

Toward Scalable Cut Vertex and Link Detection with Applications in Wireless Ad Hoc Networks

Ivan Stojmenovic, University of Ottawa

David Simplot-Ryl, University of Lille 1

Amiya Nayak, University of Ottawa

Abstract

Ad hoc networks are expected to have some critical connectivity properties before partitioning. Timely partition prediction signals action for improving fault tolerance and performing some data or service replication so that the network can continue functioning after partition does occur. This article surveys existing prediction concepts and discusses their scalability, simplicity, correctness, speed, communication overhead, and applications. Existing centralized algorithms declare an edge or a node as critical if its removal will separate the network into several components. Several localized definitions of critical (or cut) nodes and links, and removable nodes, are demonstrated to be simple, useful, and scalable. A node is critical if the subgraph of p -hop neighbors of node (without the node itself) is disconnected. A link is critical if its endpoints have no common p -hop neighbors (assuming that the link between them does not exist). Definitions are extended toward local k -connectivity. The false positives mostly occur when alternative routes exist but are relatively long, and therefore may not provide satisfactory service in applications. Therefore, localized protocols provide faster and often more reliable partition warnings for possible timely replication decisions. This conceptual advance provides ingredients for establishing and restoring biconnectivity.

The concepts discussed here are applicable to other networks, but are most useful for wireless ad hoc networks (WAHNS). This includes infrastructureless node formations for disaster relief, conference, wireless office, and battlefield, and wireless robot, actuator, and sensor networks connected to infrastructure. Wireless sensor networks monitor the environment and may be assisted by wireless actuator or robot networks that may act on both sensors and environment.

Connectivity of WAHN is a minimal requirement for their functionality in many applications. However, their decentralized and self-organizing nature creates the presence of critical nodes and links, whose failures may partition the network or create communication bottlenecks. Some algorithms [1] attempt to improve the communication without avoiding or delaying partitioning (*post-partitioning* approaches). For example, losing connection to certain destination in TORA routing protocol indicates possible partitioning and action to reconnect the network. This can, however, be too late in many scenarios, and our goal is to provide partition signs sufficiently ahead of actual partition so that the network is either reconfigured or ready to recover (proactive approaches).

A node or link is *critical* if its removal will disconnect the graph into two (or more) separate components. A critical node is often referred in literature as the *cut vertex*, and these two terms will be used interchangeably. Suppose we have a user E and a server L (Fig. 1.). When user E requests a ser-

vice or data from L , the request follows the path $ECBAJLL$. Nodes A , B , and J , and link AB are critical, since removal (or failure) of any of them will partition the network. Node E may initiate some actions, such as replicating data from L in its own memory, or look for alternative service in different part of network, to withstand possible partition.

The depth first search (DFS) algorithm was used to detect critical nodes and links in [2, 3]. It is a centralized algorithm, and can be also implemented in globalized distributed manner. A centralized algorithm requires that a node should be aware of global topology. For WAHNS, this method is not scalable since it involves a quadratic (in number of nodes) communication overhead in order to update link information when nodes are moving, or change their status from active to sleeping and vice versa.

A network is called *biconnected* if the removal of any node does not disconnect it. Moving some nodes (with minimal distance traveled) to biconnect the network so that the network does not have any critical link or node is a difficult problem, even for a centralized solution [4], and especially so if further constraints (e.g., covering an area) are added. In concrete applications, some other related concepts need to be introduced. For large networks, solutions where nodes *declare themselves as critical based on limited locally available knowledge* are needed. The primary goal of this article is to survey such solutions and discuss their properties. Each of them has natural accuracy limitation. A link/node that appears locally

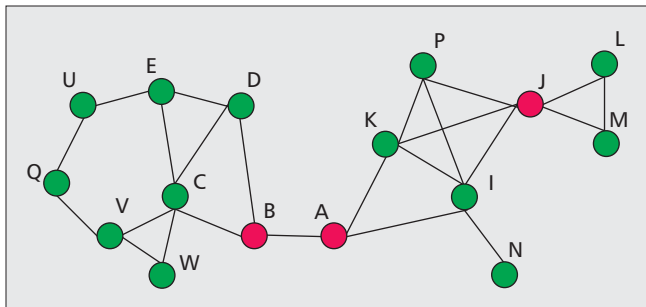


Figure 1. Critical nodes A , B , and J , and critical link AB .

critical may in fact be globally non-critical when more network information is added. Existing applications [4–8], however, demonstrate that there exist a satisfactory trade-off between the communication overhead to gather limited local knowledge and practical usefulness of derived criticality conclusions. The *excess* work caused by declaring more critical nodes is reasonable, and has a side effect of improving service quality.

Localized algorithms are distributed in nature and resemble greedy algorithms, where simple local behavior achieves a desired global objective. In a localized algorithm each node makes a decision to determine critical nodes or links based only on limited local knowledge. Existing applications show every possible partition can be detected by localized algorithms, therefore *greatly reducing communication overhead and the speed of detecting*, allowing network to replicate data or service in time if/when needed.

The next section describes centralized solutions and corresponding applications for data replication. We then discuss localized concepts and algorithms for detection of critical links/nodes, and localized detection of k -connectivity. We conclude this article by discussing some other possible applications and open issues and challenges.

Centralized Detection of Cut Vertices and Links and Data Replication

A straightforward algorithm for detecting critical nodes may consider, for each node A , the subgraph obtained by its removal and removal of all its adjacent edges, and testing whether this subgraph is connected. If it is not connected, the corresponding node A is a critical node. As a consequence, node that has only one neighbor is not critical (e.g., node N in Fig. 1).

A faster centralized algorithm for detecting critical nodes was described in [2, 3], who used DFS. While executing DFS on an undirected graph, we start at an arbitrarily chosen node, which becomes the root. We keep traversing fresh edges and mark nodes as visited; on the way we keep pushing nodes into a stack data structure. This process is continued until we reach a node, which is only connected to already visited nodes. At this point we keep backtracking up to a vertex, which has edges connecting them to nodes which have hitherto not been visited. With a little thought it can be seen that such a node will always be a critical node of the graph. Alongside the identification of the critical node, it is easy to pop the downstream nodes from the stack into a set, which corresponds to a bi-connected component. Since the above steps can be executed during DFS in the same pass, identification of critical nodes takes only linear time in number of nodes.

Critical link is the link connecting two critical nodes so that when this link is eliminated from the graph, the graph becomes disconnected. Note that two neighboring critical nodes may have alternate path between them. In [3] once critical links in WAHN are detected, two ways are proposed to delay or avoid

their failure: changing the trajectory of one or both nodes forming the critical link and bringing another node to reinforce it. Increased delivery rates were reported due to prolonged network connectivity.

Authors in [9] aimed at enhancing data access in WAHNS by detecting partitions in it and replicating data. Each node of a connected group knows the behavior of other members in the group, because each node computes its velocity, and spreads that information to other nodes. It is then possible to predict when a node will leave its group. The *node-to-leave* picks another node of the group to be a host of the data and performs a data replication on that node. Each node predicts when the partition will occur. However, a positioning system is needed. Next, the network is continuously and relatively highly loaded due to information exchange among nodes. While general ideas in [9] are good, the essential details are missing. One can even assume the whole connected network to be a group, since it also satisfies the vague definition given. The protocol for predicting link breakage, based on predicting future locations and connectivity of two nodes, is described in detail. However, there is no description of any protocol to detect group partitioning. For example, a pair of neighboring nodes leaving the group together does not trigger replication, since the protocol requires partitioning from each remaining group member.

Three replica allocation methods, assuming that each item is not updated, are proposed in [10]. These methods take into account the access frequency and the status of network connection. One of the methods groups mobile hosts into biconnected components by a centralized algorithm that is periodically run. In the algorithm each mobile host broadcasts its ID. One coordinator in each connected component is then deciding about biconnected components, and informs other nodes. Several more centralized partition detection and data replication algorithms are reviewed in [1, 11].

Localized Cut Vertices and Links and Data Replication

We first define what the local knowledge available to nodes is and how nodes gain it. We use the notion of p -hop knowledge. Two nodes are considered to be p -hop neighbors if and only if the shortest route between them has p or less hops. Awareness of itself only is represented as 0-hop knowledge. Nodes collect p -hop knowledge by sending hello messages to its neighbors containing the graph of their $(p - 1)$ -hop neighbors. To reduce overhead during *hello* message cycle for $p = 1, 2, 3, \dots$, only nodes and edges not reported previously can be sent. Thus, 1-hop knowledge is a list of direct neighbors. We discuss primarily topological information, not involving geographic coordinates of nodes. If position information is available, then the p -hop information is obtained by transmitting merely the lists of $(p - 1)$ -hop neighbors. Starting from $p = 2$, p -hop topological information, and corresponding subgraph of p -hop neighbors, include all existing links between a p -hop and a $(p - 1)$ -hop neighbor, but not information on whether or not two strictly p -hop neighbors are connected. If position information is used, then the existing links between p -hop neighbors are also learned. The corresponding definitions of critical nodes and links using topological and positional information are same, except applying them for corresponding p -hop graphs. Details are in [1, 12]. The applications described here use value $p = 1$ with positional and $p = 2$ with topological information. Thus, localized algorithms surveyed here discover all critical nodes and links very quickly and with very small communication overhead. However, these

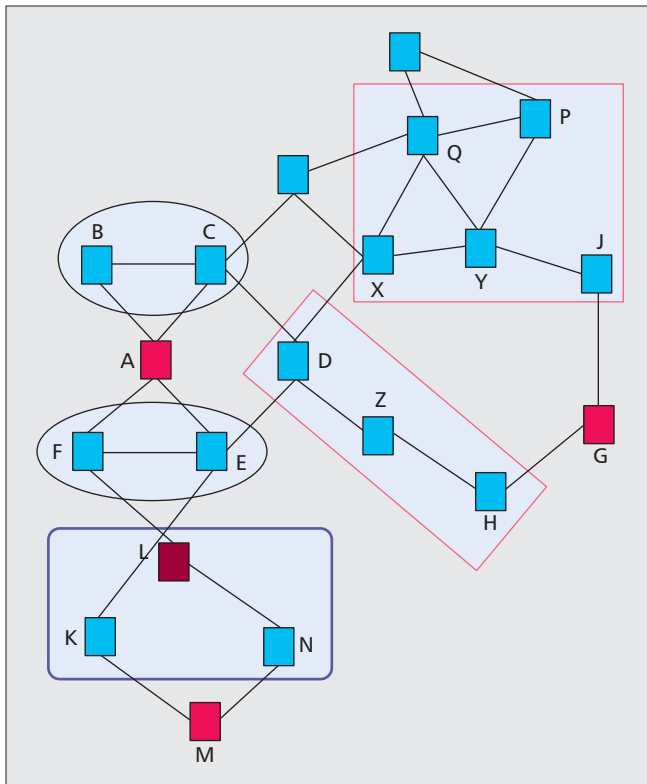


Figure 2. Nodes *A* and *M* are 1-hop critical, node *G* is 3-hop critical.

algorithms may detect some nodes and links as critical although they may not be globally critical. This is unavoidable since local knowledge only is used, and with such restriction it is impossible for a node to learn about alternate connections in different parts of the network. On the other hand, in applications, often long alternate paths do not provide satisfactory service; thus, a localized method may provide even more useful decisions.

Criticality Detection for Post-Partition Connectivity Restoration

Authors in [5, 6] have proposed algorithms to efficiently restore actor connectivity. Actors dedicate one of their neighbors to replace them in the event of failure. In [6], the neighbor with the least node degree (e.g., one with no additional neighbors) is selected; in case of a tie, the closest one; the ternary key is its ID. The authors in [6] also discuss the maintenance of biconnectivity. The approach in [5] is based on the concept of the connected dominating set (CDS) of the network [5]. A neighbor not in the CDS is preferred. These neighbors may in turn *pull* their dedicated neighbors in a cascaded fashion. In a dominating set each node either belongs to it (a dominator) or has a neighbor in it (a dominee). The authors in [5] observe that the connectivity of the network can be maintained as long as CDS is maintained by replacing a failed dominator with a nearby dominee. The CDS construction algorithm [13] requires two-hop neighbor knowledge and may be implemented without any message exchange beyond *hello* messages. If a node is a dominee, it cannot be a cut vertex since its absence will not violate the connectivity of the network [5]. A dominator node that has a dominee neighbor with degree one (no other links) will be a cut vertex because it will partition that neighbor in case of failure. Otherwise, one of the neighbors of CDS node *u* starts a local DFS to look for all the remaining neighbors of *u*. If all of them can be reached, *u* is not a cut vertex; otherwise, it

will declare itself as a cut vertex. The authors in [5] observed an increased message overhead depending on the network topology, and suggested few options to restrict the search scope. In the first option, *u* will be simply declared a cut vertex. This means all dominators without neighboring degree one dominees are cut vertices, which is often incorrect. An alternative is to perform DFS if *u* does not have any dominee within two hops. However, such dominees may exist when *u* is cut vertex (e.g., *u* has two dominee neighbors that are neighbors themselves but not to any other node except *u*) while *u* is declared non-cut vertex, thus failing the purpose. Furthermore, it is not clear which nodes are included in DFS. To avoid cyclical cascaded movement, nodes that moved once will not move again to fix the failure of the same node. In an improved algorithm, node *u* receives from all its neighbors the cost of using their closest dominee in a two-step replacement, and chooses the best one. Loops are avoided but message cost increases. Overall, the cut vertex detection algorithm misses some cut vertices or involves excessive messaging, which is completely avoided by algorithm from [1].

If a partition is to occur, PADRA [5] designates a failure handler to initiate the connectivity restoration process, localize the scope of the recovery and minimize the overhead imposed on the nodes. PADRA is further expanded to handle multiple node failures, by providing a mutual exclusion mechanism in repositioning the nodes to restore connectivity [5]. The algorithms in [5, 6] have temporary network disconnection while actors move, between failure of a cut vertex actor, and completion of the cascaded movement to restore connectivity.

The algorithm in [8] first detects the actors whose absence does not lead to any partitioning by algorithm [1], and classifies them into dominators and dominees by [13]. If the actor is critical and there is any dominee node (*v*) among neighbors, it sends a message to *v* to notify that it must replace the failed actor in case of failure. If there is not any such actor, the cut vertex sends a request message to all its neighbors and asks them to send the maximum allowable movements that they can make (towards the failed actor) while preserving connectivity to their neighbors. The cut vertex calculates whether or not the neighbors can reconnect the network in the absence of cut vertex. If not, a message is sent to the nearest neighbor of the cut vertex to replace the cut vertex in case of failure. Thus, a failed actor is not always replaced by other actors in a cascaded movement. Instead, neighbors collaborate with each other and restore the network with minimum latency.

Detection of Critical Nodes Based on *p*-hop Knowledge

For each node *A*, consider a subgraph of *p*-hop neighbors of *A*, where *A* and all its incident edges are excluded. *A* is a *p*-critical node if the corresponding subgraph of *p*-hop neighbors of *A* is disconnected [1]. Clearly, if a node is globally critical, the localized algorithm will detect it as such.

We now discuss particular cases and give some examples as illustration. For $p = 1$, if only topological information is used, no links between neighbors exist in the decision graph, therefore all nodes are declared critical. This is obviously not very helpful. However, the corresponding definition that uses positional information and possible links between neighbors is very useful (e.g., connected dominating sets [13] can be applied, as in [5]).

For brevity, we illustrate mainly definitions with topological knowledge. In Fig. 2, 1-hop neighbors of *A* can be divided into two (circled) sets $\{B, C\}$ and $\{E, F\}$, which are disconnected. Therefore, *A* is 1-hop critical. Node *A* is not 2-hop critical since its 2-hop neighbor *D* will connect two sets. Node

M is 1-hop critical since its 1-hop neighbors K and N are not connected. However, it is not 2-hop critical since its 2-hop neighbors K , N , and L create a connected subgraph (marked by a rectangle). Node G is 3-hop critical since its 3-hop neighbors are divided in two subgraphs with vertices $\{H, Z, D\}$ and $\{J, Y, X, Q, P\}$ (the two sets are enclosed by polygons) which are disjoint.

Teams of multiple mobile robots may communicate with each other using WAHNS. Fault tolerance in communication can be achieved by making the communication network bi-connected. The authors in [4] presented a localized protocol for constructing a fault-tolerant bi-connected robotic network topology from a connected network, in such a way that the total movement of robots is minimized. The proposed distributed algorithm uses p -hop neighbor information to identify cut vertices. Critical head robots are cut vertices having one or more neighbors that are not cut vertices and therefore mobile. They direct one or two non-cut nodes to move and bi-connect their neighborhood. Simulation results show that the total distance of movement of robots decreases significantly (e.g., about 2.5 times for networks with density 10) with this localized algorithm when compared to an existing centralized one. Although being very successful on average, proposed localized algorithm does not guarantee bi-connectivity, may partition the network, and may even stop at connected but not biconnected stage.

The localized cut vertex concept [1] was also applied in [7], and extended to define removable nodes. A removable node is a non-cut node such that the removal of it does not generate any new cut-node in the network. To establish bi-connectivity with minimal node movement, removable nodes are moved to new locations near cut nodes, so that cut-node becomes non-cut node. The following simple algorithm recognizes removable nodes. If all neighbors are included in the same block after the removal of node i then node i is removable node. Blocks can be identified by running DFS algorithm [2] on p -hop neighborhood graph without node i [7]. Alternatively, one could run cut vertex determination algorithm [1] on $(p - 1)$ -hop neighborhood of each neighbor j , without i and j , and without j only, and check if any new cut vertex has been created. Removable node i will make node j a non-cut node if it, after movement, connects to all connected components of the local topology of node j . Movement to centers of minimal enclosing circles of node sets containing one node from each connected component of node j are considered as candidates, with closest one taken as movement destination. Figure 3 illustrates the concept and its application.

Critical Links Based on Common p -hop Neighbors

AB is a (topologically) p -critical link if the sets of p -hop neighbors of A and B (assuming that the link AB does not exist) are disjoint [1]. When $p = 1$, AB is 1-critical link if A and B have no common neighbors.

The graph in Fig. 4 contains only one globally critical link, HI . Localized algorithms will detect some more critical links. For example the link AB as 1-critical, because nodes A and B have disjoint 1-hop neighbors (C and D). The link XY is declared as 2-critical, because the 2-hop neighbors of X (H, G, I , circled) and Y (D, C, B, E , enclosed in a quadrilateral), are disjoint. Nevertheless, the link XY is not 3-critical because the 3-hop neighbors of X (H, I, O, T, S, G, F) and Y (D, C, A, B, E, F) are not disjoint. The link KL is 3-critical, because the 3-hop neighbors of K (O, P, M, T, S, I, H) and L (R, Q, N), both enclosed in quadrilaterals, are disjoint.

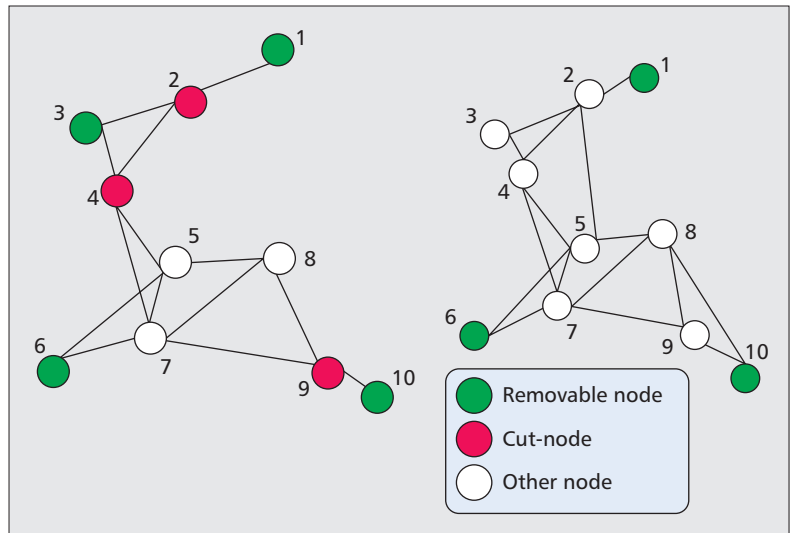


Figure 3. Cut nodes, removable nodes, and minimal movement of nodes 1, 3, and 10 to bi-connect.

This definition of critical link [1] can be rephrased in the way it was applied in [14] to decide when to apply data replication. Node A sends replica to a neighbor B whenever A detects only a single route to B , i.e., when A cannot detect a loop including itself and B . This means that loops of length up to $2p$ hops can be detected since there exists a node belonging to p -hop neighborhood of both A and B . In Fig. 3, node F in the loop $XHGFEYD$ belongs to the 3-hop neighborhood of both X and Y .

k -connectivity

The network or graph is k -connected if it remains connected after the removal of any $k - 1$ nodes. This is a desirable characteristic for reliable communication and load balancing. Fault tolerance in WAHN increases with increasing k for which the network is k -connected. The timely recognition of k -disconnectivity is important, to perform some data or service replication or sensor replacement or activation, before network becomes fault intolerant, leading to partitions.

Existing algorithms for testing k -connectivity are centralized. The algorithm [2, 3] can be extended to test global 3-connectivity of graph G as follows. For each node X in G construct graph $F = G - \{X\}$ (eliminate X and all edges ending in X from G). Run DFS based 2-connected criterion on F . If F is 2-disconnected for any X then G is 3-disconnected.

We describe three definitions and corresponding protocols for deciding local k -connectivity, based on minimum degree of all p -hop neighbors, k -connectivity of p -hop neighborhood, and $(k - 1)$ -connectivity of all p -hop neighbors and subgraph without the node itself [12]. For some definitions (e.g., the first one given here), global k -connectivity may imply local k -connectivity at each node. However, for other definitions, globally k -connected networks may have nodes, which are locally k -disconnected. This is due to additional connectivity to local neighbors being available by long routes in the whole network.

In the first local neighbor detection (LND) algorithm [12], each node verifies whether or not itself and each of its p -hop neighbors have at least k neighbors. In the second local critical node detection (LCND) protocol [12], it also tests if the subgraph of its p -hop neighbors is k -connected. The third local subgraph connectivity detection (LSCD) protocol [12] is based on communications between neighboring nodes to exchange local decisions starting from $k = 1$. All nodes declare themselves locally 1-connected. For $k = 2, 3, \dots$, iteratively, local decisions are propagated to p -hop neighbors. If node A is $(k - 1)$ -connected, all its p -hop neighbors are $(k -$

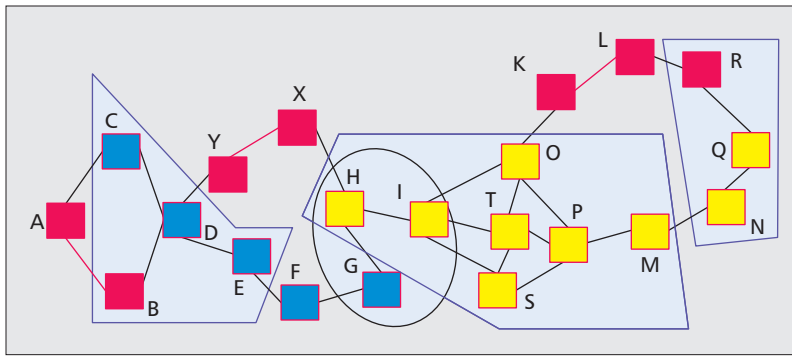


Figure 4. Critical link HI, 1-critical link AB, 2-critical link XY, 3-critical link KL.

1)-connected, and the graph consisting of p -hop neighbors of A (excluding A) is $(k - 1)$ -connected, then node A declares its neighborhood as k -connected.

Conclusions

We believe that the usefulness of the introduced definitions and corresponding localized algorithms can be demonstrated by additional applications. With suitable modifications and additional criteria, the proposed localized algorithms for detection of critical links and nodes can be applied to service replication. In this scenario, a particular route between a client and a server node is monitored for the presence of critical links or nodes. If such a node or link is detected, an alternate service in the network is searched, or the service is replicated. The proposed localized partition detection schemes may also be applied in sensor network scenarios. Sensors that detect their criticality or existence of critical links may report to a monitoring center, asking for deployment of additional sensors in the area, or waking up some nearby sleeping sensors. Localized partition detection is also useful for designing sensor activity scheduling protocols.

It is an interesting graph-theoretical problem to study the properties of the introduced definitions for a variety of network families. While experiments in [1, 12] show encouraging results with low false positives, there are no theoretical bounds for them.

Acknowledgments

This research is partially supported by NSERC Discovery grants and NSERC CRDPJ386874-09 grant on reliable and secure QoS routing and transport protocols for mobile ad hoc networks.

References

- [1] M. Jorgic *et al.*, "Localized Algorithms for Detection of Critical Nodes and Links for Connectivity in Ad Hoc Networks," *Mediterranean Ad Hoc Net. Wksp.*, Bodrum, Turkey, June 2004, pp. 360–71.
- [2] R. Tarjan, "Depth First Search and Linear Graph Algorithms," *SIAM J. Comp.*, vol. 1, no. 2, 1972, pp. 146–60.
- [3] D. Goyal and J. Caffery, "Partitioning Avoidance in Mobile Ad Hoc Networks using Network Survivability Concepts," *Proc. IEEE ISCC*, Taormina, Italy, July 2002, pp. 553–58.
- [4] S. Dasef *et al.*, "A Localized Algorithm for Bi-Connectivity of Connected Mobile Robots," *Telecommun. Sys.*, vol. 40, no. 3–4, Apr. 2009, pp. 129–40.
- [5] A. A. Abbasi, M. Younis, and K. Akkaya, "Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks," *IEEE Trans. Parallel Distributed Sys.*, vol. 20, no. 9, Sept. 2009, pp. 1366–79.
- [6] K. Akkaya *et al.*, "Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility," *IEEE Trans. Comp.*, vol. 59, no. 2, 2010, pp. 258–71.
- [7] Z. Yan *et al.*, "Fault-Tolerance in Wireless Ad Hoc Networks: Bi-Connectivity through Movement of Removable Nodes," *IEEE Trans. Vehic. Tech.*, to appear.
- [8] A. Zamanifar, O. Kashefi, and M. Sharifi, "AOM: An Efficient Approach to Restore Connectivity in Wireless Sensor and Actor Networks," *Int'l. J. Comp. Net. & Commun.*, vol. 1, no. 1, Apr. 2009, pp. 61–72.

- [9] S. H. Shah, K. Chen, and K. Nahrstedt, "Cross-Layer Design for Data Accessibility in Mobile Ad Hoc Networks," *Wireless Personal Commun.*, vol. 21, no. 1, Apr. 2002, pp. 49–76.
- [10] T. Hara and S. K. Madria, "Data Replication for Improving Data Accessibility in Ad Hoc Networks," *IEEE Trans. Mobile Comp.*, vol. 5, no. 11, Nov. 2006, pp. 1515–32.
- [11] A. Derhab and N. Badache, "Data Replication Protocols for Mobile Ad-Hoc Networks: A Survey and Taxonomy," *IEEE Commun. Surveys & Tutorials*, vol. 11, no. 2, 2nd qtr. 2009, pp. 33–51.
- [12] M. Jorgic *et al.*, "Localized Detection of k -Connectivity in Wireless Ad Hoc, Actuator and Sensor Networks," *IEEE ICCCN*, Hawaii, Aug. 2007.
- [13] F. Dai and J. Wu, "An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks," *IEEE Trans. Parallel Distrib. Sys.*, vol. 15, no. 10, Oct. 2004, pp. 908–20.
- [14] H. Hayashi, T. Hara, and S. Nishio, "A Replica Allocation Method Adapting to Topology Changes in Ad Hoc Networks," *DEXA 2005, LNCS 3588*, 2005, pp. 868–78.

Biographies

IVAN STOJIMENOVIC [F'08] (ivan@site.uottawa.ca) received his Ph.D. degree in mathematics. He has held regular and visiting positions in Serbia, Japan, the United States, Canada, France, Mexico, Spain, the United Kingdom (as Chair in Applied Computing at the University of Birmingham), Hong Kong, Brazil, Taiwan, and China, and is a full professor at the University of Ottawa, Canada, and an adjunct professor at the University of Novi Sad, Serbia. He has published over 250 papers, and edited seven books on wireless, ad hoc, sensor and actuator networks, and applied algorithms with Wiley. He is editor of over a dozen journals, Editor-in-Chief of *IEEE Transactions on Parallel and Distributed Systems* (since January 2010), and founder and Editor-in-Chief of three journals (*Journal of Multiple-Valued Logic and Soft Computing*, *International Journal of Parallel, Emergent, and Distributed Systems*, and *Ad Hoc & Sensor Wireless Networks*). He is one of about 260 computer science researchers with h-index of at least 40 and has over 7000 citations. He received three best paper awards and the Fast Breaking Paper for October 2003 from Thomson ISI ESI. He is a recipient of the Royal Society Research Merit Award, United Kingdom. He is an IEEE Communications Society Distinguished Visitor, 2010–2012. He received the Excellence in Research Award of the University of Ottawa in 2009. He has chaired and/or organized over 60 workshops and conferences, and served on over 200 program committees. He was program co-chair at IEEE PIMRC 2008, IEEE AINA '07, IEEE MASS '04 and '07, EUC '05 and '07–10, AdHocNow '08, IFIP WSAN '08, WONS '05, MSN '05 and '06, and ISPA '05, and '07; founded workshop series at IEEE MASS, ICDCS, DCOSS, WoW-MoM, ACM Mobihoc, IEEE/ACM CPSCoM, FCST, and MSN; and is/was Workshop Chair at IEEE INFOCOM 2011, IEEE MASS '09, and ACM MobiHoc '07 and '08.

DAVID SIMPLOT-RYL (David.Simplot@lifl.fr) received a graduate engineer degree in computer science, automation, electronic, and electrical engineering, and M.Sc. and Ph.D. degrees in computer science from the University of Lille, France, in 1993 and 1997, respectively. In 1998 he joined the Fundamental Computer Science Laboratory of Lille (LIFL), where he is currently a professor. He received the Habilitation degree from the University of Lille in 2003. His research interests include sensor and mobile ad hoc networks, mobile and distributed computing, embedded operating systems, smart objects, and RFID technologies. Recently, he mainly contributes to international standardization of RFID tag identification protocols in partnership with Gemplus and TagSys companies. He writes scientific papers, book chapters, and patents, and he received the Best Paper Award at the Ninth International Conference on Personal Wireless Communications (PWC '04) and at the Second International Conference on Mobile Ad-Hoc and Sensor Networks (MSN '06). Since 2008 he has been the scientific chair of INRIA Lille Nord Europe Research Centre. He is an editor of several journals, including *IEEE Transactions on Parallel and Distributed Systems*.

AMIYA NAYAK (anayak@site.uottawa.ca) received a B.Math. degree in computer science and combinatorics and optimization from the University of Waterloo in 1981 and a Ph.D. degree in systems and computer engineering from Carleton University in 1991. He has more than 17 years of industrial experience, working at CMC Electronics, Defense Research Establishment Ottawa (DREO), EER Systems, and Nortel Networks in software engineering, avionics and navigation systems, simulation, and system-level performance analysis. He is on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *International Journal of Parallel, Emergent, and Distributed Systems*, the *International Journal of Computers and Applications*, *International Journal of Computer Information Technology and Engineering*, and *International Journal of Computing and Information Science*. Currently, he is a full professor in the School of Information Technology and Engineering (SITE) at the University of Ottawa. His research interests are in the areas of mobile ad hoc and sensor networks, fault tolerance, and distributed systems/algorithms, with more than 140 publications in refereed journals and conference proceedings.