SUSTAIN: An Adaptive Fault Tolerance Service for Geographically Overlapping Wireless Cyber-Physical Systems *

Gholam Abbas Angouti Kolucheh and Qi Han
Department of Mathematical and Computer Sciences
Colorado School of Mines, Golden, CO 80401
{gangouti,qhan@mines.edu}

ABSTRACT

As wireless sensor networks (WSNs) technologies are becoming more mature and the use of WSNs for different cyber-physical applications are becoming increasingly popular, it is now possible to set up more than one WSN in the same area. However, constant failures may occur in these WSNs, causing a network to be partitioned into several parts. Nodes in the parts without the sink will lose the opportunity to report their data, significantly degrading the performance of the supported cyber physical applications. In this paper, we propose SUSTAIN, a fault tolerance support algorithm that adapts to the network conditions and exploits the nodes in other neighboring networks as relays to resume communication for the faulty network. Our simulation results demonstrate that SUSTAIN increases the average packet delivery ratio in the network sometimes by more than 40% with an affordable amount of delay and cost overhead.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network management, Network monitoring

General Terms

Reliability, Algorithms

Keywords

wireless sensor networks; fault tolerance; network protocol; cyber physical systems

1. INTRODUCTION

Cyber-Physical Systems (CPS) use cyber components to monitor and control the physical world, and then use the status of physical world to further influence the design of cyber infrastructures. We are especially interested in wireless CPS where cyber components are wirelessly connected. Wireless sensor networks (WSNs) technology is one kind of commonly used cyber elements in emerging wireless CPS. With the increasing popularity of WSN deployment, more than one WSN may be deployed in the same geographical area to serve different physical applications. For instance, Sustainable Bridges [2] and Cartalk 2000 [17] are two physical applications with two dedicated WSNs deployed in the same area. The former one monitors bridges in a cost-effective manner using sensor nodes in order to detect structural defects as soon as they appear, and the latter is a cooperative driver assistance system that uses information captured by sensor nodes to provide ad-hoc warnings related to transportation problems like traffic jam, accidents, and lane or highway merging. Each WSN is deployed for a specific functionality and is isolated from the other. In other words, these two WSNs do not collaborate with each other. This type of different WSN deployment in geographically overlapping areas can be found in other applications as well. For instance, one WSN may be deployed for energy management and the other one used for surveillance of the same physical building; or one WSN is used for road traffic monitoring and the other one is used for loophole detection for the same segment of a major road.

While using WSNs for wireless CPS brings lots of benefits, WSNs are failure prone, so fault management techniques need to be in place in order to keep the deployed WSNs operational. This has spurred a large amount of research in the design of fault detection, isolation, diagnosis, and recovery algorithms for WSNs. However, to the best of our knowledge, those protocols and mechanisms are designed to deal with faults in dedicated WSNs without exploring the potential to be assisted by other WSNs deployed in the similar geographical area. We believe providing the techniques to exploit the resources in geographically overlapping wireless CPS will significantly improve the performance of future CPS.

In this paper, we propose SUSTAIN, an adaptive fault tolerant service that solicits help from other WSNs deployed in geographically overlapping areas to provide best possible service to CPS at hand. SUSTAIN is designed to support data collection applications where all nodes in the network send data to the sink. SUSTAIN adapts to fault severity in current CPS: when data delivery can go through smoothly in current CPS, no help from neighboring CPS is requested; however, when partitions are identified in the current network, SUSTAIN constructs a data delivery path by using nodes in other WSNs as relay nodes. Simulation results

*This work is supported in part by NSF grant CNS-0915574.

ARM'2010, November 30, 2010, Bangalore, India.

show that SUSTAIN significantly increases packet delivery ratio while incurring affordable overheads such as packet latency and cost in terms of number of hops of the packets.

The rest of the paper is organized as follows. Problem definition along with assumptions of this work is presented in Section 2. Section 3 contains a complete discussion of SUSTAIN. Section 4 discusses our simulation studies and experimental results. In Section 5, we review existing fault management techniques in WSNs; and Section 6 concludes the paper.

2. PROBLEM DEFINITION

We refer to WSNs deployed in the same area for different purposes, as *neighbor WSNs*. Although the nodes in neighbor WSNs are programmed in different ways to serve different applications, some nodes are close enough to each other and they are in the transmission range of each other. We consider a representative scenario where all nodes in a network need to send their data to the sink. As a result of node or link failures, a network may be partitioned to disconnected parts, so the nodes isolated from the sink will not be able to communicate with the sink. The sink will hence lose the data from those nodes completely. The objective of SUSTAIN is to resume the communication between the sink and the disconnected parts of the network by using nodes in neighbor WSNs as relays.

Without loss of generality, we refer to the network being considered for fault tolerance support as WSN1 and the other network supporting it as WSN2. Note that there can be more than one supporting network for WSN1, but the same techniques can be applied. We assume that each WSN in the area is initially connected and the full topology of each WSN is available at its sink. We also assume that all nodes have the same transmission range and many nodes are in the transmission range of nodes in a neighbor WSN. The focus of the work is to provide fault tolerance support, instead of detecting faults in WSNs which has been investigated in prior work as detailed in Section 5.

3. MAIN DESIGN OF SUSTAIN

Before discussing the detailed design of SUSTAIN, we describe the following terms used throughout the paper. A node in the transmission range of at least one node in neighbor WSNs is called an *FT node*. For each `FT` node, one of the nodes within its transmission range in the neighbor WSN is designated as its *FT_peer node*. After the network is partitioned, one `FT` node in each part is designated as *FT_selected node* which serves as the access point of all nodes in this part and other nodes use it to communicate with other parts of the network. We also make a distinction between the part including the sink (referred as *sink-included part*) and the parts not including the sink (referred as *no-sink parts*) when the network is partitioned.

SUSTAIN consists of three main mechanisms: discovering network partitions, connecting sink-included part and each no-sink part, and managing partitioning in no-sink parts. Figure 1 is the flow chart of the algorithm.

- Identifying network partitions. Any fault reported to the sink triggers SUSTAIN to determine the number of network parts if the fault has led to any partitioning in the network. In the mean time, SUSTAIN also identifies which part each node belongs to.

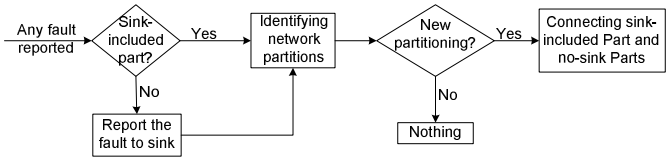


Figure 1: The flowchart of SUSTAIN

- Connecting sink-included part and each no-sink part. For each no-sink part, SUSTAIN tries to construct a communication path to the sink-included part by using FT nodes. A sequence of nodes in the neighbor WSN will be part of the newly constructed path.

- Managing partitioning in no-sink parts. After making sure all no-sink parts have paths available for communicating with the sink, other nodes in no-sink parts may still fail resulting in new disconnected parts. To avoid consuming resources in internal sensor nodes, faults in these no-sink parts are reported to the sink where network partitions are identified at the sink in a similar way as described in the first mechanism.

3.1 Identifying Network Partitions

When a failure is detected in any of the nodes or links in the network, it will be reported to the sink. The sink, then, examines whether the failure will cause network partitions. To do this, the algorithm considers the network as a graph with nodes as the vertices and communication links as edges. Basically, the algorithm determines whether the failure results in an *articulation set*. An articulation set is an extension of the concept of an articulation point in graph theory whose removal disconnects the graph [7]. We consider an articulation set as a set of nodes and links whose deletion increases the total number of partitions in the network.

To identify partitions, faulty nodes and all of edges incident to them, as well as faulty links are first deleted from the graph, then the connectivity of the result graph is checked using depth first search (DFS) algorithm [7]. If the graph is still connected, then the network is not partitioned. Otherwise, the network is partitioned, so the algorithm will further identify all the partitions and nodes belong to those partitions. For instance, in Figure 2, the failure of node i causes three partitions: P1 is the sink-included part, P2 and P3 are no-sink parts.

In addition, all `FT` nodes are identified for each part by each node sending out beacons. If a node can hear the beacons from a neighbor WSN, it is marked as an `FT` node. For each `FT` node, a corresponding `FT_peer` node is also selected. If an `FT` node receives beacons from more than one node in the other WSN, it selects the first one as its `FT_peer` node. Figure 3 demonstrates an example in which there is one sink-included part and one no-sink part in WSN1. WSN2 is its supporting neighbor network. All of the `FT` nodes and their `FT_peer` nodes are identified and grouped together.

3.2 Connecting sink-included Part and no-sink Parts

After discovering partitioning in the network, this mechanism will be triggered to connect each no-sink parts and the sink-included part. First, one `FT` node in the sink-included part is selected by the sink as the `FT_selected` node to be

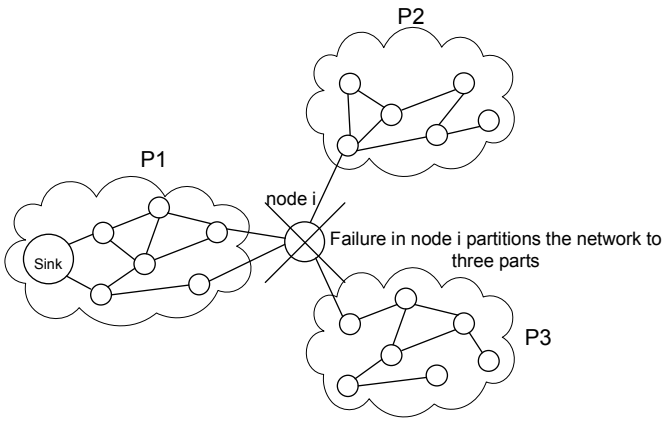


Figure 2: If node i fails, the sink can identify all three partitions.

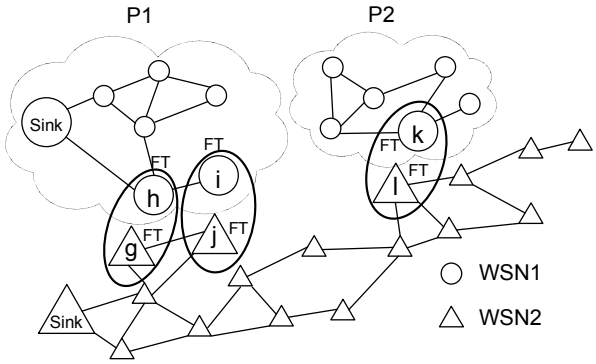


Figure 3: A network is partitioned into two parts. `FT` and `FT_peer` nodes are identified for each part and grouped together.

used for communication with no-sink parts. To select an `FT_selected` node, the sink broadcasts a message in the sink-included part and looks for `FT` nodes. Any `FT` node that receives this message responds to it. The sink marks the sender of the first response as the `FT_selected` node and sends it an acknowledgment. Other responses are all discarded by the sink. The `FT_selected` node marks itself as `FT_selected` node after receiving this acknowledgment. This `FT_selected` node functions as an access point, so all receptions and transmissions between this partition and other no-sink parts will go through this node (e.g., node i in Figure 3).

Second, a path will be constructed between this `FT_selected` node and an `FT` node in a no-sink part using nodes in a neighbor WSN. To find the path, the sink sends a route request message to the `FT_peer` node, requesting a path to the no-sink part. This message is different from a regular routing message which seeks a path to a destination node. Instead, this message indicates that the sink is looking for an `FT` node in a specific no-sink part. As the `FT_selected` node (e.g., node i) receives this message from the sink, it will forward the message to its `FT_peer` node in WSN2 (e.g., node j). Node j broadcasts this message to its neighbors in WSN2. Each node that receives this message saves the id of the sender before broadcasting the message to its neighbors. By saving the id of the sender, a reverse path is built. If this message is received by any `FT` node in WSN2 whose `FT_peer` node belongs to a no-sink part (e.g., node l), the node will send a response to the `FT_peer` node (e.g., node j) of the `FT_selected` node using the created reverse path. This response message also will be overheard by the `FT_peer` node in WSN1 (node k) of this `FT` node in WSN2 (node l). As the `FT` node in WSN1 (node k) receives this message, it marks itself as the `FT_selected` node and the path is complete. Result of this process is depicted in Figure 4.

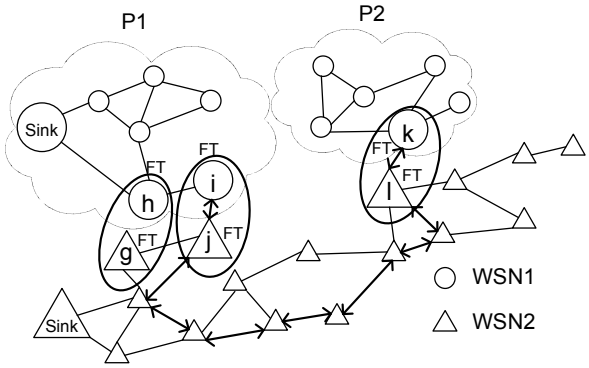


Figure 4: A path between two parts is created using nodes in a neighbor network. Bi-directional arrows represent the created path between the parts of the partitioned network.

All nodes in the no-sink parts select the `FT` node that belongs to the created path as the `FT_selected` node. Any node in these parts which has any data to send, transmits it to this `FT_selected` node and the `FT_selected` node will send the data to the sink using the created path. As a result, each node in a no-sink part should have a route to the `FT_selected` node in this part.

3.3 Managing Partitioning in no-sink Parts

In the situation depicted in Figure 2, the exact number of partitions is known at the sink node. However, there are situations where the exact number of partitions is not known at the sink. For example, in Figure 5, if node i fails at time t_i and is reported to the sink immediately, but node j fails after t_i, then the report of node j's failure cannot reach the sink and the sink does not have enough information to discover all the partitions. In other words, the exact number of network parts is not known at the sink. Also, it is possible that the no-sink parts are further partitioned. Because there is not any sink node in these parts and all nodes are resource-constrained, we choose not to execute the partition identification and path construction processes on these nodes. Instead, SUSTAIN deal with potential partitioning in no-sink parts without help of the sink.

As mentioned in the previous mechanism, one `FT` node in each no-sink part is designated as the `FT_selected` node. This `FT_selected` node serves as the access point of this no-sink part. The topology of this part is then transmitted to the sink along with any faults reported to the `FT_selected` node. The sink will run the two processes described in the previous two subsections to provide fault tolerance for this no-sink part.

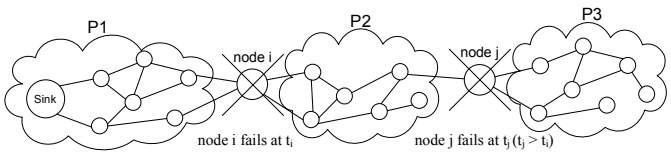


Figure 5: An example where the number of partitions is not known at the sink: If node j fails after node i, the sink does not have any information about part P_3.

4. PERFORMANCE EVALUATION

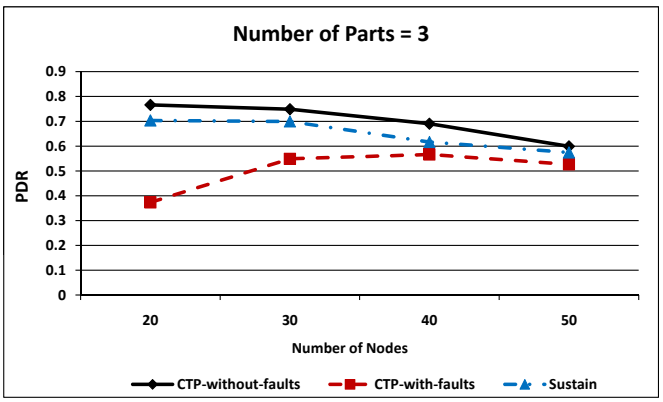
In prior work, there is not any similar fault tolerance supporting method that collaborates with neighbor WSNs, so we use two baseline methods to evaluate the efficiency and scalability of SUSTAIN. Both baseline methods use the Collection Tree Protocol (CTP) [3]: (i) "CTP-without-faults" in which there is no partitioning in the network, serving as the upper bound of the performance; (ii) "CTP-with-faults" in which the network is partitioned. CTP does not deal with network partitioning and delivers packets to one of accessible roots in the network.

Three performance metrics are used: packet delivery ratio (PDR), cost, and delay. PDR is the ratio of the total number of received distinct packets to the total number of sent packets; cost is the average number of hops that packets travel from sources to the sink; and delay is the average amount of time it takes for packets to be delivered from sources to the sink.

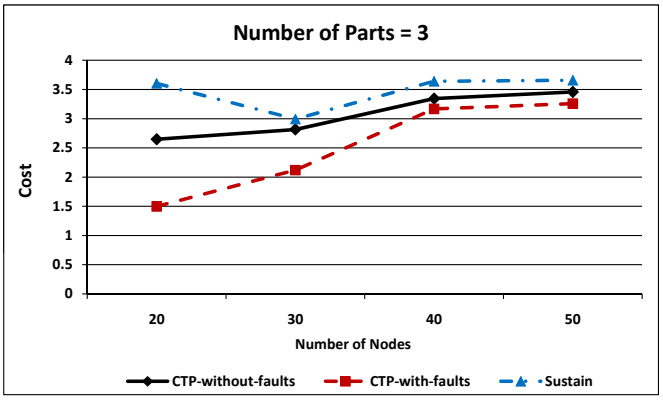
We implemented SUSTAIN in nesC [1] and ran simulations with TOSSIM [11]. We used the default models of TOSSIM for radio and MAC layers. In all simulation scenarios, there are two WSNs (WSN1 and WSN2) and partitioning occurs only in WSN1. For WSN1, star topologies are used with the sink node located in the middle of network. All nodes send data to the sink every 8 seconds and to avoid collisions, the starting time is randomized. Since faults that do not make an articulation set are typically handled by existing fault tolerance mechanisms and SUSTAIN's focus is on network partitions caused by faults, in our evaluation, we only simulate faults that will lead to network partitions. The impact of network size and number of faults on the performance (i.e., PDR, cost and delay) are separately evaluated in our simulation.

Impact of network size. Figure 6 demonstrates the performance when the total number of nodes increases from 20 to 50 with the total number of parts being 3. We observe that the increase in network size leads to decreasing PDR and increasing cost and delay. This is because data traffic and network density increase when there are more nodes in the network, resulting in more packet collisions, larger queueing delay at intermediate nodes, and more number of hops for delivered packets. We also observe that the PDR of SUSTAIN is lower than CTP-without-faults but higher than CTP-with-faults, while delay and cost are higher for SUSTAIN since SUSTAIN needs to take extra effort in engaging the neighbor WSN.

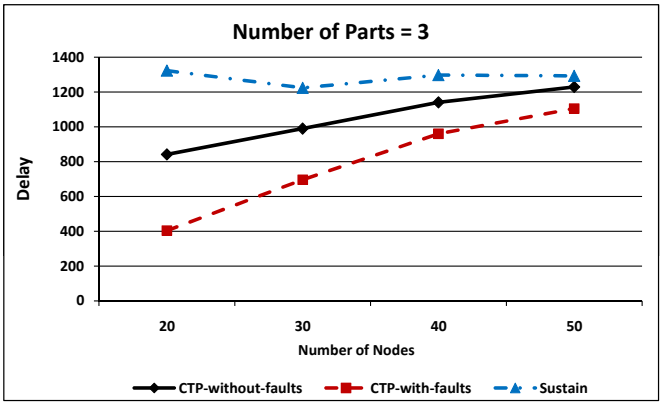
Impact of number of faults. Figure 7 shows the impact of fault severity on the performance when the network has 40 nodes and the number of partitions generated increases from 1 to 5. The first data point in each sub-figure is actually



(a)



(b)

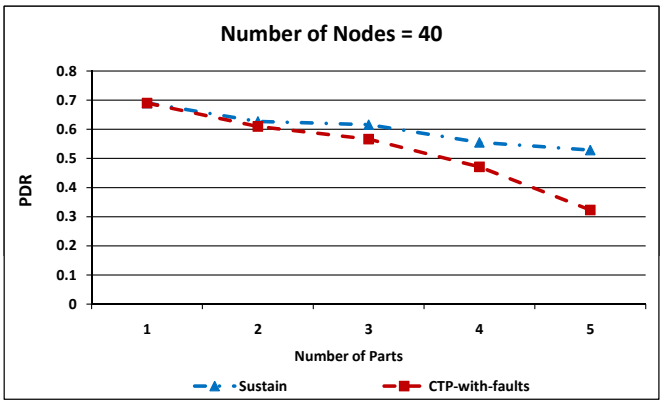


(c)

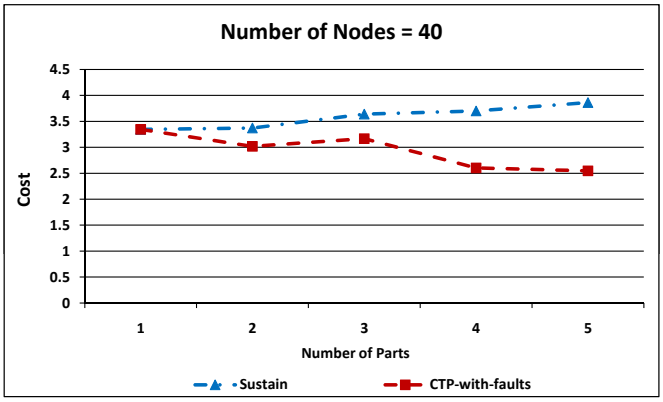
Figure 6: Impact of network size (total number of parts is 3)

the performance of CTP-without-faults. As expected, as the number of parts increases, PDR decreases while delay and cost increase. However, the amount of PDR decreasing is smaller than that of CTP-with-faults and this difference is more significant for a large number of partitions. This is because with more partitions, more nodes get disconnected from the sink and cannot deliver data to the sink. The higher PDR of SUSTAIN comes at the price of higher cost

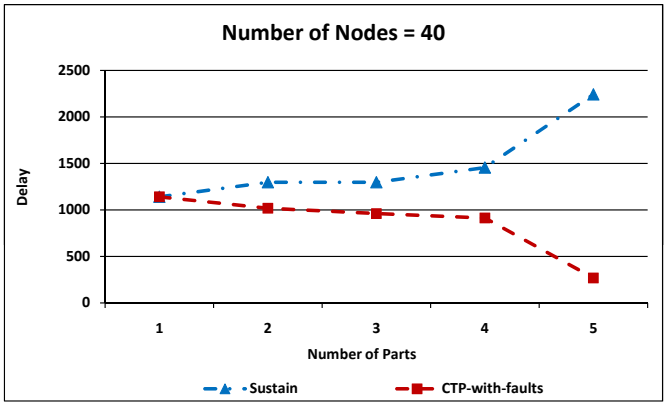
and latency, since it involves more work in order to build a communication path using nodes from a neighbor WSN.



(a)



(b)



(c)

Figure 7: Impact of fault severity (total number of nodes is 40)

Performance summary. In summary, SUSTAIN mitigates the effect of partitioning in the network. It increases average PDR in the partitioned WSN at a moderately increased cost and packet latency.

5. RELATED WORK

Techniques dealing with faults in WSNs can be classified into fault prevention, fault detection, fault isolation, fault identification, and fault recovery [14]. The work presented in this paper is more closely related to fault detection and recovery, so we review these two areas in a bit more detail in this section.

In fault detection techniques [4, 5, 8, 13, 15, 20], faults in the network are classified into two main types [9]: *functional fault* and *data fault*. Functional fault means that a node completely breaks down and it does not function normally and does not generate or forward any data in the network. In situations where a node generates invalid data, a data fault has occurred. With detected alarms, fault isolation and identification processes diagnose and determine the real causes. When the sink does not hear from a particular part of the routing tree, it is unknown whether it is due to failure of a key routing node, or failure of all nodes in a region. A fault tracing protocol has been proposed [19] to differentiate between these two cases. Sympathy [16] monitors regular network traffic which is assumed to be frequently generated by each non-failed node: sensor readings, synchronization beacons, routing updates, etc. Sympathy treats the absence of monitored traffic as an indication of faults. It uses metrics traffic generated at the nodes to localize the failures. These metrics include connectivity metrics, flow metrics, and node metrics. SUSTAIN does not detect faults, instead, any of these fault detection and tracing techniques can be applied to trigger the execution of SUSTAIN.

Fault recovery techniques are used to treat faults, i.e., reverse their adverse effects. In general, faults can be (1) discovered and recovered within the sensor network; or (2) concealed at the sink after collecting and analyzing the sensor readings [14]. Fault tolerant protocols have been developed for sensor data collection (i.e., upstream delivery from sensor nodes to the sink) [18, 10] or sensor data dissemination (i.e., downstream delivery from the sink to sensor nodes) [12, 6]. However, all of these protocols aim to operate in a dedicated WSN. We haven not seen any work that uses neighbor WSNs for fault tolerance. This is exactly the objective and focus of SUSTAIN.

6. CONCLUSION

In this paper, we proposed SUSTAIN, an algorithm to support fault tolerance in geographically overlapping WSNs. In other words, SUSTAIN works for situations in which there are two or more WSNs deployed in the same area for different CPS. Simulation results indicate that SUSTAIN increases the average PDR with an affordable overhead, so it is especially beneficial to cyber-physical applications where the percentage of received data is more important than slightly increased latency and number of hops for packets. Future wireless CPS will heavily rely on WSNs and deploying more than one WSN for different CPSs in the similar geographical area will also become very common, so adaptive fault tolerance services such as SUSTAIN will fully demonstrate its effectiveness in supporting emerging CPS.

7. REFERENCES

[1] http://nescc.sourceforge.net.
[2] http://www.sustainablebridges.net.
[3] Tinyos 2 tep 123: The collection tree protocol. http://www.tinyos.net/tinyos-2.x/doc/txt/tep123.txt.

[4] L. Balzano and R. Nowak. Blind calibration of sensor networks. In *Proceedings of the sixth IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[5] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach to in-place sensor calibration. *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, pages 301–316, 2003.

[6] J. Considine, F. Li, G. Kollios, and J. Brers. Approximate aggregation techniques for sensor databases. In *Proceedings of The 20th IEEE International Conference on Data Engineering (ICDE)*, pages 449–460, 2004.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, second edition, 2002.

[8] J. Feng, S.Megerian, and M. Potkonjak. Model-based calibration for sensor networks. *The second IEEE international conference on sensors*, pages 737–742, 2003.

[9] S. Guo, Z. Zhong, and T. He. Find: faulty node detection for wireless sensor networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 253–266, 2009.

[10] Q. Han, I. Lazaridis, S. Mehrotra, and N. Venkatasubramanian. Sensor data collection with expected reliability guarantees. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, pages 374–378, 2005.

[11] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the first international ACM conference on Embedded networked sensor systems (SenSys)*, pages 126–137, 2003.

[12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[13] E. Miluzzo, N. Lane, A. Campbell, and R. Olfati-Saber. Calibree: A self-calibration system for mobile sensor networks. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems (DCOSS)*, 2008.

[14] L. Paradis and Q. Han. A survey of fault management in wireless sensor networks. *Journal on Network and Systems Management*, 15(2):171–190, 2007.

[15] N. Ramanathan, L. Balzano, M. Burt, D. Estrin, T. Harmon, C. Harvey, J. Jay, E. Kohler, S. Rothenberg, and M. Srivastava. Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks. Technical Report CENS-TR-62, Center for Embedded Networked Sensing, 2006.

[16] N. Ramanathan, K. Chang, L. Girod, R. Kapur, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pages 255–267, 2005.

[17] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz. Cartalk 2000: Safe and comfortable driving based upon inter-vehicle-communication. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV)*, volume 2, pages 545–550, 2002.

[18] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. Esrt: Event-to-sink reliable transport in wireless sensor networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc)*, pages 177–188, 2003.

[19] J. Staddon, D. Balfanz, and G. Durfee. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 122–130, 2002.

[20] K. Yedavalli and B. Krishnamachari. Sequence-based localization in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 7(1):81–94, 2008.