Advanced Topology Analysis in Three Wireless Community Networks

Michele Pittoni

2014 - VI

Contents

1 Introduction

Wireless Community Networks (WCNs), a particular kind of wireless mesh networks, have become more and more popular in recent years. These networks are different from traditional ones in various aspects, such as the link nature and the arising topologies. For this reason, generic network protocols may not be well suited for WCNs and much research effort is being dedicated to the development of new routing protocols for this networks and for measuring their efficiency.

This works provides an introduction to the topic of Wireless Community Networks and a description of the three networks which are subsequently analysed. Also provided is an introduction to OLSR, the routing protocol which is used in the analysed networks. The biggest part focuses on the analysis of some efficiency metrics for the chosen networks and a comparison with some well known random network models.

2 Wireless Community Networks

Wireless Community Networks have existed since 2000 and have gained popularity with the decrease of the cost for Wi-Fi equipment. The idea behind them is simple but powerful: leveraging Wi-Fi technology to create a network which is owned by citizens instead of corporations. The basic component of a WCN is the node, which usually consists of a router with some wireless interfaces attached. Most nodes have one or more directional interfaces to communicate with other nodes and some also have an omnidirectional interface to serve as an Access Point (AP).

Because of their nature of not asking permissions, it is difficult to determine precisely the number of active WCNs as there is no central registry to look at.

However, there is a Wikipedia page[1] which, albeit incomplete, lists 262 WCNs at the time of writing. The dimensions of such networks are varied, from just a handful of nodes to nearly 25,000 as in Guifi[2], which is probably the world's largest WCN.

The three WCNs which are analysed later are Ninux, Funkeuer Wien and Funkfeuer Graz. The study considers 50 snapshots of the networks taken from … to … .

Ninux

Ninux[3] is the largest italian WCN. It was started in 2001 in Rome and now consists of about 250 active nodes, located in different "Ninux islands" all over Italy. The analysis considered only the biggest connected island, with 132 nodes and 154 links ($\langle k \rangle \simeq 2.333$).

OLSR is used as the routing protocol inside islands, while the islands are connected together using tunnels.

FFWien

FunkFeuer[4] is the collective name of different WCNs in Austria. FunkFeuer Wien[5] (FFWien) is the biggest, with 237 nodes and 433 links ($\langle k \rangle \simeq 3.654$) and it covers, according to the website, 1/3 of Wien (Vienna).

FFGraz

FunkFeuer Graz[6] (FFGraz) is the "smaller sister" of the FFWien network, situated in the homonymous city. It consists of 144 nodes and 199 edges ($\langle k \rangle \simeq 2.764$).

Both FFWien and FFGraz also use OLSR as a routing protocol. It should be noted, however, that the versions of OLSR used in practice in WCNs do not behave exactly as the specification of the protocol mandates. The next section

[1] ("List of Wireless Community Networks by Region. Wikipedia, the Free Encyclopedia" 2014)

[2] http://guifi.net

[3] http://wiki.ninux.org/

[4] http://www.funkfeuer.at/

[5] http://www.funkfeuer.at/Vienna.206.0.html?&L=1

[6] http://graz.funkfeuer.at/

discusses both OLSR and the differences between the standard and the real cases.

3 OLSR summary

Optimized Link State Routing (OLSR) is a link state routing protocol, standardized by the IEEE in RFC 3626[7] and designed to have a better performance on wireless mesh and ad-hoc networks than traditional protocols for wired networks. The most peculiar feature of OLSR is a more efficient flooding technique which allows to propagate topology information using a fraction of the traffic required by simple uncontrolled flooding. Reducing overhead traffic for routing is key in wireless networks where links can become congested easily and even more so in WCNs where the nodes are usually low-power devices.

The key concept of OLSR is the use of multi-point relays (MPRs) to achieve a more efficient distribution of routing information. After each node selects its own MPRs between its neighbours, the other neighbours will not rebroadcast topology information from it. This means less processing load and less unnecessary traffic for the network. The selection of MPRs works in two steps: neighbourhood discovery and relay selection.

In order to select its MPRs, a node must first know its 2-neighbourhood. This is achieved by receiving `HELLO` messages from the first neighbours, which communicate their respective neighbourhood. After receiving the `HELLO` messages of all its neighbours, a node knows its 1-neighbours, 2-neighbours and to which 2-neighbours every 1-neighbour is connected.

Using this information, the node can select a set of its 1-neighbours such that every node in the 2-neighbourhood is at 1 hop from the set. Formally, call $N_1(u)$ the set of 1-neighbours, $N_2(u)$ the set of 2-neighbours, select $S(u) \subseteq N_1(u)$ such that

$$\forall v \in N_2(u) \exists s \in S(u) \text{ st. } v \in N_1(s)$$

This requirement essentially means that each node in the strict 2-hop neighbourhood can be reached through a MPR. The protocol specification suggests that the MPR set of each node should be as small as possible, but does not require it. Once a node has selected its MPRs, it will signal its choice in the

[7] Jacquet (2003)

subsequent HELLO messages. Each MPR registers its choice, adding the selector node to its MPR Selector Set, and uses this information when deciding whether to forward a packet or discard it.

Link quality

OLSR implements a mechanism to avoid using “bad” links (i.e. links which are usually too weak but may let HELLO messages pass from time to time). Since HELLO messages are transmitted at a regular interval, each node knows how many of them to expect from each neighbour over a period of time. Comparing this with the number of received messages it computes a measure of the Link Quality (LQ). This metric was originally only used to decide if a link was reliable enough to use. New versions of OLSR have put more importance on link quality.

It is common in WCNs to use the ETX metric to express link quality. ETX stands for Expected Transmission Count and was proposed in De Couto (2004). It indicates the expected number of transmissions (including retransmissions) required to successfully deliver a packet.

In OLSR, ETX is derived directly from LQ. HELLO messages contain the calculated values, so each node has for every link two measures: its own (LQ) and its neighbour’s (NLQ). Since each packet transmission requires an acknowledgement, the estimated probability of success is $\mathrm{LQ} \cdot \mathrm{NLQ}$. ETX is calculated as

$$\mathrm{ETX} = \frac{1}{\mathrm{LQ} \cdot \mathrm{NLQ}} \tag{1}$$

4 Robustness analysis

The first metric analysed is the robustness of the network. The chosen methodology is a variation of the percolation problem described in Chapter 16 of (Newman 2010).

Defining robustness is not trivial. While it is intuitive that removing nodes or links affects the network ability to successfully transport information, it is not obvious how this ability can be quantified. Moreover, nodes and links in a network are note all equal, neither in the impact of their removal nor in their probability of failure in the real world.

The first concern is traditionally addressed in literature by considering the connected components of the remaining graph. Specifically, the metric is defined as the ratio

$$S = \frac{\text{size of the largest connected component}}{\text{size of the original graph}} \tag{2}$$

Then the inequalities of nodes and link must be taken into account. This is a more complicated matter because there is not a single correct solution. The approach largely depends on the real world situation to be analysed. For example, to evaluate the robustness of a network to random equipment failures the nodes can be removed in a random order. On the other hand, to simulate a targeted attack scenario the nodes with highest degree can be removed first, assuming attackers will direct their action to cause the highest possible damage. Other node or link metrics can be used in the same way, to simulate other scenarios.

Here different criteria have been used to gather a broad set of data. Specifically, nodes were first removed randomly with uniform probability, as in the classic percolation problem. Then they were removed following the order based on the following criterias.

Degree The degree of a node is the number of edges connected to that node

Betweenness centrality The beetweennes centrality of a node is the fraction of the shortest paths between any two other nodes that pass through that node

Closeness centrality The closeness centrality of a node is the mean lenght of the shortest paths from that node to every other node

Analysed networks

In addition to the three WCNs, some graph have been generated based on known random graph models. The chosen models, explained below, are the Erdős-Rényi random model and the Barabási-Albert preferential attachment model. The reason for the choice is that a very different behaviour is expected with respect to the various robustness metrics, as described in (Albert, Jeong, and Barabási 2000). The goal is to compare the well known behaviour of these topologies to the real ones of the WCNs. The implementation of these models provided by NetworkX has been used.

Erdős-Rényi random graph

(NetworkX generator: `fast_gnp_random_graph`)

The random graph model originally proposed by Erdős and Rényi is also called $G(n, M)$ model, since it consists in the uniform random selection of a graph from the set of all graphs with n nodes and M edges.

The model used here is a variaton first introduced by (Gilbert 1959), called the $G(n,p)$ model. The algorithm starts form a graph with n nodes and no edges. Then, for each unordered pair of nodes $\{i,j\}.i \neq j$, the edge ij is added with probability p.

The $G(n,p)$ models has some interesting properties which are not obvious at a first look. For example, the number of edges is not known as in the $G(n,M)$ models, but the expected number of edges can be determined to be $\binom{n}{2}p$. Another important aspect is connectedness: for $p > \frac{(1+\epsilon)\ln n}{n}$ the graph will almost surely be connected, while for $p < \frac{(1-\epsilon)\ln n}{n}$ it will almost surely have isolated vertices.

Finally, the degree distribution has the form

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

Barabási-Albert graph

(NetworkX generator: `barabasi_albert_graph`)

A scale free network is a network whose degree distribution follows a power law of the form $p_k = Ck^{-\alpha}$.

A method for generating graphs with a power law degree distribution, using a preferential attachment mechanism, was devised by A. L. Barabási and R. Albert in (Barabási and Albert 1999). This is the method implemented by NetworkX. Given a target number n of nodes and a parametes m which controls the density of the network, the algorithm starts from a graph with m nodes and no edges. Then other nodes are added and from each new node m edges are created. The new edges are attached preferentially to the nodes with higer degree. This continues until there are n nodes in the graph, meaning the final graph will contain $(n-m)m$ edges.

Practical considerations

To obtain meaningful results, the syntetic graphs need to be comparable with the real topologies at least in some aspects, the most obvious being average degree. In order to achive this, the parameters of the generators must be adjusted remembering that:

- for the Erdős-Rényi model, the expected average degree is

$$\langle k \rangle = \frac{2\binom{n}{2}p}{n} = \frac{2}{n}\frac{n!}{2!(n-2)!}p = \frac{1}{n}\frac{(n)(n-1)(n-2)!}{(n-2)!}p = (n-1)p \qquad (3)$$

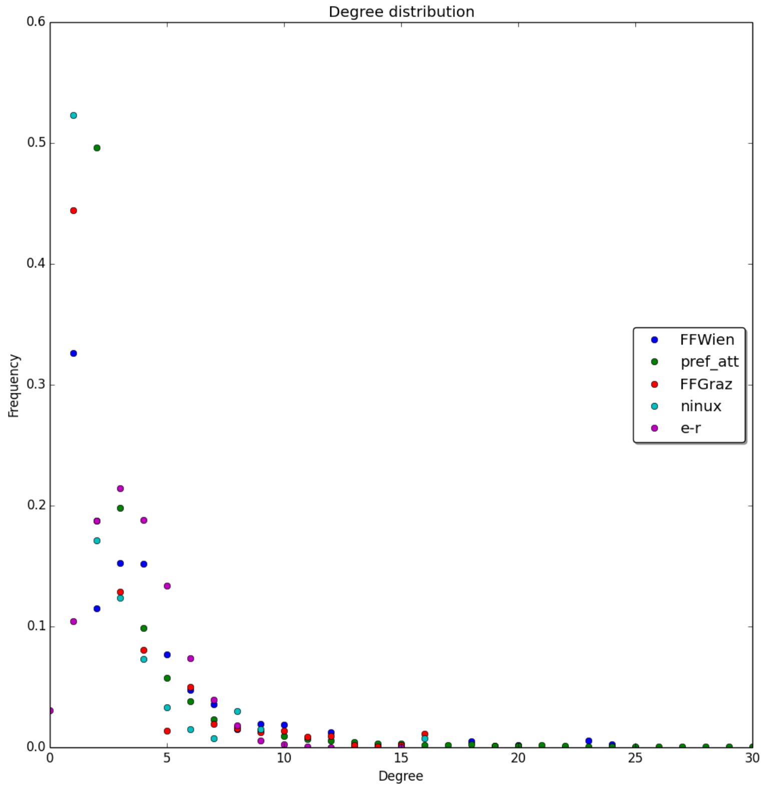
- for the Barabási-Albert model the exact average degree is known

$$\langle k \rangle = \frac{2(n-m)m}{n} \tag{4}$$

In the second case, since $m \in \mathbb{N}$, the value of $\langle k \rangle$ can not be controlled precisely once fixed n.

Results

Degree distributions



As expected, the degree distributions of the three WCNs are quite similar to that of the Barabási-Albert graph, while the Erdős-Rényi graph has a binomial distribution.

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,[8] 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).

[8] 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

Albert, Réka, Hawoong Jeong, and Albert-László Barabási. 2000. "Error and Attack Tolerance of Complex Networks." Nature.

Barabási, Albert-László, and Réka Albert. 1999. "Emergence of Scaling in Random Networks." *Science* 286 (5439): 509–12. doi:10.1126/science.286.5439.509. http://www.sciencemag.org/content/286/5439/509.

De Couto, Douglas SJ. 2004. "High-Throughput Routing for Multi-Hop Wireless Networks." PhD thesis, MIT. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.117.3059&rep=rep1&type=pdf.

Gilbert, E. N. 1959. "Random Graphs." *The Annals of Mathematical Statistics* 30 (4): 1141–44. doi:10.1214/aoms/1177706098. http://projecteuclid.org/euclid.aoms/1177706098.

Jacquet, Philippe. 2003. "Optimized Link State Routing Protocol (OLSR)." January. http://tools.ietf.org/html/rfc3626.

"List of Wireless Community Networks by Region. Wikipedia, the Free Encyclopedia." 2014. June 10. https://en.wikipedia.org/w/index.php?title=List_of_wireless_community_networks_by_region&oldid=612342893.

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.

Newman, Mark. 2010. *Networks: An Introduction*. Oxford University Press, USA.