance, for p = 0.9, NA_{SURF} = 0.85146, while $P_{case2} + P_{case3} \approx 10^{-6}$. Upon further increasing p, the sum $P_{case2} + P_{case3}$ will become even smaller, until it reaches zero, as p tends to 1.

Tools for computing availability in more complex networks

In more complex network topologies with large number of nodes, it is difficult to derive analytical expressions for their network availabilities. Therefore, we have developed a tool that implements both the exact algorithm and as well as heuristic 1 described in the section on computation of network availability. This tool can be used to calculate the network availability of any given topology.

However, even when the heuristic is used, the computation time can increase beyond the acceptable level. For this reason, for very large networks (dense networks with more than 100 nodes) we have developed a Network Availability Simulator (NAS). NAS is an event-driven simulator that, for a given network topology and link availability, generates the failure and repair events, and subsequently checks for network connectivity. As a large number of simulation runs (10^5) is performed, results obtained are statistically valid, leading to an accurate estimate of the network availability.

To evaluate these two tools, we compare the network availability results derived analytically, with the results obtained with the algorithm tool and the NAS tool. For the SURFnet topology, assuming a link availability p = 0.9, these results are listed in Table 1.

Network availability	Analysis	Algorithm tool	NAS tool
Both crowns connected	0.85146		
One crown connected	5.4×10^{-7}		
SURFnet	0.85146	0.85146	0.85086

Table 1 Comparison between network availabilities derived analytically, with the algorithm tool and with the NAS simulator

The results in Table 1 demonstrate that both tools generate mutually almost identical results that are also very close to the analytical ones. This indicates that the developed tools can be deployed for computing network availability in topologies to which analytical methods cannot be applied.

Conclusions

In this paper we first discussed different network availability definitions and proposed network connectivity as the unambiguous one. Further we discussed algorithmic methods to obtain network availability values in a given network. The analytical expressions for various (simple) network topologies have been derived and explained. The utility of these results has been indicated by their application to the real-world SURFnet topology. Finally, two tools for computation of network availability in large and complex networks have been presented. We have shown that results obtained with these tools match the analytically derived results remarkably well.

Acknowledgments

This work is partially supported by the IST-2004-026600 DESEREC project (Dependability and Security by Enhanced Reconfigurability), by the Netherlands Organization for Scientific Research (NWO) under project number 643.000.503, and by the GigaPort project, which is led by SURFnet and funded by the Dutch Ministry of Economic Affairs under grant number BSIK03020.

References

1. R.K. Ahuja, T.L.Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, 1993.
2. R. Bhandari, Survivable Networks: Algorithms for Diverse Routing, Springer, 1999.
3. H. Nagamochi, K. Nishimura and T. Ibaraki, Computing all small cuts in an undirected network, *SIAM J. Discrete Math.*, Vol. 10, no. 3, pp. 469-481, August 1997.

Version: November 2007