Adding throughput and link usage as metrics to the network robustness simulator

Student: Sergio Gomez Cosgaya
Supervisors: Carmen Mas Machuca, Jose Luis Marzo Lazaro
Home institution: University of Girona (Spain)
Host institution: Technical University of Munich (Germany)
Working group: Technology-related disasters (WG3)

From April 3, 2017 to April 11, 2017

Motivation

This STSM aims at extending an existing network robustness analysis tool developed by UdG[1] to consider some networking metrics by integrating routing algorithms developed by TUM[2] . In this way, the extended tool is be able to evaluate the robustness of any network for different attack models and using various routing algorithms. The new integrated networking metrics are the throughput and the link usage.

1 Previous background

This section introduces the previous work performed by UdG and TUM, giving a basic background of topics that help the understanding of what is done during the STMS.

1.1 Network robustness simulator, UdG

The BCDS (Broadband Communication and Distributed Systems) research group at UdG has been working for some years in analysing how robust a network is, that means how the robustness is affected against an attack. To achieve this, some metrics are used with different considered failure scenarios that can help to understand how the network reacts against different kind of attacks.

[1] University of Girona
[2] Technical University of Munich

1.1.1 Robustness metrics

The available metrics in the simulator are organized in different groups, depending on what they can tell us regarding the robustness. There are **structural**, **fragmentation**, **connectivity** and **centrality** metrics.

First, structural metrics measure the classical parameters of a graph. Some of them measure the density (such as nodal degree) and others the size, such as the diameter. All considered structural metrics are:

- Average nodal degree
- Heterogeneity
- Average shortest path length
- Diameter
- Efficiency
- Effective resistance
- Number of spanning trees
- Clustering coefficient
- Assortativity
- Symmetry radio
- Largest eigenvalue

Fragmentation metrics address the number of components of a network. It should be taken into account that they are useless for connected networks (i.e. they do not measure anything until the network is unconnected). Available fragmentation metrics are:

- Largest connected component
- Fractional size largest component
- Average two terminal reliability
- Degree of fragmentation

Connectivity metrics measure how difficult is to break the network when removing elements. In contrast to fragmentation metrics, they can not be used when the network is fragmented. All available connectivity metrics are:

- Edge connectivity

- Vertex connectivity
- Algebraic connectivity
- Natural connectivity

Centrality metrics can measure how important are the elements in the graph, i.e. if they are at the middle of most of the paths. All available centrality metrics are:

- Degree centrality
- Node betweenness centrality
- Edge betweenness centrality
- Closeness centrality
- Eigenvector centrality

1.1.2 Failures scenarious

Depending on the generated kind of attacks over a network, different scenarios are obtained. In our simulator, it is possible to attack nodes or edges using **random**, **targeted** and **epidemics** attacks:

- Random attacks:
  In random attacks, attacked elements are selected randomly, even if elements attacked are nodes or edges. Theoretically, this method is the less effective one, cause randomizing doesn't look for a way to make more damage to the network.

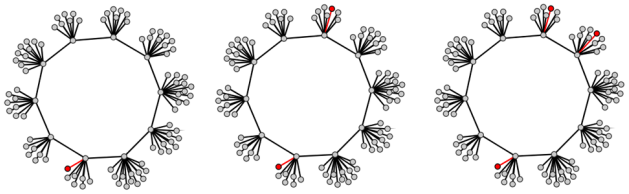

Figure 1: Random attack to nodes example. Each node is selected randomly

- Targeted attacks:
  Unlike randomizing, targeting attacks look for the most important elements in network and select them. For example, the node with a larger

node's betweenness centrality is selected when attacking nodes and the edge with a larger edge's betweenness centrality should be selected in case of attacking edges. Obviously, the parameter used to select the desired elements can be changed.

In the targeted method there are two options. On one hand, there are **targeted simultaneous** attacks where all attacked elements are selected in one step (e.g. the 10 nodes with more node's betweenness centrality), on the other hand, **targeted sequential** attacks where attacked elements are selected step by step sequentially looking for the most important element of the network every time after attacking some other before (e.g. look for the node with more node's betweenness centrality after removing previously other elements).

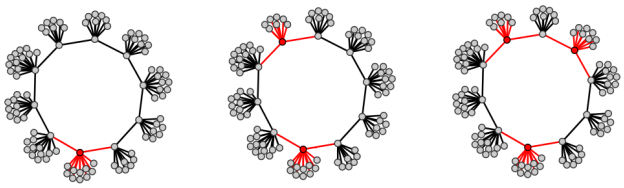

Figure 2: Targeted attack to nodes example. Each node is selected by its importance inside the network. Note that although it can looks like random it selects the central nodes.

- Epidemic attacks:
  When using the epidemic method, the attack simulates an epidemic spread over the network, that is selecting randomly the first infected node and then sequentially select nodes that are connected to an infected node following a probabilistic function. This method only works with node attacks because it has no sense to have infected edges.

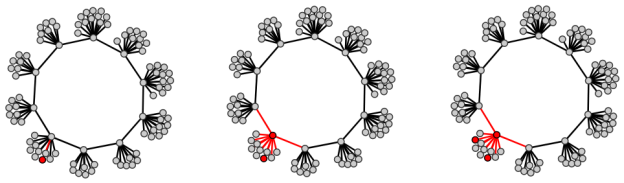

Figure 3: Epidemic attack to nodes example. Next selected node has to be connected to an infected node.

1.1.3 The robustness surface

The simulator provides as a result a robustness surface for each experiment. That is, a set of attacks combinations (defined as M) to a single network with a number of elements attacked (defined as P). The resulting robustness surface will be a PxM table:

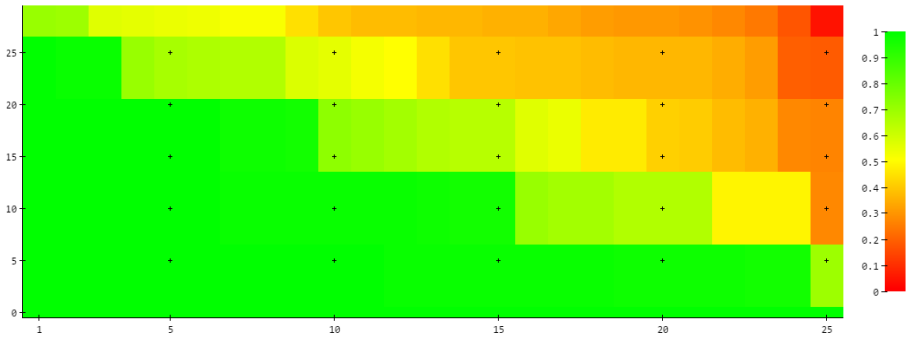

Figure 4: Robustness surface sample. X axis represents samples number (M) and Y axis represents the amount of elements attacked (P).

Each cell will tell us the robustness value[3] of that combination M and the number of elements attacked p (e.g. if P is 30% of elements attacked, there will be a p row when attacking 1%, 2%, etc.). Also, it is colourized within a red-green range[4] which helps comparing in a fast way two or more simulations.

1.2 Proposed routing and networking metrics, TUM

The Department of Electrical and Computer Engineering of TUM has been working developing the algorithm for computing the survived flows and the link utilization inside a network when some elements are removed from it. TUM's work is based on SNDLib[5] networks and demands data.

1.2.1 Optimization algorithm

A network is described as a graph $G = (V, E)$, where V denotes the nodes of the network that can generate or switch the traffic, and E represents the communication links. The links (i,j) have a limited bandwidth C_{ij}. The flows in the traffic demand are described as $d = (s_d, t_d, c_d)$, where s_d and t_d denote

[3]Robustness maximum value is 1, which it means the network is still totally robust and it is not damaged. When damaging it, robustness value decreases and values equal to or less than 0 mean the network is fully damaged and it should not be operative

[4]The range starts at 0 with red colour and ends at 1 with green colour. White colours mean robustness value is less than 0

[5]SNDlib is a library of test instances for Survivable fixed telecommunication Network Design. Official website: http://sndlib.zib.de/

the source and destination of the traffic demand, and c_d denotes the data rate of the flow. Binary variables $z_{ij,d}$ indicate if the edge (i, j) carries flow d, and x_d indicate if the flow d is accepted.

The objective is to maximize the throughput of the survived flows, while minimizing the sum of each demand's path length. Small weight is assigned to the number of used links, to ensure shortest possible paths $\epsilon \ll 1$.

$$\max \sum_{d \in D} x_d c_d - \epsilon \sum_{ij \in E} \sum_{d \in D} z_{ij,d} c_d$$

Capacity and flow conservation constraints have to hold.

$$\sum_{d \in D} z_{ij,d} c_d \leq C_{ij}; \forall (i,j) \in E;$$

$$z_{ij,d} \leq x_d; \forall (i,j) \in E; \forall d \in D;$$

$$\sum_{ij \in E; i=n} z_{ij,d} - \sum_{ij \in E; j=n} z_{ij,d} = s_{n,d} x_d - t_{n,d} x_d; \forall n \in V; \forall d \in D;$$

We also prevent the flow split, by allowing at most one incoming and one outgoing link to be used by the same flow:

$$\sum_{ij \in E; j=n} z_{ij,d} \leq 1; \forall n \in V; \forall d \in D;$$

$$\sum_{ij \in E; i=n} z_{ij,d} \leq 1; \forall n \in V; \forall d \in D;$$

Where s_d and t_d are helper functions defined as:

$$s_d = \begin{cases} 1, \text{if } n \text{ is a source of flow } d \\ 0, \text{otherwise} \end{cases}$$

$$t_d = \begin{cases} 1, \text{if } n \text{ is a destination of flow } d \\ 0, \text{otherwise} \end{cases}$$

1.2.2 Throughput

When optimization algorithm finishes, it is possible to calculate the network throughput metric value that is defined as the sum of all accepted flows:

$$Throughput = \sum_{d \in D} x_d c_d$$

Throughput metric represents numerically the whole quantity of data carried through the network.

1.2.3 Link usage

Link usage average metric value can be calculated from optimization result. It is defined as the sum of the throughput carried by every edge divided by the sum of edges:

$$LinkUtilization = \sum_{d \in D} \sum_{ij \in E} z_{ij,d} c_d / \sum_{ij \in E} C_{i,j}$$

Average link usage represents the usage percentage of edges, saying if network near to its maximum capacity or if it is not in use.

2 Description of the work done

The main goal of the STSM was to add throughput and link usage metrcis from TUM inside UdG simulator. To achieve it, some steps had to been done. Main steps started in manually pass the data and ended to integrate TUM's code into simulator's code.

2.1 Integration process

Before detailing every step, it is important to understand how all final process work. First, a network which is attacked with one of the methods explained below is selected. Then, there is a failure scenario from which all desired metrics are calculated (including throughput and link usage) for every PxM combination and finally from these metrics the robustness surface is calculated as explained in [MCSS+14].

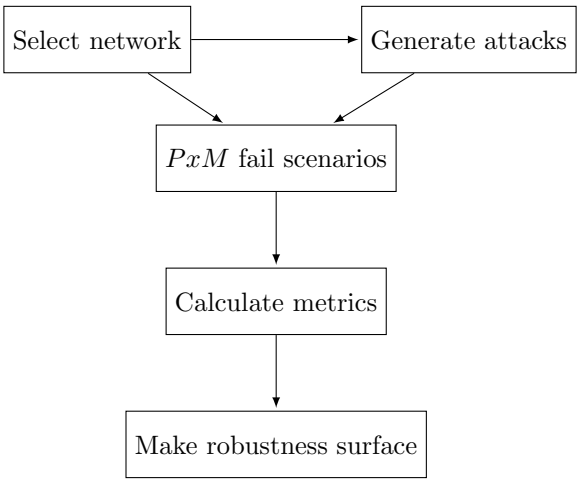

Figure 5: Main process steps. For generating fail scenarios both selected initial network and generated attacks are used, then each fail scenario has its own network which it will always be the initial network less the elements attacked.

2.2 Manual process

We started implementing the process shown in Figure 5 manually. Initially, in UdG, fail scenarios were generated and saved into a **CSV** file, then these attacks were sent to TUM so they could execute the algorithm to calculate the throughput and link usage values. When the execution finished, TUM saved all the results of fail scenarios throughput and link usage metrics in another **CSV** file and sent it back to UdG. Finally, UdG added these results into their simulator to complete the experiment. The process has been depicted in Figure 6.

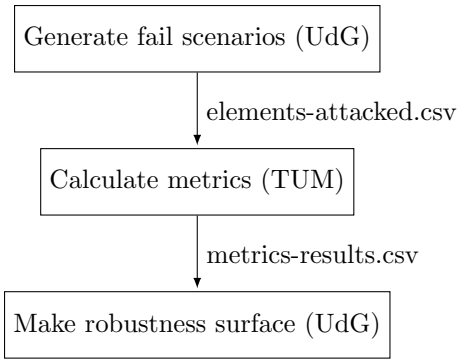

Figure 6: Manual process steps.

Obviously, this process had drawbacks as the time lost when sending the CSV files from each one to each other, but it was necessary to check that the process could be done successfully. When it worked and those relevant expected results were achieved, e.g. comparing how demands are affected in a network against different attack types, the next step was adding these metrics into the simulator.

2.3 Integrating TUM code into the simulator

The main work done during the STSM was related to integrate TUM algorithm code into our UdG simulator. It had some difficulties because the simulator core is written in R[6] whereas TUM throughput and link usage based routing algorithms are written in Python[7]. Although the conversion from Python to R was initially considered, it was discarded in order to facilitate future code updates. So finally, the decided process is the following:

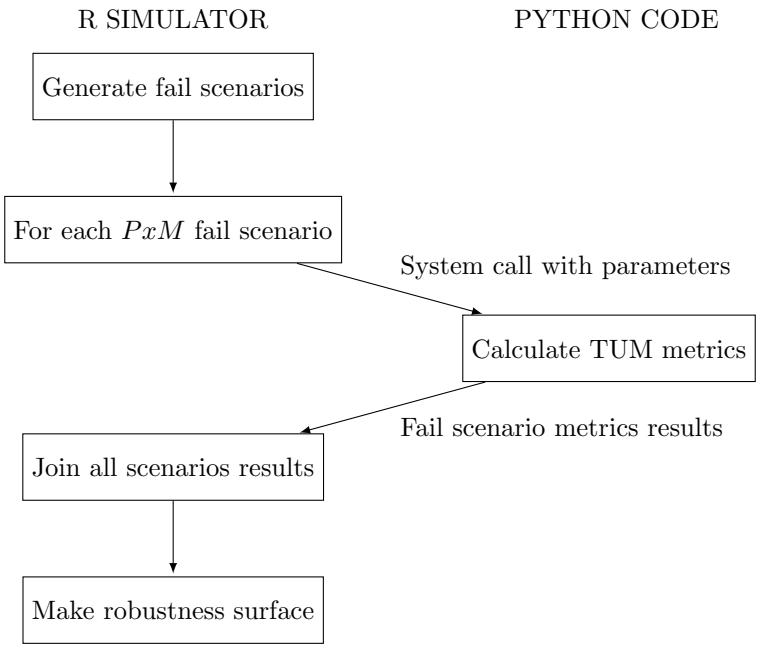

Figure 7: Interaction between R simulator and Python TUM script.

The R simulator generates all fail scenarious and for each one makes a system call to execute the Python script with some parameters such as the path to the

[6] R is a free software environment for statistical computing and graphics. Official website: https://www.r-project.org/

[7] Python is a widely used high-level programming language for general-purpose programming. Official website: https://www.python.org

network file inside the server and the list with all elements attacked of the fail scenario. Then the Python algorithm is executed returning the throughput and link usage to R simulator letting it resume the process.

2.4 Defining different metrics and routing strategies

After the code integration, some time was spent to analyse other metrics and routing strategies related to TUM work done that could be relevant inside a robustness background. TUM algorithm calculated throughput and link usage metrics based on the **number of survived flows** using a **full restoration algorithm**. We decided to add the capability of measure these metrics using different metrics aspects and routing the flows with others methods, some of them were implemented during the STSM and some others have been set as future work.

2.4.1 Traffic related relevant metrics

In this section, traffic related relevant metrics to use and implement into our simulator are detailed:

- Number of survived flows:
  Here goal is to maximize the number of survived flows, which can route more flows (and drop less) but the total sum of throughput routed can not be optimal, e.g. if you have many low demand flows, then other high demand flows would be discarded.
- Survived throughput:
  Maximizing the sum of survived throughput can help routing most of the demanded data but many flows can be discarded in case they have a low demand value.
- Link utilization:
  When maximizing the link utilization, it is intended to distribute all traffic among all edges in the most homogeneous way.
- Path extension:
  An other option is to minimize the average paths length, so our solution will be the one of paths with less hopes between source and destination node.

Maximizing the number of survived flows was the first developed goal metric by TUM and the survived throughput option was implemented during the STSM. Link utilization and path extension options are going to be implemented in a near future.

2.4.2 Routing algorithms

We can use several routing algorithms to route flows through the network. During the STSM two of them were implemented and another one has been marked

as next routing algorithm to implement. Also, other routing algorithms have been thought to include into the simulator in future steps. The main routing algorithms are:

- Unprotected:
  This routing algorithm doesn't try to reroute any flow after it is been removed. Obviously it is the worst case but it has no execution cost.

- Full restoration:
  When using full restoration a routing of **all flows** for each fail scenario, that means removing all previously routed flows and reroute all through the network after some network elements have been removed. It is the most efficient algorithm but the execution cost/time is a relevant drawback.

- Partial restoration:
  The partial restoration algorithm is a more efficient version of the full restoration algorithm. It improves the execution time by not routing again all flows in every fail scenario by routing only affected flows with available edges capacity after network elements removal.

Unprotected and full restoration algorithms were successfully implemented during the STSM and the partial restoration algorithm developing has been starting and the end of stay. As it is said, there are other algorithms to be implemented in future steps. One example of them is the **shared protected** algorithm where two non related paths[8] share the same protected path[9].

3 Preliminary results

As a simple demonstration of the work done, an execution over some networks as Abilene, cost266, Germany50 and India35 available in SNDLib repository[10] was done and the robustness surface using some UdG metrics and both throughput and link usage TUM metrics was generated:

[8] Two paths are not related if they don't share any node or edge between them

[9] A shared path is a pre-calculated path that will be used automatically when default path is not available

[10] SNDLib official repository: http://sndlib.zib.de/home.action

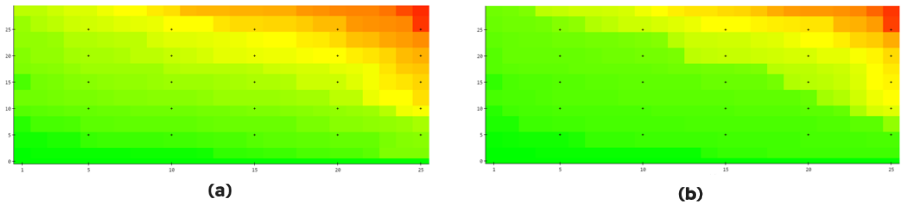

Figure 8: Comparison of simple execution results using unprotected routing (a) and full restoration routing (b). Both executions are over the cost266 network against random attacks to nodes. X axis represents samples number (M) and Y axis represents the amount of elements attacked (P).

In figure 8 (b) can be seen the robustness surface generated using the unprotected routing algorithm and at right there is the robustness surface resulting when routing with the full restoration algorithm. Obviously the first one (a) has a worse result (less green means less robust) because unprotected routing does not recourse any failed flow.

4 Future work

The main future work is adding to the simulator the metrics and routing methods detailed in previous sections. Then in order to write a publication, a more in depth robustness analysis of networks against the mentioned attacks in this document will be done using the new added throughput and link usage metrics.

A more in depth analysis about how different networks react to different attacks using the proposed throughput and link usage metrics calculated will be carried out. Using the obtained results will help to write the mentioned publication .

References

[AJB00] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[DC04] Anthony H Dekker and Bernard D Colbert. Network robustness and graph topology. In *Proceedings of the 27th Australasian conference on Computer science-Volume 26*, pages 359–368. Australian Computer Society, Inc., 2004.

[MCSS+14] Marc Manzano Castro, Faryad Sahneh, Caterina Scoglio, Eusebi Calle Ortega, and Josep Lluís Marzo i Lázaro. Robustness surfaces of complex networks. *Scientific Reports, 2014, núm. 4, P. 6133*, 2014.

[MCTP+13] M. Manzano, E. Calle, V. Torres-Padrosa, J. Segovia, and D. Harle. Endurance: A new robustness measure for complex networks under multiple failure scenarios. *Computer Networks*, 57(17):3641 – 3653, 2013.

[SH+10] James P.G. Sterbenz, David Hutchison, Egemen K. etinkaya, Abdul Jabbar, Justin P. Rohrer, Marcus Schller, and Paul Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8):1245 – 1265, 2010. Resilient and Survivable networks.

[SSYS10] Ali Sydney, Caterina Scoglio, Mina Youssef, and Phillip Schumm. Characterising the robustness of complex networks. *International Journal of Internet Technology and Secured Transactions*, 2(3-4):291–320, 2010.