Advanced Topology Analysis in Three Wireless Community Networks

Michele Pittoni

2014 - VI

Contents

1 Introduction

2 Wireless community networks

Ninux

FFWien

FFGraz

3 OLSR summary

4 Robustness analysis of synthetic topologies

Scope

Analysing the robustness of different kinds of graphs of known topology.

Methodology

For each kind of topology generate and analyse 30 different graphs, then compute an average of the results.

Topologies

- Random network (`fast_gnp_random_graph`)
- Scale free network with a power law distribution (`barabasi_albert_graph`)

Computing strategies

Different strategies have been used to test the robustness under different conditions. All the computations were done by removing nodes one at a time and computing the size of the giant cluster of the remaining graph. The order in which nodes were removed varied between strategies.

Random removal

The first strategy was to randomly select the order of the node removal, to simulate random independent failures of network components. This test was repeated 30 times per graph, with a different order each time.

Ordered removal

To simulate an attack on the most strategic nodes of the network, the nodes were removed ordered by different metrics, such as:

- degree

- betweenness centrality
- closeness centrality
- ratio of degree over clustering coefficient

The metrics were calculated on the original graph and used to order the nodes. The metrics and the order were not updated after the removal of each node, in order to avoid increasing the complexity too much. For these metrics, only one test was done on each graph (since the order would not have changed anyway).

5 Message propagation analysis

The importance of routing

The robustness of a network is based on a static analysis of the connectivity of the network graph when removing nodes or links. A communication network, however, is a dynamic system where information needs to move between nodes. Moreover, the decentralised nature of computer networks means that the complete topology of the network is not necessarily the topology used to transmit information, depending on the routing protocol used for the network.

Given this, in order to understand the behaviour of a communication network we need to study the behaviour of its routing protocol with different underlying topologies. We are interested in the phase of topology discovery, where each node receives information on the existence of the other nodes in the network and (part of) the route to reach them.

Topology discovery in link state routing protocols is usually performed with each node flooding the network with some kind of `hello` message. The possible variations are the flooding policy and the contents of the message. The most popular routing protocols used in mesh networks behave as follows:

- B.A.T.M.A.N. uses the simplest possible flooding (each node just performs a duplicate detection to avoid loops) and the `hello` message contains the sender address and a sequence number (for the duplicate detection)
- OLSR employs a more sophisticated flooding mechanism based on MPRs and the `hello` message contains the whole neighbourhood of the sender
- versions of OLSR used in practice usually force each node to be an MPR,[1] thus having an hybrid behaviour with flooding as in B.A.T.M.A.N.

Problem definition

The network is represented by a weighted graph $G = (N, E)$, where weights represent the probability of losing a packet on each link (we use the ETX metric of OLSR for this purpose).

[1] Maccari (2013)

Each node creates a message with information on its neighbourhood and propagates it to each neighbour. Each node also propagates the message it receives, with a simple duplicate detection based on the sender to avoid loops.

Further iterations of the analysis will consider a subset of nodes generating and propagating the messages and a different protocol for loop avoidance.

Given the above situation, we define

$$T_u = \forall v \in N.\ \text{“node } v \text{ has a route to node } u\text{”}$$

$$R_u = \forall v \in N.\ \text{“node } u \text{ has a route to node } v\text{”}$$

Determine the probability of T_u and R_u for each node u in the graph.

Methodology

The propagation of a message with duplicate detection can be simulated with a Breadth First Search (BFS) over the graph. The most important variation is that before traversing an edge a random number is generated and compared to the packet loss probability of that link, to check if the transmission succeeds. During the BFS, the simulation keeps track of which nodes received the message and based on the content of the message determines the couples of nodes which have a known route between themselves. The search is repeated for every node as the starting point. The union of the results is then used to verify T_u and R_u.

The random simulation is run 1000 times to gather a significant figure of the probability of T_u and R_u.

The propagation for a node is as follows in pseudocode:

function propagate(Graph g, Node u)

```
message_sender <- u
message_content <- neighbourhood_of(u)
q <- Queue()
route_knowledge <- set()
u.visited <- True
for v in u.neighbours append (u,v) to q
while q is not empty do
    pop (u,v) from q
    n <- random()
    if (not v.visited) and (n   weight(u,v))
        v.visited <- True
        for i in message_content
            add (i,v) to route_knowledge
        for w in v.neighbours append (v,w) to q if w   u
return route_knowledge
```

The function is run for each node in the graph an the results are collected.

function propagate_all(Graph g)

```
rk <- set()
t, r <- array()
for u in g
    rk <- rk   propagate(g, u)
for u in g
    if (u,v)   rk   node v   u
        t[u] <- 1
    else
        t[u] <- 0
    if (v,u)   rk   node v   u
        r[u] <- 1
    else
        r[u] <- 0
return t, r
```

function run_simulation(Graph g, Integer n)

```
pt, pr <- array()
for n times
    t, r <- propagate_all(g)
    pt += t                                 % sum each element
    pr += r                                 % "
divide each element of pt by n
divide each element of pr by n
return pt, pr
```

Rationale

The feasibility of calculating T_u and R_u exactly has been evaluated. However, the computational complexity of this approach seems very high and likely does not justify its use in place of the Monte Carlo simulation.

Going into the details, the probability of a message propagating from node u to node v is easily calculated by… ~~summing the probabilities of success for every simple path between the two nodes.~~ With this result for every possible destination $v1 \ldots vn$ of a message transmitted by u, it's theoretically possible to calculate T_u but it's not easy: the events "reaching v1"…"reaching vn" are not independent, so the probability of "reaching every node" is not the product of their probabilities.

For example, if w is in any simple path from u to v, the probability of success between u and v changes if it is known that node w has been reached.

$P(v) = P(w) \cdot P(v|w) + P(\neg w) \cdot P(v|\neg w)$

The value of $P(v|w)$ and $P(v|\neg w)$ is not so obvious:

- $P(v|w)$ is the sum of the probabilities of the subpaths $w \to v$ for every simple path uv
- $P(v|\neg w)$ is the sum of the probabilities of success for simple paths from u to v excluding the paths that contain w

This must be computed for every w that appears in at least one simple path $u \to v$. Again, this computation must be repeated for every possible source-destination pair u, v.

Developments

Save the paths in order to figure out which one is the best (between the ones that succeeded in the simulation)

- No neighbours (B.A.T.M.A.N.)
- MPR (see ninux-topology-analyzer for MPR solver)

6 Conclusions

Bibliography

Maccari, Leonardo. 2013. "An Analysis of the Ninux Wireless Community Network." In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, 1–7. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6673332.