Content Based Routing with Directional Random Walk for Failure Tolerance and Detection in Cooperative Large Scale Wireless Networks

Pierre Leone, Cristina Muñoz

Content Based Routing with Directional Random Walk for Failure Tolerance and Detection in Cooperative Large Scale Wireless Networks

Pierre Leone and Cristina Muñoz

Computer Science Department, University of Geneva
Battelle Bâtiment A, route de Drize 7, 1227 Geneva , Switzerland

Abstract. In this article we develop around the idea that Failure Tolerant and Detection systems can be built using similar mechanisms that the ones used to implement Publish/Subscribe systems. Then, we propose Directional Random Walks (DRW) as a heuristic mechanism to propagate information in wireless networks in order to match publications with subscriptions. The efficiency of this heuristic is validated with numerical experiments.

1 Introduction

Our position that motivates this article is the fact that event based mechanisms are appropriate building tools for the implementation of Failure Tolerance and Detection systems. We remind that Publish/Subscribe systems are used in networks to match the publication of information at a node with a subscription to similar information at another node. The matching is done independently of the network structure and should rely only on the content of publications and subscriptions. Here, we do not delve into the description of how we represent the information. We assume only that we have the possibility to exchange and recognize this representation through message passing [5, 11]. In order to support this position we review some well known strategies on the topics of failure tolerance and detection and we describe how they can be implemented on the top of a Publish/Subscribe system.

However, the implementation of classical Publish/Subscribe systems requires to identify some particular nodes as broker nodes that are connected to each other [12, 7, 16, 6], i.e. compose a functional subnetwork, share the information about all the publications and subscriptions and are responsible for implementing the matching process. This characteristic is an issue in large scale wireless networks. Actually, the size and structure of the broker nodes must adapt to the scale of the network in order to manage appropriately the demands.

Moreover, it is classical in wireless networks to minimize the energy consumption and this leads to consider protocols/algorithms that balance energy consumption [8, 9]. The identification of broker nodes leads to over-use these nodes which is the opposite to what we aim for.

A last issue that we consider in this article is the fact that classical implementations of Publish/Subscribe systems are done on the top layer of a routing algorithm. To illustrate this point, we mention the use of a Distributed Hash Table (DHT). Given a piece of information and its description, the DHT translates the description of the information as an address, typically an IP address, and the information is stored/retrieved at the node with the closest address in the network [14, 3, 15]. However, building a routing protocol that uses the addresses in large scale wireless networks is a challenge. The solution of this point leads to the development of a **content based routing** algorithm. Guided by these considerations, our goal in this article is to present an efficient strategy to:

- Match publications with subscriptions. Imagine a node in the network that publish some information and another one that is subscribed to receive such information. Then, it is necessary to build a path in the network that links these two nodes.
- The process must be scalable in the sense that it must not be too sensitive to the total number of nodes in the network. Moreover, it must support multiple publications and subscriptions, i.e. the system offers the possibility to build groups.
- The path system that results from the process of linking nodes that are interested in matching publications/subscriptions must share the load, i.e. the number of publications/subscriptions managed by the nodes must be balanced.

The algorithm to build paths targets wireless networks because the heuristic that we use to disseminate the information in the network use the geometrical connectivity property of the nodes, i.e. the nodes that are close-by are connected, see below the Unit Disk Graph (UDG) model. However, we argue that the extension to more complex network topologies is realistic[4]. This could be achieved by considering the embedding of the communication graph in a general surface [20, 21].

The rest of this paper is organized as follows: Section 2 discusses about the implementation of failure tolerance and detection systems on the top of a Publish/Subscribe layer. Section 3 provides the information related to the design of the content based routing algorithm. Section 4 evaluates the performance of our method. Finally, Section 5 summarizes the main characteristics and results of the heuristic proposed.

2 Implementation of Failure Tolerance and Detection Systems

In the following we present how we can implement failure tolerant and detection systems using Publish/Subscribe mechanisms. Our aim in this article is to focus on the presentation of the content based routing algorithm. Further work will be dedicated to validate the efficiency of the protocol in the framework of failure tolerant and detection systems.

A major tool used to implement failure tolerant systems is to duplicate the available resources at different places in the network [18]. This requires some coordination between the nodes to ensure the consistency property. Moreover, it must be assured that the primitives used are the ones that make possible the group membership management, i.e. the nodes must be able to list the members of the group, membership must be dynamic. A simple system offering the basic functionalities is built if all the nodes that are designed to be part of the group send a publication or a subscription message that identify to which group they want to be part of. The content based routing algorithm matches the subscriptions with the corresponding publications and each node gets the list of all the members of the group. Actually, a node may get the list twice: when the algorithm matches the subscription of the node, i.e. the node receives the list of the corresponding publications, and when when the algorithm matches the publication of the node, i.e. the node receives the list of the corresponding submissions. We emphasize that besides providing the list of the corresponding nodes, the algorithm also provides the routing paths towards the nodes.

Another way to implement failure detection is to use gossip algorithms. Nodes that are running the critical processes gossip periodically information relative to the process' state. Gossip algorithms are used to disseminate the information and make available the relevant information to the nodes that check the process' state [19]. We argue that our content based routing algorithm can be used instead of the gossip algorithm. Conceptually this is clear because both algorithms are used to disseminate the information. From the point of view of performance, in particular in harsh environments, it will be interesting to compare the performance of the algorithms. The algorithm that we present here does not manage the failure of the nodes that compose the routing path. Of course, this is a key point of gossip algorithms and further work will be dedicated to the comparison and investigation for merging the strategies. We mention that complex failure detectors combine gossip algorithms with group membership algorithms [17, 13].

3 Content Based Routing Algorithm

We start by presenting our heuristic algorithm for matching publications with subscriptions. A classical assumption in the field of wireless networks is to consider that the nodes are deployed in a region that is planar or can be approximated locally by a plan. Moreover, the wireless links satisfy a proximity assumption in the sense that nodes that are connected are supposed to be close to each others. Of course, this is valid in a first approximation but anyway this is a classical approximation that leads to the definition of the **Unit Disk Graph** (UDG). To each node u we associate a range of communication r_u and node u can transmit to node v if the distance between u and v, $d(u, v)$, satisfies $d(u, v) < r_u$. In general, the range of communication is uniform and denoted by r. We hope to handle more general topologies using topological graph embeddings [1, 2].

In this setting, the heuristic algorithm tries to propagate the information following a straight line. Because nodes are located in a plane, if the publisher

and the subscriber disseminate the information in a straight line the two lines cross. In our research, the straight lines are approximated by Directional Random Walks that go hop-by-hop and try to go straight. The node where the two Directional Random Walks meet makes the correspondence between the publication with the subscription. It then stops the dissemination of information and sends back messages to the publisher and the subscriber to notify about the matching. We refer generically to both of them as the initiators. Along the path of the directional Random Walks, information is stored in order to make possible sending messages back to the initiator. Indeed, a node that is visited for the *first time*, memorizes the description of the information together with the ID of the node that transmitted the information first. When messages are transmitted back to the initiator this ID is used, i.e. while a node receives a piece of information to route, it checks in its routing table the ID that it has associated to the information. The use of this method ensures that the path that goes back to the initiator is **loop-erased**.

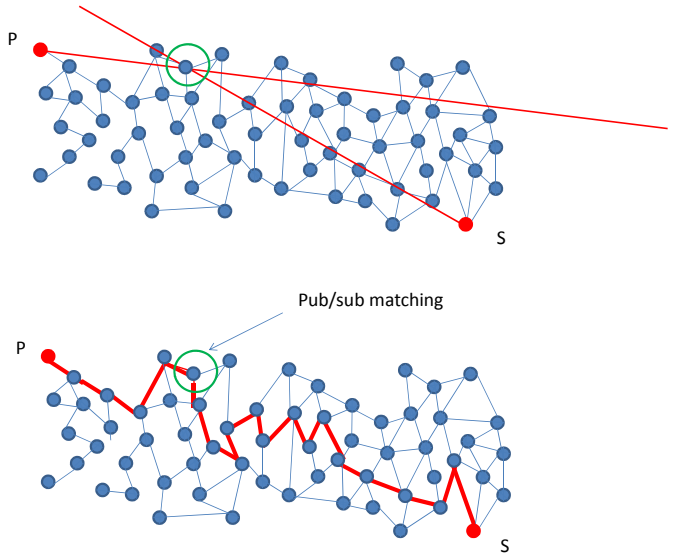

Fig. 1. Illustration of the heuristic leading to disseminate the information using Directional Random Walks. On the top of the Figure, the two nodes P and S corresponding to the publisher and subscriber of the information are shown. The information is disseminated in a straight line. At the node where the two lines intersect (the Rendez-vous point) the node matches the publication with the subscription. Messages are sent back to P and S to inform about the matching. The intermediate nodes memorize the trace of the information to route data subsequently. At the bottom of the Figure, an example of a Directional Random Walk, an approximation of a straight line.

To propagate the information following a straight line, we use the fact that the communication graph is approximated with a Unit Disk Graph. Assume that a node x propagates the information to node y that must choose a subsequent

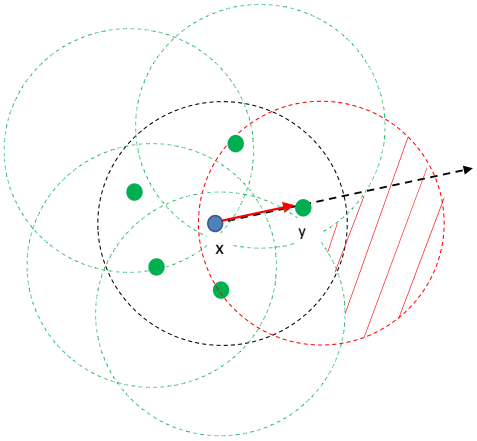

Fig. 2. Local construction of the Directional Random Walk. Node y received the information to disseminate from node x. Node y selects a subsequent node by favoring the nodes located in the hatched region. Computationally, this is done by selecting the node z with the less 2-hops paths from x to z.

node that is close to the straight line $\overline{xy}$, see Figure 2. We assume that node y knows the two hops neighborhood. Formally, we denote $\mathcal{N}(y)$ as the nodes that are connected to y, i.e. $\mathcal{N}(y) = \{z \mid d(y,z) < r\}$ and y knows the set of nodes $\cup_{z \in \mathcal{N}(y)} \mathcal{N}(z)$. On Figure 2, we represent the nodes x, y and 4 unlabeled nodes that are neighboring nodes of x. The heuristic to force y to propagate the information following a straight line is that y chooses a next node that is not a neighboring node of x and that has as few as possible neighboring nodes in common with x. This amounts to choose a node that belongs to the hatched area.

Unfortunately it can be that this area does not contain any node. To handle this situation, node y considers all the nodes $z \in \mathcal{N}(y)$ and counts the number of 2-hops paths from x to z, denoted n^y_{xz}.

$$n^y_{xz} = \mid \{v \mid v \in \mathcal{N}(x) \cap \mathcal{N}(z)\} \mid$$

Node y selects as next node to disseminate the information the neighboring nodes $z \in \mathcal{N}(y)$ with minimal n^y_{xz}. It breaks the ties with uniform probability. This mechanism is complemented with two heuristic mechanisms that favor the propagation in straight line and make sure that the Directional Random Walk eventually visits all the nodes. The pseudo code of the algorithm proposed is presented on Figure 3. The first mechanism is to increase the weight of the variable n^y_{xz} if $z \in \mathcal{N}(x)$ to force the information to get away from x, see the initialization on Figure 3. The second one is that after node y transmits to z, the variable n^y_{xz} is incremented to make possible the selection of another path if the data come back to y through x again, see line 4. This ensures that the Directional Random Walk eventually visits all the nodes.

The properties of the Directional Random Walk are the following:

Construction of the Directional Random Walk using the 2-hops neighboring nodes
Initialization : Each node y initializes a variable
$n^y_{xz} = \mid \mathcal{N}(x) \cap \mathcal{N}(z) \mid + deg(x) * 1_{z \in \mathcal{N}(x)}$, for $x, z \in \mathcal{N}(y)$

1. The initiator x selects a random node y uniformly in $\mathcal{N}(x)$.
2. **if** y is visited by the directional Random Walk the first time **then** y sets a pointer $back = x$ and associates the description of the information.
3. y selects $z \in \mathcal{N}(y)$ with minimal n^y_{xz}, breaks the ties with uniform probability.
4. y updates $n^y_{xz} = n^y_{xz} + rand$, with $rand \in [0, deg(x)]$.
5. $x = y$, $y = z$, **go to** 2.

Fig. 3. Pseudo code for building a Directional random walk. The pseudo code here is a centralized version of the algorithm. The distributed reactive implementation follows straightforwardly.

- The Directional Random Walk eventually visits all the nodes in the network.
- The backward path is loop-erased. This means that there is a direct path between publishers and subscribers.

4 Evaluation of the efficiency of the directional Random Walk (DRW)

The communication networks that we use for our numerical evaluations of the Directional Random Walk (DRW hereafter) are obtained by placing the nodes randomly and uniformly in a squared area of unit area. The communication model is defined by the range of communication r. Two nodes that are closer than the range of communication can communicate. The graph we obtain in this way is the Unit Disc Graph (UDG). We conduct numerical validation for 1500, 2000, 2500, 3000 nodes with a range of communication r=0.04. The average number of neighbors is 7.5, 10, 12.5, 15. Notice that before starting the simulations we check that the communication graph is connected, if not, we redraw a random graph. Notice that, under these conditions, it is hard to obtain connected networks with less than 1500 nodes. To evaluate the performance of the construction of the DRW, we consider different measures:

- **Time to intersection.** Two nodes are chosen randomly and uniformly and each one initiates a DRW. We measure the time to intersection of the two DRW. The two DRWs are progressing at the same speed.
- **Impact of asynchrony**, i.e. the two DRW do not progress at the same speed. This measures the efficiency of the collaboration to build the path system, i.e. one path executes 1 step while the other one k steps with $k = 1, 10, 100$.
- **Nodes load.** We consider the basic problem of routing a permutation, i.e. to each node we select randomly and uniformly another node in the network.

Each such pair of nodes initiate two DRW (one as a publisher, one as a subscriber, this is pure convention). We measure for each node in the network the total number of paths that go through it to route the permutation.

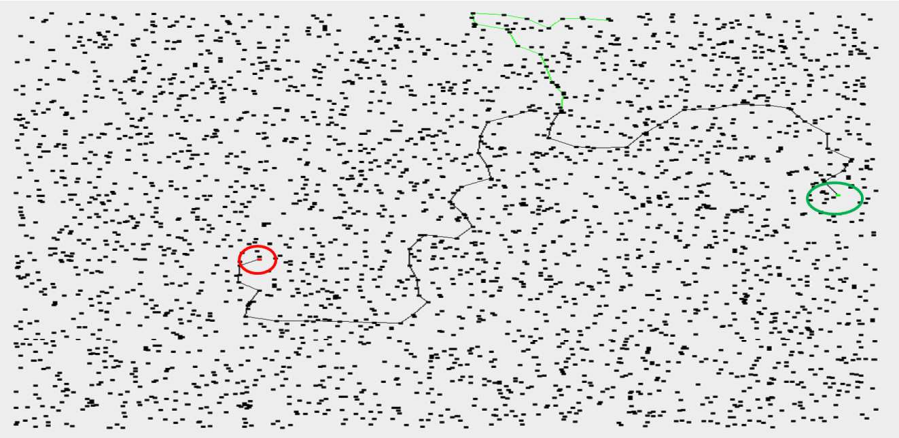

Fig. 4. An example of the building of a path. The nodes that are circled start to build two paths independently. At the intersection point the **loop-erased** path linking the two nodes is built and displayed in the figure. We can observe that on this figure that the DRW continues the exploration of the space.

4.1 Time to intersection

To measure the time to intersection we selected two nodes randomly and uniformly. The two nodes start the construction of a DRW synchronously, i.e. The two paths are constructed at the same speed, progressing step-by-step synchronously. In order to compare our strategy, we proceed to the same experiments but using (pure) Random Walks instead of the DRWs. To build a path a Random Walk selects as subsequent node any neighboring node of the current node with uniform probability.

The results with DRW are shown on the left of Figure 5 while the results with the Random Walk are shown on the right. The comparison between the two sets of results show clearly that the DRW improves the time to intersection. The median and mean are smaller with a multiplicative factor of 6-10. It is also relevant that the results obtained with the DRW are much more concentrated than the ones obtained using a Random Walk.

We conclude that **directionality** decreases the time to intersection.

It is relevant to notice that the results on Figure 5 show that the time to intersection seems to converge as we increase the density. An explanation for the occurrence of this phenomenon can be that the length of the steps decreases

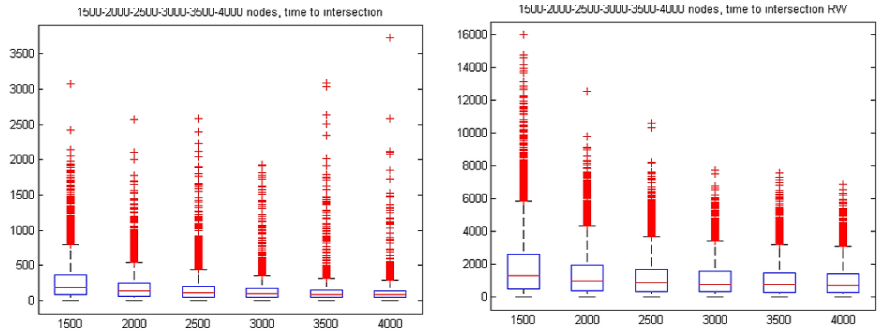

Fig. 5. Boxplots of the results of the numerical measure of the time to intersection. The number of nodes is 1500, 2000, 2500, 3000, 3500 and 4000. In the y-axis the time to intersection is the total number of steps before the intersection of the two paths. For each boxplot we generated 100 random networks and for each network we measure the time to intersection for 100 pairs of nodes chosen randomly and uniformly. On the left the results with the Directional Random Walk and on the right with the Random Walk.

in average for both Random Walks. Then as the density increases the time to intersection tends towards a limit that corresponds to a 'continuous' case (in average). The exact values can be read on Figure 6 in the column corresponding to $ratio = 1$.

4.2 The impact of asynchrony

The next set of experiments measures the efficiency of the collaboration of the publisher and subscriber in building the path between them. We then measure the time to intersection under the same conditions than in section 4.1 and we introduce asynchrony. More precisely, while one path grows of one step, the other one grows of k steps. We measure the total number of steps before intersection with $k = 1, 10, 100$. The results are presented in Figure 6 by using the DRW and the Random Walk. The numerical results show clearly that making the two path progressing simultaneously decreases the time to intersection, i.e. we measure the smallest time to intersection with $k = 1$. We observe this independence on the way the path is built. We emphasize that the numbers reported here are the total number of steps.

We conclude that the **cooperation** between the publisher and subscriber decreases the time to intersection.

4.3 Efficiency of the path system

We run a set of experiments and consider three different ways of building the paths: Directional Random Walk (DRW), (pure) Random Walk and shortest path.To consider a complete path system we route a permutation, i.e. to each node we assign a destination chosen uniformly among the set of nodes and a

node is selected only in one pair. In a pair, each node plays the role of publisher and subscriber, this means that with n nodes we build n paths (two paths per pair). The results are presented in Tables 7, 8 and in Figure 9.

We observe that the Random Walk is the best strategy to balance the load, the mean and median are very close in all situations. The DRW is also good at balancing the load, we remind that the time to intersection is incomparably better. While with the shortest path, which is centralized and not realistic but included for comparison, we obtain that few nodes are over-used and route close to half of the traffic.

We conclude that the DRW is good at **balancing** the load.

	DRW			Random Walk		
Number of nodes\Ratio	1	10	100	1	10	100
1500	192/265	267/364	431/602	1267/1842	1700/2579	2668/4264
2000	144/188	192/260	314/433	956/1327	1319/1897	2018/3061
2500	116/155	160/213	279/364	830/1156	1099/1604	1762/2599
3000	102/130	143/186	227/301	756/1056	1077/1508	1647/2447

Fig. 6. Impact of asynchronicity (difference of speeds) on the time to intersection. The values in the table are Median/Mean time to intersection. We run experiments with networks of 1500, 2000, 2500 and 3000 nodes, communication range $r = 0.04$ and $k = 1, 10, 100$. In each configuration, we generated 100 connected networks and on the top of each network we selected 100 pairs of nodes for measuring the time to intersection.

With this set of experiments, our goal is to measure the efficiency of the path system obtained with our algorithm in terms of the node's load, i.e. the number of paths that go through a node. We remind that when we construct paths a *back* pointer and the description of the information are associated and memorized by the node but just the first time that the node is selected for the path. It is then crucial that the number of paths through a node stays reasonable and accordingly to the memory constraints.

5 Conclusion and Further work

In this paper we sketch how publish/subscribe systems can be used to implement failure tolerance and detection. Then, we propose a heuristic to build such systems in wireless networks using Directional Random Walks. Our numerical findings reflect that Directional Random Walks decrease the time to intersection of the traces compared to classical random Walks. In addition to this, the cooperation of publishers and subscribers decreases the time to intersection. Finally, it

Total number of Nodes	1500	2000	2500	3000	3500
Shortest path	3/73	2/66	2/65	2/65	2/60
DRW	54/62	56/63	56/64	55/63	54/63
Random Walk	62/65	72/73	79/79	85/84	91/89

Fig. 7. Median/Mean Nodes load for routing a permutation

Total number of Nodes	1500	2000	2500	3000	3500
Shortest path	899/2	1068/2	1348/2	1725/2	1861/2
DRW	366/2	330/2	364/2	463/2	453/2
Random Walk	253/2	201/2	205/2	200/2	221/2

Fig. 8. Max/Min Nodes load for routing a permutation

has been shown that Directional Random Walks balance the nodes load almost as efficiently as pure Random Walks.

Furthermore, we suspect the existence of a limit process, as the density increases, that provides a lower bound to the time to intersection. The investigation of this limit process is of real practical interest. Indeed, such a lower bound provides an efficient criteria to decide when to stop the path because the fact that no intersection occurs means that there are no events to match in the network. Actually, the problem of stopping the Directional Random walk is not addressed in this paper that is focused on proving the efficiency of the heuristic. Further papers are planed to compare the algorithm with classical techniques. In particular, it is tempting to think merging the Directional Random walk with gossip mechanisms in order to manage node failures or mobility efficiently as well as improving the gossiping techniques. In this direction see for instance [10] that makes use of a kind of learning algorithm for gossiping. To conclude, it must be said that further work will be done in order to handle more general topologies using topological graph embeddings.

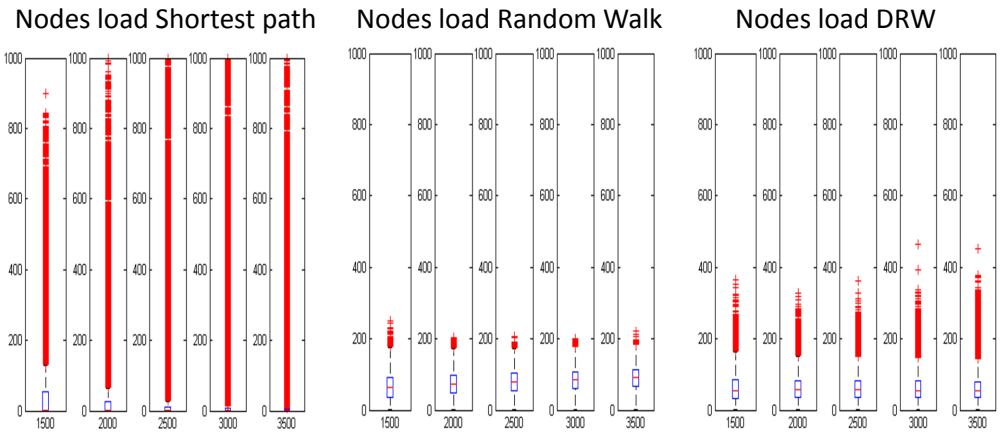

Fig. 9. Comparison of the nodes load for the three different algorithms to build the path system: shortest path, Directional Random Walk and pure Random Walk. We consider networks of 1500, 2000, 2500 and 3000 nodes. The range of communication is $r = 0.04$. For each configuration we generated 10 connected networks and on the top of each connected network we measured the nodes load for 10 different permutations. We built a total of more than 150.000 paths per configuration.

6 Acknowledgment

This work has been developed as part of the POPWiN project (Parallel Object Remote Programming for Heterogeneous Wireless Networks over IPv6) that is financially supported by the Hasler Foundation in its SmartWorld - Information and Communication Technology for a Better World 2020 program.

References

1. Maia Fraser. Local routing on tori. *Ad Hoc & Sensor Wireless Networks*, 6(3-4):179–196, 2008.
2. Maia Fraser. Local routing in graphs embedded on surfaces of arbitrary genus. *CoRR*, abs/1202.6109, 2012.
3. Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless sensor networks. *IEEE Distributed Systems Online*, 4(11), 2003.
4. Jonathan L. Gross and Thomas W. Tucker. *Topological graph theory.* Wiley-Interscience, New York, NY, USA, 1987.
5. S. Helmer, A. Poulovassilis, and F. Xhafa. *Reasoning in Event-Based Distributed Systems.* Studies in Computational Intelligence Studies in Computation. Springer, 2011.
6. Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Proceedings of the 4th International Conference*

on Mobile Data Management, MDM '03, pages 122–140, London, UK, UK, 2003. Springer-Verlag.

7. Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s - a publish/subscribe protocol for wireless sensor networks. In *COMSWARE*, pages 791–798, 2008.
8. Pierre Leone, Aubin Jarry, Sotiris E. Nikoletseas, and José D. P. Rolim. Optimal data gathering paths and energy-balance mechanisms in wireless networks. *Ad Hoc Networks*, 9(6):1036–1048, 2011.
9. Pierre Leone, Sotiris E. Nikoletseas, and José D. P. Rolim. Stochastic models and adaptive algorithms for energy balance in sensor networks. *Theory Comput. Syst.*, 47(2):433–453, 2010.
10. Meng Lin and Keith Marzullo. Directional gossip: Gossip in a wide area network. Technical report, La Jolla, CA, USA, 1999.
11. Gero Mühl, Ludger Fiege, and Peter R. Pietzuch. *Distributed event-based systems*. Springer, 2006.
12. Peter R. Pietzuch and Jean Bacon. Hermes: A distributed event-based middleware architecture. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCSW '02, pages 611–618, Washington, DC, USA, 2002. IEEE Computer Society.
13. Sridharan Ranganathan, Alan D. George, Robert W. Todd, and Matthew C. Chidester. Gossip-style failure detection and distributed consensus for scalable heterogeneous clusters. *Cluster Computing*, 4(3):197–209, July 2001.
14. Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with ght, a geographic hash table. *MONET*, 8(4):427–442, 2003.
15. Karim Seada and Ahmed Helmy. Geographic rendezvous-based architectures for emergency data dissemination. *Wireless Communications and Mobile Computing*, 10(9):1221–1237, 2010.
16. Eduardo Souto, Germano Guimarães, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, Carlos Ferraz, and Judith Kelner. Mires: a publish/subscribe middleware for sensor networks. *Personal Ubiquitous Comput.*, 10(1):37–44, December 2005.
17. Ann T. Tai, Kam S. Tso, and William H. Sanders. Cluster-based failure detection service for large-scale ad hoc wireless network applications. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, DSN '04, pages 805–, Washington, DC, USA, 2004. IEEE Computer Society.
18. Andrew S. Tanenbaum and Maarten van Steen. *Distributed systems - principles and paradigms (2. ed.)*. Pearson Education, 2007.
19. Robbert van Renesse, Yaron Minsky, and Mark Hayden. A gossip-style failure detection service. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, Middleware '98, pages 55–70, London, UK, UK, 1998. Springer-Verlag.
20. Xiaokang Yu, Xiaotian Yin, Wei Han, Jie Gao, and Xianfeng Gu. Scalable routing in 3d high genus sensor networks using graph embedding. In *INFOCOM*, pages 2681–2685, 2012.
21. Fenghui Zhang, Hao Li, Anxiao Jiang, Jianer Chen, and Ping Luo. Face tracing based geographic routing in nonplanar wireless networks. In *INFOCOM*, pages 2243–2251, 2007.