
# CONNECTIVITY-AWARE TOPOLOGY CONTROL WITH CYCLIC-LIKE STRUCTURES IN WIRELESS SENSOR NETWORKS

HUANG Zhiwei, ZHENG Zimu, LI Zhicheng and PENG Xinyi
School of Software Engineering
South China University of Technology, Higher Education Mega Center South, Guangzhou
Guangdong Province, China
Emails: zeedmood@hotmail.com

***Submitted: Aug, 15, 2014*** ***Accepted: Nov. 10, 2014*** ***Published: Dec. 1, 2014***

***Abstract- Connectivity offers the foundation for achieving required quality of service in all WSN applications. By the concept of cyclic-like topologies proposed and their property proved, we find a new searching method based on cyclic paths on nodes and their combination, which cuts down the cost of ensuring bi-connectivity in this problem. On the basis of graph and probability theory, our centralized connectivity-aware algorithm with cyclic-like topologies and computation of link reachability is proposed. Results of the experiments show that, our topology control algorithm could be more satisfactory than that of another two algorithms in different scales of networks.***

**Index terms*:* Wireless Sensor Network, connectivity, topology control, fault-tolerant, cyclic-like topologies, bi-connected.**

# I. INTRODUCTION

Wireless sensor networks are becoming increasingly common in recent years [1][2], and heading out to a wide variety of applications in the real world such as energy saving, object tracking and smart building issues[16-17]. The network nodes are expected to form a network in order to share data and coordinate their actions when participating in the execution of tasks [15].

Particularly, connectivity is a significant property of wireless sensor network, for a disconnected network is out of service. It is always at the heart of WSN design, analysis, and implementation as it provides the communication foundation for achieving desirable quality of service in all WSN applications [3]. In this paper, different from traditional work in sensor network which only focuses on traditional connectivity and interference power limitation, we introduce cyclic-like topologies and probability in this problem. Similarly as what was proposed in CR network [4], we also divide connectivity into topological connectivity and physical connectivity; but we modified the definition of the later one to adapt cases in wireless sensor network. In topological connectivity, bi-connectivity is a widely applied for design of fault tolerant wireless sensor networks [5]. Then our problem to address is as below:

Given a network in which every node is using the maximum power for transimition, the targeted physical connectivity of induced network, our task is to achieve bi-connectivity and targeted physical connectivity in this network; and our objective is to minimize the average transmitting power of nodes and the number of components.

No matter topological or physical connectivity has been studied in traditional wireless ad hoc networks in previous literatures, thus those related works are divided into these two categories.

Topological connectivity mainly focuses on the topologies of network, which contributes to overall robustness. 1-connectivity of ad hoc network is investigated in [6], while k-connectivity of a wireless ad hoc network is considered in [7][8][9]. Then a robust topology control algorithm was also proposed for a better fault tolerant WSN, which is usually made sure by k-connectivity [10]. Particularly, bi-connectivity as a basic requirement of k-connectivity, is a widely applied for design of fault tolerant wireless sensor networks [11][12].

Works showed above have two common characteristics and are called Routine Topological-Connectivity-Aware algorithms (RTCA), which all ensure the bi-connectivity of a certain

topology by checking the connectivity without one node which may have a heavier overhead and do not take physical connectivity into account. Physical connectivity in this paper refers to the probability of successful communication on links or paths which have transmission opportunity between nodes. In RTCA, the physical connectivity on a link or a path is simply treated as a variable in Boolean: connected (valued 1) or not (valued 0). However In practical environments, pairs of nodes can be not ideally fully connected but always with reachable lossy links; accordingly, better grained physical connectivity will allow a transmitter to not only connect more nodes but also produce a better topology. Moreover, the correlation of adjacent nodes is taken into account and lossy links are better utilized, thus the communication of network is becoming more efficient than before [13]. Both QoC (Quality of Communication, similar with physical connectivity) and QoS (Quality of Service) requirements are considered by the algorithm CSR in [14], which performs 250% better than another algorithm; but regardless of topological connectivity.

On one hand in these works, the bi-connectivity of a certain topology is traditionally ensured by checking the connectivity without each one node, which is exactly based on the definition but with relatively heavy overhead. In this method, when a node is removed, we need $O(n^2)$ to establish a BFS tree to check the connectivity of the remained network. Then since every one node should be removed once, the overall time cost of this traditional checking function could be as high as $O(n^3)$.

On the other hand, the number of topologies to check could be very large. Admittedly CSR may deal well with physical connectivity. however if certain requirements on topological connectivity are additionally considered in CSR, these and the origin requirements on physical connectivity often could not be both at the same time achievable on the whole network. In this case to acquire both two categories of requirements, the whole network has to be divided into various sub-networks. Then since the requirements are strict and the number of possible topologies is large, we need to search subsets of the whole network in a particular order. However, topologies do not have a strictly incremental bi-connectivity on nodes. That is to say, bi-connectivity has no strict correlation with the number of nodes. For example, if a 3-node topology is bi-connected, a 4-node topology based on it is may be or may not be bi-connected; if a 3-node topology is not bi-connected, a 4-node topology based on it still may be or may not be bi-connected. In the worst

case, we may have to check for all possible subsets in the network, whose number could be very large.

Thus in terms of the checking function and searching method, previous works still need to be improved to address our problem to achieve both topological and physical requirements. If we do not follow any rules to search for required topologies, there will be a number of sub-networks for us to check, which means we have to call the heavy function to check the bi-connectivity in various sub-networks again and again when considering requirements on both connectivity, which probably brings unbelievably high cost on time and energy.

Therefore in this paper, different from traditional work in connectivity of WSN which focuses on only topological or only physical connectivity, we manage to balance these two requirements.

## II. PRELIMINARY

As mentioned above, previous works still need to be improved for a better network achieving both topological and physical requirements. The challenge of the former requirement lies in the high cost to go through possible topologies and pick up the best one [6-12]; at the same time the latter requirement is always ignored but exists in engineering [13, 14].

By proposing cyclic-like topologies, we omit the heavy checking function and find a new searching method based on the minimum cyclic paths on nodes and their combination, which cuts down the cost of ensuring bi-connectivity in this problem and ensure the topological requirement; and by utilizing the concept of probability like previous work, we make physical connectivity better grained. A Connectivity-aware Topology control algorithm with Cyclic-like Structures (CTCS) is then proposed to ensure the requirements of both topological and physical connectivity. CTCS uses a topological optimized algorithm based on Cyclic-like topologies to cut down the cost of the traditional checking function of bi-connectivity. First we introduce this new definition and prove its property of bi-connectivity.

**Definition 1** Cyclic-like topologies

It refers to a kind of topologies meeting any one of the two conditions:

(1) It contains only one single node or two connected nodes;

(2) For every node in it, there exists at least one path which is called cyclic path that starts from it and also ends at it. A cyclic path contains more than two nodes and no overlapping nodes besides the starting/end node. There is more than one node overlapped between cyclic paths.

We next show that the cyclic-like topology is equal to the bi-connected topology by respectively proving the necessity and sufficiency of this proposition. We have the following lemma first.

**Lemma 1** In a cyclic-like topology, there are at least two node-disjoint paths between two nodes $v_i, v_j$ which belong to two different cyclic paths $P_i, P_j$.

Proof: According to the Definition 1, every cyclic path overlaps with at least another cyclic path to two or more nodes. Below we prove Lemma 1 with mathematical induction on different (direct and indirect) overlapping cases of $P_i, P_j$.

If there is not any other cyclic paths between $P_i$ and $P_j$, and $P_i$ directly overlaps with $P_j$ to two or more nodes, i.e., $v_1, v_2$. Since $v_1, v_2$ both belong to different cyclic paths, there must be such a path $p_{12}$ between $v_1, v_2$ without $v_i, v_j$. Then the path between $v_i$ and $v_1$ belongs to $P_i - p_{12}$ is denoted as $p_{i1}$. Similarly we have $p_{i2}, p_{j1}, p_{j2}$. Next, the path $p_{i1j}$ without duplicated nodes could be obtained by connecting $p_{i1}$ and $p_{j1}$ with endpoint $v_1$. Similarly, we can get the path $p_{i2j}$ by connecting $p_{i2}$ and $p_{j2}$ with endpoint $v_2$. Because without $v_i$, $p_{i1}$ does not overlap with $p_{i2j}$ and without $v_j$, path $p_{j1}$ does not overlap with $p_{i2j}$, obviously without $v_i$ and $v_j$, $p_{i1j}$ will not overlap with $p_{i2j}$. Thus there are at least two node-disjoint paths $p_{i1j}$ and $p_{i2j}$ between two nodes $v_i, v_j$ when $P_i$ directly overlaps with $P_j$.

If there is at least one cyclic path between $P_i$ and $P_j$, and $P_i$ indirectly overlaps with $P_j$ (to two or more nodes), i.e., $P_i$ overlaps with $P_1$ to at least two nodes $v_{i11}, v_{i12}$, $P_1$ overlaps with $P_2$ to two or more nodes $v_{121}, v_{122}$, $P_2$ overlaps with $P_3$ to not less than two nodes $v_{231}, v_{232}$, …, $P_{k-1}$ overlaps with $P_k$ to at least two nodes $v_{(k-1)k1}, v_{(k-1)k2}$, and finally $P_k$ overlaps with $P_j$ to not less than two nodes $v_{kj1}, v_{kj2}$, then for mathematical induction we assume that there are at least two node-disjoint paths between any two nodes $v_i, v_k$ which belong to two different cyclic paths $P_i, P_k$ and below we prove that there are at least two node-disjoint paths between any two nodes $v_i, v_j$ which belong to two different cyclic paths $P_i, P_j$.

Path $p_{kj12}$ denotes the path between $v_{kj1}, v_{kj2}$ without $v_j$ and overlapping nodes of $P_{k-1}$ and $P_k$, *i.e.*, $v_{(k-1)k1}, v_{(k-1)k2}$; similarly we define the one between $v_{(k-1)k1}$ and $v_{(k-1)k2}$ as $p_{(k-1)k12}$. Path $p_{(k-1)k1}$ could be obtained by collect the nodes between $v_{(k-1)k1}$ and $v_{kj1}$ in $P_k - p_{(k-1)k12} - p_{kj12}$; similarly $p_{(k-1)k2}, p_{kj1}, p_{kj2}$ could also be achieved. Like how we got $p_{i2j}$ in previous proof, we can get the path $p_{(k-1)j1}$ by connecting $p_{(k-1)k1}$ and $p_{kj1}$ with

endpoint $v_{kj1}$, so as the path $p_{(k-1)j2}$. Again, these two paths are not overlapped with each other except their endpoint $v_j$.

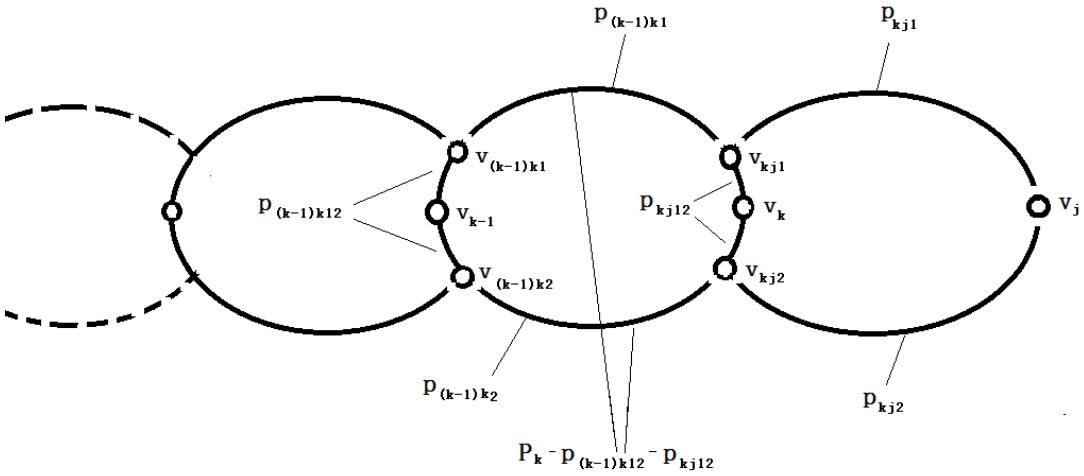


Fig.1 A sketch for indirectly overlapped cyclic paths

According to our assumption, there are also at least two node-disjoint paths between any two nodes $v_i, v_{(k-1)k1}$ which belong to two different cyclic paths $P_i, P_k$; we define one of the two paths which does not overlap with $p_{(k-1)k12}$ except endpoint $v_{(k-1)k1}$ as $p_{i(k-1)k1}$; It can connect to $p_{(k-1)j1}$ with endpoint $v_{(k-1)k1}$ and finally $p_{ij1}$ are obtained. Similarly, we can get another path $p_{ij2}$ by analyzing paths on $v_{(k-1)k2}$. Obviously $p_{ij1}$ and $p_{ij2}$ are not overlapped with each other except endpoints $v_i, v_j$, thus there are at least two node-disjoint paths $p_{ij1}, p_{ij2}$ between $v_i, v_j$ when $P_i$ indirectly overlaps with $P_j$.

Therefore in a cyclic-like topology, there are at least two node-disjoint paths between two nodes $v_i, v_j$ which belong to two different cyclic paths $P_i, P_j$.

**Theorem 1** A $N$-node ($N$>0) cyclic-like component is bi-connected.

Proof: if $0 < N < 2$, that is a cyclic-like component with one or two nodes and it is obviously bi-connected.

If $N > 2$, as below we first prove that every node pair $v_i, v_j$ (i ≠ j) is connected with at least two distinct paths in which there is no overlapping nodes.

If the cyclic path of $v_i$ contains $v_j$, then there is a path like $v_i, v_1, \ldots, v_j, \ldots, v_2, v_i$ without overlapping nodes besides $v_i$ and that is to say, there are two distinct paths between $v_i$ and $v_i$ without any overlapping nodes, i.e., $v_i, v_1, \ldots, v_j$ and $v_j, \ldots, v_2, v_i$.

If the cyclic path of $v_i$ does not contain $v_i$, then these two nodes belong to different cyclic paths. According to Lemma 1, we can always find at least two node-disjoint paths between $v_i$ to $v_j$ respectively from different cyclic paths. Therefore every node pair $v_i, v_j$ (i ≠ j) is connected with at least two distinct paths without overlapping nodes, which means the corresponding topology is bi-connected when $N > 2$.

Overall, A $N$-node ($N$>0) cyclic-like component is bi-connected.

**Theorem 2** A $N$-node ($N$>0) bi-connected component is cyclic-like.

Proof: If $0 < N < 2$, for a bi-connected component with one or two nodes, it is obviously cyclic-like.

If $N > 2$, as below we first prove that for every node in the component, there exists at least one path that starts from it, also ends at it and contains more than two nodes and no overlapping nodes besides the starting/end node. According to the definition of bi-connected topologies, there are at least two node-disjoint paths between every two nodes $v_i, v_j (i \neq j)$ in a bi-connected component, i.e., $v_i, v_1, \ldots, v_j$ and $v_i, v_2, \ldots, v_j$. If we connect these two paths on the endpoint $v_j$, a cyclic path that starts from $v_i$ and ends at it can be obtained, i.e., $v_i, v_1, \ldots, v_j, \ldots v_2, v_i$. Obviously the path contains more than two nodes and there is not any duplicated node except endpoints. Thus if the topology is in fact a cyclic path, obviously it is cyclic-like according to the definition.

Below we consider the cases with nodes that constitute more than a cyclic path in a bi-connected topology. Assume that there is such a cyclic path $p$ in bi-connected topologies, which does not overlapped with any other cyclic paths or only overlapped to one node. Then with proof by contradiction we can show that assumption is not valid and prove that in all bi-connected topologies all cyclic paths are overlapped with at least another one to two or more nodes.

Because a bi-connected topology is connected, the cyclic path $p$ must connect to other nodes and since there are more nodes than a single cyclic path in this topology, such “other nodes” must exist. Links that connect two cyclic paths but do not belong to any of them are called bridge.

According to the proof above, all nodes belong to at least one cyclic path, thus $p$ must connect to at least another cyclic path by bridges or overlap with at least another cyclic path.

Assume that path $p$ is connected to another cyclic path $q$ by the bridge. If they are connected with two or more different bridges, and then among these bridges there must be at least one pair of different bridges which do not share the same endpoint with each other; otherwise the topology will become disconnected when the same endpoint is removed, which is absurd according to the assumption that the topology is bi-connected. But if there is such a pair of different bridges that do not have shared endpoint and connect path $p$ and $q$, then a cyclic path $r$ that contains the pair of bridges will always occur between path $p$ and $q$; $r$ will obviously overlap with $p$ and $q$ to two or more nodes, which is also absurd according to our assumption. Thus if $p$ connects to $q$, then they are connected by only one single bridge. If $q$ also connects to or overlaps with other cyclic path(not $p$) that connects to $p$, then a cyclic path will also occur containing those connected links between cyclic paths; this new cyclic path also overlap with $p$ to two or more nodes, which is also absurd according to our assumption. But if the cyclic path $q$ does not connect to or overlap with other cyclic paths than $p$, then when any one endpoint of the single connecting bridge between $p$ and $q$ is removed, $p$ and $q$ will no longer be connected, which is also absurd according to the assumption of bi-connectivity. Thus $p$ cannot connect to $q$ with the bridge, so as other cyclic paths.

Then in bi-connected topology, cyclic paths can only overlap with each other instead of connected to each other by the bridge. Assume that $p$ overlap with $q$to only one node. To ensure bi-connectivity, there must be an overlapping chain between $p$ and $q$, i.e., $q$ overlaps with $\mathbf{r}$, $\mathbf{r}$ overlaps with $\mathbf{s}, \ldots, \mathbf{x}$ overlaps with $p$. Also to ensure bi-connectivity, the node to which $p$ overlaps with $q$ must not be the same as the one to which $p$ overlaps with $\mathbf{x}$; then a cyclic path containing these overlapping nodes will occur, which overlaps with $p$ to two or more nodes and is absurd according to the assumption. Thus $p$ can only connect to $q$ by the single overlapping node, which is again absurd according to the assumption of bi-connectivity. That is to say, $p$ cannot overlap with $q$ to only one node.

Therefore, the assumption that there exists a cyclic path $p$ that does not overlap with any other cyclic path $q$ to two or more nodes is not valid and in bi-connected topologies, there is more than one node overlapped between cyclic paths.

Overall, A -node ($N$>0) bi-connected component is cyclic-like. □

## III. CTCS OVERVIEW

a. Minimum cyclic path based on a certain node

Based on cyclic-like topologies, we now propose algorithm 1 to search for a minimum cyclic path based on a single node $i$.

Given nodes and possible links between them, a minimum cyclic path could be obtained by performing a variant of Breadth First Search (BFS), which takes $i$ as root; all links can only be searched for once and all paths are recorded; all searched nodes are marked and its child node will not be visited again. After this BFS, the first path which meets any one of following conditions is the minimum cyclic path $p$ on a single node $i$ that we are searching for:

1) The endpoints of the path $p$ are $i$.
2) A path $p$ that can be combined from a pair of half-cyclic paths and contains the single node $i$. A pair of half-cyclic paths means two paths $r$ and $q$ having the same leaves as each other. The path $p$ can be obtained by connecting $r$ and $q$ on node $i$ and deleting all of the duplicated nodes except their endpoints.

When started, Algorithm 1 put the child nodes of the targeted node $i$ into a queue; each time it takes one node out to process and put its children into the queue, which implements a BFS, until all links are visited or a path that meets any of the two conditions are found. During the process, the algorithm reserves all paths. If any duplicated nodes other than $i$ are found, the corresponding half-cyclic-paths are combined and check whether the deduced cyclic path meets the condition.

Algorithm 1 is formally presented as follows:

---

Algorithm 1, to search for a minimum cyclic path based on a single node $i$

---

**Input:**

$G_{max}$, graph for reference

$root$, the node to search for the smallest cyclic path based on, which is the root of the BFS tree of Algorithm 1

**Output:**

$p$, the smallest cyclic path based on node $root$

1: Set all nodes unvisited
2: $start \leftarrow root$
3: Check $G_{max}$ and put neighbors of $start$ into the queue $BFSnodes$ as elements of nodes to visit
4: Put $start$ into the array $BFSpaths$ as the first element of paths to reserve
5: Set $start$ visited
6: Do
7: For every neighbor $neighbor$ of $start$
8: Delete the edge between $neighbor$ and $start$ in $G_{max}$ //no duplicated edges
9: Put $neighbor$ into the array $BFSpaths$ as the an element of paths to reserve
10: If $neighbor$ is unvisited
11: Set $neighbor$ visited
12: Else if $neighbor$ is visited
13: If $neighbor$ is $root$
14: Return the processing path as output
15: End if
16: Combine half-cyclic paths and acquire the deduced cyclic path $c$
17: If $c$ contains $root$
18: Return the path $c$ as output
19: End if
20: End if
21: $start \leftarrow$ the first element dequeued by queue $BFSnodes$
22: End for
23: While $BFSnodes$ is not empty

---

Next we show that Algorithm 1 can deduce minimum cyclic paths based on a certain node, before which we have a lemma first.

**Lemma 2** If node $i$ belongs to a bi-connected topology, the path obtained from Algorithm 1 is a cyclic path containing node $i$.

Proof: Since node $i$ belongs to a bi-connected topology, according to Theorem 2, there must be a cyclic-like topology that contains $i$, which means there must be a cyclic path containing node $i$. By Algorithm 1, when a duplicated node is found, we can obtain a cluster of paths ϱ (having at least two paths) from it to node $i$. Since the result of BFS is a tree instead of a graph, every path achieve from Algorithm 1 will not contain duplicated nodes except endpoints. Therefore if the path that we acquired has the endpoints of node $i$, then obviously it is a cyclic path containing $i$; otherwise we connect two paths in ϱ with node $i$ and delete all duplicated nodes, then we achieve a path that have no duplicated nodes except endpoints, which is also a cyclic path according to our definition. Finally since node $i$ is in the path according to condition (2) in Algorithm 1, thus it is also a cyclic path containing node $i$.

**Theorem 3** If node $i$ belongs to a bi-connected topology, the path obtained from Algorithm 1 is the minimum cyclic path containing node $i$.

Proof: According to Lemma 2, the path $c$ acquired from Algorithm 1 is a cyclic path containing node $i$. Next we use proof by contradiction to show that this path $c$ is the minimum cyclic path containing node $i$.

Assume that $c$ is the first path to be found in Algorithm 1 and is not the minimum path containing $i$. Since node $i$ belongs to a bi-connected topology, according to Theorem 2, there must be a cyclic-like topology that contains $i$, which means there must be a cyclic path containing node $i$ and a minimum one $c_{min}$ also exists. Thus the number of nodes in $c_{min}$ must be smaller than that in $c$, i.e., $|c_{min}| < |c|$. Because Algorithm 1 is a variant of BFS, the path that Algorithm 1 obtains is in BFS tree and the length of it is gradually increased by 1 while the depth of BFS tree raised by 1. Since the difference in different length of path must not be less than one, the leaves of shorter paths like $c_{min}$ will always be closer to the root and will be quicker to be found than longer paths like $c$.

Although every link can only be used once in Algorithm 1, shorter path obviously will be obtained first other than longer ones, thus the links in the minimum cyclic path will be acquired first and then Algorithm 1 can make sure the minimum cyclic path could be found without its links occupied. Thus $c_{min}$ is the path that meets the condition of Algorithm 1 and is made sure to be found first other than $c$, which is absurd according to our assumption that $c$ is the first path to be found. Therefore the path $c$ obtained from Algorithm 1 is at the same time the minimum path containing $i$.

b. Complex cycles based on combination of minimum cyclic paths

Then we introduce Algorithm 2 to establish more complicated cycles which comes from multiple cyclic paths. A cyclic topology also includes combination of different simple cyclic paths if and only if any cyclic path among them are overlapped with at least another to two or more nodes. Thus Algorithm 2 goes like this: first it chooses a pair of cyclic-like topologies to combine and check whether they are overlapped with each other to at least two nodes. If so, a new topology (a complex cycle) is created from these two topologies by connecting them together and deleting all duplicated nodes. Then the new one will be put into the set of cyclic-like topologies to wait for another possible combination. According to our definition of cyclic-like topologies, we can easily acquire Theorem 4. The proof of Theorem 4 is straightforward and hence omitted.

**Theorem 4** Complex cycles obtained from such combination of two different cyclic paths or other cyclic-like topologies are still bi-connected.

When it comes to compute the physical connectivity of a component, we used a multi-round threshold-based algorithm mentioned in our previous work. In each round, possible paths with one cell-distance longer are computed and synchronously selected; those link paths having a lower performance of connectivity than threshold are cut and others are allowed to grow until the whole reachability is satisfied [10].

---

Algorithm 2

---

Input:

$c_1$, one of the cyclic-like topologies to be combined

$c_2$, another one of the cyclic-like topologies to be combined

σ, all found cyclic-like topologies before Algorithm 2

Output:

σ, all found cyclic-like topologies after Algorithm 2

1: If $c_1$ overlaps $c_2$ with more than two nodes

2: $c \leftarrow c_1 \cup c_2 - c_1 \cap c_2$, $c$ is the complex cycle created by combination of $c_1$ and $c_2$

3: Compute the physical connectivity of $c$

4: If the physical connectivity of $c$ is more than overall threshold

5: Insert $c$ into σ

6: Return true

7: Else

8: Return false

9: End if

10: Else

11: Return false

12: End if

---

c. Connectivity-aware Topology control algorithm with Cyclic-like Structures

Finally we propose our overall CTCS algorithm. First we pick out a cyclic-like component having the least power cost and sort other components in increasing order of degree of itself. Degree of a component means the number of link from this component to other components. Then the algorithm try to combine the component that has the least power cost and other sorted components. Because of the definition and maximality of bi-connected components, components that we obtained should not have duplicated nodes and we delete the duplicated ones. Repeat the above steps until all components are processed or the number of components remains unchanged.

---

Connectivity-aware Topology control algorithm with Cyclic-like Structures

---

Input:

$G_{max}$, graph for reference

Output:

$G$, graph whose nodes are allocated power

1: $G \leftarrow V(G_{\max})$

2: σ←∅

3: Set all nodes unvisited in $G$

4: For every node $n$ in $G$

5: If $n$ is unvisited

6: $c$←Algorithm 1 ($G_{max}$, $n$)

7: Set all nodes in $c$ visited

8: σ←σ ∪ $c$

9: End if

10: End for

11: Do

12: $c_{min}$←the path not processed yet and having the least power cost in σ

13: Set $c_{min}$ processed

14: $Q$ ←paths that not yet processed in σ-$c_{min}$

15: Sort $Q$ in increasing order of degree of components

16: For every cyclic-like component $c_{another}$ in $Q$

17: Algorithm 2($c_{min}$, $c_{another}$, σ)

18: End for

19: Delete components in σ that have duplicated nodes as previous ones

20: While |σ| is changed or there is path not yet processed in σ

## IV. EVALUATION

In this section, we compare the performance of our algorithm CTCS to another two approaches named RTCA and CSR [14].

Table.1 comparison of CTCS and another two algorithms (RTCA and CSR)

| | Topological Connectivity | Physical Connectivity | Power Control |
|---|---|---|---|
| RTCA | √ | × | × |
| CSR | × | √ | × |
| CTCS | √ | √ | √ |

As explained in Section 1, RTCA ensures the bi-connectivity of a certain topology by checking the connectivity without each one node and do not take physical connectivity into account; meanwhile CSR ensures only physical connectivity and regardless of topological connectivity; our proposed algorithm CTCS manage to balance both requirements.

In our experiments a variable N_nodes is simulated; It reflects the number of nodes in deployment area whose value varies from 0 to 450. The thresholds of physical connectivity are both set 0.4 in CTCS and CSR, which is slightly lower than the average reachability (0.5) of each single link to somehow ensure the existence of result. As the environment gradually changed, we observe how the three algorithms perform. Every experiment is run as much as 10 times and we take the average as its final result.

Note that the operation time for RTCA is already longer than one day when the number of nodes increasing to 150, which is too long to run multiple times and the experiment result may not be accurate enough, thus the related data for RTCA is omitted when the number of nodes not less than 150.

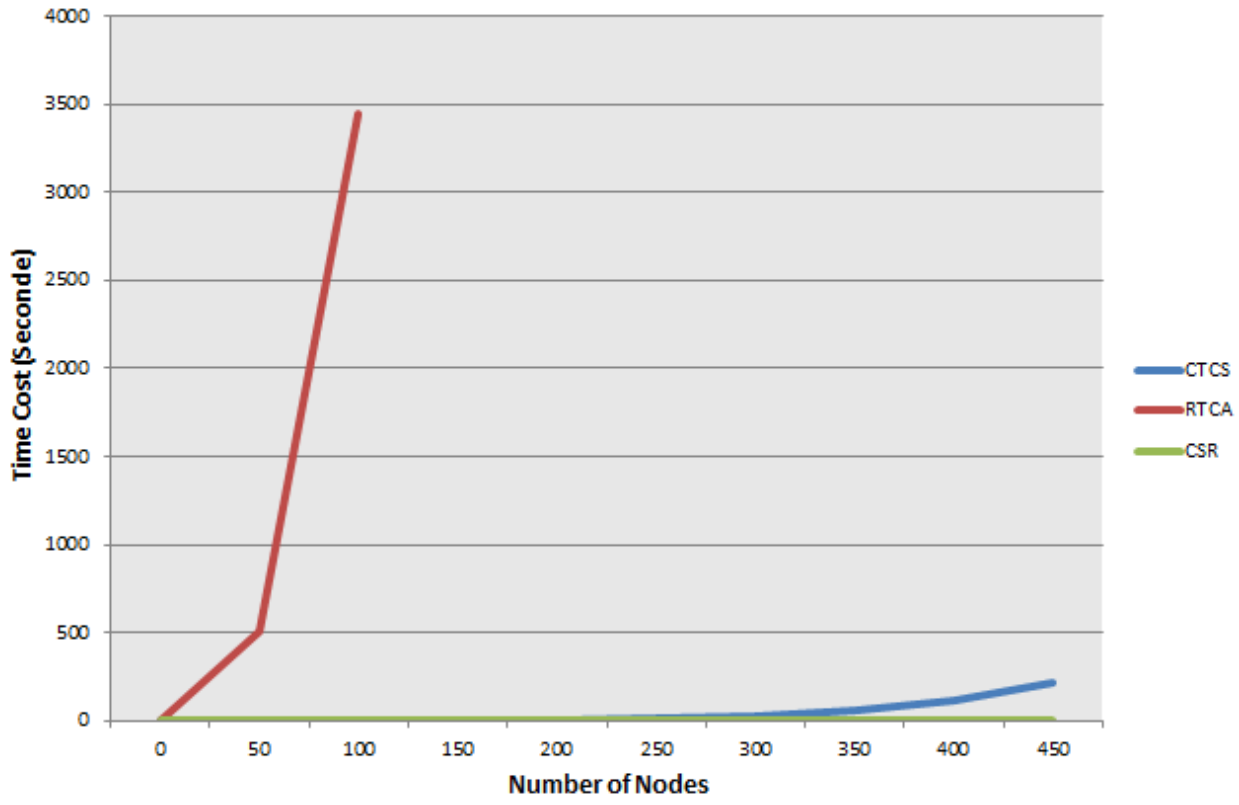


Fig.2 Time Cost of CTCS, RTCA, CSR against Number of Nodes

In the first series of experiments, the number of nodes increased and the corresponding performance on time cost of the three algorithms are listed in Fig.2. As the number of nodes raised, all operating tines of these algorithms also increased. Especially RTCA, the time goes up to 511 seconds when 100 nodes and even more than one day when 150 nodes. That is because as the number of nodes increases, the number of possible subsets that is bi-connected sharply raised. Thanks to the property of cyclic-like topologies, when ensuring bi-connectivity CTCS do no need to compute possible subsets and check for bi-connectivity. Therefore CTCS goes up much slower than the RTCA do. But when the scale of nodes grows more, more combinations of smallest cyclic paths are needed which costs more time. CSR has the lowest operation times which are always below 10 seconds, since it only focus on physical connectivity and the deduced topology of which do not need to be processed by the high-cost checking function, but of course its deduced topologies have the lowest robustness among the three algorithms.

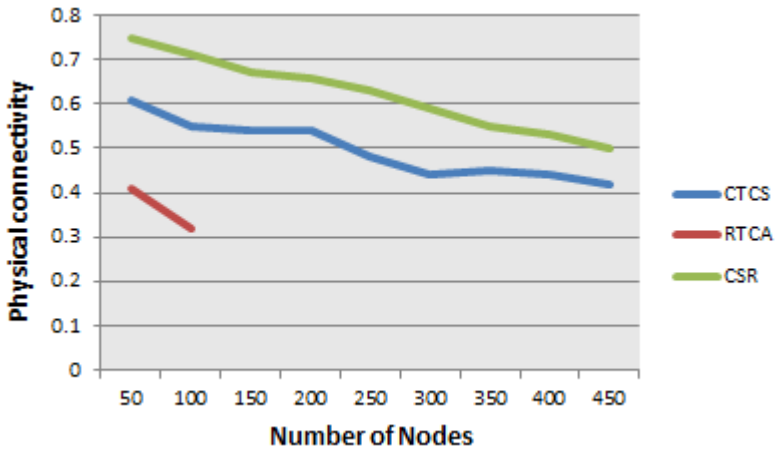


Fig.3 Physical Connectivity of CTCS, RTCA and CSR against Number of Nodes

In the second series of experiments (Fig.3), as the number of nodes grows the corresponding physical connectivity of the three algorithms comes down. That is because when the scale of the network increases, the average length between any pair of nodes raised. Since the possibility for successful communication of each link is always lower than 100%, thus the reachability of every two nodes drops. Though there is only two pair of available data for RTCA, i.e., (50, 0.41) and (100, 0.32), because the result will be even lower as scale increases, the follower result of RTCA will be very likely to be much lower than CSR and CTCS. CSR always has the better

performance than another two since it only focus on physical connectivity. CTCS comes lower than CSR but the difference between them seems to be increasingly slightly smaller since the threshold is set 0.4 when configuration and bots results will gradually close to it.

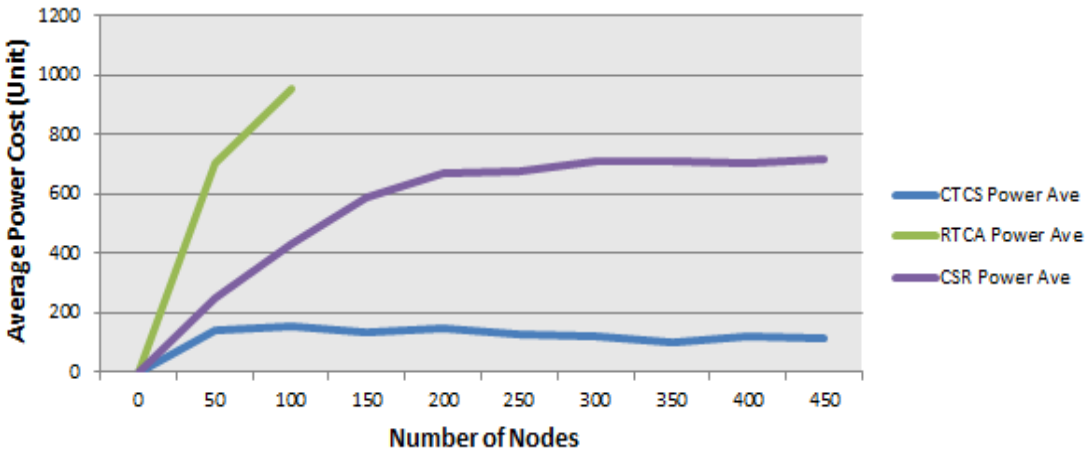


Fig.4 Average Power Cost of CTCS, RTCA and CSR against Number of Nodes

In the third series of experiments (Fig.4) we focus on the average power cost on each node. As the number of nodes in the network increases, the power of RTCA and CSR both grows since they do not have any power control. When the scale of the network rises, the average distances on paths between nodes increases and it needs more power to communicate. Especially RTCA, it increases so fast that in the case of 20 nodes the average power even reaches over 500 units. CSR grows fast at first and become gradually stable at around 700 units when the number of nodes gets larger, which is always lower than RTCA. That is because CSR tend to create more reachable and stable paths, which means longer (and thus not that stable) links that needs more power are less likely to be established than RTCA. The last one CTCS costs far less power than any other two algorithm, which is always less than 200 units; because when combination processes CTCS take the links using less power in priority, which keeps the average power cost at a relatively very low level.

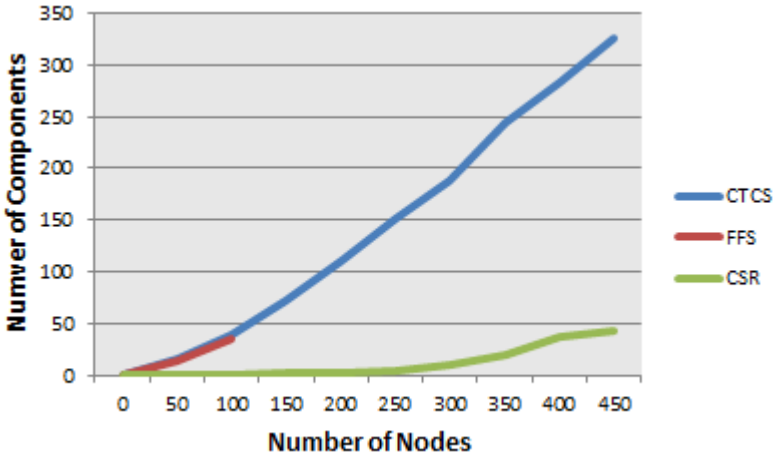


Fig.5 Number of Components of CTCS, FFS, CSR against Number of Nodes

The last series of experiment (Fig.5) shows that for fixed topological and physical connectivity requirements, the number of components grows after the number of nodes raises. That is because when the scale of the network increases, the cost of change the whole network into a single connected topology with required property also goes up or even become impossible, which stop components to combine and get large and therefore increasingly more components remain in the deduced topologies. The result of CTCS and FFS should at the same meet both two categories of requirements other than one in CSR, which is in fact much tougher and hence leaves much more components.

## V. CONCLUSIONS

In this paper, we have investigated the problem of providing the network a better performance on topological and physical connectivity. CTCS is introduced to meets both requirements while cutting down the time cost during operation. The key idea is to investigate a new topology named cyclic-like topologies and use a physical connectivity control function for probabilistic topology control, which eventually strikes a better balance of topological and physical connectivity requirements.

## REFERENCES

[1] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” Ad Hoc Networks, vol.3, no.3, pp.325–349, 2005.

[2] I. F. Akyidiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, “A survey on sensor networks,” IEEE Communications Magazine, vol.40, no.8, pp.102–114, 2002.

[3] Wang, Yun, Brendan M. Kelly, and Xiaolong Li. "On the network connectivity of wireless sensor networks following a random and non-uniform distribution." Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on. IEEE, 2013.

[4] Liu, Jianwei, et al. "Connectivity of two nodes in cognitive radio ad hoc networks." Wireless Communications and Networking Conference (WCNC), 2013 IEEE. IEEE, 2013.

[5] H. Liu, A. Nayak, and I. Stojmenovic, “Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks,” Guide to Wireless Sensor Networks, S. Misra et al. (eds.), Springer-Verlag (London), 2009.

[6] P. Gupta and P. Kumar, “Critical power for asymptotic connectivity”, in Proc. IEEE CDC 1998.

[7] O. Dousse, F. Baccelli, and P. Thiran, “Impact of interferences on connectivity in ad hoc networks”, IEEE/ACM Trans. Netw., vol. 13, no.2,pp. 425-436, Apr. 2005.

[8] F. Dai and J. Wu, “On constructing k-connected k-dominating set in wireless ad hoc and sensor networks”, J. Parallel Distrib,vol. 66, no.7, pp. 947 - 958, 2006.

[9] X. Jia, D. Kim, S. Makki, P.-J. Wan, and C.-W. Yi, “Power assignment for k-connectivity in wireless ad hoc networks”, in 24th IEEE Conference on Computer Communications, INFOCOM’05, March 2005, pp. 2206-2211.

[10] Xijun Wang; Min Sheng; Mengxia Liu; DaosenZhai; Yan Zhang, RESP: A k-connected residual energy-aware topology control algorithm for ad hoc networks[C]. Wireless Communications and Networking Conference (WCNC), IEEE, 2013: 1009, 1014.

[11] H. Liu, A. Nayak, and I. Stojmenovic, “Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks,” Guide to Wireless Sensor Networks, S. Misra et al. (eds.), Springer-Verlag (London), 2009.

[12] S. Das, H. Liu, A. Nayak, and I. Stojmenovic, “A Localized Algorithm for Bi-Connectivity of

Connected Mobile Robots," Telecommunication Systems, vol. 40, no. 3, pp. 129-140, 2009.

[13] Yunhuai Liu, Lionel Ni and Chuanping Hu, "A Generalized Probabilistic Topology Control for Wireless Sensor Networks", Selected Areas in Communications, IEEE Journal on Volume:30, Issue:9, Page(s):1780-1788, 2012.

[14] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, and S. C. Mukhopadhyay, "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes" IEEE SENSORS JOURNAL, VOL. 13, NO. 10, OCTOBER 2013, pp. 3846-3853.

[15] Zimu Zheng, Zhiwei Huang, Zhicheng Li, Xinyi Peng. "A Quality-Aware Relay Node Placement Algorithm to Connect Disjoint WSN Segments with Topology Reorganized.", Sensors & Transducers, Vol. 170, Issue 5, pp. 122-131, May 2014.

[16] Senel, Fatih, Mohamed Younis, and Kemal Akkaya. "Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks." Vehicular Technology, IEEE Transactions on 60.4 (2011): 1835-1848.

[17] N.K.Suryadevara, M.T.Quazi and S.C.Mukhopadhyay, Intelligent Sensing Systems for measuring Wellness Indices of the Daily Activities for the Elderly, proceedings of the 2012 Eighth International Conference on Intelligent Environments, Mexico, June 1-3, 2012, pp. 347-350

[18] Varun Ramchandani, Kranthi Pamarthi, Shubhajit Roy Chowdhury. "Comparative Study of Maximum Power Point Tracking using Linear Kalman Filter & Unscented Kalman Filter for Solar Photovoltaic Array on Field Programmable Gate Array.", The International Journal on Smart Sensing and Intelligent Systems, Vol. 5, no. 3, pp. 701 – 716, Sep 2012.

[19] N. K. Suryadevara, S. C. Mukhopadhyay. R.K. Rayudu and Y. M. Huang, Sensor Data Fusion to determine Wellness of an Elderly in Intelligent Home Monitoring Environment, Proceedings of IEEE I2MTC 2012 conference, IEEE Catalog number CFP12MT-CDR, ISBN 978-1-4577-1771-0, May 13-16, 2012, Graz, Austria, pp. 947-952.

[20] Anuj Kumar, I. P. Singh and, S. K. Sud. "An Approach Towards Development of PMV Based Thermal Comfort Smart Sensor.", The International Journal on Smart Sensing and Intelligent Systems, Vol. 3, no. 4, pp. 621 – 642, Dec 2010.