A Simple Algorithm for Fault-Tolerant Topology Control in Wireless Sensor Network

Jianhui Zhang*, Jiming Chen*, Yu Wang†, Yang Xiao‡ and Youxian Sun*
*State Key Lab of Industrial Control Technology, Zhejiang University, P.R.China
Email: {jhzhang,yxsun}@iipc.zju.edu.cn; jmchen@ieee.org
†Department of Computer Science, University of North Carolina at Charlotte, USA
Email: wangyu@ieee.org
‡Department of Computer Science, The University of Alabama, USA
Email: yangxiao@ieee.org

***Abstract*—To preserve network connectivity is an important issue especially in wireless sensor network, where wireless links are easy to be disturbed and tiny sensors are even easy to fail accidently. Therefore, it is necessary to design a fault-tolerant network. A feasible method is to construct a k-vertex connected topology. In this paper, we consider k-connectivity of wireless network and propose a simple global algorithm (GAFT$_k$) which preserves the network k-connectivity and reduces the maximal transmission power (TP). The average degree expectation of the topology generated by GAFT$_k$ is $O\left(\frac{(k+3)^2}{4}\right)$. Based on GAFT$_k$, we propose a simple local algorithm (LAFT$_k$) which preserves k-vertex connectivity while maintaining bi-directionality of the network. Simulation results show that GAFT/LAFT have better performance than other current fault-tolerant protocols.**

I. Introduction

Fault tolerant topology control (TC) is important in wireless networks, especially in wireless sensor network (WSN), since sensor nodes and wireless links are easy to fail and power and computational resource is limited. Many topology control algorithms have been designed to maintain network connectivity while reducing energy consumption and improving network capacity [1]. A few of them have considered the fault-tolerance [2], [3], like CBTC$_k(\alpha)$ [4].

The CBTC$_k(\alpha)$ is an extension of CBTC from [5] to provide fault tolerant topology. In CBTC(α), every vertex increases its transmission power (TP) until either the maximum angle between its two consecutive neighbors is at most α (a constant depended on k) or its maximal power is reached. Therefore, the TP would adjust to a excessive value.

Fault tolerant Local Spanning Sub-graph (FLSS$_k$) can preserve k-connectivity while maintaining bi-directionality [6]. It is based on a min-max optimal centralized algorithm, Fault tolerant Global Spanning Sub-graph (FGSS$_k$). Whether FGSS$_k$ is k-connected that need be tested by using network flow techniques (NFT) [7], which results in much high complexity and cost of the algorithm. Furthermore redundant edges would be added to the topology when FLSS$_k$ locally constructs a subgraph.

There have been several research efforts recently on theoretically studying the necessary condition for k-connected topology [8] [9] and devising approximation algorithms to construct such topologies [4]. All of above fault-tolerant topology control algorithms either use unrealistic assumptions (i.e. the locations of all nodes and the exact distance among them are known) or involve complex calculation.

In this paper, we proposal two TC algorithms with simple calculation to preserve the network k-connectivity. The main contributions of this paper include: (a) the topologies constructed under GAFT$_k$ and LAFT$_k$ preserve the network k-connectivity; (b) the time complexity of both GAFT$_k$ and LAFT$_k$ is low, so that nodes with limited computational and power capacity can still afford these algorithms efficiently; (c) the resulting topology can be converted into one with only bidirectional links.

The rest of the paper is organized as follows. Section II introduces our models and assumptions. Section III presents GAFT and its performance analysis, while Section IV gives its distributed implementation LAFT. Simulation study on both GAFT and LAFT is provided in Section V. Section VI concludes the paper.

II. Models and Assumptions

Assume that the nodes in network are uniformly deployed in a $c \times c$ area. Each node has an omni-directional antenna, which can adjust its TP discretely. Its transmission radius is denoted as r and its maximal value is denoted as r_{max}. An undirected simple graph $G = (V, E)$ is applied to analyze our TC algorithm, where V is the set of all nodes in the network and E is the edge set. $|V| = n$ and $|*|$ is the size of the set $*$. $d(N_u, N_v)$ denotes the received signal strength indication difference (RD) between nodes N_u and N_v. So GAFT/LAFT do not depend on any radio propagation model. A unique address *ID* is assigned to each node. The one-hop neighborhood of a node N_u, denoted as $H^1_{N_u}$, is the set of

This work is supported by the Nature Science Foundation of China under Grant 60604029, 60702081, Nature Science Foundation of Zhejiang Province under Grant Y106384,Y107309, Joint Funds of NSFC-Guangdong under Grant U0735003, the SRFDP under Grant 20050335020, the R&D Program of Zhejiang Province under Grant No.2007C31038 and 111 Projects under Grant B07031; The work of Yu Wang was supported in part by the National Science Foundation under Grant NeTS-NOSS-0721666 and by funds provided by the University of North Carolina at Charlotte;Prof. Xiao's work was partially supported by the National Science Foundation under the grant CNS-0716211.

neighbors that N_u can directly reach by the maximal TP. We also assume that $G(V,E)$ is k-connected.

Receiver can know the transmission power P_t of transmitter, which is beforehand included in the ***hello*** message. And the receiving power P_r can be calculated from RSSI tested at the receive antenna. So RD = $P_t - P_r$ and it can indicate the distance d between two nodes.

III. Global algorithm for Fault Tolerant–GAFT

The basic idea of GAFT is growing the k-connected subgraph from a single node step by step and in each step a node inside the subgraph invites new nodes joining the subgraph. There are two criterias for the selection of new nodes: (1) adding the new nodes (or links) can improve the connectivity of the subgraph to satisfying k-connected requirement; (2) if the subgraph already satisfies connectivity requirement or multiple links can improve the connectivity, select those closet nodes. In GAFT, V_k is the set of nodes currently in the subgraph, while V_f is the set of those who already connect to at least k neighbors in the subgraph (thus no need to be considered again). Assume GAFT begins from node N_0. Initially, $V_k = N_0$ and $V_f = \varnothing$. Then N_0 connects with its k closest neighbors. If defining N_i and its closet connected neighbors as a set S_{N_i}, we can obtain S_{N_0}. Then we merge S_{N_0} into V_k, i.e. $V_k := \cup S_{N_0}$. Since N_0 already has k neighbors in V_k, we then include N_0 in V_f. If all node are in V_f, i.e. $V = V_f$, GAFT terminates. Next, node N_i ($N_i \in V_k \cap \overline{V_f}$) tries to connect with its k closest neighbors. Notice, N_i selects the neighbors who can improve the connectivity of the subgraph (nodes in V_k) with higher priority. If $S_{N_i} \subseteq V_k$, GAFT has to find another node, N_j ($N_j \notin V_k$), which is closest to nodes in V_f. Notice this can guarantee the subgraph (V_k) is growing. $V_k := \cup\{N_j\}$ and $V_f := \cup\{N_i\}$. If $S_{N_i} \nsubseteq V_k$, N_i only selects those neighbors in V_k (thus $S_{N_i} = S_{N_i} \cap V_k$). Then GAFT selects the remaining $k - |S_{N_i}|$ from those nodes who are not in V_k and are closest to nodes in V_f. Assume these nodes are N_k. GAFT let $V_k := \cup\{N_k\}$. Notice that in this case N_i would not be included in V_f, since it still need $k - |S_{N_i}|$ connections. This whole process will terminate until all nodes in V_f. The detailed algorithm is given as Algorithm 1. Theorem 1 guarantees the G_k obtained by GAFT is k-connected.

Theorem 1. *Let G=(V, E) be a graph on V=$\{N_0, N_1, \cdots, N_{n-1}\}$ with n≥k+1. G$_k$(V, E$_k$) is obtained by GAFT. If G(V, E) is k-connected, then G$_k$ is also k-connected.*

Proof: If $n = k + 1$, both of G and G$_k$ are complete graphes. When $n \geq k+2$, suppose there is a set $C \subset V$ with $|C| \leq k-1$ so that $G_k - C$ is disconnected. Let V_1 and V_2 be two distinct connected components of $G_k - C$. V_1 and V_2 are still k-connected since G is. Suppose nodes in V_1 join V_k earlier than those in V_2. Vice versa.

Before any node in V_2 joins V_k, $V_k \subseteq V_1 \cup C$. Since GAFT always invites new node into V_k, there must be one moment at which a node $N_i \in V_2$ is invited into V_k. If $S_{N_i} \subseteq V_k$, At least one neighbor of N_i is in V_1 since $|C| < k$. N_i must connect to the neighbor then V_1 and V_2 are connected. The theorem is proved. Therefore there must be $S_{N_i} \nsubseteq V_k$. Then GAFT runs $S_{N_i} = S_{N_i} \cap V_k$ and $S_{N_i} \subseteq C$ (otherwise V_1 and V_2 are connected). Next N_i connects to $k - |S_{N_i}|$ neighbors closet to V_k. Since these $k - |S_{N_i}|$ neighbors respectively connect to nodes in V_k, there are still k links between V_k and N_i. Because $|C| < k$, there must be at least one link between N_i and V_2. V_1 and V_2 are connected.

If C separates G_k into more than two disjoint components, the above proof can be applied to every pair of components then the k-connectivity of G_k can be obtained. ■

Algorithm 1: GAFT

```
   Input  : $G(V,E)$, a simple connected graph;
   Output: $G_k(V,E_k)$, a $k$-connected spanning subgraph of $G$;
1  begin
2      $E_k := \varnothing$; $V_k := S_{N_0}$; $V_f := \varnothing$;
3      while $N_i \in V_k$ & $N_i \notin V_f$ do
4          $N_i$ selects its $k$ "closest" neighbors, i.e. $S_{N_i}$;
5          if $S_{N_i} \subseteq V_k$ then
6              Find a node $N_j \notin V_k$, which are closet to a node in $V_f$;
7              $V_k := \cup\{N_j\} \cup S_{N_i}$;
8              $E_k := \cup\{e(N_i,N_l)\}$, $N_l \in S_{N_i}$;
9              $V_f := \cup\{N_i\}$;
10             Continue;
11         else
12             $S_{N_i} = S_{N_i} \cap V_k$;
13             Connect to $k - |S_{N_i}|$ neighbors closest to nodes in $V_k$;
14     Adjust all nodes' TP according to algorithm 2;
15 end
```

GAFT only needs RD. The number expectation of nodes in the maximal cover range of a node is $n_e \triangleq n\pi(\frac{r_{max}}{c})^2$. So each node only needs to communicate with n_e nodes, less than n. It would reduce much calculation.

To adjust the TP, several heuristic algorithms have been proposed, such as LINT and LILT [10]. These algorithms also can be used here but we propose a quite simple one as shown in Algorithm 2 in order to to reduce the number of communication messages. Each node can obtain the RDs from the ***hello*** messages ofcits neighbors . According to these tested distance, each node arranges its neighbors into a non-decreasing order and then calculate requisite TP level according to Algorithm 2. An ***affirm*** message is ultimately broadcast to inform its neighbors of the final TP level. Each node only broadcasts two messages, which decreases power consumption and interference.

Theorem 2. *After running TPA, G$_k$ is bi-directionally connected and k-connected only if G is.*

In fact, TPA only adds some edges on G_k as shown in Algorithm 2 and makes G_k bi-directionally connected. So TPA doesn't decrease the connectivity of G_k.

Before giving out Theorem 3, we show a fact. Suppose that node N_u has m ($m \geq k$) neighbors in its maximal transmission range. The probability (named as $P(m,l)$) that N_v directly connects with l neighbors out of the m ones is $P(m,l) = \binom{m}{l}/\binom{n}{l}$ since there are totally n nodes in the network.

Algorithm 2: TP adjustment–TPA

```
Input  : A set $V$ containing $n$ nodes;
Output : $G(V,E)$, a simple connected graph $G$;
1  begin
2      for $N_i$,$i \leftarrow 0$ to $n-1$ do
3          Broadcast a Hello message with maximum TP level $P_{tmax}$.
4      if $N_i$ receives Hello messages then
5          Detect $P_{rN_j}$ of these messages from $N_j,\ j \in N_i^1$ && $j \neq i$ and calculates $d(N_i,N_j)$;
6          Sort its neighbors in non decreasing list $L_{N_i}$;
7      if $N_i$ need communicates with the first k nodes in $L_{N_i}$ then
8          Calculate: $P_{tN_i} = P_{tmax} + P_{tmax} - P_{rN_k}$ ;
9          Broadcast $P_{tN_i}$ by an Affirm message;
10     Receive the Affirm messages from its neighbors and know the $P_{tN_j}$ of its neighbors;
11     Set its adjusted TP level as $max\{P_{tN_i}, P_{tN_j}\}$;
12 end
```

Algorithm 3: LAFT

```
Input  : $G(V,E)$, a simple connected graph;
Output : $G_k(V,E_k)$, a $k$-connected spanning subgraph of $G$;
1  begin
2      $E_k := \varnothing$; Each node $N_u$ sets sets $V_k^{N_u} = S_{N_u}$ and $V_f^{N_u} = \varnothing$;
3      Each node $N_u$ collects the information about $H^1_{N_u}$ and $H^2_{N_u}$;
4      Finds the $k$ closest neighbors, i.e. $S_{N_u}$;
5      while $N_i \in V_k^{N_u}$ && $N_i \notin V_f^{N_u}$ do
6          if $S_{N_i} \subseteq V_k^{N_u}$ then
7              Find another neighbor $N_j$ ($N_j \notin V_k^{N_u}$) closet to $V_k^{N_u}$;
8              $V_k^{N_u} := \cup\{N_j\}$; $V_f^{N_u} := \cup\{N_i\}$;
9          else
10             $S_{N_i} := \cap V_k^{N_u}$;
11             Connect with $k - |S_{N_i}|$ neighbors closet to $V_k^{N_u}$.
12         $E_k := \cup\{e(N_i,N_l)\}$, $N_l \in S_{N_i}$;
13     Adjust all nodes' TP according to algorithm 2;
14 end
```

Theorem 3. *After running GAFT, the average degree expectation of G(V, E) is* $O\left(\frac{(k+3)^2}{4}\right)$.

Proof: Suppose N_0 firstly starts GAFT, so $V_k = S_{N_0}$ and there are $k+1$ nodes in V_k. When N_i connects with its k nearest neighbors, it may add extra neighbors to other nodes in V_k so the "extra degree" is created. If we consider the "extra degree" to be in the degree of N_i, the total weight of G_k would not change. The degree expectation of N_i is $E(d_i) = k + \sum_{j=1}^{k} j \cdot P(|V_k|, j)$. The first node N_0 does not add extra neighbors because V_k is empty when it joins. So the average degree expectation is

$$\begin{aligned} E(d) &= \frac{1}{n}\sum_{i=1}^{n} E(d_i) = \frac{1}{n}[k + \sum_{i=2}^{n} E(d_i)] \\ &\leq \frac{1}{n}[nk + (k-1)\sum_{j=1}^{k} P(k+1,1)j + \sum_{l=k+1}^{n}\sum_{j=1}^{k} P(l,1)j] \\ &= \frac{k^2+5k}{4} + \frac{k^2+n-k-2}{n^2} \leq \frac{k^2+5k}{4} + \frac{k^2+n}{n^2} \\ &\leq \frac{k^2+5k}{4} + \frac{(n-1)^2+n}{n^2} \leq O\left(\frac{(k+3)^2}{4}\right) \end{aligned} \qquad (1)$$

Notice that V_k grows from $k+1$ to n when nodes join it one by one. ■

Theorem 3 says that each node needs $k + \frac{1}{n}(k+2)O(n-k)$ neighbors on average. The result is close to the Penrose's proof that the node degree is much close to the connectivity when the number of nodes is larger enough [9]. It indicates that a k-connected subgraph can be obtained when each node uses its local information to link its neighbors properly.

IV. Local algorithm for Fault Tolerant–LAFT

In this section, a distribution algorithm LAFT to implement the GAFT is given out. LAFT takes $O(n_e \log n_e)$ steps and totally $O(n)$ messages to construct a k-connected topology.

In Algorithm 3, the two-hop neighborhood of a node N_u, denoted as $H^2_{N_u}$, is the set of nodes in $H^1_{N_u}$ can directly reach by using the maximum TP. Each node N_u has two sets $V_k^{N_u}$ and $V_f^{N_u}$. The former is the set of nodes currently in the subgraph, while the later is the set of nodes who already connect to at least k neighbors in the subgraph (thus no need to be considered again). Each node also uses two messages to obtain the information about one-hop and two-hop neighbors. These messages contain the ID and P_t among nodes. Subsequently each node runs Algorithm 3 to determine which neighbors to connect with. After that, nodes run Algorithm 2 to adjust its TP level. We can easily find that the message complexity of the whole network is $O(n)$ by running Algorithm 3.

Before proving the topology constructed by LAFT is k-connected, we firstly prove Theorem 4. In $G = (V,E)$, suppose there are a node N_u and its two-hop neighbors N_v, $N_v \in H^2_{N_u}$. By defining a node set $V_{N_u} = \{N_u\} \cup H^2_{N_u}$ and a edge set E_{N_u}, which contains all edges connected with the nodes in V_{N_u}, we can obtain a local graph $G_{N_u} = (V_{N_u}, E_{N_u})$. After running LAFT, a k-connected local subgraph is obtained. The k-connected local subgraph is defined as $L_{N_u} = (V_{N_u}, E^k_{N_u})$.

Theorem 4. *If the connectivity of* G_{N_u} *is* m*, then the connectivity of* L_{N_u} *is* k *when* $m > k$ *and is* m *when* $m \leq k$.

Proof: When LAFT runs in the local graph G_{N_u}, it becomes GAFT. So LAFT can surely construct a m-connected subgraph according to Theorem 1 when the connectivity $m \leq k$. LAFT can construct a k-connected topology when $m = k$. $m > k$ is a more sufficient condition. Therefore LAFT can surely construct a k-connected topology when the connectivity $m > k$. ■

Theorem 5. *Let* $G = (V,E)$ *be a graph on V=*$\{N_0, N_1, \cdots, N_{n-1}\}$ *with n≥k+1. L*$_k$*(V, E*$_k$*) is obtained by LAFT. If G(V, E) is k-connected, then L*$_k$*(V, E*$_k$*) is also k-connected.*

The proof is very similar to that for FLSS [6].

Theorem 6. $L_k(V, E_k)$ *is bi-directionally connected.*

It is very similar to Theorem 2 because LAFT also runs TPA, which adds some edges for those unidirectional edges.

From Algorithm 3, we can obtain the complexity of LAFT is $O(n_e \log n_e)$ because $|S_{N_u}| \leq n_e$. That means each node can finish the calculation of LAFT by taking at most $O(n_e \log\, n_e)$ steps. Li uses NFT to construct topologies in both the FGSS_k and FLSS_k algorithms [6]. The time complexity of FGSS_k is $O(m(n+m))$ and can be improved to $O(m)$ for $k \leq 3$, where m is the number of edges. For a k-connected topology, $m \geq \frac{kn}{2}$. It is obvious that NFT makes FGSS_k and FLSS_k much complex.

V. Simulation

The simulation for performance evaluation of GAFT/LAFT is given out with the Omnet++ simulation tool. Comparison with other fault-tolerant protocols will be given.

The radio receive sensitivity is at least -98 dBm. The output power levels of radio ranges from -20 dBm to 5 dBm in steps of 1 dBm. $r_{max} = 261.195\ m$. $c \times c = 1000 \times 1000\ m^2$. Variable numbers of nodes from 100 to 400 are deployed in the area in steps of 10 or 20. Each data point is the average of 50 simulation samples.

Logical neighbors–degree. Logical neighbors refer to those whom one node has direct linkages with. Although one node has more accesses to other nodes with bigger transmission radius, it incurs more interference. Degree is a good indication of the link access. Fig.1 shows the average logical neighbors

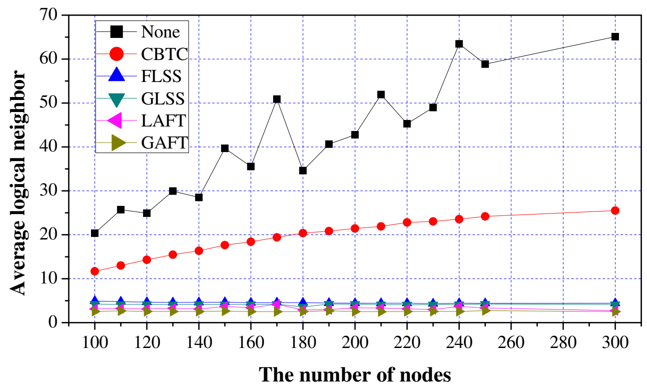
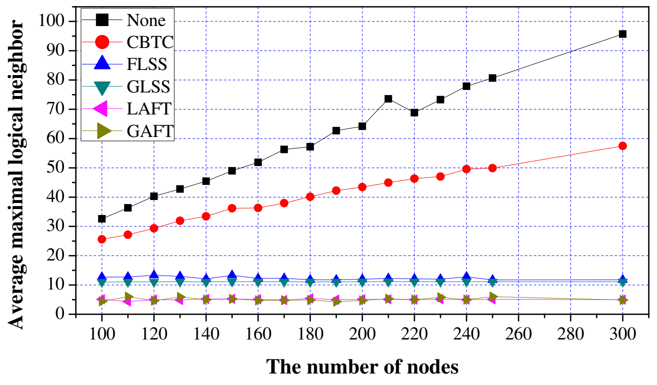
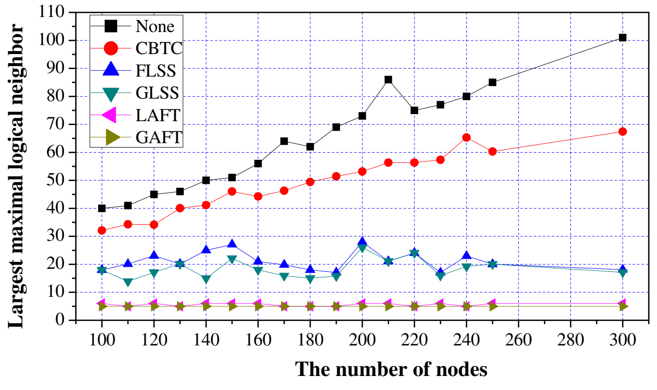


Fig. 1. Comparison of None, CBTC($\frac{\pi}{3}$), FGSS_2, FLSS_2, GAFT_2 and LAFT_2 with respect to average degree ($k = 2$)

of the topologies constructed by $\text{CBTC}_k(\frac{\pi}{3})$, FLSS_2, GLSS_2, LAFT_2 and GAFT_2. When None indicates that no topology control is implemented, the average degree increases dramatically with the number of nodes. Those of None and $\text{CBTC}_k(\frac{\pi}{3})$ is much higher than both GLSS/FLSS and GAFT/LAFT and that of GLSS/FLSS is slightly higher than GAFT/LAFT. The average degree under $\text{CBTC}_k(\frac{\pi}{3})$ also increases with the number of nodes. The $\text{CBTC}_k(\frac{2\pi}{3k})$ has the coverage constraints on each individual cone, which makes each node create redundant linkages in order to keep at least one neighbor in each cone. And the redundant linkages increases with the node density. Furthermore, each node has at least $2\pi/\frac{2\pi}{3k} = 3k$ neighbors to keep the whole network k-connectivity. But GAFT/LAFT and GLSS/FLSS only link with visible neighbors. So their average degrees are very close to each other.



(a) Average maximal node degree



(b) Largest maximal node degree

Fig. 2. Comparison of None, CBTC($\frac{\pi}{3}$), FGSS_2, FLSS_2, GAFT_2 and LAFT_2 with respect to the maximal node degree ($k = 2$)

In Fig.2, the average maximal degree and the largest maximal neighbors are given out from the topologies derived under $\text{CBTC}_k(\frac{\pi}{3})$, FLSS_2, GLSS_2, GAFT_2 and LAFT_2. The value under $\text{GAFT}_2/\text{LAFT}_2$ are significantly smaller than those under None/$\text{CBTC}_k(\frac{\pi}{3})$ and slightly smaller than those under $\text{GLSS}_2/\text{FLSS}_2$. With increasing of the number of nodes, the value under $\text{GAFT}_2/\text{LAFT}_2$ almost change little and their performance improvement becomes more remarkable.

The proportion of logical neighbors in physical ones. In order to link with some neighbors, others is also included in the transmission range. So the inference occurs. All these neighbors are called physical neighbors. The links with logical neighbors are necessary to establish a topology, but interference on other neighbors should be as low as possible. Therefore the ratio between logical neighbors and physical ones is also an important parameters to indicate the interference, as shown in Fig.4. Most of the ratios of GAFT are close to 80 percentage while most of the ratios of LAFT are close to 10 percentage. These results are coincident with those in Fig.3. Topology under GAFT has better spatial reuse than that under LAFT.

We give out the physical neighbors under 2, 3 and 4-connectivity by GAFT/LAFT in Fig.3. The average physical neighbors of LAFT under different connectivity are slightly bigger than those of GAFT because GAFT knows the global

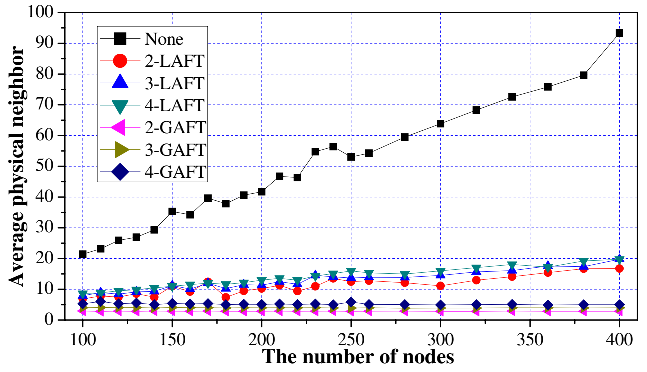


Fig. 3. Comparison of GAFT$_k$ and LAFT$_k$ with respect to the physical node degree ($k = 2, 3, 4$)

information. But LAFT can not know information out of two hops so some nodes may link some farer neighbors. Therefore the average physical degrees slightly increase as the number of nodes increases in Fig.3. Under different k, the physical neighbors do not change much. GAFT/LAFT are robust under different connectivity.

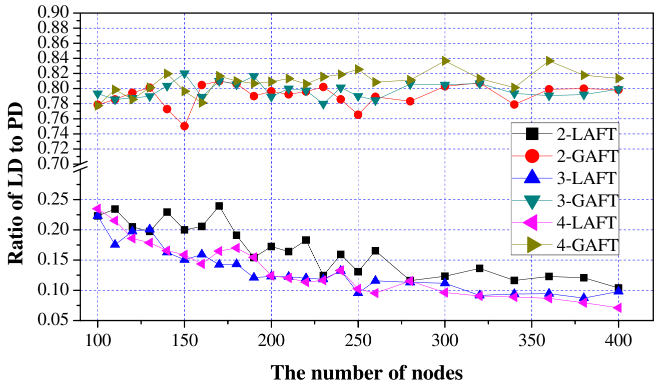


Fig. 4. Comparison of GAFT and LAFT with respect to the ratio of LD to PD, where LD and PD refer to logical degree and physical degree

TP level. In order to clearly show the results of our

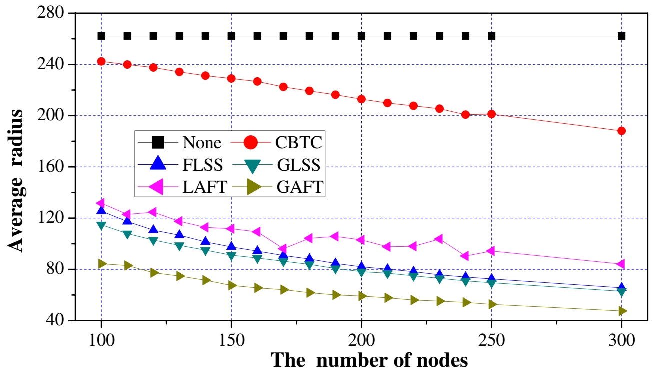


Fig. 5. Comparison of None, CBTC$_k(\frac{\pi}{3})$, FGSS$_2$, FLSS$_2$, GAFT$_2$ and LAFT$_2$ with respect to the average radius ($k = 2$)

algorithms, we theoretically transform TP into radius distance and compare None, CBTC$_k(\frac{\pi}{3})$, FGSS$_2$, FLSS$_2$, GAFT$_2$ and LAFT$_2$ when $k = 2$. Note that it doesn't mean that GAFT/-FALT assume the antenna pattern is a perfect disk although other algorithms do. In the Fig.5, the radiuses of GAFT$_2$ are obviously lower than those of None, CBTC$_k(\frac{\pi}{3})$ and LAFT$_2$ and slightly lower than those of GLSS$_2$ and FLSS$_2$. The radiuses of LAFT$_2$ are obviously lower than those of None and CBTC$_k(\frac{\pi}{3})$ and slightly higher than those of GLSS$_2$ and FLSS$_2$. But it does not decrease the performance of LAFT since GLSS/FLSS should examine the network connectivity before each edge is admitted to the topology, which would lead to great calculation and data transmission.

VI. Conclusion

In this paper, a k-connected TC algorithm GAFT is proposed to preserve k-fault tolerance. Then a distributed implementation LAFT of GAGT is presented to adapt to some self-organized wireless network. These two algorithms need simple calculation and result in k-connected topology.

We consider that the transmission range is irregular and variable and do not adopt UDG model. GAFT/LAFT use RD to determine the linkage relation with its neighbors. The real distance does not effect the two algorithm. The algorithm complexity of GAFT/LAFT is lower and they use only a few messages.

References

[1] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comp Surveys*, vol. 37, no. 2, pp. 164–194, 2005.

[2] F. Wang, M. T. Thai, Y. Li, X. Cheng, and D. Du, "Fault-tolerant topology control for all-to-one and one-to-all communication in wireles networks," *Transactions on Mobile Computing*, vol. 7, pp. 322–331, 2008.

[3] M. Cardei, S. Yang, and J. Wu, "Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 545–558, 2008.

[4] M. Bahramgiri, M. Hajiaghayi, and V. S. Mirrokni, "Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multi-hop networks," in *Proceedings of IEEE Eleventh International Conference on Computer Communications and Networks (ICCCN)*, 2002, pp. 392–397.

[5] L. Li, J. Y. Halpern, P. Bahl, Y. M. Wang, and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks," in *Proceeding of ACM Symposium on Principles of Distributed Computing (PODC), Newport, RI, USA*, 2001, pp. 264–273.

[6] N. Li and J. C. Hou, "FLSS: a fault-tolerant topology control algorithm for wireless networks," in *Proceedings of the 10th ACM/IEEE Annual International Conference on Mobile Computing and Networking (Mobi-Com), New York*. ACM press, 2004, pp. 275–286.

[7] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM Journal on Computing*, vol. 4, no. 4, pp. 507–518, 1975.

[8] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network," in *The Fifth ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MOBIHOC),Lausanne, Switzerland*, 2002, pp. 80–91.

[9] M. D. Penrose, "On k-connectivity for a geometric random graph," *Communications of the ACM*, vol. 15, no. 2, pp. 145–164, 1999.

[10] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *Proceedings of the IEEE Conference on Computer Communications, Tel Aviv, Israel*, vol. 2, 2000, pp. 404–413.