Loop Avoidance for Fish-Eye OLSR in Sparse Wireless Mesh Networks

Yasir Faheem, Jean Louis Rougier

Loop Avoidance for Fish-Eye OLSR in Sparse Wireless Mesh Networks

Yasir Faheem
TELECOM ParisTech. Paris, France
faheem@telecom-paristech.fr

Jean Louis Rougier
TELECOM ParisTech. Paris, France
rougierj@telecom-paristech.fr

Abstract— **The use of Fish eye scoping has been introduced to reduce the overhead of the OLSR routing protocol. This simple method is based on reducing the scope (TTL) of some topology updates, thus giving routers a precise view of their close neighborhood and a more and more approximate view of farther nodes. Fish Eye OLSR (OFLSR) has been showed to have excellent scaling properties and low network overhead. However, if deployed in relatively sparse networks, this scoping limitation of topology updates can result in long living routing loops, thus limiting the potential applications of such mechanisms in some practical wireless mesh networks.**

In this paper, we address the transient mini-loop problem due to fisheye scoping. We first analyze the occurrence of mini-loops. We discuss potential solutions and propose a pragmatic and distributed off-line heuristic, which allows each router to compute "safe" scope for topology updates. With our method, every mesh router calculates in advance the minimum TTL value that avoids mini-loops at the "scope" boundary --- optimal scope that will be set for generating topology update message whenever a neighbor link lost is detected. Simulations show that the proposed algorithm drastically improves safety of Fish Eye OLSR while still retaining its scaling and performance properties.

Terms— **Fish OLSR, transient loops, Wireless Mesh Networks.**

I. Introduction and Rationales

Wireless Mesh Networks (WMNs) are operated or self-organizing wireless networks capable of relaying data over multi-hops. Based on recent advances in wireless networks, in particular 802.11 capacity increase, such networks have turned out to be a very economical alternative in a vast range of environments like last mile communications, backhauling, safety networks, etc. Usually, WMN are based on MANET[1] routing protocols, mostly for self-adaptation and ease of deployment. In this paper, we concentrate on proactive routing protocols such as OLSR [1], in which the whole[2] network topology is stored in each node, from which a routing table for all potential destinations is computed. The network topology is maintained through topology control messages, broadcasted throughout the network regularly or optionally whenever a change in the network topology is detected.

In this context, OLSR is a well-known link state routing protocol, optimized for the wireless ad-hoc network environment. As compared to legacy link state routing instances (in wired networks), OLSR brings in particular specific neighbourhood discovery (with hysteresis mechanisms for link quality estimation), reduced topology flooding overhead in dense networks, though the use of MPRs (Multi-Point Relays). More details can be found in [1].

Many works have considered means to further reduce overhead of proactive link-state protocols such as OLSR. In particular, one of the simplest and most elegant idea was brought with Fisheye State Routing [2, 3], which increases scalability by reducing the scope (TTL[3]) of some topology updates. For instance, a router may send every two update with TTL=2, every four update with TTL=4 and every eight update with global TTL. With this protocol, routers have thus a precise view of their close neighborhood and a more and more approximate view of farther nodes. OFSLR [4] has been defined to combine OLSR optimizations (in particular MPRs) with Fisheye TC broadcast scoping. Several studies both analytically [5] and by means of simulations [6, 7] show that this protocol is highly scalable.

There are also many on-going works on very scalable ad-hoc routing in different environments (MANET, sensor networks, VANET[4]), considering alternative routing paradigms (geographic routing, etc.). These promising results may provide potentially even better scalability. However, we concentrate on OFSLR as a pragmatic short-term solution, which can be directly used in existing WMN. In fact, our original focus is not network size but frequent topology broadcasts due to fast mobility of some nodes [8], in which OFLSR brings a great interest. However, introduction of OLSR in sparse networks sorted out to cause routing loops, as explained in the next section.

II. Problem Statement: Mini-Loops

With links state technology, whenever a link breaks or there is an update in the network, routing loops can occur during a short period of time, as a normal phenomenon during network convergence. This is due to the difference in transmission delays and processing times of routing updates, which cause different routers to have temporarily different view of the current topology. These so-called *micro-loops* can exist from several milliseconds to several seconds depending upon the nature and parameters of the routing protocol being deployed

[1] MANET: Mobile Ad-Hoc Network.
[2] or a partial but sufficiently consistent view of the network topology.
[3] TTL: Time To Live
[4] Vehicular Ad-Hoc Network.

978-1-4244-3375-9/09/$25.00 ©2009 IEEE

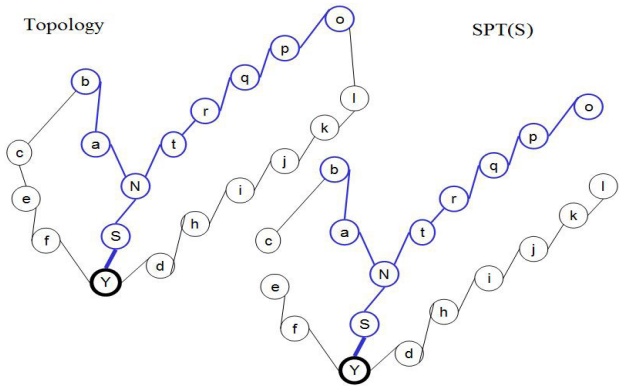


Figure 1: Topology view and Shortest Path Tree of source S rooted at S.

[9]. With Fisheye scoping, network convergence is reduced to a given TTL boundary. Thus, instead of normal network convergence, the loops forming at the scope boundaries will last until next topology update is generated with larger scope. In figure 1 for instance, if link *Y-S* becomes unavailable and *S* sends its topology update with *TTL=2*, only nodes *N, A* and *T* will be aware of the topology change. All traffic towards Y will redirected to *A* by router *N* and to *B* by router *A*; but *B* may send back this traffic to *A* as its best route towards Y can still be through *A* (remember than *B* is not in the TTL scope). Statistics on such loop occurrence in some random topologies are given in section IV.

We will speak of *mini-loops*, highlighting their longer duration (typically tens of seconds, depending of the fish eye flooding parameters). Even if quite obvious, this problem was not previously mentioned to the authors' knowledge; probably since mini-loops do not occur in *dense* networks (typical simulation framework in ad hoc networking). For economical and performance reasons (e.g. use of directional antennas), we have seen, however, that current deployed WMN can be *sparse* (i.e. without any warrantee to have 2 hop alternate routes at each node for instance). A mechanism to safely configure fish eye mechanism is thus required in this context.

III. PROPOSED ALGORITHM & BACKGROUND INFORMATION

In this paper, we propose a simple algorithm, which allows computing the best scope that can be safely used to upgrade information of a given link. We can use it as follows: when a mesh router is not performing more critical tasks, it calculates in advance the safe minimum scope value for each of its adjacent link[5]. This value can be used with fish eye to reduce the scope of part of the topology flooding, without causing mini-loops. We proposed this *distributed offline* approach believing that mesh routers usually do not have power consumption constraints (contrary to ad-hoc/sensor networks).

We first state some *SPT* (Shortest path tree) properties on which our work is based. Note that we benefited for this from previous works on loop-less convergence in OSPF networks, like [10], which have some similarities. Let suppose that a mesh router S wants to determine safe TTL value for neighbour *Y*. Upon link breakage *S-Y*, the routers that get affected are the ones that have link $S \rightarrow Y$ or $Y \rightarrow S$ in their *SPT* [10]. They will re-compute their SPTs to reroute their traffic. Hence a loop will occur when an effected router reroutes its traffic towards an adjacent router that lies on the same effected branch ($S \rightarrow Y$) of *rSPT(Y)* (reverse Shortest Path Tree rooted at Y) and that is not yet aware of the topology change. Determining which routers have link *S-Y* in routing tables at node *S* has complexity $O(n^3)$ (e.g. Floyd Warshall). Then *S* would need to identify alternate routes that each router would to be use in case of a link failure. If every node *S* has an average degree of *m* (so *m* links), it would require calculations of at least $O(mn^3)$ to determine the safe TTL scope value — which is quite complex. Here we propose an approximate approach with lower complexity, even if it does not eliminate mini-loops with 100% confidence.

We have identified that almost all routing loops mainly occur: (i) at the scope boundary, (ii) for a destination which is directly adjacent to the failing link (e.g. towards *Y* for link *Y-S*). We can thus narrow our search to these most frequent events and thus reformulate the problem as follows. From the point of view of router *S,* the *sub-branch* $S \rightarrow Y$ of *rSPT(Y)* gives all routers that are using *S* to reach *Y* ($S \rightarrow Y$) [10]. We want to determine the nodes on branch (*Y-S*) of *rSPT(Y)* at minimum hop distance, such that, if they are informed with that particular hop distance scope, there will be no mini-loop.

The algorithm requires the following *pre-computations*. First the legacy *SPT* computed by OLSR need to be slightly modified. First, in order to detect all potential mini-loops, we need a *dijkstra ECMP, i.e.* considering ECMP paths[6]. It other words, whenever a node has two equal cost next hops to reach a particular destination they are both saved in the SPT structure. Note that the resulting SPT structure is now an acyclic graph and not a tree. The second modification is that dijkstra ECMP shall compute a reverse SPT at the same time. Note that, as links are symmetric in OLSR, both *SPT* and *rSPT* are the same, so no further computation time is required. We only require an additional tree structure for *rSPT* to be stored in memory: The reverse tree gives information on parent nodes for each node: e.g. in Figure 2, node *b* has two parents: *a* and *c* in *rSPT(Y).*

The algorithm works then as follows: Suppose router *S* wants to determine the TTL value of topology update to be sent when *S* detects loss of neighbour *Y*. This computation will be done by node *Y* and then will be transmitted to *S* through some hello message extensions for instance. As stated before, we consider *rSPT(Y)* is available at node *Y*, which then further calculates *rSPT_broken(Y)*, considering link *S-Y* broken[7]. This will give the shortest path of all nodes towards *Y* after link breakage. To detect loops, we compare the parent node of every two neighbour nodes on sub-branch *S-Y* in *rSPT(Y)* and *rSPT_broken(Y).* If the parents of two nodes are pointing towards each other there will be a mini-loop between those

[5] In fact with OLSR, only the MPR selectors need to do this computation.

[6] Original dijkstra algorithm used in OLSR does not consider ECMP (Equal Cost Multi-Path)

[7] An incremental SPF algorithm may be used to further reduce complexity.

nodes (if this link was at the fish eye scope boundary). Consider node *N* in Figure 2, the parent of node *N* in *rSPT_broken(Y)* is node *a* whereas parent of node a in *rSPT(Y)* is *N*. It means that *parent(parent_broken(N)) == N* and a mini loop would occur if TTL for link *S-Y* was set to 1 (i.e. the position of *N* w.r.t. *S*).

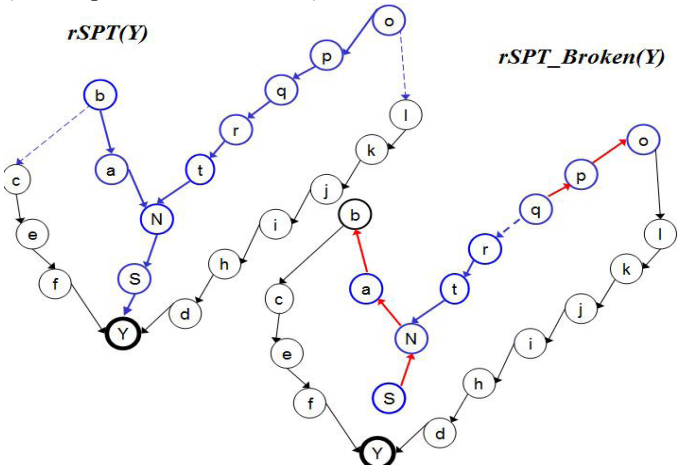


Figure 2: rSPT(Y) and rSPT_broken(Y)

The algorithm then proceed as follows: We start at hop count *1* in rSPT(Y) and initialize *ReqTTL* value to *0*. Consider all nodes in *Y-S* sub-branch at distance *h=1*: if loops are found at *h = 1*, *ReqTTL* will be set the hop count of the looping node that has higher distance from *S*, e.g. *TTL=2* corresponding to node *a*, when checking loop *N* at distance *h=1* (*N* looping with *a*). Now *h* is increased by *1* and we proceed with nodes in sub-branch *Y-S* of *rSPT(Y)* at distance *h=2*. If any node at hop *2* is detected to be involved in a loop, increase *ReqTTL* as described above. Keep on searching for loops till maximum hop depth of the affected branch. At this point, the *ReqTTL* value is to be set as the smallest[8] *TTL* value to be used for subsequent topology update generated by *S* upon *S-Y* link changes. In **Figure 2**, *ReqTTL = 6* due to looping nodes *p* and *o*. If the topology is denser, *ReqTTL* will be much smaller which is mostly the case as shown by simulations (Figure 2 is a pathological case used for illustration).

Note that while this simple DFS[9] processing on branch *Y-S* of *rSPT(Y)* takes place, we find successive values of "safe" *TTLs* (i.e. values for which no loop will occur). Values of *TTL=0, 1*, and *2* lead to loops. *TTL=3* is safe with no loop as both *b* and *r* are aware of the failure of link *Y-S*. Then, *TTL=4, 5* again induce loops (at node *q-p* and *p-o* respectively). Finally *TTL=6* is safe (and corresponds to the maximal depth of the affected branch, which stops the algorithm). To summarize, *TTL=3* could be considered as a "safe" (loopless) alternative. However, as introduced earlier, we must remark that we used an approximate approach were only the most common loops (occurring on nodes on *rSPT(Y)* and for destination *Y*) are seeked. However, more complex loop scenarios can occur (for destinations behind Y for instance and so forth). Obviouly, simulations show that the larger the chosen TTL value, the smaller the probability of occurrence of loops. We have thus chosen to stop the loop detection process only when reaching the maximum branch depth (and not as soon as a the first loop-less TLL value is found). We will see from the simulation results detailed in the next section approach that the probability of mini-loop occurrence is reduced to a negligible level using the proposed approach. This is achieved with a complexity order of $O(m.n^2)$ (i.e. one *rSPT_broken(x)* for each neighbour x).

MINI-Loops Psuedocode

```
ReqTTL ← 0;
procedure MINI-LOOPS ( )
    Calculate rSPT(Y)
        // already available as each node already has its own SPT. There is no need to calculate it.
    Calculate rSPT_broken(Y)
        // calculate reverse SPT rooted at Y considering link S-Y broken
        //For all nodes on branch S-Y of rSPT(Y) do
    for hop=1:maxhop (rSPT(Y))
        list[x] ← all nodes at distance h
            if | parent (parent_broken(x))= = x |
                Temp_TTL ← max(distance( parent(x)) ,distance ( parent_broken(x) ) )
                        //distance of these nodes in original SPT(S)
                If | ReqTTL < Temp_TTL |
                    ReqTTL ← Temp_TTL;
    End for loop
end procedure
```

IV. Simulation Results

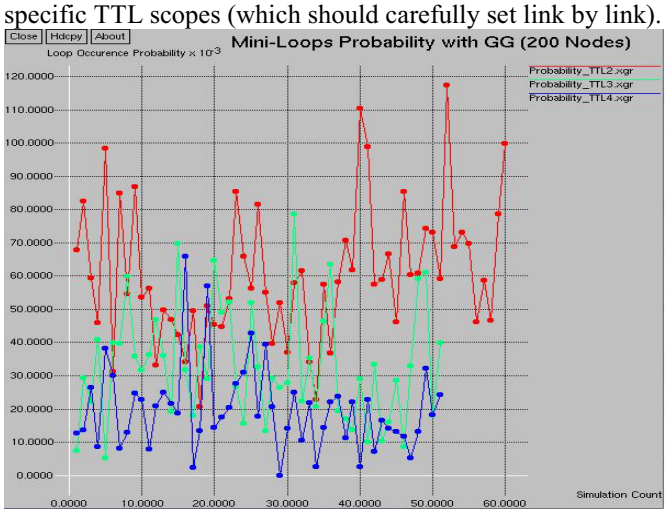
All the simulations are performed in Matlab. The first aim was to a) determine the occurrence probability of boundary loops of various scopes in different topologies, and b) determine how the destinations get affected with these loops. For this study, topologies are generated from Relative Neighbourhood Graphs (RNG) to Gabriel Graphs (GG) and Delaunay Triangulations. Gabriel graphs have an average degree of 4 where as RNG topologies are sparser with an average degree of 2.56. Delaunay triangulation gives denser topologies with average degree of 6. 50 random topologies each with an average 200 nodes were generated for each graph type.

First we analyzed the occurrence probability of mini-loops and their effect when a TTL of 2 (as specified in OFLSR) is used in RNG, GG and Delaunay topologies. Loops are detected by exhaustive comparisons of routing tables (before and after a random link failure), computed by Floyd-Warshall algorithm. RNG topology obviously being the most sparse, loops occur in almost 50% of link breakages making on average 25% destinations inaccessible. 7% of link breakages in GG topology can result in mini-loops at TTL boundary, making up to 4.76% destinations inaccessible. Delaunay graphs are generates a triangulation hence no mini loop can ever occur, as verified in our simulation. In this case, if some node S detects S-Y link breakage it always has a downstream neighbour node X to which it can reroute its traffic without creating loops.

Since GG topologies give relatively sparse topologies, it was chosen to analyse mini-loops effects and how the proposed

[8] When global TTL is not used, depending on FishEye configuration.
[9] DFS: Depth First Search

algorithm performs on them. Figure 3 shows the effect of using different OFLSR TTL scopes over 50 simulations. It clearly shows that mini-loops occur quite regularly for all TTL values, though they occur less often for larger TTL values. With TTL 3, about 3.3% of link breakages in a network create mini-loops loosing access to 3.7% destination nodes. In short, we see that if OFLSR is to be used in such *sparse* topology, proper care should be taken in sending TC messages with specific TTL scopes (which should carefully set link by link).



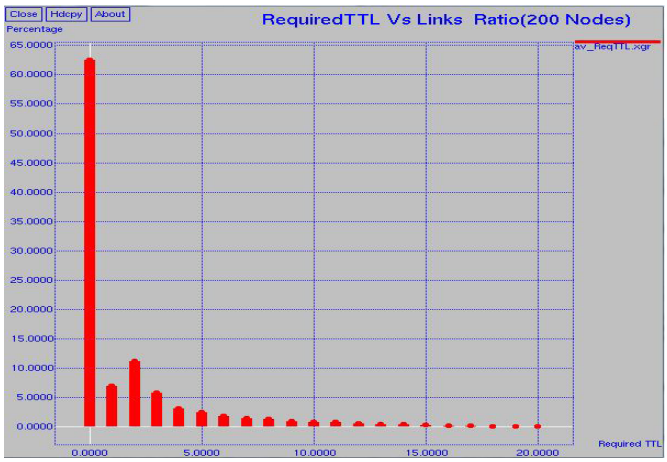


Figure 4: Required TTL vs. Link Ratio (200 Nodes)

Figure 4 shows the results obtained by the proposed algorithm in *GG* topologies. It represents the percentage of links requiring various *TTL* values ranging from 1 to some higher value depending upon topology. 62.5% of link breakages never result in a mini-loop (*TTL=0*). This occurs whenever the neighbourhood of the failing link is dense enough, i.e. when node *S,* detecting the *S-Y* link loss, has a downstream neighbour N_d which is carrying traffic towards *Y* and its descendents without using *S*. Remark that such neighbour N_d of *S* is referred to as a *Loop Free Alternate* in IP Fast Reroute techniques[10]. Also, link breakage does not cause any transient loops in cases when *S-Y* link is the only available path to access *Y* and other routers through *Y*. We are thus able to identify that such link breakages do not result in any transient loop even without any topology update, which means we could even **decrease** OFLSR overhead in may cases. About 7% of link breakages require a TC message with a TTL 1 and 11.2% of link breakages require a TTL of 2. This percentage then goes on decreasing with the increase in Required TTL value. We can observe that almost 1% of link breakages may require a TTL of as high as 10.

Finally, we should remember that the TTL values shown are computed with an approximate method, only detecting the most probable loops (i.e. lying on *rSPT(Y)* with Y as destination). However, using these values reduces the loop occurrence to a minute value of 0.066% in these simulations (w.r.t. to 6% with default OFSLR with minimal TTL set to 2).

[10] See Works on IP Fast Re-Route at the IETF Routing Area Working group for more details.

V. Conclusion

In this paper we discussed the potential interest of OFSLR for use in wireless mesh networks but identified that this protocol can cause long-living routing loops in sparse topologies. We then analyzed the occurrence and impact of such mini-loops and discussed potential solutions. An approximate, low-complexity algorithm eliminating most routing loops occurring at fisheye scope boundary was provided. Simulations on some random graphs highlighted the efficiency of the approach. Further works are however still needed. More exhaustive simulations should be conducted in order to better analyze the flooding overhead of such mechanisms. A formal analysis of the rare loops that are not avoided by the proposed mechanism is required, and potential solutions should be investigated. We believe that this study validated the approach of setting up "safe" TTL value for topology updates link by link in order to avoid mini-loops. We have also showed that a simple heuristic with low complexity can compute such values with excellent reliability.

VI. References

[1] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot: Optimized Link State Routing Protocol, RFC 3626, version 11, IETF, October 2003.

[2] G Pei, M Gerla, TW Chen, Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks, Communications, 2000. ICC 2000. 2000 IEEE International Conference

[3] E.Johansson, K.Persson, M. Skold and U. Sterner, An Analysis of the Fisheye Routing Technique in Highly Mobile Ad Hoc Networks, Vehicular Technology Conference, 2004.

[4] Thomas Heide Clausen, "Combining Temporal and Spatial Partial Topology for MANET routing – Merging OLSR and FSR", in Proc. of IEEE WPMC'03, Yokosuka, Japan. October 2003.

[5] C Adjih, E Baccelli, T Clausen, P Jacquet, Fish Eye Scaling Properties, IEEE Journal of Communication and Networks (JCN), 2004

[6] Jawei Chen, YZ Lee, D Maniezzo, M Gerla, Performance Comparison of AODV and OFLSR in Wireless Mesh Networks, IFIP Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), 2006

[7] Dang Nguyen & Pascal Minet, Scalability of the OLSR Protocol with the Fish Eye Extension, Networking, 2007. ICN'07. Sixth International Conference

[8] H. Mansoor Ali, A.M. Naimi, A. Busson, V. Vèque: An efficient link management algorithm for high mobility mesh networks. MOBIWAC 2007.

[9] U Hengartner, S Moon, R Mortier, C Diot - Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, Marseille, France, 2002

[10] P Francois, O Bonaventure, Avoiding Transient Loops during IGP convergence in IP networks, INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies